SESSION

NOVEL SYSTEMS DESIGN AND APPLICATIONS + POWER EFFICIENCY AND MANAGEMENT

Chair(s)

TBA

Livestock Management System

James Foulkes[1], Peter Tucker[2], Mariflor Caronan[3], Rebecca Curtis[4], Leslie G. Parker[2], Chris Farnell[2], Brett Sparkman[2], Guoqing Zhou[2], Scott C. Smith[2], and Jingxian Wu[2]
Department of Electrical Engineering, Rose-Hulman Institute of Technology, Terre Haute, IN[1]
Department of Electrical Engineering, University of Arkansas, Fayetteville, AR[2]
Department of Electrical Engineering, Northern Arizona University, Flagstaff, AZ[3]
Department of Electrical & Computer Engineering, Missouri University of Science and Technology, Rolla, MO[4]
foulkejw@rose-hulman.edu, p.n.tucker@gmail.com, mdc259@nau.edu, rcr2f@mst.edu, lgparker@uark.edu, cfarnell@uark.edu, bsparkma@uark.edu, gzhou@uark.edu, smithsco@uark.edu, wuj@uark.edu

***Abstract*—With the rising demand of the already large cattle industry, new techniques are being employed to aid in the tracking and monitoring of cattle herds. The wireless monitoring system described in this paper sets forth the framework for a large scale monitoring system to aid in the health and wellbeing of such herds. The system employs individual ear tags on each cow that monitored important vital signs such as core temperature, heart rate, and blood oxygen levels. Static access points were used in the network to continually track individual cattle movement as well as relay cattle health data back to the farmer's computer. This low power wireless system was successfully constructed and tested with overall favorable results. Sensor data was successfully sent back to the computer to be displayed in a graphical user interface along with the positioning information determined through triangulation for each cow. Future alterations to the ear tags would provide a more reliable product for marketable cattle monitoring systems.**

Keywords- Bovine, Cattle, Ear Tag, Livestock Monitoring, Triangulation, Vital Signs, Wireless Sensor Network, XBee, ZigBee

I. Introduction

In 2010, the U.S. consumed approximately 26.4 billion pounds of beef, putting the retail equivalent value of the entire beef industry around 74 billion U.S. dollars [1]. For this level of consumption to be maintained, each cattle farmer must maintain an average herd size of around 100 or more heads of cattle [2]. Yet in 2010, due to factors such as bovine respiratory disease and other illnesses, the industry also suffered a significant loss of 1,234,500 cattle [3]. This results in considerable economic costs due to antibiotic treatment, losses due to death, and reduced herd performance.

II. Research Objective

In order to offer farmers an efficient method of managing their livestock from the comfort of their homes, this project aims to employ a low power wireless sensor network to relay health and location data from the herd of cattle back to the farmer's computer. Where a farmer may have difficulty managing the herd 24 hours a day, the developed system would be able to track and monitor the well-being of each cow continually, and report all data back to a central PC. Proposed sensors would monitor pulse rate, temperature, respiration, and location information, in order to alert the farmers of any abnormalities, such as cattle leaving the specified grazing areas, early signs of illnesses, critical levels of body temperature or heart rate, and many others concerns related to the well-being of cattle.

III. Five-Phase Research Approach

A. Phase 1

In order to develop a working knowledge of the problem faced by farmers, research was done to lay out a basic method for monitoring bovine vital signs and location, and then relaying that information back to the farmers. As the first phase in this project's five phase process, which guides the research from hardware development through the implementation stage, information on bovine health was obtained through online articles, published papers, and personal conversations with local farmers and animal science professors. Using information gathered from these interviews and articles, specific vital signs were chosen to be monitored using sensors to detect the early signs of disease.

B. Phase 2

In the second phase of this project, the focus was on defining device specifications and initial device selection. This includes sensors, mounting equipment, microprocessors, wireless transceivers, and a base station, all configured for low power consumption. Several factors were considered throughout this process, such as power management, hardware compatibility, cost effectiveness, intrusion to the animal, and ease of overall use.

C. Phase 3

The third phase of the project involved three simultaneous tasks to build a prototype of the system that transmits data from a cow through access points to a base station.

1) Development of mobile units and an ad hoc network

At the lowest level of the network, the wireless transceivers on the cow ear tags serve as the mobile units in the network. Together these mobile units form an ad hoc network through mesh networking, which automatically finds the shortest route to the nearest access point. The minimum distance routing scheme reduces the transmission distance of each datagram, which reduces noise and overall transmission errors in the system. Each mobile unit collects sensor information and an accompanying distance vector for the transceiver's routing

table to be framed with a small dedicated microprocessor, and transmitted out through the transceiver.

2) Development of access points and static network

The second task involved designing and building the access points to the basic service sets defined earlier as the ad hoc network. These access points collect the sensor information from the mobile units as defined through the ZigBee protocol and then route it through the extended service set to the base station. Again, each access point maintains a routing table defined using a distance vector routing scheme that allows the network to utilize the shortest route.

This process included a method for triangulation, which was done through the process of cell splitting. By setting the network up in this cellular fashion, the mobile units end up between at least three of the receivers at any given time. This allows for the towers to use signal strength measurements to triangulate the signal.

3) Implementation of a Graphical User Interface

The third task involved configuring a computer as a base station and building a graphical user interface (GUI). This receives the data from each mobile unit, displays it to the user in an easy to use format, and stores the data for use in a later phase of the project.

D. Phase 4

Due to the large area and scale of some ranching operations the extended service set must be completely composed of wireless units requiring their own power sources. The fourth phase of the project involved research and implementation of power systems for both the mobile units and the access points. Research was also done on energy production and storage systems for regulation and distribution of power to th,e transceivers and supporting components. An ideal power system does not require any replacement by the farmer for at least the lifetime of the cow, but the framework was set up for a failsafe mechanism to alert the farmer if the unit was close to failure or has experienced a power failure before the expected lifetime.

E. Phase 5

The final phase of the project was developing algorithms for data interpretation. This involved using the data from the mobile units to interpret when something is out of the norm, and alert the farmer. These alerts focus on issues such as cows escaping from the fence and early signs of disease. This makes the information easier to understand and use by making qualitative results from quantitative data.

IV. BACKGROUND

A. Measuring Cow Vitals

Previously, cattle health has been determined by visually assessing the cattle or a manual inspection by a veterinarian. Due to the amount of demand for beef and dairy cattle, it is essential for the farmer to have a quick and easy, as well as efficient, means of monitoring their cattle. In recent years, sensory devices have begun to be utilized to monitor cattle vitals such as temperature, heart rate, and respiration, since these have been determined to be the best indicators of early disease.

1) Temperature

One sensory device used to measure the cow's temperature is called a FeverTag, which is a tympanic thermometer device pinned to the ear with a probe inserted in the lower ear canal. This device flashes an indicator light when the temperature is greater than a set temperature such as 103.6°F [4]. Another sensory device used commercially is the CorTemp bolus, a large pill-like device, placed in a second stomach near the heart called the reticulum to measure core body temperature [5].

2) Heart Rate

Not only has the CorTemp bolus been used to measure core body temperature, but it has also been used to measure heart rate. This bolus has been designed to identify the beginning of each pulse using a small waterproof microphone so that the times between consecutive pulses can be determined and then converted into a pulse rate [6]. Heart rate has also been previously monitored using a Polar heart belt, which acquires an animal's heart vector using a standard set of electrodes. This makes the Polar heart belt impractical for long term usage [5].

3) Respiration

Not many sensors have been developed to directly measure the respiration of cattle, but one device that has been used to measure cattle respiration utilizes a thermistor attached to a nose stud in the animal's nostril. The temperature of the thermistor increases with respect to the ambient temperature as the animal exhales. The respiration rate can then be calculated by recording the number of times per minute the temperature rises and falls [7].

B. Tracking Unit Locations

The most common methods of guiding and tracking remote systems are based on the idea of triangulation. Triangulation is the process of determining the location of a point by measuring the time difference of arrival of a signal to three different receivers. Currently, the most common usage of a triangulation like technology is in GPS systems, which determine a position based on information from multiple satellites.

V. SYSTEM SELECTION

A. Cattle Temperature

Multiple temperature sensors were considered for measuring a cow's core body temperature through the ear, anus, or within a bolus. Due to the fact that the nearest location from the ear tag to get a reading for the core body temperature is the ear canal, a probe inserted at least two inches into the ear was the most convenient and least invasive method. The sensor chosen was similar to that used in the previously mentioned FeverTag. It was imperative that this probe be inserted into the lower ear canal deep enough for consistent readings without creating a nuisance to the cow. To prevent the probe from dislodging from the ear due to the cow's movement, the rigidity was strengthened by wrapping wire around the probe's length. In addition, the probe was selected due to it having an adjustable resolution for obtaining

a more accurate temperature and a waterproof housing to prevent contamination from the elements.

B. Reflective Pulse Oximeter

Low oxygen saturation of the hemoglobin or an abnormal heart rate may be early indicators of bovine illness. A common method of monitoring these vitals in humans is with a transmissive pulse oximeter. It is typically placed on the finger where it transmits light through the tissue, oscillating between an infrared light-emitting diode (LED) and a red LED, to a photo diode on the underside of the finger. Where the changing absorption of light from a single LED will indicate the pulsing of veins, two different wavelengths will be absorbed differently due to the oxygen in the blood, enabling the measurement of oxygen saturation by a light intensity comparison. The only feasible tissue on cattle to place a transmissive pulse oximeter would be the ear, however due to the hair on the backside that can alter data, a reflective pulse oximeter is necessary. The light from the LEDs is sent through the surface of the skin, where light reflects from the superficial vasculature back to the photo diode.

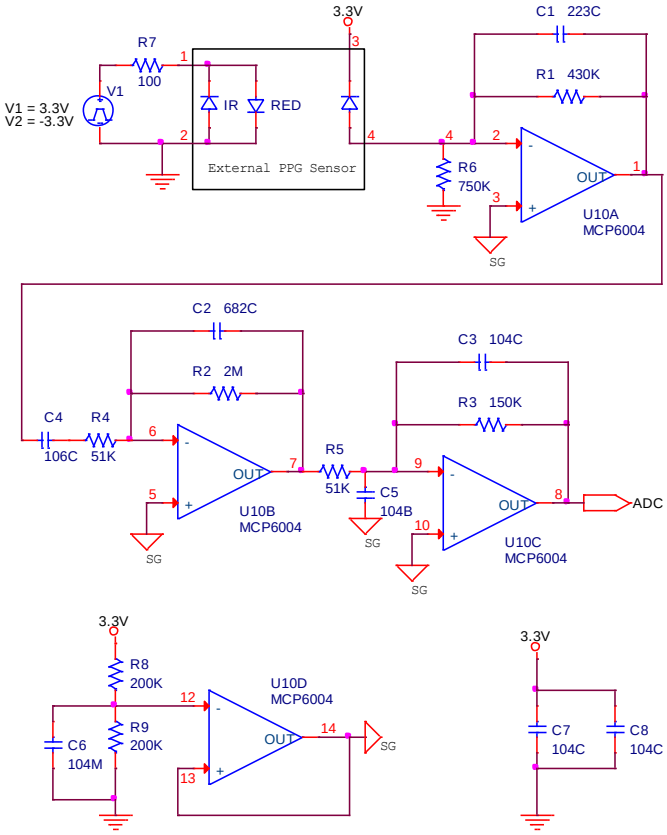


Figure 1 - Pulse Oximeter Schematic

The reflective pulse oximeter chosen featured a dual emitter and a photo detector embedded in a small chip, less than a centimeter squared in size and shown along with the sensing circuit in Figure 1. Since the device was intended for being used long term on a cow, the lower power and smaller size was more convenient for the ear tag.

C. Environmental Humidity Sensor

Humidity negatively affects the signal strength, which increased data loss in the communication between wireless components. After reviewing different sensors that measure outdoor temperature and humidity, a digital relative humidity and temperature sensor was chosen. The humidity level acquired on the farm can be used to examine the signal strength dissipation. Also, the recorded temperature data serves as a baseline for the cattle temperatures throughout the day.

D. Solar Panel and LiPo Rider

To run a remote sensory network the end points have to run off of a sustainable energy source since the nodes may not be easily accessible on a regular basis. This requires the nodes to incorporate a battery supply to sustain a level charge for the circuitry, while also having a method for recharging the batteries for extended operating durations to avoid unwanted disruptions. To recharge the batteries on the cows', solar panels were chosen over piezo-electric energy generation because solar power would give a predictable amount of power throughout the day for each cow, whereas any type of vibration or movement device would give off an unpredictable amount of power for each cow. Solar power was determined to be the only viable source of sustainable energy for the access points due to the impractical nature of other options such as wind power.

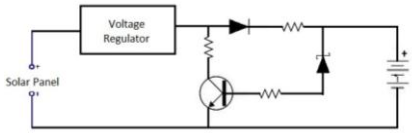


Figure 2 - Ear Tag Recharging Circuit

The solar panels on the ear tags were used to recharge small lithium ion button cell batteries through the circuit designed and shown above in Figure 2. The solar panels on the access points, however, were used to charge the batteries through a commercially available board called a LiPo Rider, which allows larger batteries to be charged. Each of these circuits allows the solar panels to run the transmitter directly in the event that the batteries are drained, and maintain a regulated voltage level.

E. XBee

In order to reduce the impact on the cows' normal activities, it was decided that all the sensorial information would be transmitted back to a central location through a wireless network. Due to the various sizes of farms, this wireless link could vary anywhere between a couple feet to a couple miles and be effected by all sorts of weather conditions. The XBee device, a wireless transceiver, was chosen because it works in all these scenarios with a maximum operating radius of two miles. XBees are also a good choice for this application because they are physically small, slightly larger than a quarter, and fit well within the space of an ear tag. The XBees also have a maximum transmitting power of 67mW, which makes them ideal for the low power endpoints of a remote sensor network. The conservation of power by these devices allows them to run in situations where they may not interact with people more than twice a year when the cows are being weighed and vaccinated. An added benefit of the XBee devices is that they follow the ZigBee protocol and have the

capability of setting up their own ad hoc networks, which made networking simpler.

Among the Xbee devices available, the programmable variant was chosen due to the sensor selection and special network requirements. The selected temperature sensors were one wire devices, which required the data collection to be done through the additional microprocessor on the programmable XBee devices. The ADC on this additional microprocessor was also used to sample the waveforms from the pulse oximeter's analog circuitry and place it into a digital frame.

VI. Design and Implementation

A. Communication Links

As previously mentioned, XBee devices work off of a ZigBee networking scheme that divides the mobile units in the network into three main categories as determined by their function in the network. These three categories under the ZigBee protocol are the coordinator, router, and end device.

Due to the definition of these components in the ZigBee protocol, the networks all followed the star topology layout, and at the center of this layout is the network coordinator. The network coordinator sets up the network and allows other units to join the network assigning them network addresses unique to the coordinator's internal routing table.

The only links usually made with the coordinator directly are through routers, which like coordinators never sleep. Routers also act as the link between ad hoc networks of end devices and the coordinator, relaying the data from each device to the coordinator, and likewise from the coordinator to the end devices.

End devices, however, do in fact have the ability to sleep for periodic cycles and communicate during their uptime. These end devices, the third part of the networking scheme, are the outer most part of the star topology, and ideally communicate through the routers to the coordinator or to each other.

B. Ear Tags

The main role of each ear tag, shown in Figure 3, was to collect information from the cow's vital sensors and relay those readings back to the coordinator at the farmer's computer. The data collected by the microprocessor on the programmable XBee was packaged together to send the combined packet back to the coordinator.

Whereas the main focus of the end devices is to collect data from the sensors and feed that information to the coordinator, the end devices also had to interact with the routers, or access points, so that the position of the cows could be triangulated. This was done through a frame acknowledgement to an AT request sent by an access point.

With large quantities of data being transmitted and received by the XBee devices, power management became an issue of great importance. To reduce the power consumption, the cows would be sampled only a few times an hour, during which time the XBees would be on. During the rest of the hour the XBee devices could be put in a sleep cycle so that the battery power would be conserved.

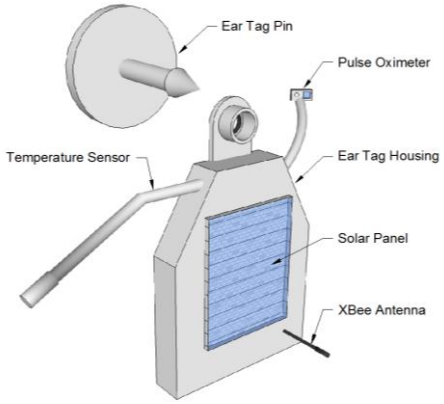


Figure 3 - Ear Tag Diagram

C. Access Points

Generally, the purpose of a router is to serve as a switch or a range extender for a network so that the devices can communicate more effectively. Yet, in this network application, the access points, which acted as routers, were placed in fixed locations and used to triangulate the cows' positions through a maintained line of sight transmission. Each of these routers serves as an access point to the basic service set by routing the sensor information back to the coordinator through the shortest transmission path, so that a connection is always maintained between the cow and the coordinator. These access points were intended to be laid out in a hexagonal format, as in Figure 4, to maintain the possibility for at least three connections at all times and aid in the setup of large triangulation schemes. The setup of the network in this fashion also allows for the distance attenuation curve to be generated dynamically at the beginning of each triangulation routine. Instead, for the scope of this project the main access point was built with a temperature and humidity sensor to add a loss term to the propagation loss model when building the reference curve.

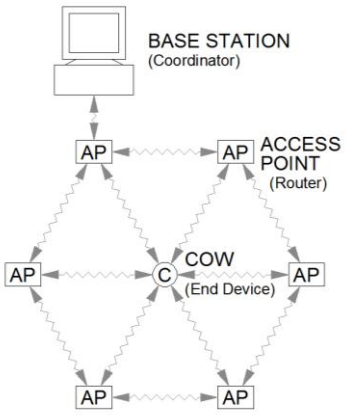


Figure 4 - Network Layout

D. Base Station

As the coordinator, the base station at the farmer's computer creates and hosts the network, which initially entails assigning every member of the network a 16-bit network address that could be used to create packet routes through the network. This position in the star topology doesn't have to be

filled by a programmable XBee, since the serial communication with the base computer could parse the frame and pull out the appropriate data packets. For the sake of speed, a programmable XBee was used to print out the values associated with each part of the data packet to the serial port along with a text header that could be parsed out as the values were read into the computer. As part of setting up the network, the coordinator sends the main access point the hardware addresses of the cows within the pasture, so that the main access point can then coordinate the AT-Command requests for the received signal strength from each of the routers to each of the cows.

E. Tracking the Cattle

When installing this system on a farm, the GPS locations of each access point are taken. Due to the fact that GPS coordinates don't account for the curvature of the earth, standard triangulation techniques give skewed results when using GPS coordinates directly. For this reason the Universal Transverse Mercator coordinate system is used to give a system of projected planar coordinates in terms of eastings and northings, which can be easily used to calculate the correct triangulated position. This conversion is done by the computer when entering the GPS coordinates into the computer as illustrated in Figure 5 below.

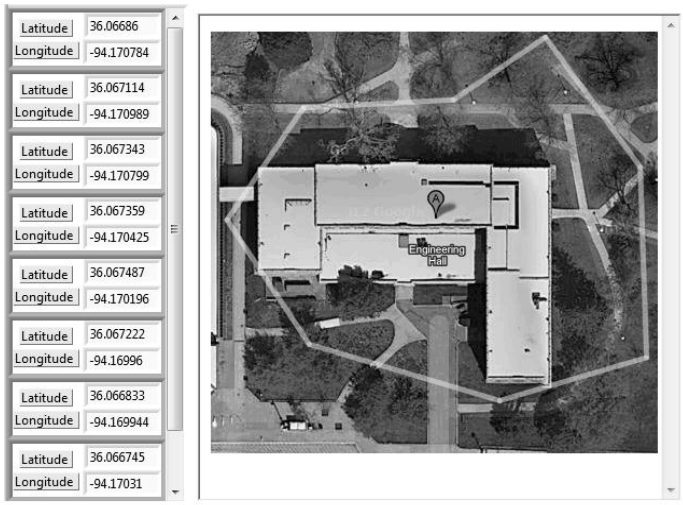


Figure 5 - Fence Parameter Setup

The equations used to compute the location of the cows in this project were derived by first making the assumption that around each of the access points was a sphere that had a radius determined by the strength of the signal it received. This distance is found empirically from a reference curve calculated during a calibration routine, as shown in Figure 6.

By solving the equations of each sphere for the point of intersection, the location of the common cow could be determined.

General form:

$$d^2 = x^2 + y^2 + z^2$$

$$d^2 = (x - x_a)^2 + (y - y_a)^2 + (z - z_a)^2$$

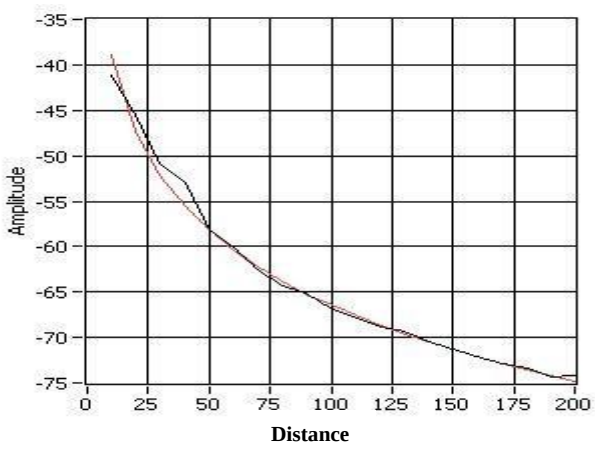


Figure 6 - Propagation Attenuation Curve

Inclusion of three circles, as height is not a factor:

$$d_a^2 = (x - x_a)^2 + (y - y_a)^2 \quad (1)$$

$$d_b^2 = (x - x_b)^2 + (y - y_b)^2 \quad (2)$$

$$d_c^2 = (x - x_c)^2 + (y - y_c)^2 \quad (3)$$

Subtract eq. 2 and 3 from 1:

$$d_a^2 - d_b^2 = (x - x_a)^2 - (x - x_b)^2 + (y - y_a)^2 - (y - y_b)^2$$

$$d_a^2 - d_c^2 = (x - x_a)^2 - (x - x_c)^2 + (y - y_a)^2 - (y - y_c)^2$$

Expand and collect terms:

$$d_a^2 - d_b^2 = (x^2 - x^2) - 2xx_a + x_a^2 + 2xx_b - x_b^2 + (y^2 - y^2) - 2yy_a + y_a^2 + 2yy_b - y_b^2$$

$$d_a^2 - d_b^2 = (x_a^2 - x_b^2) + (y_a^2 - y_b^2) + 2(x(x_b - x_a) + y(y_b - y_a)) \quad (4)$$

$$d_a^2 - d_c^2 = (x_a^2 - x_c^2) + (y_a^2 - y_c^2) + 2(x(x_c - x_a) + y(y_c - y_a)) \quad (5)$$

$$V_4 = \frac{(d_a^2 - d_b^2) - (x_a^2 - x_b^2) - (y_a^2 - y_b^2)}{2} = x(x_b - x_a) + y(y_b - y_a)$$

$$V_5 = \frac{(d_a^2 - d_c^2) - (x_a^2 - x_c^2) - (y_a^2 - y_c^2)}{2} = x(x_c - x_a) + y(y_c - y_a)$$

$$V_4 = \frac{(d_a^2 - d_b^2) + (x_b^2 - x_a^2) + (y_b^2 - y_a^2)}{2}$$

$$V_5 = \frac{(d_a^2 - d_c^2) + (x_c^2 - x_a^2) + (y_c^2 - y_a^2)}{2}$$

$$V_4 = x(x_b - x_a) + y(y_b - y_a)$$

$$\frac{V_4 - y(y_b - y_a)}{(x_b - x_a)} = x$$

$$V_5 = x(x_c - x_a) + y(y_c - y_a)$$

Substitute in for x:

$$V_5 = \frac{V_4(x_c - x_a)}{(x_b - x_a)} + y\left(-\frac{(y_b - y_a)(x_c - x_a)}{(x_b - x_a)} + (y_c - y_a)\right)$$

$$V_5 - V_4\left(\frac{x_c - x_a}{x_b - x_a}\right) = y\left(\frac{(y_c - y_a)(x_b - x_a) - (y_b - y_a)(x_c - x_a)}{(x_b - x_a)}\right)$$

Solve for y:

$$y = \frac{V_5(x_b - x_a) - V_4(x_c - x_a)}{(y_c - y_a)(x_b - x_a) - (y_b - y_a)(x_c - x_a)}$$

Substitute solution for y into equation for x:

$$x = \frac{V_4 - y(y_b - y_a)}{(x_b - x_a)}$$

This position is then logged in the computer and saved so that the interactions of sick cattle can be tracked back to determine how the disease has spread and which cattle may be affected by the illness.

F. Power Supply

When defining the battery supply required by each member in the network, several key factors had to be taken into account. Among these factors were relative network usage by each of the network members as well as the availability to renewable energy. For example, the access points were not constrained by size or weight, so they were free to have larger batteries and solar panels for recharging the batteries. The ear tags, however, were constrained by both size and weight requirements so that they would not cause ear drop in the cattle, which presents false signs of depression. Also, the solar panel chosen to recharge the batteries had to be powerful enough to supply the required current, but had to fit on the front of a standard ear tag to prevent excessive damage.

Access points handle the majority of the traffic in the network as they coordinate the AT requests required for triangulation as well as serve as the link between the end devices and the coordinator. For this reason, the batteries in the access points were chosen to meet the daily requirements for transmission in the event that the solar panels failed to recharge the battery at all. The value used to determine the relative transmissions for the day was calculated from the rated transmission power of the XBee devices and the total number of projected transmissions in a lossy environment with an equivalent projected error rate. As a result of having a larger battery to fill this demand, a larger solar panel was chosen to charge the battery through the LiPo rider board.

In addition to the ear tags being constrained by size and weight, they also sent and received fewer packets than the access points individually. As a direct result of this, the batteries could be smaller and therefore lighter weight. Yet, by having a smaller battery, the circuit has a smaller reserved power supply and relies more heavily on the ability for the batteries to be recharged periodically. A solar panel, being the most portable and reliable method of recharging the battery, is still limited by the need for sunlight to create current flow, which cannot be controlled on a cow that moves into shade to cool off. For this reason a balance was attempted to be made between the daily demand for transmissions by the ear tags and the amount of average solar energy projected to be harvested during the daylight hours.

G. Software Integration

The Graphical User Interface (GUI) provides the farmer an easy to use method of viewing the statistics of the herd and logging the data along with additional information pertaining to symptoms and treatments regarding the cattle.

1) Live Data Display

As data is read into the GUI it is displayed on a front panel with each cow's position in the field as shown in Figure 7. A scrolling capability is provided to display each cow's most recent statistics in the table above the field, as also seen in Figure 7. This function can also be accessed by moving the cursor over the desired cow. By clicking on each cow, the individual statistics and history are displayed in a new window as shown in Figure 8.

In the front panel, when a cow's vital falls outside of the norm, the color of the corresponding dot changes from green to yellow, representing the cow's current status and warning the user of possible ailment. Similarly, a cow's status color will change to red and send an alert to the user when its condition is critical and needs to be manually inspected for illness.

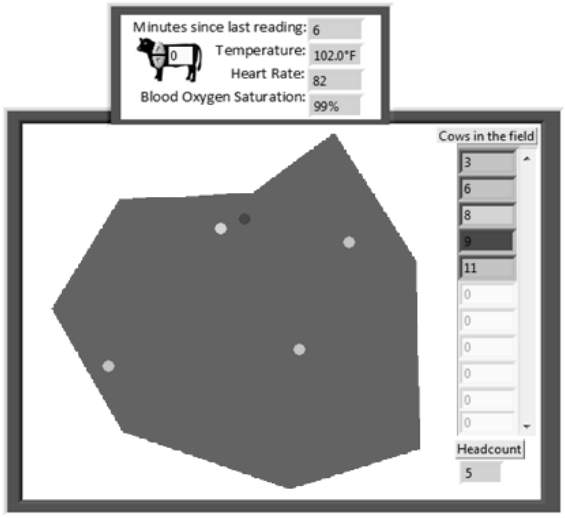


Figure 7 - Herd Monitoring Window

In the case that more than one cow's vitals are falling outside the norm, the GUI enables the user to track the history of the cows' movements. This could help determine possible points of contamination such as a similar watering hole or food source.

2) Data Logging

In order to provide a useful account of a cow's medical history, vitals are saved hourly for a seven day period. All proceeding vitals measured and recorded have a daily average going back for three months. Along with each individual cow's information, an average of the entire herd is recorded in a similar method. Saving the herd's average vitals allows the user to compare that of the individual to a norm of others in a similar environment, in addition to comparing cow temperatures with the corresponding outdoor temperature as illustrated in Figure 8.

Many situations may be presented where the farmer has a need to share the data he has logged in the program with others that don't have access to the cow's history. For instance, when the farmer interacts with a veternarian or a member of the Center for Disease Control, there is a method for printing out the health data associated with each cow as well as where the cows have been. This data is presented in an easy to read format that could also be given to buyers upon

purchase of the cattle to ensure that the health information can be tracked in a responsible manner.

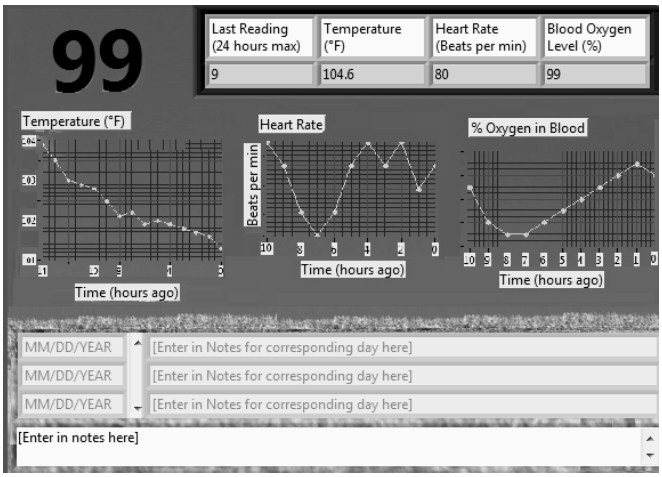


Figure 8 - Individual Cow Statistics

VII. Results and Discussion

Overall, the system worked as intended, although some alterations could be made for improvement. The tympanic temperature sensor gave accurate and consistent readings through all tests. However, when tested in the field on actual cows, the probes tended to fall out of the ear canal easily due to their unpredicted and excessive ear movements. Future models should incorporate thinner, longer, and stiffer probes to prevent dislodging due to ear movement. Probe length should change per individual cow due to differing ear sizes and tag placement.

The pulse oximetry sensor on the ear tags went through several stages of development. A pulse oximeter was built with 5 millimeter LEDs and a similar sized photodiode, which was run through a basic current-to-voltage converter and single stage low pass filter to give a recognizable pulse waveform for both light frequencies to be sampled by the microprocessor; however, this circuitry would have had to have a multiple stage filter to add to the already large LED diode combination. For this reason a smaller LED diode integrated sensor was obtained from APMKorea to reduce the overall footprint of the device. This sensor supplied a signal to a four stage op-amp filter supplied along with the device, as shown in Figure 1. Despite much debugging and redesigning the circuitry accompanying the device in an attempt to better tune the filtered signal, the sensor failed to yield a consistently recognizable pulse waveform.

Field tests were initially done on creating an empirical propagation loss model to be utilized in triangulating the location of the ear tags. This test consisted of creating a curve from received signal strength at known distances from the receiver and then fitting this data with a logarithmic curve as shown in Figure 6. This allowed for future alteration of the curve due to other loss factors such as changes in relative humidity. When the data gathered from the received signal strengths were gathered from the access points, the fitted curve was interpolated to find the distance that corresponded to the respective signal strength. By this method the value for the relative distance between the ear tag and the access point was determined and then used in the triangulation equations in Section VI.E to determine the actual location of the ear tag. This process was done during testing at a higher resolution by using an average of data from combinations of three access points from a set of five unique access points. Based on this test the averaged result gave us the location of the ear tag within a range less than a foot.

VIII. Future Direction and Conclusion

To make this product more useful to farmers, other sensors could also be incorporated into the monitoring network to help track additional factors not considered in the scope of this project. For example, an XBee could be incorporated into the farmer's scale to track individual cattle growth through the early stages of its life. RFID chips could also be integrated into the system for additional benefits. In many areas, RFID tags are already utilized to track cattle through the transactions and transfers between farms. The incorporation of RFID tags in this network device would allow the health information of individual cows to follow them from farm to farm. This would allow all health information to be uploaded and tabulated in a large database that could aid in the tracking of bovine diseases throughout the cattle industry and improve the cattle vaccination process.

This wireless cattle monitoring project proved to be a viable proof of concept design that could aid in the tracking of health in cattle herds. Used by both farmers and researchers, a system based on this networking scheme could be utilized to more fully understand the health and natural patterns associated with cattle and many other species. This could one day be used as a vital link in a system of tracking and prevention of disease among livestock on a national level.

IX. References

[1] (2012, April 25). *U.S. Beef and Cattle Industry: Background statistics and Information* [Online]. Available: http://www.ers.usda.gov/news/BSECoverage.htm

[2] (2009, Sept. 28). *Cattle: Background* [Online]. Available: http://www.ers.usda.gov/Briefing/Cattle/Background.htm

[3] (2011, May 12). *Cattle Death Loss* [Online]. Available: http://www.usda.mannlib.cornell.ude/MannUsda/viewDocumentInfo.do?documentID=1625

[4] J. T. Richeson *et al.*, "Evaluation of an Ear-Mounted Tympanic Thermometer Device for Bovine Respiratory Disease Diagnosis," Arkansas Animal Science Department, Fayetteville, AR, 2011, pp. 40-42.

[5] K. Smith *et al.*, "An Integrated Cattle Health Monitoring System," in *EMBS Annual International Conference*, NY, 2006, pp. 4659-4662.

[6] A. Martinez *et al*., " Ingestible Pill for Heart Rate and Core Temperature Measurement in Cattle," in *Engineering in Medicine and Biology Society Annual International Conference*, New York City, NY, 2006, pp. 3190-3193.

[7] L. Nagl *et al.*, "Wearable Sensor System for Wireless State-of-Health Determination in Cattle," in *Engineering in Medicine and Biology Society 25th Annual International Conference*, Cancun, MX, 2003, pp. 3012-3015.

An Emergency Medical Service in IPv6 Network Environment

Tseng-Yi Chen[1], Hsin-Wen Wei[2], Ying-Jie Chen[1], Wei-Kuan Shih[1]
[1]Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
[2] Department of Information Management, Tamkang University, Taipei, Taiwan

Abstract - *Internet of Things (IoT) has been widely studied over the past decade. Recently, many research results of IoT, related to emergency system, smart building and medical system, are published. The key research issue of IoT applications is the ability to interact with physical world through computation, communication, and machine control. The emergency medical service is a very suitable application for IoT environment. Hence, we develop an emergency medical service based on IPv6 network environment. Emergency medical service is constructed by the biosensors, the healthcare information system and the notification component. The emergency medical service will process raw biological sensing data in bio-sensor module and show system's end-user the meaningful bio-information to notify them. In advance, medical service system also adapts the features of a speaker module. The role of the speaker module is regarded as a health alarm device. The emergency medical service can diagnose user's health according to the pulse information collected from biological sensor. Alert sound will be broadcasted by speaker module, if the emergency medical service detects abnormal situation from the sensing data. After alert sound is played, the speaker plays audio of each step according the standard first aid procedures, to tell peoples who are near to patients what to do. So, patients can obtain first aid operation from the near person to have their life saved. One additional problem need to be studied is the devices control. Most medical devices need to be connected to the internet. However, the number of IPv4 addresses is not enough. It is impossible to allocate IPv4 address to all wearable medical devices. Hence, we introduce IPv6 stack into the medical devices in order to realize the machine to machine (M2M) conception of the internet environment.*

Keywords: IPv6, emergency system, wearable sensor, healthcare system.

1 Introduction

The research of the physical sensor device has garnered increasing attention owing to its technical importance in widespread applications, such as monitoring and surveillance in the military and biological detection [3,4]. Two critical issues for the development of sensor systems are the construct of innovative application and the solution of machine to machine communication. In this paper, we develop an innovative emergency medical service system. Our medical service system contains three parts: hardware, software and cloud. This medical service system integrates biosensor and speaker module into the hardware platform. The medical diagnosis service is constructed in cloud platform and the medical service application is developed on Android system. Our medical service system provides online health diagnosing service and real-time condition feedback by the medical device. If user is in the emergency status, the medical system will notify the wearable device and the wearable device will help people who are near the patient to operate basic first aid guided by speaker module when the monitored person pass out.

Sensor applications originally entail components that interact with each other through complex network protocol. Hence, proper IP protocol can reduce the problem introduced by complex network protocol. Therefore, in this research, we also port IPv6 stack [1,2] to wearable medical device. With the help of the IPv6 stack, the medical device has the ability to communicate with handheld device or cloud service server via Internet environment.

In summary, this paper has two contributions in technical and research aspect.

Technical: In this research, we propose an innovative medical service application. The application can diagnose the monitored people's status according to sensing information from the biosensors. This system also gives a feedback to monitored people according to the result of diagnosis.

Research: We port IPv6 stack to medical service device. Based on IPv6 stack, we also develop a medical service system on these devices. Therefore, all Biosensors connect to medical service system through IPv6 Protocol. The healthcare application is developed on Android device and can control the medical device through Internet protocol. The communication between application and medical device is the machine to machine concept and is the important research issue for Internet of Things.

2 Related work

Traditionally, many researches put efforts on setting up a monitoring system inside the buildings. The research [6] reports their experience on the implementation, deployment, and operation of an indoor environmental monitoring network using WSN called SensorScope. To make the sensor nodes having long lifespan and working more efficient, some researchers [7] propose a clustering-based network specific for building monitoring. They also show that WSN is conducive to build environment monitoring system. Some applications that use WSNs for smart building are mainly for the energy saving purpose [8].

Nowadays, the deployment of a building control system is getting complicated due to different communication standards exist. In general, wireless sensor networks do not equip IP protocol in their system; therefore, different network routing algorithms are deployed. So, to connect their application system into IP-based internet requires the establishment of protocol translation stack at the boundary between internet and WSN. The translation stack transforms communication packet between non-IP packets of the WSN group and the IP packets in internet. The translation stack always stores a database of the local sensor node id and the outgoing TCP/IP port. The system need to inquire the database for delivering the message to destination machine when it receives a control message and sends a collection data to sink system.

There are also some solutions for solving the problem that variety of WSN standards are used for monitoring and saving energy in smart buildings. Some researchers [9] integrate WSNs with different service communication into a 'knowledge and information services' platform to support energy management feature, which can be accessed via a Web service to support inhabitant actions for reducing energy demand. In [10], the authors not only present a web services-based approach to integrate resource constrained sensor and actuator nodes into wireless sensor networks, but also implement an API to access services on sensor nodes. Their APT follows the architectural style of representational state transfer (REST). All research mentioned above do not take account of IP convenience for wireless sensor network applications.

Therefore, many research and standard groups [5, 15, 16] try to investigate and define the standard of IP stack in wireless sensor network. Sensor networks with IP support would eliminate the difference between WSN local area network (LAN) and Internet wide area network (WAN). Some research has been proposed in the IP-based WSN, e.g. mechanisms, security, and protocols. The concept of IP stack for smart object is proposed by IP for Smart Object (IPSO) Alliance [11] in 2008. After the IPSO concept is proposed, IP-based sensor network have been implemented [12, 13, 14] recently. These works include monitoring systems, healthcare applications and smart buildings. Those applications are based on IP transmission protocol. But they do not consider the issue that IP addresses is not enough. IPv6 protocol can solve this problem.

There are many WSN OS can be selected for porting, those are TinyOS, LiteOS, and Contiki. The most popular OS for WSN is TinyOS. TinyOS [17] is based on an event-driven programing model. TinyOS signals the event handler to deal the event when external trigger occurs. However, the programming language for TinyOS, that is NesC, is not commonly used by general programmer. For this reason, we choose Contiki OS as the operating system on our target platform.

Contiki OS[18] is a small, open source and multitasking operating system. It is designed for embedded systems with small memory size. Contiki OS only consumes 2 kilobytes of RAM and 40 kilobytes of ROM. Contiki OS is also an event-driven kernel and on top of which application programs is dynamically loaded and unloaded at runtime. Figure 5 shows the architecture of the Contiki OS.

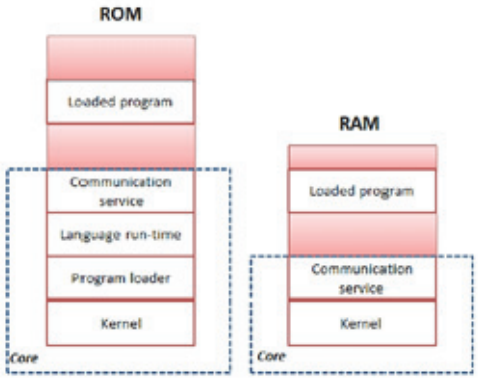


Fig. 1. Contiki system architecture

Contiki's kernel is a light-weight kernel. Contiki OS can transmit the sensing data or forwarding message through communication service. The communication service is an important service in safe building application. We build up TCP/IP stack based on the service. After choosing the operating system, this research uses uIP stack as the communication stack in WSN application.

The uIP [19] is an open source TCP/IP stack for wireless sensor object. It is widely used in the embedded systems and several platforms. It also provide IPv6 stack for wireless sensor device. After survey of embedded TCP/IP stack, we choose uIP to build on the target system. It is a light weight TCP/IP protocol in WSN applications.

Hence, we propose an integrated IoT application solution based on concept of IP-based enable WSN that includes porting uIPv6 stack to sensor object, sensor

information management system development, and speak module integration for application of escaping from disaster environment.

3 System design

The medical service system is composed of hardware platform and software stack. This section introduces the hardware and software component in emergency medical service system.

3.1 System architecture

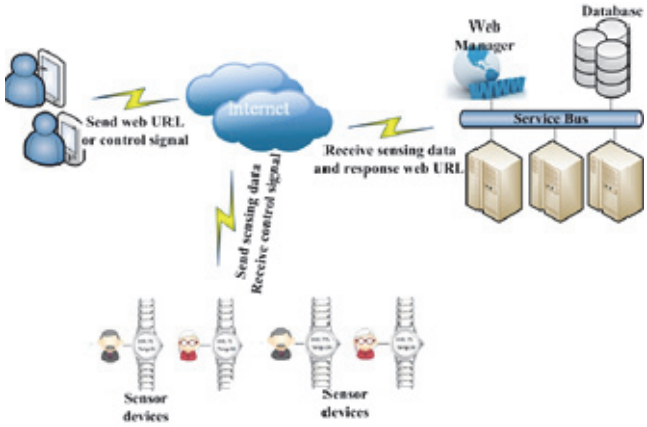


Fig. 2. Medical service system architecture

Figure 2 shows the system architecture of our emergency medical service system. We can see that our medical service system is composed of wearable health monitor device, medical service system in cloud platform and healthcare application on handheld device. The medical device system also integrates GPS module. The GPS module can locate the location of the monitored people. If the monitored people pass out, the medical service system will activate the GPS module on medical device immediately and the GPS module reports patient's location to medical service system. The medical service system will calculate a shortest path from hospital to patient's location and find the hospital that is nearest to the patient. Then, the medical service system sends the calculated path to the hospital to save the time for picking up the patient. The hospital system integration is in our future work

3.2 System Component Introduction

Hardware platform: The emergency medical service system is constructed by Piezo Buzzer sensor, temperature sensor, Speech Synthesizer Bee speaker module and ARM-based main board platform.

The Piezo Buzzer sensor is driven by square waves. It is cheaper than other pulse sensor. However, the Piezo Buzzer sensor will generate more noise in pulse detection. Hence, our medical system employs a suitable noise filtering algorithm to make pattern of pulse detection become normal. The speaker module used in our system is Speech Synthesizer Bee. It is pin-compatible with the Xbee and can be easily plugged into the sensor platform. This module is connected with Octopus II platform through UART interface.

Software stack: Our software stack is composed of system kernel, library layer and application level. We modify the system kernel in Linux in order to integrate some biosensors into embedded medical system. We also port IPv6 stack into embedded medical device. In application level, we develop a healthcare app on Android system. The healthcare apps can directly communicate with embedded medical device through IPv6 network.

4 The emergency medical service

The emergency medical service system must include wearable medical device integration, medical service deployment on cloud platform and application designed on mobile device, and the emergency medical service system based on IPv6 network.

4.1 Wearable monitor device

The wearable monitor device is combined with three components: biosensors, speaker module and ARM-based main board. The biosensors are connected to ARM-based main board with UART and IIC protocol. The biosensors include Piezo Buzzer sensor, temperature sensor and pressure sensor. These biosensors can detect the biological information for human beings. The wearable monitor device will send the detected biological information to medical service implemented on cloud platform. The speaker module is integrated with ARM-based main board through UART interface. The speaker module can play alert sound to help the monitored people to realize their health condition.

4.2 Medical service on cloud

We deploy the noise filtering algorithm and medical diagnosis system on cloud platform. The noise filtering algorithm helps our medical service system adjusting and correcting the biological information detected by biosensors. The noise filtering algorithm can help our medical diagnosis system to get more precision diagnosis. If the user is in bad health condition, the medical service sends alert to wearable device and the device notify the user the current situation by speaker. If the monitored people pass out, the medical service will send the first aid procedure step by step to wearable monitor device and also delivers the path information between the patient's location and the nearest hospital

4.3 Application designs

The application is developed on Android system. The functions of our application include real-time pulse data

monitor, health condition in history view and the wearable monitor device system configuration. The application communicates with the wearable monitor device and medical service platform through IPv6 stack

5 System demonstration

First, we describe the application scenario of our medical service system. Then, our demonstration will show the wearable monitor device with execution result, the medical service system, and the healthcare application.

5.1 Application scenario

Here are two demo cases in our demo scenario

5.1.1 The wearable monitor device could save patient's life.

The user needs to wear the medical device, such that the wearable medical device can save his life when he passes out and notify other people around the patient by speaker module in wearable device. Note that, the medical device is integrated with a speaker module and can plays first aid operation step by step when the user is in the emergency status.

5.1.2 The medical service system can conserve time to hospital.

Our wearable device equips a GPS module. Therefore, the medical system can send the location of the patient to the ambulance as soon as possible

5.2 System evaluation

5.2.1 Health monitor platform

Figure 3 shows the health monitor platform .The wearable monitor device could detect the status of people's pulse, temperature and pressure. We also integrate a speaker module into the wearable device. And the wearable device will sends the biological information of the user to the medical service platform and the healthcare application.

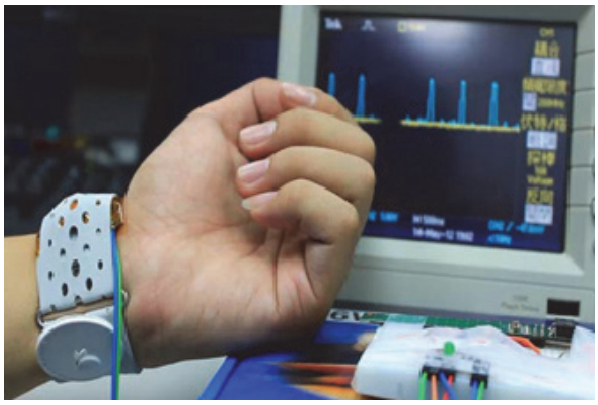


Fig. 3. The wearable monitor device

5.2.2 Medical service on cloud

We deploy our medical service on cloud platform. The medical service includes noise filtering in biological data sensing, ECG information presentation and medical diagnosis. Our medical service also could give user feedback about their health condition. Figure 4 shows our medical service system. The server display last ten records of pulse detection and shows the real-time biological information of the GUI lefthand side. User can query specific history information of the GUI righthand side

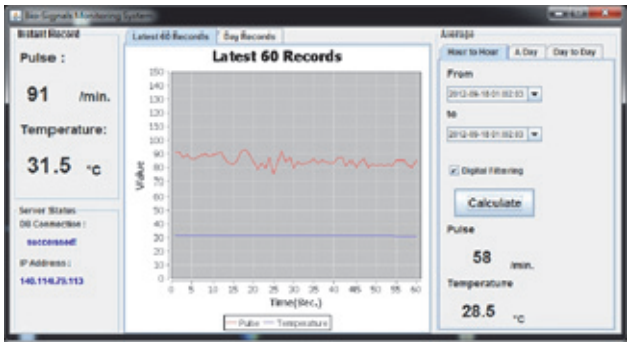


Fig. 4. The medical service on cloud platform

5.2.3 Healthcare application

Fig. 5. Healthcare application demonstration

We also develop a mobile application on Android system. Users can monitor real-timeEGC information in

application and change the view of the history data list. The application also provides the configuration of the wearable device system. All of the information communication in the application is on the IPv6 network. Figure 4 shows our healthcare application on Android system.

6 Conclusion

Handheld devices become indispensable equipment in the daily life of the people. Hence, we propose a complete medical service system. The medical service system includes wearable monitor device, medical system on cloud and healthcare application. And the medical service system is also built in IPv6 network environment. The medical service integrate speaker module to provide feedback information from medical service system. This is an innovative application in biological information system.

7 Acknowledgement

We would like to thank the National Science Council of the Republic of China (Taiwan) for financial support of this research under contract numbers NSC 101-2221-E-007-128-MY2, NSC 101-2219-E-007-007 and NSC 101-2221-E-032-067.

8 Reference

[1] S. Deering and R. Hinden, "Internet Protocol, version 6 (IPv6) specification," RFC 2460, 1998

[2] Adam Dunkels, "Full TCP/IP for 8-bit architectures," in Proceedings of the 1st international conference on Mobile systems, applications and services, MobiSys'03, pages 85-98, 2003

[3] I Lee, O Sokolsky. "Medical Cyber Physical Systems", 47th ACM/IEEE Design Automation Conference (DAC), pp. 743-748, 2010

[4] S Kannan, "Wheats: A Wearable Personal Healthcare and Emergency Alert and Tracking System", European Jounal of Scinetific Research, Vol 85 No. 3, pp 382-393, Sep. 2012.

[5] G. Montenegro and N. Kushalnagar, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", draft-ietf-6lowpan-format-01, Internet-Draft, work in progress, October 2005

[6] Schmid Thomas, Dubois-Ferrière Henri and Vetterli Martin, "SensorScope: Experiences with a Wireless Building Monitoring Sensor Network," Workshop on Real-World Wireless Sensor Networks (REALWSN'05), Stockholm, June 2005

[7] Qifen Dong, Li Yu, Huanjia Lu, Zhen Hong, Yourong Chen, "Design of Building Monitoring Systems Based on Wireless Sensor Networks," Wireless Sensor Network, Vol.2 No.9, PP.703-709, September 2010

[8] Yuvraj Agarwal, Bharathan Balaji, Rajesh Gupta, Jacob Lyles, Michael Wei and Thomas Weng, "Occupancy-Driven Energy Management for Smart Building Automation," Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy- Efficiency in Building, pp. 1-6, 2010

[9] Hanne Grindvoll, Ovidiu Vermesan, Tracey Crosbie, Roy Bahr, Nashwan Dawood and Gian Marco Revel, "A wireless sensor network for intelligent building energy management based on multi communication standards – a case study", ITcon Vol. 17, pg. 43-62, 2012

[10] Lars Schor, Philipp Sommer and Roger Wattenhofer, "Towards a Zero-Configuration Wireless Sensor Network Architecture for Smart Buildings," Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, pp. 31-36, 2009.

[11] A. Dunkels and J. Vasseur. "IP for smart objects alliance.", Internet Protocol for Smart Objects (IPSO) Alliance White paper No.2, September 2008.

[12] M. Durvy, J. Abeill´e, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A Dunkels. "Making Sensor Networks IPv6 Ready." In Proceedings of the 6th ACM conference on Embedded network sensor systems, pp. 421-422, 2008.

[13] J. Hui and D. Culler. "IP is Dead, Long Live IP for Wireless Sensor Networks." In Proceedings of the 6th international Conference on Embedded Networked Sensor Systems, Raleigh, North Carolina, USA, November 2008.

[14] B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao. "Tiny web services: design and implementation of interoperable and evolvable sensor networks." In Proceedings of the 6th ACM conference on Embedded network sensor systems, pp. 253-266, 2008

[15] IETF working group 6lowpan, "IPv6 over Low power WPAN (6lowpan)", Available: http://www.ietf.org/html.charters/6lowp ancharter.html

[16] Adam Dunkels, "The Contiki Operating System 2.x", Available :http://www.sics.se/~bg/telos/html/

[17] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, and David Culler. "TinyOS: An operating system for wireless sensor networks". In W.

Weber, J. Rabaey, and E. Aarts, editors, Ambient Intelligence. Springer-Verlag, New York, NY, 2004.

[18] Dunkels A., Björn Grönvall and Thiemo Voigt. "Contiki - a lightweight and flexible operating system for tiny networked sensors." In LCN ’04, pages 455– 462, Washington, DC, USA, 2004. IEEE Computer Society.

[19] Adam Dunkels, "uIP - A Free Small TCP/IP Stack," 2002. http://www.chuandong.com/cdbbs/upload/2007-10/11/20071011951453508.pdf

Design of a FPGA-based Virtual Bus Interface System for Reducing IOs in Hardware-in-the-Loop (HIL) Real-time Simulation

K. Strell, D. Mackellar, and Y. Jung
Electrical and Computer Engineering, Gannon University, Erie, PA, USA

Abstract - *The complex embedded systems of locomotives require a vast amount of the Hardware-in-the-Loop (HIL) testing before manufacturing. Various simulation efforts are required to commit testing of the target embedded systems. Therefore, development of the most effective systems is necessary for efficient and accurate verification of the embedded systems. Since most of the verification is complete on the software, simulator seldom uses behaviors of IO operations, which are critical in real-time applications including embedded systems for locomotives. This paper proposes a bus interface system to replacing the existing massive IO cards in an embedded system integrated in a locomotive simulator for providing reconfigurable extension while maintaining required IO specifications including IO access speed and signal fidelity. The developed bus interface system is integrated to a modified Modbus (FEBus) and the main simulator computer. The bus interface system is implemented with FPGA including the NIOS II processor core with associated other IP cores. In order to evaluate capability of the HIL simulation with the developed system, a couple of interface protocols, such as PCI over a PMC connection and Gigabit Ethernet UDP/IP, are thoroughly tested.*

Keywords: bus interface system, hardware-in-the-loop simulation, real-time simulation

1 Introduction

A control system of a contemporary locomotive comprises of heterogeneous embedded systems that communicate with each other to accomplish the desired complexity and accuracy for concurrently driving numerous actuators on the locomotive. The employed actuators are ranged from power electronic circuits to display screens. The actuators are required for sharing data between the embedded systems integrated on the locomotive to operate the desired functions.

The interface between the embedded systems is a means for efficient control and user interaction. However, it also increases the complexity of the system verification. In particular, error propagation at the system level verification is significant because an error at the system level could be often influenced by multiple components in its subsystems. Thus, inter and intra function verifications via the interface are necessary for verification of the embedded systems [1].

Given the complexity of the embedded systems, HIL simulators are the most efficient means of validating the hardware and software of these systems. However application specific simulators such as [2] and [3] would require an infeasible number of simulators to cover all of the embedded systems and would make it extremely difficult to simulate the systems together. Therefore in the locomotive simulators used currently the embedded systems in question become the UUTs and the other pieces of the locomotive are simulated in software with electrical signals interfacing the physical components to the modeled components [4].

In order to bear such increasing complexity at the system level verification, the simulators are required to cover the entire embedded systems under the same or compatible environment. In particular, our target system, a locomotive, comprises various heterogeneous embedded systems that are integrated through several partitioned interface hardware systems. Furthermore, the number of components employed in each subsystem proportionally raises the complexity and cost of the simulations.

Building a simulator is a daunting task in terms of its development time and cost. Therefore, including application specific features into a simulator is not preferable. Consequently, one of the best solutions to this issue is to maintain the simulator's capability with little or no changeability. This was evident in the first attempt at reducing IO, the Slim Sim in which the main modified Modbus (FEBus) device, the Consolidated Input Output (CIO) unit was removed completely. This yielded a simulator with reduced IO but also reduced functionality.

After the full limitations of the Slim Sim were understood there stood two distinct paths that could be taken to obtain the desired IO reduction while also maintaining the testing integrity. Both of the options involved removing at least part of the FEBus devices in the simulators. The first option was to virtualize the entire FEBus device so the control software would be run on a modeled device [5]. The second option would be to use a bus interface card [6] to mimic the IO cards on the FEBus of the CIO leaving the processing card intact to run the control software.

Section 2 describes the architecture and operation of the bus interface system. Section 3 expresses evaluation of the bus interface system with the simulation results and analysis of the simulation results. The conclusions and future work are depicted in Section 4.

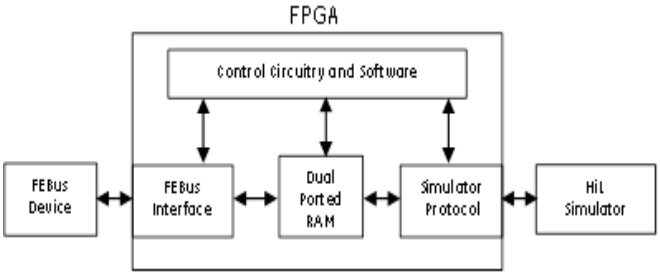


Figure 1. A Block Diagram of the FPGA-based Virtual Bus Interface System

2 Architecture of the Virtual Bus Interface System

As addressed in the previous section, the "Slim Sim" was created by eliminating the CIO unit. The SlimSim forces the control system on the locomotive to run in a reduced mode, which recognizes the CIO unit in a disabled state. The reduced mode in the simulator prevents some components from simulating at the component level. The simulator, however, retains its systems level simulation capability. More specifically, the simulator yields the ability to perform a full HIL simulation with a reduced cost if the IO portion of the CIO unit is removed but the processing card is remained.

The FEBus interface card, shown in Figure 1 as a block diagram, is composed of four major components: the FEBus interface, dual-ported RAM, simulator protocol, and the control circuitry and software. The FEBus interface is responsible for responding to the FEBus master, reading and writing the RAM when it is instructed. The simulator protocol, either the PCI interface or UDP Ethernet, is the data manager for the simulator side. It moves data according to IO point, writing outputs from the simulator into the RAM and reading inputs to the simulator from the FEBus data in RAM. The RAM is used to hold all the data for the IO points that would be controlled using the IO cards in a full HIL setup. The control circuitry and software consists of the NIOS II processor running software to manage the DMA moving data to and from the simulator protocol. In addition, it runs the software required to operate the simulator protocol, either a TCP/IP stack for UDP or a PCI driver for the reflective memory card.

2.1 FEBus for Virtual Bus Interface

After executing a simulator without CIO, it was evident this approach is a feasible solution or maybe one of the best ways for simplification. Although the simulator could be alleviate the signal congestion during the simulation, involving CIO throughout the simulation prohibits the simulator from its intended simulation capability. We realized that a means to include the processing component of the CIO unit could be a viable solution while removing the interface cards from the simulator. In the current practice, a simulator directly accesses the FEBus for reading and writing data to and from during simulation. Thus, the simulator can obtain the available CIO data without involving having any interface card.

2.2 Bus Interface Protocols for HIL Real-time Simulation

In order to transmit and receive data to and from the simulator, a preferred interface needs to be selected. This interface is also intuitively interconnect the existing simulators while satisfying important constrains including the simulator time of the IO operation via the interface. In particular, a key the interface constraint identified is timing difference between the target simulator and the interface hardware. For instance, the interface hardware provides a 10ms sampling rate. An access time of the simulator is relatively slower than the signaling time over the interface hardware. In order to overcome this issue, we adopted a reflective memory network and a dedicated Ethernet between the interface hardware and the simulator. We evaluated a couple of the interface protocols for our implementation.

The reflective memory network approach [7] can facilitate integration of the interface hardware with the simulator. A reflective memory network card seamlessly integrates and updates to the existing ring or hub by propagating the changes to its memory across to the other nodes on the network. Since data integrity is one of the key aspects for interface, data from the FEBus must be written to its dedicated ranges of the addresses. In other words, the data written via the FEBus must be not overwritten by any other simulators. However, this approach requires implementing the PCI or PCIe bus architecture (i.e., GEIP or General Standards Corp). For instance, the GEIP PCI Reflective Memory card [8] employs the PCI bus.

The PCI bus architecture requires operating in the host bridge mode, which always permits a controlling master to access the bus. In order to expedite our design process, an FPGA-based host-bridge capability is chosen. The PCI Reflective Memory card is connected to the PCI bus architecture implemented on the FPGA via a PMC connection [9]. In addition, a driver needs to be developed. As the PCI memory card is configured through assembly level programming, the driver is configured as a memory mapped variety. In this approach, one of the most critical portions in the design is to manage the PCI configuration unlikely an operating system (OS) usually take care of this kind of operation.

Another communication protocol selected is the UDP over a Gigabit Ethernet connection. This protocol is inexpensive to integrate into the simulator. In addition the protocol provides an intuitive means to utilize an interface board for monitoring data over the FEBus without any special hardware. The protocol was implemented to the same FPGA used for the PCI. The Altera's triple-speed Ethernet core and the RJ-45 connector on the Stratix III Development Board [10] were used for our implementation. Figure 2 shows the location of the RJ-45 on the development board in the blue box. The triple-speed Ethernet core [11] employs a software

driver allowing the NIOS II microprocessor to open and maintain a UDP connection in a client-server relationship with the simulator.

Figure 2. The Virtual Bus Interface System Implementation with Interface Connections on the Altera Stratix III Development Board

While the preferred interface protocol was the PCI Reflective Memory card, the design was almost unable to prototyped. The Stratix III FPGA board, however, provides dedicated pins for a PCI bus interface with 3.0V signaling. Unfortunately, the same pins are assigned for other IOs. Without ability to access the PCI pins directly, the UDP connection was chosen for the simulator interfacing protocol.

2.3 Implementation of the Virtual Bus Interface System

The virtual bus interface system developed is comprised of four components—(1) the FEBus interface, (2) the UDP simulator interface, (3) the dual-port SRAM, and (4) the NIOS II microprocessor core. The FEBus interface responds an address for each IO card in the CIO unit. Thus, the FEBus interface eliminates any additional hardware on the simulator sides. The UDP simulator interface transmits data to and receives data from the simulator by delivering the value of the IO points to and from the simulators. Dual-port SRAM permits to read and write operations simultaneously if necessary. In particular, the data at the same address need to be accessed at the same time. The NIOS II microprocessor initializes the bus interface system and controls data flow.

The primary interface logic including the FEBus and the UDP simulator interfaces, the SRAM, and the microprocessor was implemented on an Altera Stratix III FPGA development board. The FEBus connections utilize one of the two High-Speed Mezzanine Card (HSMC) interfaces. The HSMC connections are located on the left side of the development board in the red box in Figure 2. These connections were converted to standard 1" headers using a third party breakout board. The one inch headers allowed the use of standard ribbon cable which could be soldered to ribbon cable on the FEBus connector of the CIO unit.

The UDP simulator interface utilizes the Altera's triple speed Ethernet core including the MAC logic and PHY circuitry. The UDP simulator interface also exploits a software driver, which the two scatter-gather DMA cores read and write the associated Ethernet buffers. The UDP connection between the simulator and the IO board was designed for delivering a packet in every 250us.

On the other side of circuitry, the FEBus interface transmits data to and receives data from the dual-port SRAM via the address translation logic under the control of the bus master. The bus master is an IO manager for other embedded systems on the locomotive. The hardware of the FEBus was tailored to each IO slot. The developed FEBus interface responds to addresses on the bus as monitored by the bus master. The FEBus interface accesses the dual-port SRAM via the Altera's memory mapped Avalon connection.

The remaining hardware was built with Altera's COTs cores available in the Qsys tool. The Qsys tool allows the building of digital circuits in a graphical fashion and is part of the Quartus design software. Using this tool, frees the designer from having to make individual signal connections as they are grouped together logically and connected using a relatively simple graphical interface. The FEBus logic was brought into the Qsys tool for use in the design using a built in wizard for importing user defined circuitry.

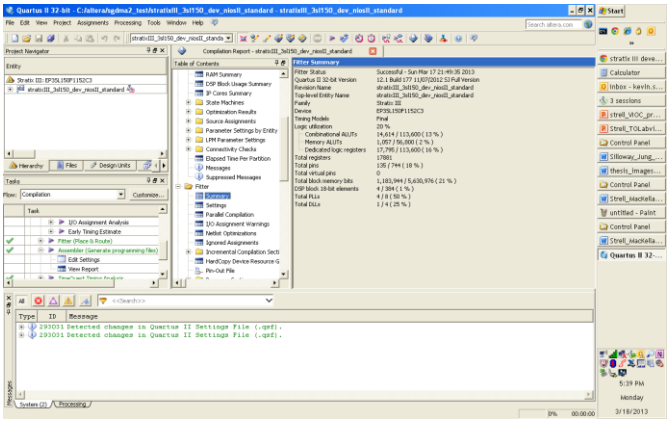


Figure 3. Synthesis and Routing Summary from Quartus

The design was built using the Quartus design software with the default timing and routing constraints. Using these settings the design easily fit into the Stratix III FPGA. Pin assignments were made using the PinPlanner feature of Quartus, enabling the setting of pins by script for some of the Altera cores. The report for the synthesis and routing is shown in Figure 3.

The microprocessor communicates to the other components in the virtual bus interface system using Altera's Avalon interconnection fabric, which provides two different types of connections, such as memory mapped and streaming. The memory mapped variety is set as default. The Avalon system also enables APIs for the memory and the DMA cores.

The software driver of the triple speed Ethernet core necessitates a DMA for the streaming mechanism.

The software program running on the microprocessor is responsible for controlling the UDP simulator interface. The software program composes the data for each packet and then sends the packet out over the Ethernet assisted by the triple speed Ethernet driver in every 250us to maintain data integrity required for the locomotive control system. The packets are delivered to the dual-port SRAM via the scatter gather DMA (SGDMA) in a block transmission mode.

The software program generates a header file with the received IO card data. This header file contains the channel and slot number of the data (on the FEBus) and its corresponding address in the dual-port SRAM. The software program utilizes the associated offsets to construct a descriptor chain for processing the SGDMA core. After the SGDMA core packs the data into a contiguous block of memory, the software program sends the data to the Ethernet core using the software driver and a UDP packet is also transmitted. A similar process takes place reading data from the simulator and placing it into the dual-port SRAM for FEBus access.

3 Evaluation of the FPGA-based Virtual Bus Interface System

The virtual bus interface system was implemented in VHDL and programmed to the Stratix III FPGA. The VHDL cores implemented are simulated via the Altera's Quartus II. The software program was verified using Altera's NIOS extension for the Eclipse design environment. Wireshark was utilized for evaluating the packet communication on the network. Each bus interface was tested separately prior to integrate the entire interface system. The FEBus interface was tested using a standalone control system with diagnostic software. The control system consists of the CIO and two smart displays where the CIO acts as the IO manager for the control system on the displays.

3.1 Evaluation of the FEBus interface with Dual-ported RAM

The FEBus interface's connection to the dual-ported RAM is a critical component in the system. This connection allows the FEBus data to be accessed by the Ethernet core and sent to the simulator. In order to test this connection a simple system of the FEBus interface and the RAM was created using the Quartus II software. Once the code was generated it was simulated in Altera's version of Modelsim by wrapping the top level design in a handwritten testbench. The testbench simulated the function of the FEBus master and executed a cycle of two reads and two writes of different data to verify the data could pass through the FEBus interface to the RAM.

Figure 4 illustrates the simulation results of the virtual bus interface system for a read operation. In the figure, the FEBus interface operates according to the implemented finite state machine. Four states are defined as (2) a command receiving state, (1) a dual-port SRAM accessing state, (3) a completion acknowledging state, and (0) an idling state for next interface operation. In the diagram the first region (A) shows the card in the idle state (0) with the master placing the address on the bus. No commands are issued during this state. The second region (B) diagrams the reception of the receive command. With the address on the bus the master issues the read command by deasserting IORC, sending the slave into the command receive state (2). During this state the slave places the address on the RAM's address line and generates the read command by asserting the read line on the dual ported RAM. In the third region (C) the slave receives the data from the memory and places it on the FEBus data line. This is the RAM interface state (1) which is followed by the acknowledge state (3). In the acknowledge state, the slave acknowledges the command and keeps asserting the acknowledgement until the command is removed from the bus. The final region (D) shows the state machine returning to the idle state (0) in which all address are reset to zero and the bidirectional data bus is put into a high impendence state. The bus is then ready for the next cycle.

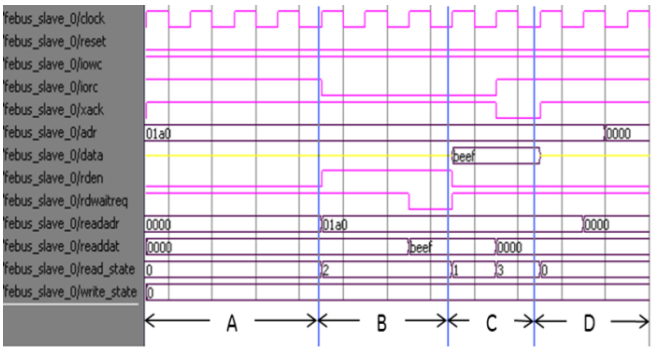


Figure 4. Simulation Result of the FEBus Interface Implemented in VHDL

3.2 A Case Study: The FE187 Digital Output Card

In order to validate the hardware and software of the Virtual IO Card work within a FEBus device the card must replicate the function of the IO cards within the FEBus device. In this test case the FE187 will be replicated to ensure the control software on the CIO can drive writes and reads to the replacement card with no ill effects. To replicate this card the Virtual IO Card will be attached to the CIO via the FEBus breakout and to the test computer via the Ethernet connection from the development board. Altera's SignalTap tool was used to capture data while testing the card with the control system.

The CIO is connected to the rest of the control system via ARCNET so the diagnostic software of the control system can be used. As mentioned above the FEBus breakout allows the HSMC connector on the Stratix III Development board to be physically attached to the FEBus device. The UDP connection is made over a 1 Gbps Ethernet connection with standard RJ45 connectors.

The FE187 card is a 75 V 16 channel digital output card that operates on the FEBus. To effectively model the function of this card the Virtual IO Card must respond to the FEBus master exactly how the FE187 would respond. In order to implement this card the Virtual IO Card must maintain a register with the value of the digital channels. The control software first reads this register and compares the value to its previous value to check if an overcurrent has occurred. If the channel is not in an overcurrent state the control software proceeds to write the card with its new data and afterwards checks again for an overcurrent. Thus the Virtual IO Card must respond to a sequence of read, write, read from the master.

With the Virtual IO Card attached to the CIO and test computer, the control system was loaded with its diagnostic software. Writes were then driven to the card using diagnostic commands built into the CIO software. The data written was then able to be seen in the dual ported RAM. In addition to seeing the data from the bus in memory, the successful return of the diagnostic write command confirms that the Virtual IO Card can replicate the FE187 digital output card.

4 Conclusions

A bus interface card for a modified version of the Modbus, the FEBus, was created to reduce the amount of IO in locomotive HIL simulators. The card was implemented on a Stratix III FPGA and consisted of four main components: the FEBus interface, UDP simulator interface, the dual ported RAM, and the NIOS II processor. The FEBus interface acted as a universal bus slave in place of the IO cards on the CIO unit, reading and writing the dual ported RAM at the request of the master. The UDP simulator interface was responsible for updating the RAM with the simulator data, writing simulator outputs to the RAM and reading simulator inputs from it. The NIOS II processor contained the software to control the UDP Ethernet and the DMA required for using it. Finally the RAM held the value of the IO points being replaced, granting both interfaces access. The FEBus interface was tested for accuracy with a standalone CIO. The UDP connection was confirmed by sending data to and from our main simulator computer to ensure the models ran correctly with the new input method. Testing in a full locomotive HIL simulator has yet to be completed at this time. Despite the full testing being incomplete, the card is still useful as a tool for software developers working on the CIO unit and other FEBus devices.

5 References

[1] L. Cheng and Z. Lipeng, "Hardware-in-the-Loop Simulation and Its Application in Electric Vehicle Development," IEEE Vehicle Power and Propulsion Conference, 2008.

[2] D. Michalek, C. Gehsat, R. Trapp, T. Bertram, "Hardware-in-the-Loop-Simulation of a Vehicle Climate Controller with a combined HVAC and Passenger Compartment Model," International Conference on Advanced Intelligent Mechatronics, 2005.

[3] H. Hao, Z. Yang, X. Guoqing, "Hareware-in-the-loop Simulation of Electric Vehicle Powertrain System," IEEE, 2009.

[4] J. Pouget and Y. Riffonneau, "Signal Hardware-In-the-Loop simulator of hybrid railway traction for the evaluation of energy mana gement," IEEE Vehicle Power and Propulsion Conference, Oct. 2012.

[5] Y. Ito, Y. Sogure, S. Oho, "Model-Based Software Validation for Automotive Control Systems," International Conference on Control, Automation and Systems, 2010.

[6] J. Chen, L. Li, L. Wang, "The Application of PROFIBUS Technology in the Fengchan River Project's Electronic Control System Reform," Fifth International Conference on Intelligent Networks and Intelligent Systems, 2012.

[7] Z. Rui, H. Shiying, "Research and Realization of Reflective Memory Network based on QNX," IEEE Vehicle Power and Propulsion Conference, 2008.

[8] M. Jovanovich, V. Milutinovic, "An Overview of Reflective Memory Systems," IEEE Concurrency, pp. 56 – 64, 1999.

[9] IEEE Standard Physical and Environmental Layers for PCI Mezzanine Cards (PMC), IEEE Standard 1386.1, 2001.

[10] Altera, Stratix III 3SL150 Development Board Reference Manual, 2008.

[11] P. Xue, H. Wang, J. Hou, W. Li, "SOPC realization of image acquisition and real-time network monitoring system," International Conference on Measurement, Information, and Control, 2012.

An Adaptive Real-Time FPGA-Based Speech Enhancement Processing System

Andrés J. Cuevas-Romano, Víctor H. Díaz-Ramírez, and Andrés Calvillo-Téllez

Instituto Politécnico Nacional - CITEDI, Avenida Instituto Politecnico Nacional 1310, Mesa de Otay, Tijuana B.C. 22510, Mexco; email: {acuevas, vhdiaz, calvillo}@citedi.mx

***Abstract*—A real-time system for robust speech processing based on a Field Programable Gate Array (FPGA) is presented. The system estimates a clean signal from a noisy speech signal using a locally adaptive algorithm based on calculation of rank-order statistics from the input signal over a moving window. The algorithm is able to improve the quality of noisy speech with no perceivable musical noise. The processing system adapts the size and contents of the moving window as well as the local estimator used to recover the clean signal at each iteration. Results obtained with the proposed system in terms of quality and intelligibility objective measures are presented and discussed when testing the system in different non-stationary noise environments. The real-time performance of the proposed FPGA system is also evaluated an discussed.**

***Index Terms*—Speech enhancement, noise reduction, audio processing, field programmable gate arrays, real-time.**

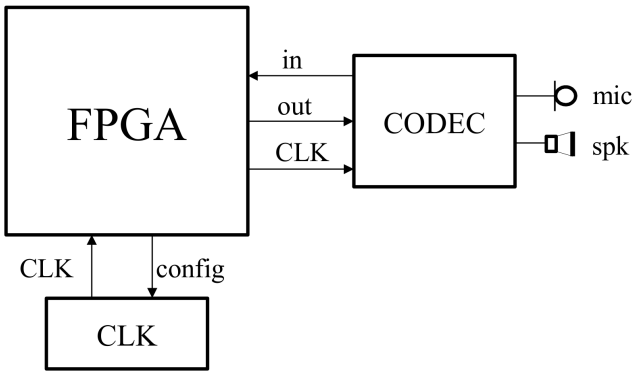


Figure 1. Schematic diagram of the proposed speech enhancement system.

I. Introduction

NOWADAYS, the demand for transmitting noise-free speech signals on mobile communication equipment has greatly increased the need for robust techniques for real-time speech enhancement. Two common techniques used for speech enhancement are given by the spectral subtraction and Wiener filtering algorithms [1]. The former approach calculates an estimate of the average noise spectrum which is subtracted from the spectrum of the noisy signal in such a manner that the average signal-to-noise ratio (SNR) is improved [2], [3]. On the other hand, the Wiener filtering is a linear filter optimized with respect to the mean-squared-error between the clean and processed signals. In this approach, an estimate of the clean speech power spectrum is estimated (locally) from the noisy speech signal [4]. Furthermore, when the noise can be described by a stationary random process then the use of wiener filtering is recommendable [5].

The spectral subtraction and the Wiener filtering algorithms are fast and reliable noise reduction techniques. However, because of both of these algorithms operate in the frequency domain, they commonly introduce spurious artifacts to processed speech resulting in annoying musical noise [1]. Furthermore, it is important to note that these algorithms assume that noise is stationary, however, in real life scenarios such as in mobile communications, noise is almost never stationary. For this reason, the use of a robust filtering is desirable. There are several successful nonlinear filters based on calculation of rank-order statistics that could be used for speech enhancement [6]. It is well known that these filters are robust and able to preserve fine signal structures. In speech enhancement, this feature can help to suppress the additive noise while preserving the intelligibility of speech. Speech processing systems can be broadly classified in single or multiple channel systems. Single channel systems use only one microphone to capture speech signals whereas multiple channel systems utilize a microphone array to capture speech from different locations.

In this work, we propose a single channel system for real-time speech enhancement which uses a nonlinear locally adaptive algorithm for robust speech processing based on a FPGA chip. The algorithm is carried out in the time-domain using a time variant moving window. The algorithm is able to improve the quality of noisy speech in terms of objective metrics and with no perceivable musical noise [7]. The processing system adapts the size and contents of the moving window as well as the local estimator used to recover the clean signal at each iteration. Therefore, a noise-free signal can be estimated using a time-variant estimator over a local adaptive neighborhood. A local neighborhood is a subset of signal elements of the sliding window, which are close in some sense to a given element [6]. The proposed algorithm is adaptive to non-stationary signal fragments and noise fluctuations.

Note that since the proposed method is implemented in the time domain then a priori estimation of the spectral distributions of signals and noise is not carried out for each frame. Thus, annoying artifacts introduced to processed speech are not perceivable [8]. A schematic diagram of the proposed system is shown in Fig. 1.

II. Speech Enhancement Algorithm

Adaptive filtering has proven to be a good choice in processing signals corrupted by non-stationary noise [9]. The proposed algorithm for speech enhancement takes advantage

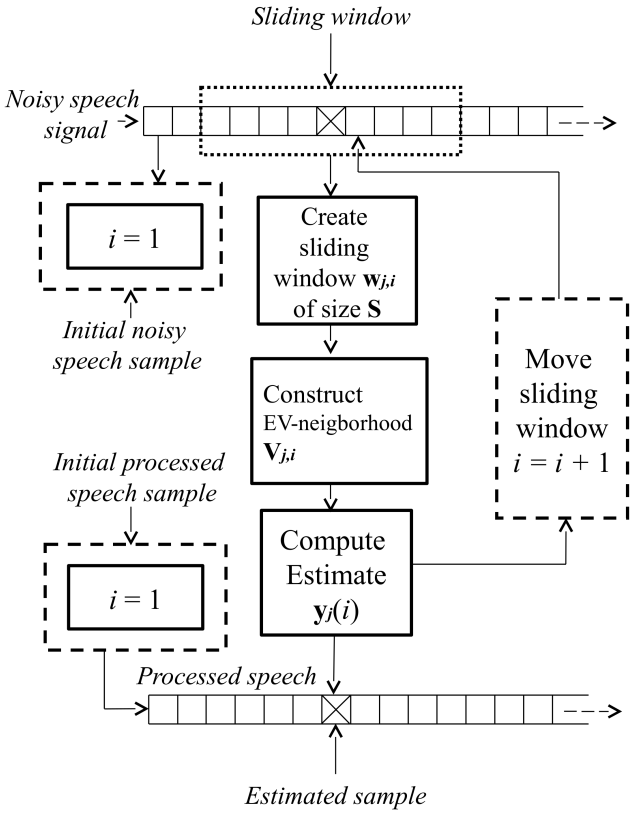


Figure 2. Block diagram of the proposed speech enhancement algorithm.

of the calculation of rank-order statistics to locally modify the parameters of a robust estimator for recovering a speech signal with an improved quality and by preserving its intelligibility [7].

The flow diagram of the used speech enhancement algorithm is presented in Fig. 2 and consists of the following steps [7]:

STEP 1: Read an initial input speech segment $\mathbf{n_o}$ with S elements assuming speaker's silence.

STEP 2: Start processing of speech segments ($j = 1$).

STEP 3: Read the jth input speech segment $\mathbf{f}_j = \mathbf{s}_j + \mathbf{r}_j$ (with N samples) to be processed. Note that the speech segment is given by superposition of the clean signal $\mathbf{s}_j$ and additive noise $\mathbf{r}_j$.

STEP 4: Create a sliding window $\mathbf{w}_{j,i}$ around the noisy element $f_j(i)$, given by

$$\mathbf{w}_{j,i} = \left[w_{j,i}\,(n) = f_j(n) : |n - i| \leq \frac{S-1}{2} \right]^T. \quad (1)$$

STEP 5: Calculate a local estimate of the SNR as follows:

$$SNR_{j,i} = \frac{\mathbf{w}_{j,i}^T \mathbf{w}_{j,i}}{\mathbf{n_0}^T \mathbf{n_0}}. \quad (2)$$

STEP 6: Calculate the parameter ϵ_v as follows:

$$\epsilon_{v_{j,i}} = k_1 \sigma_n \left[1 - \frac{1}{1 + (SNR_{j,i})^{-k_2}} \right], \quad (3)$$

where k_1, and k_2 are constant parameters which are defined within the range of (0,1]. The parameters k_1 and k_2 help us to take into account a priori information about either the spread of the signal to be preserved or noise fluctuation to be suppressed [7]. Using these two parameters a trade-off between noise suppression and introduction of artifacts to the processed speech signal can be achieved.

STEP 7: Construct an EV-neighborhood $\mathbf{v}_{j,i}$ from $\mathbf{w}_{j,i}$ as follows:

$$\mathbf{v}_{j,i} = \left[w_{j,i}(n) : w_{j,i}(i) - \epsilon_v \leq w_{j,i}(n) \leq w_{j,i}(i) + \epsilon_v \right]^T.$$

STEP 8: Compute the output estimate as $y_j(i) = \mathbf{a}^T \mathbf{v}_{j,i}$.

STEP 9: Move the window ($i = i + 1$) and evaluate if ($i \leq N$). If the result is "true" go to STEP 4. Otherwise, go to STEP 3.

III. REAL-TIME FPGA-BASED SYSTEM FOR SPEECH ENHANCEMENT

In recent years, FPGA devices have achieved a growing popularity because of their great flexibility. These devices offer the possibility to reconfigure their hardware according to the designer's needs. An FPGA can be programmed to operate with a custom hardware architecture or can be programmed to operate as a customizable reduced instruction set computer (RISC) with particular specifications. In this work we propose a real-time robust speech enhancement system using the algorithm shown in Fig. 2 which is implemented on an FPGA configured as a high-speed custom RISC processor.

The development board used for the implementation is the Altera DE-115 which has a Cyclone IV E EP4CE115 chip. The sampling rate for this system was set at 8 Khz and the input speech was coded using signed 16 bit integer data. From Fig. 1 we can observe that the CODEC captures the input speech signal through the microphone (mic) and codes each data sample according to the set sample rate. Next, the FPGA processor reads out each data sample from the CODEC and arranges them into data frames of fixed length for subsequent signal processing. Each processed data frame is sent back to the CODEC one next to another for sound playback through a speaker (spk).

In order to obtain a real-time processing system, the processing block must be fast enough to maintain a constant data flow. To obtain the best performance with the FPGA chip, we use the Nios II/f (fast) configuration. The block diagram of the proposed speech processing system is shown in Fig. 3.

For real-time operation, we must take into account that the size of the data frames in the system is a hardware dependent parameter. Observe that if the size of the data frame is too large then the system's latency will be high. On the other hand, if the size of the data frame is to small then the latency will be

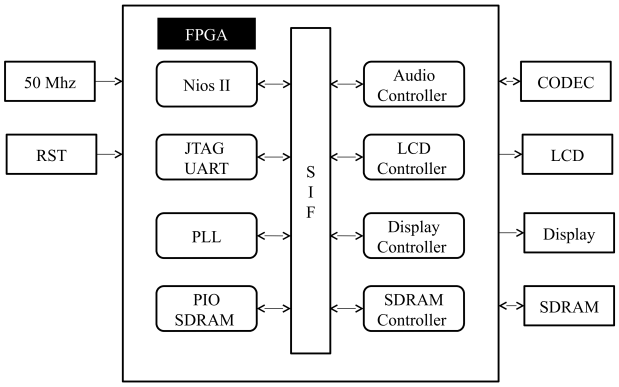


Figure 3. Block diagram of the FPGA system.

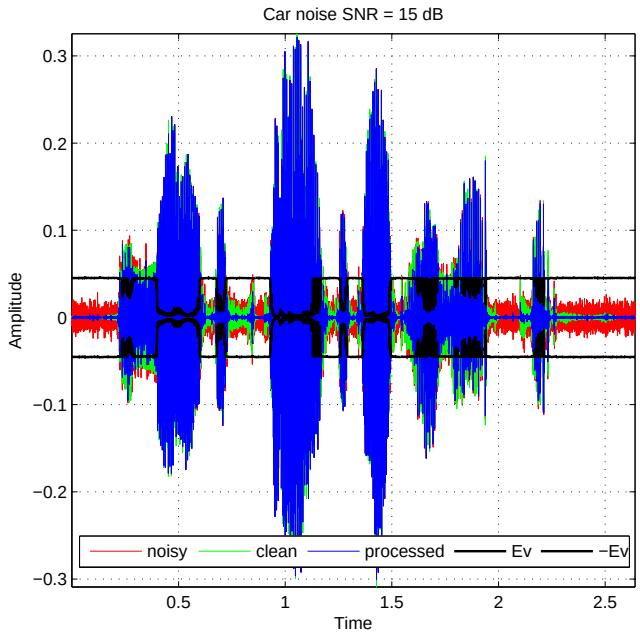


Figure 4. Example of a speech sentence corrupted with 15dB SNR car noise processed with the proposed system.

short but there will be interruptions in the output audio stream [10]. So, the length of the data frame is a trade-off parameter among the system's latency and the quality of playback stream. The size of the data frame used for the implementation is 204 samples, because the size of the sliding window is 51 samples. It is important to mention that the processing time for 204 samples captured with a sample rate of 8 Khz takes 25 ms.

IV. Results

In this section we present the results obtained with the proposed real-time system for speech enhancement. We evaluate the performance of the system in terms of objective quality and intelligibility metrics. Also, we test the real-time performance of the FPGA-based system. To evaluate speech quality we use the following metrics: perceptual evaluation of speech quality (PESQ) [11], source to distortion ratio (SDR) and source to interference ratio (SIR) [12]. The SDR metric, characterizes the effective separation between speech and background noise. The SIR, is a metric commonly used to characterize introduction of artificial artifacts such as musical noise. Speech intelligibility was evaluated by the short-time objective intelligibility (STOI) metric [13].

We tested the proposed system for speech enhancement in four different noisy environments: train, car, street and restaurant. Speech samples were taken from the NOIZEUS database [14]. The database consists of 30 IEEE sentences [15], produced by different speakers. The speech samples were originally sampled at 25 kHz and downsampled to 8 kHz. All sentences from the database were processed with the proposed system in different noise environments. Fig. 4 shows an example of one speech file from the database after processing with the proposed system when speech is corrupted with 15dB SNR car noise. Note that the system adapts well to the non stationary characteristics of speech signal and noise (see the $\pm\epsilon_v$ behavior). With 95% confidence the results obtained in terms of performance metrics are shown on tables 1-4 and Fig. 9-12. The results show a considerable quality improvement from the noisy recordings and show consistent results across the different noisy environments. The results obtained in terms of speech intelligibility (see STOI results) show a lower rating for higher noise SNR, however, this is normal behavior because it has shown [1] that all noise reduction algorithms either improve the speech quality or intelligibility. Also, note that the proposed approach yields a good performance in terms of noise reduction that is characterized by the SDR.

In Fig. 5 we see the spectrogram of the clean sentence of the database "He knew the skill of the great young actress". In Fig. 6, we see the spectrogram of the sentence used in Fig. 5 corrupted with 15 dB SNR car noise. The corrupted sentence used in Fig. 6 was processed with the proposed FPGA-based system. The spectrogram of the processed sentence is shown in Fig. 7. It can be noted that there is a considerable reduction of noise and the frequency distribution of the processed signal is very similar to the one for the clean signal shown in Fig. 5.

Table I
Performance of the proposed algorithm with 95% confidence in non-stationary train noise environment.

	15 dB	10 dB	5 dB
STOI	0.93 ± 0.03	0.87 ± 0.03	0.78 ± 0.04
PESQ	2.52 ± 0.10	2.16 ± 0.11	1.88 ± 0.15
SIR	17.18 ± 1.25	12.15 ± 0.99	7.92 ± 1.57
SDR	15.96 ± 0.55	11.22 ± 0.39	6.58 ± 0.65

Evaluation of the real-time performance of the system

For evaluation of the real-time performance of the system, we measure the processing time scheduling of the FPGA implementation in terms of the rate of data frame processing (latency). By performing these tests, we are interested in

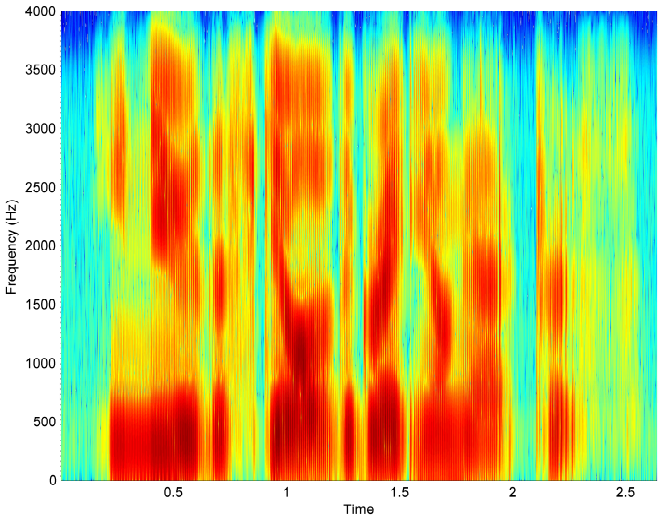


Figure 5. Spectrogram of the clean speech sentence "He knew the skill of the great young actress".

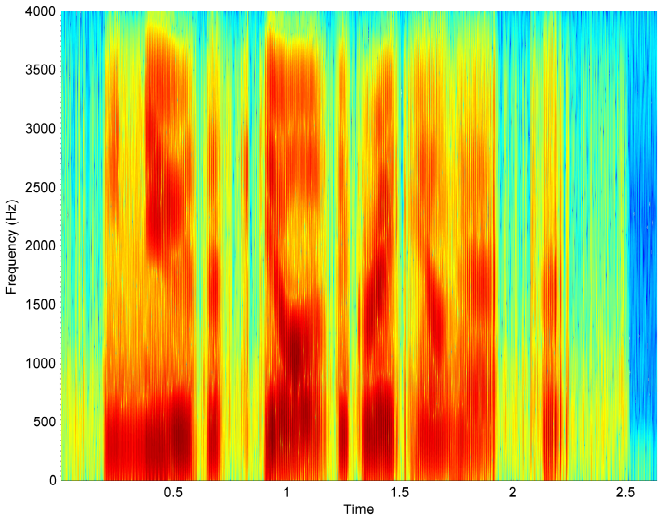


Figure 7. Spectrogram of the processed speech sentence "He knew the skill of the great young actress" when is corrupted with 15dB SNR car noise.

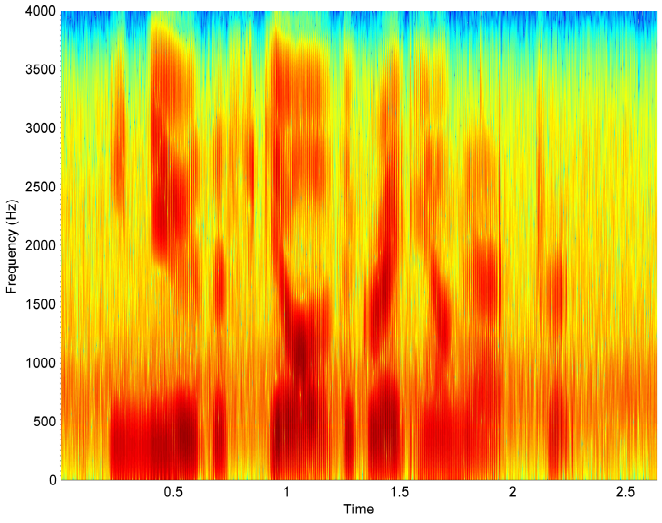


Figure 6. Spectrogram of the speech sentence "He knew the skill of the great young actress" corrupted with 10dB SNR car noise.

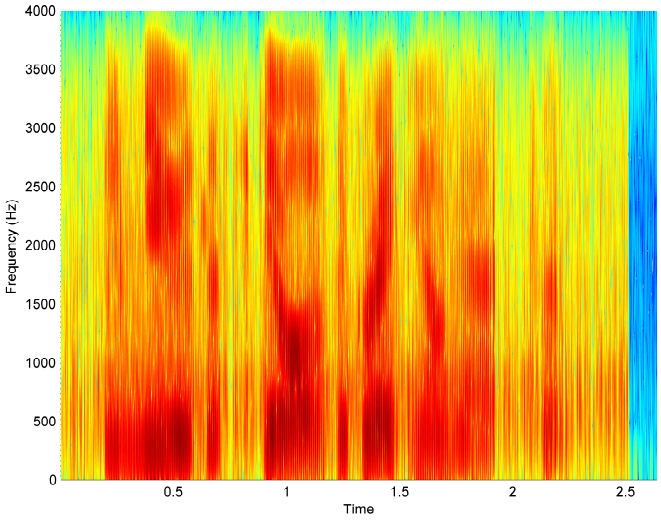


Figure 8. Spectrogram of the processed speech sentence "He knew the skill of the great young actress" when is corrupted with 10dB SNR car noise.

Table II
PERFORMANCE OF THE PROPOSED ALGORITHM WITH 95% CONFIDENCE IN NON-STATIONARY CAR NOISE ENVIRONMENT.

	15 dB	10 dB	5 dB
STOI	0.92 ± 0.03	0.84 ± 0.03	0.77 ± 0.02
PESQ	2.55 ± 0.10	2.22 ± 0.11	1.94 ± 0.12
SIR	17.74 ± 0.80	13.40 ± 0.99	8.45 ± 1.11
SDR	16.55 ± 0.55	11.09 ± 0.39	7.26 ± 0.64

Table III
PERFORMANCE OF THE PROPOSED ALGORITHM WITH 95% CONFIDENCE IN NON-STATIONARY STREET NOISE ENVIRONMENT.

	15 dB	10 dB	5 dB
STOI	0.93 ± 0.03	0.87 ± 0.03	0.77 ± 0.02
PESQ	2.53 ± 0.10	2.27 ± 0.10	1.91 ± 0.12
SIR	16.87 ± 1.25	11.53 ± 0.99	6.51 ± 1.57
SDR	16.03 ± 0.55	10.78 ± 0.39	5.97 ± 0.65

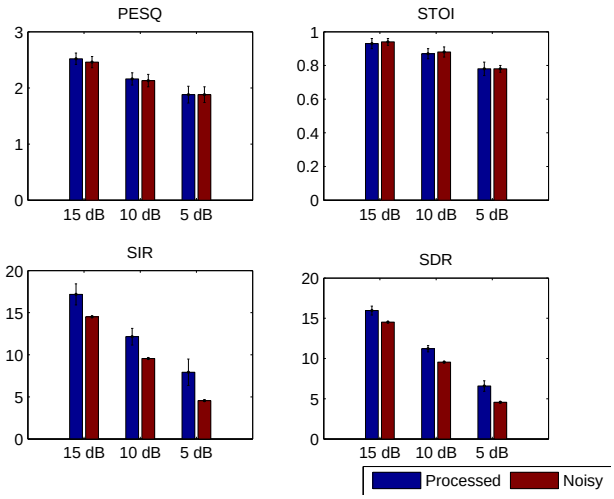


Figure 9. Performance of the proposed algorithm with 95% confidence in non-stationary train noise environment.

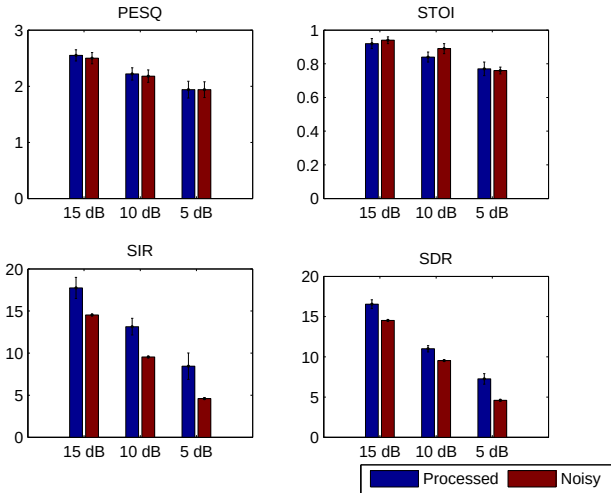


Figure 10. Performance of the proposed algorithm with 95% confidence in non-stationary car noise environment.

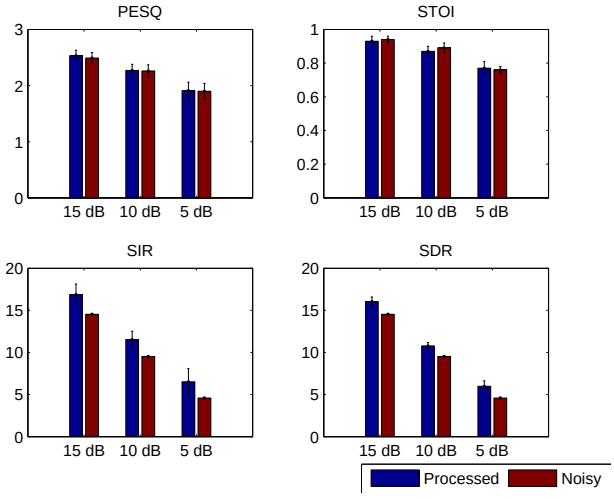


Figure 11. Performance of the proposed algorithm with 95% confidence in non-stationary street noise environment.

Table IV
PERFORMANCE OF THE PROPOSED ALGORITHM WITH 95% CONFIDENCE IN NON-STATIONARY RESTAURANT NOISE ENVIRONMENT.

	15 dB	10 dB	5 dB
STOI	0.92 ± 0.03	0.84 ± 0.03	0.79 ± 0.04
PESQ	2.55 ± 0.10	2.22 ± 0.11	1.95 ± 0.12
SIR	17.74 ± 1.25	12.40 ± 0.99	7.45 ± 1.57
SDR	15.55 ± 0.55	11.09 ± 0.39	7.26 ± 0.65

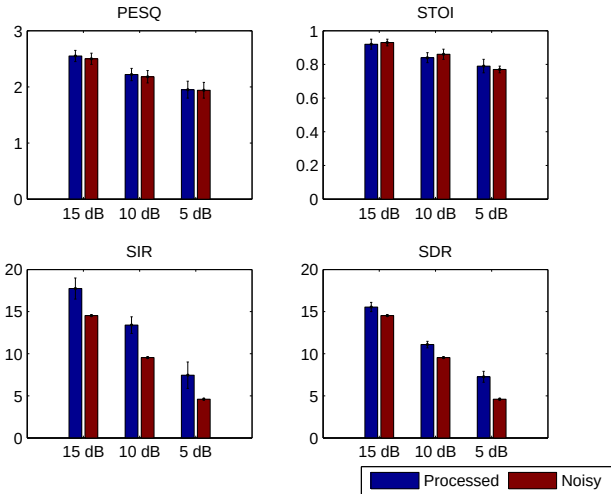


Figure 12. Performance of the proposed algorithm with 95% confidence in non-stationary restaurant noise environment.

showing that the proposed system can be efficiently used for real-time applications. Additionally, we want to show that the proposed system can improve the quality of noisy speech signals in real-time without sacrificing intelligibility and with no perceivable musical noise. To measure the latency of the FPGA implementation, a 600 Hz sinusoid tone was used as the input signal. In Fig. 13 we can see that the obtained output latency is 80 ms. When the speech enhancement algorithm is disabled, that is, when the input samples are copied from the input ADC to a memory buffer and then sent for playback, it took 64 ms (see Fig. 13). So by subtracting this latency to the latency when the algorithm is active we get 16 ms witch is the latency of the processing operations.

V. CONCLUSIONS

A locally adaptive FPGA-based system for robust speech processing was presented. The implemented algorithm is able to recover an undistorted signal from a noisy speech employing a time-variant estimator over a locally adaptive neighborhood without introducing musical noise. Based on the conducted tests, it is concluded that the system is capable of enhancing speech signals corrupted by non-stationary noise in a real-time processing scenario.

REFERENCES

[1] P. C. Loizou, *Speech enhancement theory and practice*. Taylor & Francis, 2007.

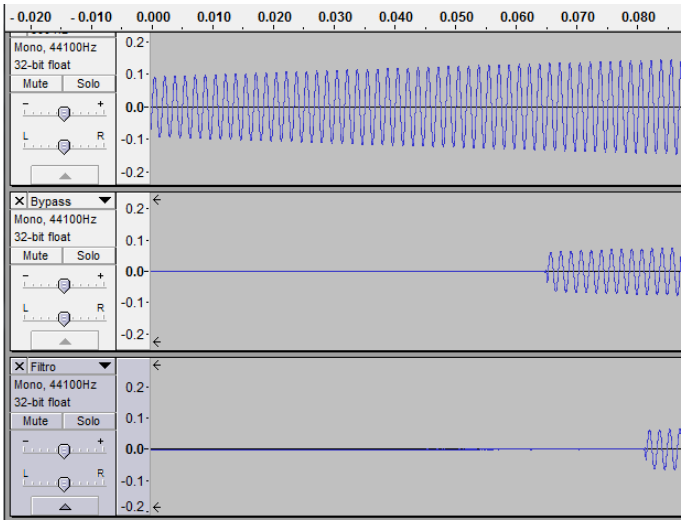


Figure 13. Measurement of system's output latencies (top) Input, (center) Bypass Output and (bottom) Processing Output.

[2] S. F. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Trans. Acoust. Speech. and Signal Process.*, vol. 27, no. 2, pp. 113–120, 1979.

[3] R. McAulay and M. Malpass, "Speech enhancement using soft-decision noise suppression filter," *IEEE Trans. Acoust. Speech and Signal Process.*, vol. 28, no. 2, pp. 137–145, 1980.

[4] C. Plapous, C. Marro, and P. Scalart, "Improved signal-to-noise ratio estimation for speech enhancement," *IEEE Trans. Audio, Speech and Lang. Process.*, vol. 14, no. 6, pp. 2098–2108, 2006.

[5] Y. Hu and P. C. Loizou, "Evaluation of objective quality measures for speech enhancement." *IEEE Trans. on Audio, Speech, and Lang. Process.*, vol. 16, no. 1, pp. 229–238, 2008.

[6] L. Yaroslavsky and M. Eden, *Fundamentals of digital optics*. Boston:Birkhäuser, 1996.

[7] V. Diaz-Ramirez and V. Kober, "Robust speech processing using local adaptive nonlinear filtering," *IET Signal Processing*, In press 2013.

[8] C. Breithaupt, T. Gerkmann, and R. Martin, "Cepstral smoothing of spectral filter gains for speech enhancement without musical noise," *IEEE Sig. Process. Lett.*, vol. 14, no. 12, pp. 1036–1039, 2007.

[9] J. M. Roger Woods, *FPGA-based Implementation of Signal Processing Systems*. WILEY, 2008.

[10] V. L. Richard Boulanger, *The Audio Programming Book*. The MIT Press, 2011.

[11] U.-T. P.862, "Perceptual evaluation of speech quality (pesq): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs, itu-t recommendation p.862," 2007.

[12] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE Trans. Audio, Speech and Lang. Process.*, vol. 14, no. 4, pp. 1462–1469, 2006.

[13] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time-frequency weighted noisy speech," *IEEE Trans. Audio, Speech and Lang. Proc.*, vol. 19, no. 7, pp. 2125–2136, 2011.

[14] Y. Hu and P. C. Loizou, "Subjective comparision and evaluation of speech enhancement algorithms," *Speech Commun.*, vol. 49, pp. 588–601, 2007.

[15] I. of Electrical, E. Engineers, I. Audio, and E. G. S. Committee, *IEEE Recommended Practice for Speech Quality Measurements*, ser. IEEE Std. The Institute, 1969. [Online]. Available: http://books.google.com.mx/books?id=blu1NwAACAAJ

Telemedicine: Issues With A Mote™ Based Remote Patient Monitoring Wireless System Design

Patrick Otoo Bobbie[1,2], **Erik Hadley**[3]
[1,3]Computer Science and Software Engineering, Southern Polytechnic State University, Marietta, GA, USA
[2]School of Informatics, Ghana Technology University College, Accra, Ghana

Abstract – *The objective of the research project is to explore the issues involved in building a wireless Electrocardiogram (ECG) remote patient monitoring system. The issues include capturing of the ECG signal data at a sufficient data rate to produce an accurate representation of the signal, and the maintaining of a reliable data stream from the ECG device to the monitoring base station. An ECG device, capable of amplifying the signal and regulating the output voltage without overloading the analog to signal converter, was built. Data was captured using the MDA300CA™ data acquisition board and the MicaZ wireless Mote™ from Crossbow Technologies™ Inc. Crossbow Technologies's MIB600CA™ base station was used for receiving wireless transmission. The MIB600CA also bridged the Ethernet connection for streaming the wireless packets to a computer via the Ethernet port. The paper discusses the hardware and software architectures and the design methodology for building the system. The results, through observations of the system's behavior and outputs, are discussed including suggestions for mitigating the challenges toward the construction of a commercially viable prototype.*

Keywords: ECG, Telemedicine, Embedded TinyOS, Mote, Remote Patient Monitoring.

1 Introduction

The purpose of this research project is to examine the issues in producing a reliable wireless ECG patient monitoring system using an analog to digital converter, the MDA300CA and MicaZ wireless motes developed by Crossbow Technologies, Inc. By using a wireless system to obtain a patient's ECG readings it allows for the patient to be independent and possibly even far away from the doctor. It is also possible for a large number of patients to be monitored from one location reducing the need for constant walking from bedside to bedside.

The ECG signal data captured by the analog to digital converter is stored in a database on a computer that is connected to an MIB600CA ™ base station, also developed by Crossbow Technologies™, Inc. A database is used for data storage because it offers the ability to store large amounts of patient data, including timestamps and ECG signal values, quickly and reliably. Further it is easy to pull the data out of the database for analysis and display.

As stated earlier, patients receive the benefit of not having to be connected to a large bedside ECG apparatus in order for a doctor or nurse to be able to monitor them. For the purpose of experimentation an assisted living center was used as a model for data capture. This model was chosen because it demonstrates how a patient can use a "wearable" device and move around in their environment all the while being monitored by a doctor or nurse. The patient gains a degree of freedom and the doctor is still able to watch over the patient through the remote monitoring interface at the hospital. Figure 1 shows a layout of an environment that fits the assisted living center model.

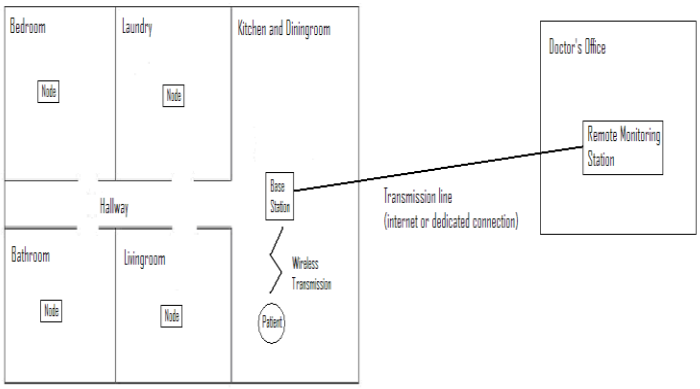


Figure 1: *An Assisted Living Center Model*

2 Related Research

A prototype remote patient monitoring system, similar to ours, has been developed for use in hospitals by Roke Manor Research. The prototype attaches sensor devices to a patient that allows them to move about the hospital and be monitored by the staff. When the patient is in a fixed location, such as their bed, the portable unit is placed in a base station to allow for more detailed monitoring. The prototype system has also had some failsafe features in that if any abnormal signals are detected the hospital staff is alerted immediately [17].

Another telemedicine device similar to the mote-based system discussed here has been developed by Tadlys. The Tadlys' remote monitoring system uses Bluetooth for its wireless communication and offers the ability to monitor multiple patients in real-time from the same computer, as well

as archive patient data to a dedicated server for later review [18].

However these existing products do not deal with issues including the optimal location to place nodes throughout the environment. How handoff between nodes occurs is also not mentioned (although this information could be proprietary to the company to exclude its publication in the literature). The systems also do not discuss how often the database is archived to free up storage space, nor how many patients can the systems track. Also not discussed is how a patient's location is tracked to dispatch help in an emergency, only that the systems monitor patients for problems and alert the physician. Lastly not mentioned is if or how the battery is monitored so that the device can maintain power and thus maintain monitoring. These are challenging issues in embedded systems design and construction, especially in the telemedicine arena.

3 Methodology

Following is the description of the design methodology of the experiemental mote-based system for remote patient monitoring. The two major considerations toward the design approach are, first, the system architecture, and secondly the hardware design. The hardware design depends on prior research, which resulted in the fabrication of an ECG board for experimental research [1]. The software design approach is two-prong: the software architecture that allows interfacing of the hardware to a backend database for data collection, and a backend software system for the analysis of the data.

Although this research currently focuses on just monitoring a patient's ECG signals, the preliminary results show the potential that could include extra sensors to monitor other signals. For example, it is possible to connect an oximeter (oxygen sensor) so that the oxygen levels in the patient's blood can be monitored as well. Also it is possible to add extra sensors to monitor the relative temperature and humidity of the patient's environment. This could be used to signal an alert to the doctor that the patient is in an environment that puts them at risk and needs to be monitored closely.

The main issues that this research deals with relating to the remote patient monitoring system are:

- Reliably capturing the ECG signal data with the MDA300CA
- Maintaining a constant packet stream as the patients move through their environment
- Reliably archiving the signal data into the database

The remote patient monitoring system contains the following elements. The electrodes attached to the patient feed into an amplifier and then a filter producing the ECG signal. The signal is then sent to the analog to digital converter, the MDA300CA, where the signal is digitized.

The MDA300CA mounts onto a MicaZ mote and the mote puts the digitized signal into packets that are sent wirelessly back through a network of MicaZ's to the MIB600CA base station. The MIB600CA connects to a computer via Ethernet cable and streams the packets to the computer. The computer pulls the signal data out of the packets and then stores them in a database. Figure 2 depicts the system architecture of the patient monitoring system.

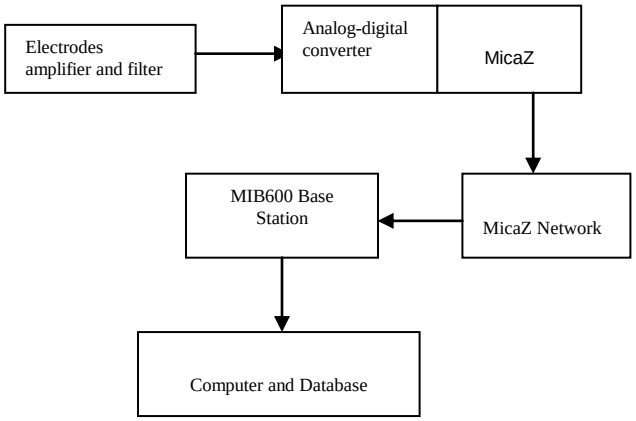


Figure 2 : *Block diagram of the remote patient monitoring system*

In the following, the two main components of the mote-based system are discussed, starting with the hardware design and implementation, followed by the software design and implementation.

4 Description of Hardware

4.1 The ECG Circuit

The ECG circuit used in this thesis consists of four componentss: the electrodes, the instrument amplifier, the low-pass filter, and a summing amplifier. The basis for this design comes from a combination of two similar designs. The first design was completed in a previous research[1]. The second design was done by a different research team [2].

The electrodes used in both designs are very simple, and consist of single strand wire with an alligator clip attached to one end. The other end connects to one of three locations on the ECG circuit. The negative lead connects to pin 1 of the instrument amplifier and the positive lead connects to pin 2. The ground lead connects to a 10Megaohm resistor that goes to ground.

Both designs utilize an instrument amplifier, specifically the AD624AD, which is connected in such a way that the signal gain is 1000. Both references [1, 2] discuss in detail why the AD624AD is an ideal choice for such a circuit.

The amplified ECG signal is output from the AD624AD, which then is fed into the second component of the circuit.. The 50Hz low-pass filter removes all frequencies above 50Hz from the signal. Most electrical devices in the United States are powered by a 60Hz AC source, such devices create

electrical noise at 60Hz. Because the cutoff for the low-pass filter is 50Hz all of the 60Hz noise from nearby electrical devices is removed from the ECG signal.

The last component in the circuit is a summing amplifier utilizing an LMC6464AIN operational amplifier. The summing amplifier is used to adjust the DC output voltage from the ECG circuit. This is because the analog to digital converter in the MDA300CA has a maximum input voltage of +2.5VDC. Therefore the summing amplifier is used to reduce the DC output voltage below the maximum 2.5VDC.

Figure 3 depicts a block diagram of the ECG circuit. (The complete circuit diagram is in [1].)

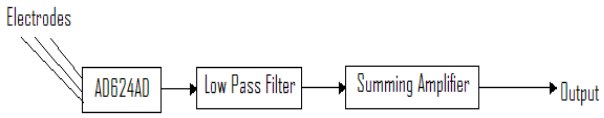


Figure 3 : *Block Diagram of the ECG Circuit*

4.2 The MDA300CA™

The MDA300CA is a data acquisition board designed for use in conjunction with either the Mica-2 ™ or MicaZ ™ series of motes developed by Crossbow Technologies Incorporated [3]. The MDA300CA contains inputs for seven different analog to digital single-ended channels, seven differential analog to digital channels, six digital input channels as well as temperature and humidity sensors and relays for turning external devices on and off. The ECG signals being output from the ECG circuit are input to analog to digital channel 0 (AD0). Figure 4 shows a picture of the MDA300CA.

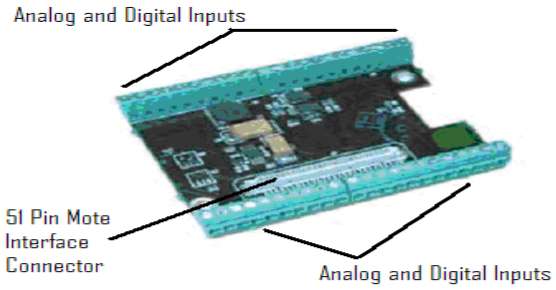


Figure 4 : *The MDA300CA ™ [4]*

4.2.1 The MicaZ ™

The MDA300CA connects to the MicaZ ™ by means of a 51-pin connector. The MicaZ ™ is programmed to poll the different input channels in the MDA300CA ™ and receive the resulting data. The MicaZ ™ can then packetize the data and transmit the packets wirelessly through a network of the MicaZ ™ motes back to a base station for processing [5].

The MicaZ ™ processor board, known as the MPR2400CA ™, consists of three major components. The first is the processor where the mote application runs, and any digital or analog I/O from the 51 pin connector is processed. The second component is the 512K of flash memory where any data that is to be stored on the mote occurs. The third component is the radio transceiver that allows the mote to communicate wirelessly with other motes and the base station [13].

The MicaZ ™ can be programmed to transmit on specific channels between 2400MHz and 2483.5MHz, with 5MHz steps in between each channel. This allows for the deployment of multiple groups of MicaZ's each performing its own tasks without each group causing transmission interference with the others. Each MicaZ ™ can also be programmed with a Group ID so that if multiple groups are deployed and are using the same transmission channel, packets that do not contain the appropriate Group ID can be rejected. Each MicaZ ™ is also programmed with its own unique ID within the group so that data from each individual mote can be identified by the base station [5]. Figure 5 is a picture of the MicaZ ™.

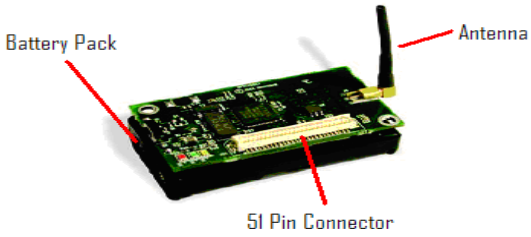


Figure 5 : *The MicaZ ™ [6]*

4.2.2 The MIB600CA ™

The MIB600CA ™ serves two functions. The first function is for the programming of the motes, and the second function is for serving as a gateway, bridging the wireless transmissions of the mote network to Ethernet [7].

The MIB600CA ™ can be interfaced with the motes in two different fashions. One method is by directly connecting each individual mote in the group, one at a time, to the MIB600CA by means of the 51-pin connector. The mote can then be assigned its unique ID and Group ID while a program is uploaded. The other method for programming the motes is by using Over The Air Programming (OTAP). As the name suggests the motes are programmed wirelessly by receiving the program either directly from the MIB600CA or from one of the neighboring motes. OTAP is very useful if the motes requiring programming are already deployed and retrieval is not feasible. It should be noted however that OTAP requires the motes to have a specific program called GoldenImage preloaded for the OTAP to work. Therefore before deployment each mote has to be connected to an MIB600CA ™ and programmed through the 51-pin connector [8].

The MIB600CA ™ bridges the wireless transmissions to Ethernet by connecting via its 51-pin connector to a mote with its node ID set to zero. This mote is loaded with software specifically designed to receive the packet streams from nodes corresponding to the Group ID of the mote, all other packet

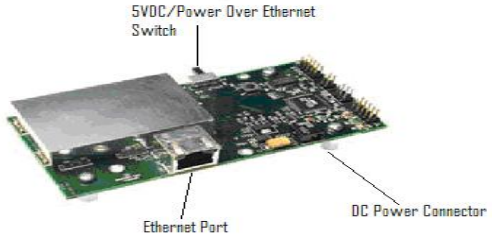


Figure 6 : *The MIB600CA ™ [9]*

streams are ignored. The MIB600CA then take the received packets and transmit them through the Ethernet port to a connected computer. The computer can then peform the desired analysis of the data [7]. Figure 6 depicts the MIB600CA™.

5 Software Description

5.1 TinyOS ™

TinyOS™ was developed as an open-source operating system specifically for wireless sensor networks. TinyOS ™ has a component library which includes such things as network protocols, distributed services, and drivers for sensors or data acquisition. Everything is built out of components which grants each user the flexibility to use things as is, or alternatively take advantage of the open source nature of each component and build their own custom components for custom applications [10].

Each application in TinyOS ™ uses a concurrency model based on tasks and hardware event handlers. As the application executes, each task runs to completion, tasks cannot preempt each other. The hardware event handlers execute due to some form of hardware interrupt and will also run to completion. Hardware events can preempt a task, or other hardware events. Any command or event that is to be executed as a hardware event handler must be declared using the keyword "async" [11].

Because preemption does occur it is possible to have a data race. Some of these data races can be found at compile time and are displayed as warnings to the user when the application is compiled. One of the best ways to avoid any data races is to have any accesses to shared data done within an "atomic" statement using semaphores, as any code within the atomic statement cannot be preempted. The user is cautioned against overuse of the atomic statement as it can prevent the execution of other tasks or handlers and affect performance or even cause the system to fail [11].

5.2 nesC ™

The TinyOS ™ applications, libraries, and components are all written with the nesC ™ language. The nesC ™ language was developed for use in embedded systems and is designed to support the concurrency model of TinyOS ™. The nesC ™ language uses a C-like syntax, but has some distinct features that separate nesC™ from C, as is required by the concurrency model [11].

Each nesC component is either a "module" or a "configuration." The modules contain the application code and the interfaces to the module. The configurations are used to create a connection between each of the modules in an application. The process of connecting modules via a configuration is referred to as "wiring" [11].

A nesC application is compiled using the "ncc" compiler which is based on the gcc compiler. Although it is possible to create the executable from the command line using "ncc," normally a *makefile* is created that runs the compiler and produces the binary file to be uploaded to the target device [11].

5.3 XSensorMDA300™

When the user installs TinyOS ™ and all the related mote applications onto their project development computer there are a large number of sample applications installed for each of the different sensor boards for the Crossbow series of motes. One of these sample applications is XSensorMDA300.

The XSensorMDA300 application consists of a module XSensorMDA300M.nc, a configuration XSensorMDA300.nc, and a header file sensorboardApp.h. The XSensorMDA300 application was written for the MDA300CA sensor board. The application polls each analog and digital input channel as well as all the other sensor devices on the board at defined intervals. The defined intervals can be found in the "implementation" declaration of the module file, for example:

#define ANALOG_SAMPLING_TIME 90

Where the time is given in tenths of a second, and so in the above statement sample 90 means that the analog channels will be sampled every nine seconds.

The resulting data is placed into one of four different packets. Each packet contains only some of the sensor results, for example packet one contains the resulting values for the first seven analog to digital channels. As each piece of data is placed into the packet a counter is incremented. When that counter equals a defined value the packet has been filled and

is ready for transmission. And example code fragment of the packet being filled is shown below.

```
switch (channelType) {
    case ANALOG:
        switch (channel) {
        case 0:
            packet[1].data[DATA_START+0]=data &
                                        0xff;
            packet[1].data[DATA_START+1]=(data
                                        >> 8) & 0xff;
            atomic {msg_status[1] |=0x01;}
            break;
            …
```

The program uses two switch-case control statements, first to determine which of the channel types was polled analog, digital, temperature, etc and then to determine which channel of that type that was polled. In the above example when analog channel 0 is polled the upper eight bits of resulting data is inserted into packet 1 in the location DATA_START+0, the very first location for data. The lower eight bits are placed in location DATA_START+1. The program then executes an atomic statement that cannot be preempted where it increments the fullness counter for packet 1.

5.4 The Modified XSensorMDA300 ™ for the ECG

For the purpose of capturing the ECG signals, it is not necessary to monitor all of the data input channels or other sensor devices on the MDA300CA. Therefore the existing XSensorMDA300 ™ application was modified so that only one analog to digital channel would be monitored at the desired sampling frequency of 100Hz. This was accomplished by changing the defined analog sampling time as shown in section 3.3 from 90 to 0.01. Also the packet length was shortened so that the packet is just long enough to transmit the value for the one channel making the transmission time shorter, and improving overall the throughput/capacity of the MDA300CA. This was accomplished by changing another defined value in the "implementation" declaration. The value for the constant MSG_LEN was changed from 29 to 8.

The unaltered program would sample all channels according to the defined sampling times. In the modified program all of the channel sampling declarations were removed except for the one channel that would have the ECG device connected to it, analog to digital channel 0.

A new variable was declared called PCKT_CNTR, which is used in an additional switch-case statement for filling the packets. When only one channel is being sampled it becomes necessary to add an extra control so that each of the four packets the XSensorMDA300 application can be filled separately. Therefore PCKT_CNTR is initially set to 1 so that on the first sample the first packet is filled. While the first packet is being filled PCKT_CNTR is incremented by one so that the next sample is placed in packet 2, and so on until packet 4 is filled and PCKT_CNTR is reset to 1.

5.5 The MoteListener Application

The MicaZ ™ on the MIB600CA ™ is loaded with a program called TOSBase. TOSBase is part of the TinyOS ™ install package and performs the function of a server. As the packets come to the base station from the deployed network the TOSBase application takes those packets and retransmits them out through the Ethernet port on the MIB600CA ™ to any client application that has a TCP connection to the MIB600CA ™.

Crossbow™, the company, provides tools for receiving and logging packets from the MIB600CA ™ that are designed to work with the sample programs provided. Since the packets that are being received by the base station are custom made it was necessary to create an application that could establish a TCP connection to the MIB600CA ™ and then parse the data from the packets as they come in. This application would then have to take the data from each packet and insert that data into a database for archiving/storing, becoming available later for retrieval, analysis, and display.

In the design of our system, the application, MoteListener, was written in C# and the database used for storing all of the ECG data was Microsoft SQL Server. (The choice of the SQL server allowed a front-end graphics module to be developed to interface with the ECG system for the display of streamed data in real-time using a Visual Basic platform.) As the Microsoft .NET package allows for programs written in different languages within the .NET package to be combined into one application, C# appeared a good choice for the back end system as it is possible to combine the two applications into one executable in the future. Further, the SQL database software was chosen because it supports complex operations/functions such as triggers, stored procedures, etc.

The MoteListener application performs three functions. The first is to establish a TCP connection to the MIB600CA ™. For experimentation purposes, the MIB600CA ™ has a hard-coded IP address of 192.168.0.43, and the default MIB600CA port is 10002. Once the connection is established the application performs its next function in the form of receiving each packet as the MIB600CA ™ transmits them. These packets are then parsed so that the identifying number of each mote and the corresponding ECG data values are extracted. The hexadecimal value of the ECG data is then converted to an integer, and then the integer is converted to a floating point millivolt value by use of a formula provided in the MDA300CA ™ users guide [12].

$$Value\ mV = 2.5 * (ADC\ Reading / 4096)$$

Once this value has been computed the application performs its last function by inserting the resulting ECG signal values and node ID's into the database.

The MoteListener application inserts several hard-coded values into the database along with the ID of the mote transmitting to the base station and the millivolt value of the ECG signal. The entire insert statement is as follows:

sqlDataAdapter1.InsertCommand.CommandText="INSERT INTO Analysis(AnalyID, Temperature, Humi, SignalVoltage, Enviro, BatteryVoltage, Name) VALUES ("+nodeNum+", 80, .60, "+newData+", 'Good', 2.5, 'Douglas Adams')";

The values for Temperature, Humi, Enviro, BatteryVoltage, and Name are currently (for simplicity) all hard coded. These values are included because it is possible for a future version of the ECG remote patient monitoring system to have additional sensors to monitor relative temperature and humidity, the battery voltage of the MicaZ, etc. It is also a relatively trivial task to use something like a case-switch selection statement to insert a patient's name based on the mote ID received from the packet. Therefore it was decided that the database and insert command should be designed with future systems in mind.

6 Experimentation and Observations

6.1 Direct Wireless Connection from the ECG to the Base Station

This section describes the verification procedures taken to ensure that the system was transmitting data from the mote connected to the ECG circuit back to the base station the following experiment was performed. It was observed that to program any of the motes it is first necessary to connect the MIB600CA ™ to the computer that the applications would be loaded from via an Ethernet crossover cable.

The MicaZ ™, connected to the ECG, was programmed with the custom XSensorMDA300 ™ application that was discussed above. Once programming was complete the MDA300CA ™ was mounted to the mote. For testing purposes the mote with the MDA300CA ™ was assigned node ID 4. The input for analog to digital channel 0 was then connected to the output of the ECG circuit, and the ground connection for the ECG circuit was connected to a ground terminal on the MDA300CA ™.

A second MicaZ ™ was programmed with TOSBase and assigned the node ID 0. This MicaZ ™ remained connected to the MIB600CA ™ to function as the server. Before any of the packets could be recorded it was necessary to turn on the SQL server since not doing so would cause a SQL exception to occur in the MoteListener application. Once the SQL server was running the MoteListener application was executed. Finally the volunteer patient is connected to the ECG circuit and the circuit is turned on.

It was then observed that there was data being transmitted from the ECG node back to the base station where the data was getting stored in the database verifying that the system was functioning properly. The limitations of this implementation are discussed in the following.

6.2 Limitations and Known Issues

6.2.1 Limitations of the Direct Wireless Connection

The direct wireless connection from the ECG node to the base station is known to have the following issues. First, the transmission rate for the packets was too slow to keep up with the sampling rate of the analog to digital channel. The sampling frequency of the analog to digital channel is 100Hz, but the incoming packet stream was significantly slower.

The packet transmission rate was estimated from the console print-statements, by visually counting the packets as they are displayed over a period of time. In this mode, it was observed that far less than 100 packets per second arrived at the base station. Therefore when the data captured by the MDA300CA was displayed the voltage values couldn't be properly interpreted as there were not enough samples to provide an accurate ECG reading.

The second limitation of the direct wireless connection was that there is a sever-related limitation placed on the mobility of the patient connected to the ECG. The specifications Crossbow provides for the MicaZ ™ state that the effective transmitter range indoors is 20-30 meters and outdoors is from 75-100 meters [13]. This is clearly not a long enough range for a patient to move around freely.

6.2.2 Limitations of the ECG System

The biggest problem faced with the ECG system was maintaining good skin contact with the electrodes. The electrodes initially provide good contact and provide good signal data either for display on an oscilloscope or for capture by the MDA300CA ™. However, over a short period of time the center conductor of the electrodes would pull away from the skin breaking the contact and thus providing no ECG signal. If the electrodes are pressed against the skin the signal could be read.

6.2.3 Possible Issues with the MDA300CA ™

There is a potential problem with the MDA300CA ™ sensor board. In previous designs from Crossbow(tm), the MDA300CA ™ was used in experiments with voltages being fed into the analog to digital channels. Upon examination of the MDA300CA ™ userguide and data sheets it was found that the maximum input voltage to the analog to digital channels is 2.5VDC [12]. For some of the experiments conducted earlier, the voltages fed into the analog to digital channels was in excess of the maximum, which could possibly damage an MDA300CA ™ and therefore providing inaccurate readings.

6.2.4 Possible Solutions for Known Issues

The problem of the packet throughput rate not being fast enough to handle the sampling frequency of the analog to digital channel could possibly be solved by using a data buffer. If the analog to digital channel samples for a given

period of time, one second for example, were all placed into a buffer then the entire buffer could be emptied into a packet and then transmitted as one-second worth of sample data back to the base station. This idea of using a buffer is similar to what was done in a previous work when the data was captured and sent over a wired network [1].

The buffer allows the base station to receive a larger sample that can be stored and displayed immediately. The drawback to using a buffer is that there will be gaps between packets where no signals are being received. Therefore there would be gaps when the patient's ECG signals are not being monitored. Also if the transmit-task is occurring then the task of sampling of the analog to digital channel could not occur meaning that there are gaps where the patients signals would not be recorded. By keeping the size of the buffer small, it is possible to have relatively small gaps or amounts of time between transmissions so that as little data as possible is not transmitted back to the base station.

The issue of maintaining good skin connectivity with the electrodes could be solved by using an elastic material such as that used for an elastic waistband. If the patient were to place waistband material around their chest and over the positive and negative electrodes then the waistband material would provide constant pressure on the electrodes pressing them against the skin. Since it is these authors' experience that every time the ECG signal disappeared and the electrodes were pressed against the skin the signal returned then it is believed that the elastic waistband material would work to solve the problem.

7 Conclusions

The use of wireless technology in medical care is a worthwhile endeavor. This research, following several similar projects, demonstrates the possibilities that wireless technology offers much to telemedicine application development. The ability to remotely monitor a patient is not only useful for the assisted living center model discussed earlier in this paper, but is also useful for the monitoring of patients in a remote location far away from a doctor's office or hospital.

During the course of the research the focus was on producing an ECG device that would feed signals into the analog to digital converter and then reliably streaming the signals back to the base station. This led to the discovery of two new issues, the first being the packet transmission rate that is too slow to provide an accurate ECG sample, and the second being that it is difficult to maintain good skin contact with the ECG electrodes. Also still missing from the system is the ability to multihop the data through a wireless mesh network back to the base station.

Although this research has advanced the prototype ECG remote patient monitoring system there is still more that can be done to make the system both reliable and deployable. The work done should be considered another step towards the goal of a deployable, commercial quality product.

8 References

[1] Chaudary Zeeshan Arif. "Wi-Fi Compatible ECG Monitoring & Interpretive Assistant", Masters Thesis, Southern Polytechnic State University, August 2004

[2] Carlson, Shawn. "Home is Where the ECG is", *Scientific American*,http://www.sciam.com/article.cfm?articleID=000C74E4-5172-1C74-9B81809EC588EF21, June 2000.

[3] Mica2 Data Acquistion Module (MDA300) Product Information ;http://www.xbow.com/Products/productsdetails.aspx?sid=77

[4] MDA300CA - http://www.xbow.com/Products/productsdetails.aspx?sid=77

[5] MicaZ ZigBee Series (MPR2400) Product Information http://www.xbow.com/Products/productsdetails.aspx?sid=101

[6] MicaZ Picture - http://www.xbow.com/Products/productsdetails.aspx?sid=101

[7] Ethernet Gateway (MIB600) http://www.xbow.com/Products/productsdetails.aspx?sid=90

[8] Wireless Sensor Networks Seminar Course book, Crossbow Technology Inc, November 2005

[9] MIB600CA Picture http://www.xbow.com/Products/productsdetails.aspx?sid=90

[10] TinyOS Mission Statement http://www.tinyos.net/special/mission

[11] Tutorial Lesson 1: Getting Started with TinyOS and nesC http://www.tinyos.net/tinyos-1.x/doc/tutorial/lesson1.html

[12] MTS/MDA Sensor and Data Acquisition Board User's Manual Document 7430-0020-04, January 2006 http://www.xbow.com/Support/Support_pdf_files/MTS-MDA_Series_Users_Manual.pdf

[13] MicaZ Datasheet - http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf

[14] [AHRI] - Aware Home Research Initiative http://www.awarehome.gatech.edu/projects/index.html

[15] Code Blue: Wireless Sensor Networks for Medical Care http://www.eecs.harvard.edu/~mdw/proj/codeblue/

[16] Qureshi, Asfandyar. Shoeb, Ali. and Guttag, John. "Building a High-Quality Mobile Telemedicine System using Network Striping over Dissimilar Wireless Wide Area Networks", Computer Science and AI Laboratory Massachusetts Institute of Technology, 2005 ; http://nms.lcs.mit.edu/papers/embc05-qureshi-horde.pdf

[17] Roke Manor Research Develops Wireless Patient Monitor, Roke Manor Research. April 30, 2003. http://www.roke.co.uk/news/article.asp?id=53

[18] Remote Monitoring Systems, Tadlys. 2004 ;http://www.tadlys.com/Pages/Product_content.asp?iGlobalId=3

FPGA Implementation of a Compact Genetic Algorithm using Cellular Automata Pseudo-Random Generator

Marco A. Moreno-Armendáriz[1], Carlos A. Cruz-Villar[2], Victor H. Ponce-Ponce[1]
[1]Centro de Investigación en Computación, Instituto Politécnico Nacional,
Av. Juan de Dios Bátiz s/n, México D.F., 07738, México.
email: *mam_armendariz@cic.ipn.mx*
[2]Sección de Mecatrónica, CINVESTAV, Av. Instituto Politécnico Nacional 2508,
Apdo. Postal 14 740, México D.F., México

Abstract— *In this paper the design and implementation of the compact Genetic Algorithm (cGA) and a Cellular Automata-based pseudo-random number generator on a Field Programmable Gate Arrays (FPGA) is accomplished. The design is made using the Hardware Description Language, called VHDL. Accordingly, the obtained results show that it is viable to have this searching algorithm in Hardware to be used in real time applications.*

Keywords: *Compact Genetic Algorithm; Cellular Automata; FPGA design.*

1. Introduction

Genetic Algorithms (GA) are very well known optimization techniques originally proposed in [1]. GA have demonstrated to be successful in the solution of complex numerical and combinatorial optimization problems, for single and multiple objectives [2] simulating natural evolution over populations of candidate solutions. GA handle a set of potential solutions instead of only one, but GA must evaluate the objective function several times, thus one of their main disadvantages is the high computing time required to solve complex problems. Some strategies have been proposed to deal with this drawback; for example, parallel implementations, efficient operators design, and hardware implementations, among others.

On the other hand, Compact Genetic Algorithms (cGA), are a kind of probabilistic model-building genetic algorithms or Estimation Distribution Algorithms (EDA) [3]. cGA operates on probability vectors by replacing the variation operators (crossover and mutation) that describes the distribution of a hypothetical population of solutions. It is known that the cGA obtains solutions of the same quality as the simple GA with binary representation and uniform crossover but with the advantage of an important reduction in the memory requirements, i. e. it only needs to store the probability vector (instead of the entire population). [4]. Therefore, the cGA may be useful in memory-constrained applications such as evolvable hardware [5].

In this paper, a novel and efficient design of a cGA in a hardware platform is shown. This design presents the following features: modularity, concurrency, minimal resource consumption, real time execution, and high scalability properties.

Nowadays there exists different computing machines to implement parallelism, nevertheless in this study we looking for a portable and autonomous evolvable hardware to be used in real-time tasks [22], therefore we select a FPGA to develop our design.

Also, it is important to mention that the population sizes in GA are problem-dependent, however in most of the cases they are from hundreds to thousands of individuals. Some exceptions are called "micro Evolutionary Algorithms" which have especially small population sizes, i.e. from 3 to 5, but in most of the cases they need additional mechanism to maintain the diversity.

The Genetic Algorithms are inherently parallel mainly due to their population-based nature. Thus, each individual of the population could be executed in a parallel manner. However, the compact Genetic Algorithm (cGA) has not a population, instead it only has a Probability Vector (PV), then, and its parallelization is slightly different.

2. Related Work

In this section we will present some of the most representative works including those related to Evolvable Hardware as well as those studies related to the compact Genetic Algorithms.

The term *Evolvable Hardware* is a research area that includes both, the design and implementation of Evolutionary Algorithms into hardware platforms to execute specific tasks; as well as the usage of Evolutionary Algorithms to generate hardware designs.

We will refer only to the former, that is, those works where the authors designed and implemented Evolutionary Algorithms into hardware. More than that, we will

refer only to the *Compact Genetic Algorithms* (cGA) (i. e. excluding other kind of Evolutionary Algorithms).

In 2001 Aporntewan and Chongstitvatana [10] proposed a cGA implementation in a FPGA using the language Verilog (Hardware Description Language). The authors showed that their design runs 1000 times faster than its software version executed in a workstation. The design is composed of five modules: random number generator, probability register, comparator, buffer, and fitness function evaluator. It is based on three basic operations: addition, subtraction, and comparisons. The probability vector updating is executed in parallel. It was tested on the Max-One problem with 32 bits and implemented in a Xilinix FPGA 23.57 MHz. using 15,210 gates and a population size equal to 256. This cGA executes one generation per three clock cycles for the Max-One problem. For other more complicated problems, one generation would take $3+e$ clocks, where e is the number of clocks used to evaluate the fitness function.

In [11], [9], the authors argued the cGA is very weak to optimize problems interesting for Evolvable Hardware's researchers. So, to overcome this issue, they proposed some modifications to the cGA by incorporating *elitism*, *mutation* and *resampling*. They demonstrated it increases cGA exploration capabilities without increasing the hardware complexity. It was implemented with VHDL in a XC400 BORG and Virtex xc2v1000 whose architecture is composed by the following five modules: Random number generator, registers for the vector and mutation probabilities, buffer (2-ports RAM storing the individuals), modules INC/DEC that update each probability vector element and, a register indicating which is the winning individual. It was tested using a Max-One problem with 32-bits and 255 individuals. Their results were compared against the ones in [10], reporting similar results with a very small increment of the consumed resources. Furthermore, the algorithm was implemented on Software and tested on the De Jong functions [12]. Additionally, the authors conducted experiments on some dynamic optimization problems.

In [13] a Cellular Compact Genetic Algorithm implemented in a FPGA was proposed. It consists of a set of identical cGA. Each is called a *cell* and interacts only with its four neighbours. Each cGA cell exchanges probability vectors with its neighbours in an asynchronous schema. The probability vectors are combined by using an equation proposed by the authors. They argued the Cellular cGA parallelization is straight forward and suited to be implemented in a FPGA. They experimented with the Max-One problem and two numerical optimization problems demonstrating that their proposal is better than the simple cGA.

Other interesting works were published in [5], [14], [15] where the authors implemented different versions of the cGA. They developed a real-valued version of the cGA obtaining solutions with the same quality of those found by binary cGA with reduced computational costs. Furthermore, they implemented the non-persistent elitist cGA in the same micro-controller used to implement the cascade control of an induction motor drive to self-tune its velocity and position. The cGA is executed on-line and the motor's sensors allow evaluating the fitness function. This work outperforms alternative schemata obtained with linear design techniques.

In [16] a novel architecture for a massive parallelization of the cGA is proposed. This schema presents three main advantages: low synchronization costs, fault tolerance, and high scalability. This architecture was simulated and tested in several processors (in software) obtaining an almost linear speed-up with a growing number of processors.

Only few publications about the analysis of the cGA can be found in specialized literature. Among them we can mention the following: The work published in [17] presents a rigorous runtime analysis of the cGA for pseudo-boolean functions on n variables, and proves that not all linear functions have the same asymptotic runtime. In [18] the authors modelled the cGA as a Markov process and approximated it by an ordinary differential equation with a small learning step. The differential equation was studied to determine its convergence and stability properties.

As a applications of GA algorithms, in [6] a methodology based on a genetic algorithm (GA) to automate the design of combinational logic circuits in which the aim is to minimize the total number of gates used is presented. In [7], a GA is used for evolving and testing new rules for intrusion detection, and finally in [8] the major problem of the computational requirements for optimizing the place and route operations of a VLSI circuit is studied, the authors investigates the feasibility of using reconfigurable computing platforms to improve the performance of CAD optimization algorithms for this problem.

3. Preliminaries

The Genetic Algorithms (GA) presented in [1] make use of a population of 100 or more individuals to search for a solution, eventually one of them is selected as final solution. For the case of the cGA (see Algorithm 1) the individuals of the population are not used directly, in really are represented through the probability vector PV, then two possible solutions ($ind1$ and $ind2$) are generated to find the final solution, which remains in PV.

3.1 The Compact Genetic Algorithm

The compact Genetic Algorithm (cGA), originally proposed in [4], is the simplest algorithm from the Estimation of Distribution Algorithms family [3] whose main purpose is a simplification of the Genetic Algorithm. The cGA generates two possible solutions through an estimated probabilistic model of the original population instead of using traditional recombination and mutation operations and it is operationally equivalent to the order-one behaviour of the simple GA with uniform crossover. The length l of the PV must be able to represent the length that has the desired solution.

The values of PV can be represented as real values $p_i \in [0,1], \forall i = 1, ..., l$ where l is the chromosome's length (binary string). It measures the proportion of '1' alleles in the i-th locus of the simulated population [19]. The cGA initializes the PV with 0.5 each of its elements. Two strings are generated using these probabilities. The fitness values for each generated individual are computed, then the PV is updated based on these strings. This process is repeated until the PV converges (to 0 or 1). The final PV is the solution found by the cGA.

The overall of the algorithm is as follows: The cGA initializes the PV to 0.5, that is $p_i = [0.5], \forall i = 1, ..., l$. Next, the strings $ind1$ and $ind2$ are generated according to the probabilities in PV. The fitness values of the strings $ind1$ and $ind2$ are compared and, the string with better fitness is named *winner* and the other is called *loser*. Then, if (winner[i] != loser[i]), the $PV[i]$ will be updated as follows: if (winner[i] = 1) then $PV[i]$ will be increased by $1/n$, otherwise, $PV[i]$ will be decreased by $1/n$. n is the population size that the cGA is emulating. Note that if (winner[i] =loser[i]), the $PV[i]$ will not be updated. The loop is repeated until each $PV[i]$ becomes zero or one. Finally, PV represents the final solution. The pseudo-code of the cGA is shown in Algorithm 1. It is assumed a maximization problem.

Algorithm 1 The compact Genetic Algorithm (cGA)

```
1:  INPUT: Population size n, chromosome length l (binary string)
2:  for i = 1 to l do
3:     {/*Initialize a Probability Vector*/}
4:     PV[i]:=0.5;
5:  end for
6:  while PV has not converged do
7:     {/*Generate two possible solutions from the vector PV*/}
8:     for i = 1 to l do
9:        {/* generate(PV[i]) returns an 1 with probability PV[i], or 0 otherwise.*/}
10:       Ind1[i]:=generate(PV[i]);
11:       Ind2[i]:=generate(PV[i]);
12:    end for
13:    if fitness(Ind1) > fitness(Ind2) then
14:       winner := Ind1;
15:       loser := Ind2;
16:    else
17:       winner := Ind2;
18:       loser := Ind1;
19:    end if
20:    {/*Update the PV towards the better one*/}
21:    for i:=1 to l do
22:       if winner[i] ≠ loser[i] then
23:          if winner[i]=1 then
24:             PV[i]:= PV[i]+1/n;
25:          else
26:             PV[i]:= PV[i]-1/n;
27:          end if
28:       end if
29:    end for
30: end while
31: OUTPUT: PV
```

4. Hardware Design

The proposed design accomplishes the following features:

- Modularity and Scalability: The complete design of the cGA is divided into components that perform a specific task. So that, it can be reused for future developments.
- Concurrency: Due to the nature of the algorithm, there are some parts that can be implemented in parallel, for example the updating of the probabilities vector and the individuals generation.
- Minimum resource consumption: In the proposal we explore different techniques for the generation of pseudo-random numbers.
- Real-time performance: Thanks to the characteristics of the selected technology it is possible to use the design for real time applications.

The complete design is conformed of five components, and the manner to joint them by using a Finite State Machine.

4.1 Hardware Components for the cGA

Considering that these designs (cGA and EcGA) will be tested with 32-bits functions (see subsection 5.1), the following components are customized to accomplish this case. Nevertheless, these could be modified to work with any size of this kind of functions. Considering this, the new components are explained next:

1) ***Pseudo-Random Number Generator (PRNG)***: This component generates the random numbers which are essential for the creation of the individuals. Any known random generator algorithm can be used. However, others will strongly impact on the overall design. We considered implementing the Cellular Automata Based Generator, explained next.

- **Cellular Automata based Pseudo-Random Number Generator (CA-PRNG)**: The idea of using Cellular Automata as generators of pseudo-random numbers was proposed in [20], where the authors used an one-dimensional cellular automata with n cells. The automata's states are '0' and '1'. In [23] a set of CA-PRNG with lengths from 4-bits to 53-bits was proposed. It used the transition rules known as Rule 90 and Rule 150 (so named according to Wolfram [20]). Rule 90 is $a_i(t+1) = a_{i-1}(t) \oplus a_{i+1}(t)$ and the Rule 150 is $a_i(t+1) = a_{i-1}(t) \oplus a_i(t) \oplus a_{i+1}(t)$, where $a_i(t+1)$ is the next state of the cell i, $a_{i-1}(t)$ is the present state of the cell $i-1$ (left neighbour), $a_i(t)$ is the present state of cell i (current cell), and $a_{i+1}(t)$ is the current state of cell $i+1$ (neighbour on the right).
  For example, Rule 0101 means cell '1' will use Rule 90, cell '2' will use Rule 150, cell '3' Rule 90 and cell '4' Rule 150. So, where there is a 'zero' Rule 90 is used, and where there is a 'one' Rule 150 applies. Cell memory (which stores the automata state) can be implemented with a synchronous flip-flop type D, and each rule is implemented with a LUT or a simple combinatorial function.
  For the case of the cGA (see Algorithm 1) with a chromosome length of 32-bits, we selected a population size of 50 individuals, then the lower number of necessary bits are 6. Therefore, a size of 6-bits for the vector PV is established.
  On the other hand, the initialization of PV is crucial to achieve an adequate behaviour of the algorithm. Thus, for the cGA version the design starts with a 48-bits CA-PRNG proposed in [23] (whose cycle length is $2^n - 1$), then two substrings of 6-bits are extracted to obtain 1-bit for $ind1$ and $ind2$. Now, since there are 48-bits is possible to use the other 6 substrings of 6-bits to obtain in total 4-bits for $ind1$ and $ind2$. Finally, another three CA-PRNG are required to complete the total 32-bits for $ind1$ and $ind2$, see Figures 2-3.

Some of the published papers [9]-[13], argue that any type of PRNG works fine for the cGA hardware design. However, it does not seem to be completely true, considering that the available resources in hardware are very limited. In this manuscript we present two different PRNG, showing that they directly affect the FPGA's resources consumption. As a result, it is necessary to do a carefully selection of the type of PRNG to be implemented. The PRNG should occupy the hardware components as less as possible, allowing more liberty to the cGA design, particularly, the fitness function.
Further, to maintain the FPGA's computations simple, a CA-PRNG results a better option because both, the cGA and the CA-PRNG, use binary representation. Hence, the comparisons between their values can be done without performing any extra computation.

2) ***Individuals Generator (IndGen)***: This component is responsible for the generation of a pair of bits, one bit for each individual; for this example, since the strings had a length of 32-bits, then it would have 32 components of this type. IndGen compares a pseudo-random number with an element of the PV. If the pseudo-random number is less than or equal to the corresponding PV element, then the generated bit is '1', or '0' otherwise. Notice that PV stores float numbers into $[0, 1]$ that must be increased/decreased with steps $1/n$ (n is the population size, see Lines 21 and 23 in Algorithm 1).On the other hand, the PV elements must be compared against the random numbers. Hence, the length of the binary chains representing each element of PV are the same length as the random numbers. In the case of CA-PRNG, it is necessary to utilize a substring of 6-bits to emulate a population with $n = 50$ individuals because it must store $n + 1$ possible numbers ($0/n$, $1/n$,..., n/n). For the MS-PRNG, the length of the PV was selected in 32-bits, however another size may be used.
3) ***Fitness Evaluator (FEv):*** This component is used to evaluate the fitness function. Its design depends on the objective function to be optimized. This block can be optimized according to the designer's ability to implement complex functions of segmented or parallel mode.
The input vectors are the two strings $Ind1$, $Ind2$, and RES as an output signal, it generates a '1' if the ability of the individual $Ind1$ is greater than or equal to that of the individual $Ind2$; or '0' otherwise.
4) ***Probability Vector Updater (PVU):*** The task performed by this component is adding or subtracting a certain amount ($1/n$ where n is the population size equal to 50) to each element of the PV (see Lines 21 and 23 in Algorithm 1).
If the 6-bit substring of CA-PRNG is utilized, then the PV elements are increased/decreased by one unit. For the case of 32-bit MS-PRNG the step is equal to 42949672.
To update the PV elements, three signals are received: (1) the winning individual (op), (2) the

bit from individual one of the $i-th$ position (ind1[i]), and (3) the bit from individual two of the $i-th$ position (ind2[i]). The necessary condition to update the $i-th$ element of PV is that ind1[i] and ind2[i] are different. If this is the case, then an addition or subtraction is applied to that element. This updating process is computed in parallel[1].

5) ***Probability Vector Checker (PVC):*** This component verifies if all the elements of the PV vector are '1' or '0'. When this condition is reached, the register RDY_PVC is equal to one, indicating that the cGA has finished.

4.2 Connecting the components

The complete design of the proposed architecture is illustrated in Figures 1, 2, 3.

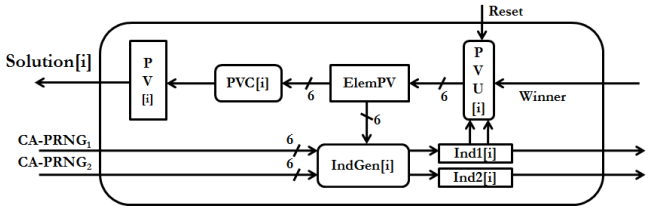


Fig. 1: One bit element of the cGA proposed design.

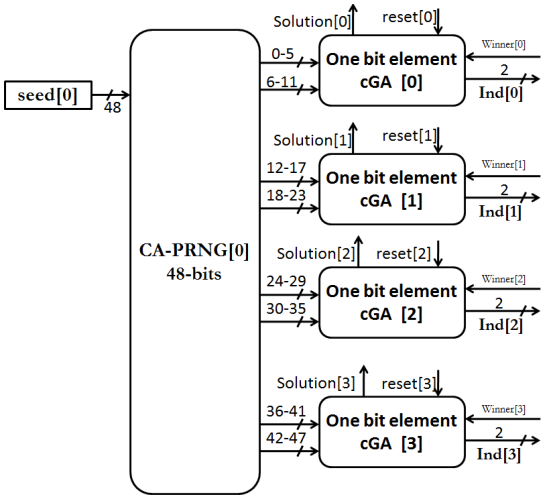


Fig. 2: Four bits element of the cGA proposed design.

Each pair of registers *seed*[i] contains the seed as input to the CA-PRNG[i] (there are two versions: 48-bits and 44-bits), which generates two 6-bits random numbers for cGA design. Then, IndGen[i] receives these two random numbers and the PV[i], to generate two bits corresponding to the registers Ind1[i] and Ind2[i].

[1]It is important to note that the values in PV must be kept into the allowed range.

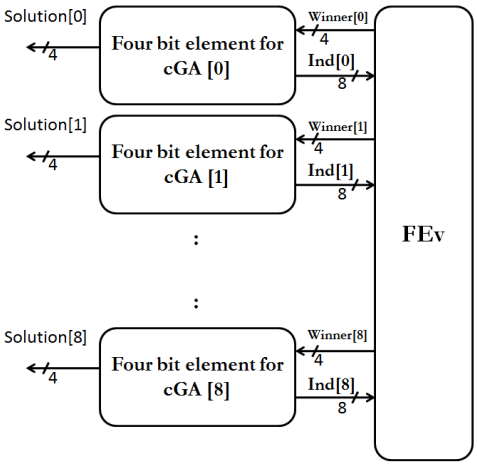


Fig. 3: 32-bits element of the cGA proposed design.

After that, the two bits are used as input to FEv. Then, FEv receives all the pair of bits, that make up two complete individuals, and to determine which of them is the winner. With this information, the component PVU[i] updates its corresponding PV[i] value. Then, the component PVC[i] verifies the PV[i] convergence. In that case, the whole PV is given as the algorithm output.

In order to synchronize all the components, a Finite State Machine (FSM) is used [13]. The usage of this machine will guarantee the correct behaviour of the whole design. The FMS is defined by two functions: the first calculates the next state of the system, and the second, the output. A Mealy Machine is applied, which uses a clock as a synchronizing signal for transitions. The FMS is shown in Figure 4.

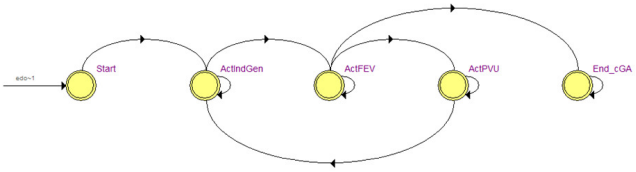


Fig. 4: Finite State Machine (FSM) of the proposed design.

5. Experimental results

5.1 Test functions

To evaluate the performance of the cGA in hardware, we use five types of test functions: Max-One, Min-One, Hamming-1431655765, Hamming-858993459 and Hamming-477218588. The so-called Hamming experiments consist of the minimization of the Hamming distance to the corresponding binary chain; for example,

the Hamming-1431655765 indicates minimizing the distance to the chain '01010101010101010101010101010101' represents the decimal number 1431655765.

Hardware	
Altera Cyclone II EP2C70F896C6 with a 50 MHz clock, and the Altera's Quartus II tool was used for designs using VHDL language.	
Total logic elements	$1,791$ out of $68,416 \approx 3\%$
– Total combinatorial functions	$1,790$ out of $68,416 \approx 3\%$
– Dedicated logic registers	754 out of $68,416 \approx 1\%$
Total registers	754
Embedded Multiplier (9-bits)	0 out of $300 = 0\%$
Software	
cGA programmed in C++, executing in AMD Athlon 64 X2 Dual Core 4000+ at 2.09 GHz with 1 GB RAM and Ubuntu 8.04	

Table 1: Experimental conditions

	HW	**SW**
	cGA	cGA
Max-One		
Iterations	694	672
Execution time	82.41 μs	4090 μs
Min-One		
Iterations	849	833
Execution time	101.4 μs	4830 μs
ReverseHamming-1431655765		
Iterations	657	628
Execution time	74.22 μs	3860 μs
ReverseHamming-858993459		
Iterations	712	695
Execution time	84.33 μs	4270 μs
ReverseHamming-477218588		
Iterations	691	686
Execution time	78.6 μs	4180 μs

Table 2: Comparison between the hardware and software

5.2 Discussion

Here some important issues about the Hardware implementation are discussed.

- It is obvious that the execution time depends on the hardware clock frequency. However, a manner to measure the performance independent of the hardware velocity, is obtaining the number of clock cycles consumed by one algorithm iteration. Thus, in this implementation, one algorithm iteration requires 6 clock cycles, no matter the clock frequency utilized. For this specific hardware platform, with a frequency of 50 Mhz., the number of algorithm iterations per second is computed with

$$Gen_{second} = \frac{Frecuency(Hz)}{clock_{generation}} \quad (1)$$

  Then, it can perform 8.33 millions of generation per second.

- The goals established in Section 4 were successfully achieved. The Modularity, Scalability and Concurrency were implemented since each bit of the individuals are generated in separate components. This allows increasing the size of the binary chain very easily. By the use of the CA-PRNG we reduce the consumption of the FPGA resources and at the same time adequate pseudo-random numbers are used in the cGA algorithms.
- The minimal resource consumption stands for all proposed hardware components without consider the resources required for the FEv component (this block can be optimized according to the designert's ability to implement complex functions), for the tested FEvt's in this paper only 3% of the FPGA resources are used. However, we carefully analyze all aspects of the hardware design of the remains components such as: data representation (fixed or floating point) and arithmetic representation. These aspect are very important in order to avoid use all of the resources on the FPGA. So, It is important to maintain a compact cGA design, because it could be attached to other components, such as a Neural Network-based controllers for a real application, resulting necessary to reach a high performance design that consumes as less FPGA's resources as possible.
- Each PB element stores a value into the range [0,1] that is utilized as the probability to generate an 1 (or complementary a zero).To maintain the FPGA's computations simple, each PV element is compared directly against the random number just generated. Thus, it seems convenient that both, the PV values and the generated random numbers, are into the same range.

  On the other hand, the cGA implemented in this work uses binary representation, which means that each PV element uses binary representation; therefore it becomes necessary to apply a discretization of its values. Considering that the cGA algorithm updates the PV values by increments/decrements of $1/n$, (where n is the size of the emulated population) it would be enough if the discretization contains the values $0/n, 1/n, 2/n, ..., n/n$. In most of the cases, the population sizes n go from 50 to 100, therefore, each element of PV could be represented with 6 bits (with $n \in$ [50,..., 64] or 7 bits (with $n \in$ [65,...,100]). The cGA and the CA-PRNG seem to be very suitable to work together, especially for their implementation on an FPGA device, mainly because they both use binary representation, which avoids further expensive computations.
- Finally, since the execution time (see Table 2) is in the order of microseconds a real-time implementa-

tion is very feasible, some successful published works are in [14], [15] and [22].

6. Conclusion and Future Work

While it is true that any known pseudo-random numbers generator would work, here we found that the MS-PRNG occupies almost all the resources with a relatively small binary chain. Thus, it could be useful only with very small problems. However, if the Cellular Automata-based PRNG (CA-PRNG) is applied, the design utilizes only 3% out of the FPGA resources (see Tables 1-2).

For the kind of problems whose objective function can be represented with binary numbers, our design is suitable to be used in real applications. So, if the complexity of the fitness function is bigger, then the designer needs to explore different ways to maintain a good balance between the velocity and the amount of FPGA resources to be used.

The previously published cGA hardware implementations [10], [5], [14], [15], [11], [9], [13] do not present a detailed description of their designs, i. e. they do not show the components structure (I/O ports) or the manner to synchronize them. Furthermore, the presented design utilizes 6 clock cycles per one iteration.

As a future work we plan to use these algorithms for tuning intelligent controllers, like the fuzzy visual control developed by our research group in [21]. This is a successful case where the implementation of the whole system in hardware gives an excellent performance in real time. However, one of the biggest problems is the fuzzy tuning. Therefore, we want to use our hardware version of the cGA algorithms as a self-tuning tool for these kinds of controllers. Another successful application was presented in [22].

7. Acknowledgements

The authors wish to thank to Instituto Politécnico Nacional (SIP-IPN grants 20131514 and 20130086, COFAA-IPN y PIFI-IPN), CINVESTAV and the Government of Mexico (SNI y CONACYT) for providing necessary support to carry out this research work.

References

[1] J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan, 1975.

[2] C.A. C. Coello, D. Veldhuizen, and G. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems.* Kluwer Academic, 2002.

[3] P. Larranaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation.* Kluwer Academic Publishers, 2001.

[4] G.R. Harik, F.G. Lobo, and D.E. Golberg, "The compact genetic algorithm", vol. 3, no. 4, pp. 287–297, 1999.

[5] F. Cupertino, E. Mininno, and D. Naso, "Elitist compact genetic algorithms for induction motor self-tuning control" in *IEEE Congress on Evolutionary Computation, CEC*, 2006, pp. 3057–3063.

[6] C.A. Coello Coello and A.D. Christiansen and A. Hernández Aguirre, "Towards automated evolutionary design of combinational circuits", *Computers and Electrical Engineering*, 27(1), 1-28, 2000.

[7] Z. Banković and D. Stepanović and S. Bojanić and Octavio Nieto-Taladriz, "Improving network security using genetic algorithm approach", *Computers and Electrical Engineering*, 33(5-6), 438-451, 2007.

[8] S. Coe and S. Areibi and M. Moussa, "A hardware Memetic accelerator for VLSI circuit partitioning", *Computers and Electrical Engineering*, 33(4), 233-248, 2007.

[9] J.C. Gallagher, S. Vigraham, and G. Kramer, "A family of compact genetic algorithms for intrinsic evolvable hardware", *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, 2004.

[10] C. Aporntewan and P. Chongstitvatana, "A hardware implementation of the compact genetic algorithm" in *Proc. 2001 IEEE Congress Evolutionary Computation*, 2001, pp. 624-629.

[11] J. C. Gallagher and S. Vigraham,"A modified compact genetic algorithm for the intrinsic evolution of continuous time recurrent neural networks" in *Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann Publishers, 2002, pp. 163-170.

[12] K. A. DeJong, "An analysis of the behavior of a class of genetic algorithms", Ph.D. dissertation, 1975.

[13] Y. Jewajinda and P. Chongstitvatana, "Fpga implementation of a cellular compact genetic algorithm" in *NASA/ESA Conference on Adaptive Hardware and Systems*, 2008, pp. 385-390.

[14] F. Cupertino, E. Mininno, E. Lino, and D. Naso, "Optimization of position control of induction motors using compact genetic algorithms" in *32nd Annual Conference on IEEE Industrial Electronics, IECON*, 2006, pp. 55–60.

[15] ——, "Compact genetic algorithms for the optimization of induction motor cascaded control" in *Electric Machines & Drives Conference, 2007. IEMDC '07. IEEE International*, vol. 1, 2007, pp. 82-87.

[16] F.G. Lobo, C. F. Lima, and H. Martires, "An architecture for massive parallelization of the compact genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, vol. LNCS 3102. Springer, 2004, pp. 412-413.

[17] S. Droste, "Not all linear functions are equally difficult for the compact genetic algorithm" in *Genetic and Evolutionary Computation Conference (GECCO-2005)*, vol. 1, 2005, pp. 679-686.

[18] R. Rastegar and A. Hariri, "A step forward in studying the compact genetic algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 277-289, 2004.

[19] C. W. Ahn and R. S. Ramakrishna, "Elitism-based compact genetic algorithm", *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 4, pp. 367-385, August 2003.

[20] P.D. Hortensius, R.D. McLeod, and H.C. Card, "Parallel random number generation for vlsi systems using cellular automata", *IEEE Transactions on Computers*, vol. 38, no. 10, pp. 1466-1473, 1989.

[21] M.A. Moreno-Armendáriz, C.A. Pérez-Olvera, F. Ortiz Rodríguez, E. Rubio-Espino, Indirect Hierarchical FCMAC control for the ball and plate system, *Neurocomputing*, 73, 2454-2463, 2010.

[22] E. Mininno, F. Cupertino, and D. Naso, "Real-Valued Compact Genetic Algorithms for Embedded Microcontroller Optimization", *IEEE Transactions on Evolutionary Computatation*, 12(2), 203-219, 2008.

[23] P.D. Hortensius , R.D. McLeod , W. Pries, D.M. Miller ,H.C. Card, Cellular Automata-based Pseudorandom Number Generators for Build-In Self-Test, IEEE Transactions on Computer-aided Design, 8(8), 842-859, 1989.

Design And Implementation Of An Embedded Obstacle Detector For The Visually Impaired

Ihekweaba Chukwugoziem[1], Nwachukwu-Nwokeafor K.C[1], Ihekweaba Ogechi[1]

[1]Computer Engineering Dept. Michael Okpara University of Agriculture, Umudike, Abia State Nigeria.

ABSTRACT: Physical disability has been a major set back to individuals, their families and subsequently national development. Visual impairment is a major physical challenge that has rendered many helpless. This paper showcases the need for the visually impaired (the blind) to detect obstacles beyond knee level at a long range and in real-time. It further presents current works within the scope of the project. Also, common obstacle detection techniques are elucidated. Finally, the system hardware development, software requirements, specifications, conclusions and recommendations were presented.

Keywords; Obstacle detection, microcontroller, transmitter, assembler, astable..

1.0. INTRODUCTION.

Obstacle detection technology is widely used globally in many areas and aspects of life within the areas of domestic operations, manufacturing, medicine, as well as space activities. In these fields, the obstacle sensors are used for detection, counting and avoidance as well, both for moving and stationary objects. The basic techniques used, involve ultrasonic sensors and infrared sensors which offer varied attendant coverage ranges, design specifications, operational principles and development techniques. In the past, efforts have been made by the General Motors Cooperation (GMC), USA, to protect a driver from running over objects or obstacles in the blind spots when backing up. An advertisement aired in the spring, of 1984 displayed a car equipped with an object detector of a small bicycle parked behind a car. Also, DELCO Electronics came up with the Near Object Detector System (NODS) for automobiles with poor rear visibility. Several other works have been done in this area with the introduction of Radar (40GHz), Infrared, Lasers, and Ultrasound Techniques. However, the radar approach was found to be the most effective over others due to itôs immunity over several environmental factors such as; rain, ice, snow, dirt, etc. Other related works include, the Tuvie; an electronic replacement for the White Cane which uses a distance sensor to inform(using voice) an end user of detection through a Bluetooth earpiece. Another scenario involves the use of a live camera to convert grayscale images into sounds referred to as sound capes. In the Prosthesis, a substitute for vision with audition, a device similar to the voice but with a head-mounted Video camera is used for capturing images and sending them to a standard digitizing board in a computer. Also, the Isreali Radar for the blind which uses two video cameras and a high source to provide audio feedback were demonstrated with the sensor vest which uses ultrasonic sensors in a vest. In this system, ultrasonic sensors installed in a vest and shirt lead to vibration which is coordinated and processed to yield a feedback on the direction in which an obstacle is located. Generally, all these techniques with their attendant disadvantages had in no doubt made their contributions. It thus becomes imperative that basic tools for obstacle detection which include, the digital cameras working with the aid of the Laser, common Radar, Microwave-based Radar, Digital Laser range finders, and Infrared sensors are deployed in this system development.

2.0. MATERIAL AND METHODS.

The hardware components used in this work include; sensors, resistors, capacitors, transistors, Light emitting diodes and the microcontroller. The resistor is a two- terminal electronic component that produces a voltage across its terminals that is proportional to the electric current through it, in accordance with Ohmô law; V= IR [1]. The device was used to regulate current and voltages at various points on the circuit. The design involved the calculations of various parameters which include; resistance, the tolerance, the maximum working voltage and the power rating. The capacitor, another passive [2] electronic component consists of a pair of conductors separated by a dielectric (insulator) [3]. An ideal capacitor is characterized by a single constant value, the capacitance, measured in farads. This is the ratio of the electric charge on each conductor to the potential difference between them. In this system, the Capacitors were used at the power supply for blocking direct current while allowing alternating current to pass, this was necessary for smoothing the output of the power supplies. The transistor, an active component was used as a simple amplifier. A major component deployed in this system is the microcontroller. The Microcontroller, a digital[4] device, usually contains four several to dozens of general purpose input/output pins (GP10). A GP10 pin is software configurable to either an input or an output state. When GP10 pins are configured to the input state they are often used to read sensors or external signals in real time control operations[8]. . A less common feature on some microcontrollers is a digital-to-analog converter (DAC) that allows the processor to output analog signals or voltage levels. The 89C51 is a 40-pin digital[8], IC. Thirty-two pins are needed for the four I/0 ports. To provide for the other microcontroller control signals [10], [11], [12], most of these pins have alternative functions, which are shown in figure 1, below;

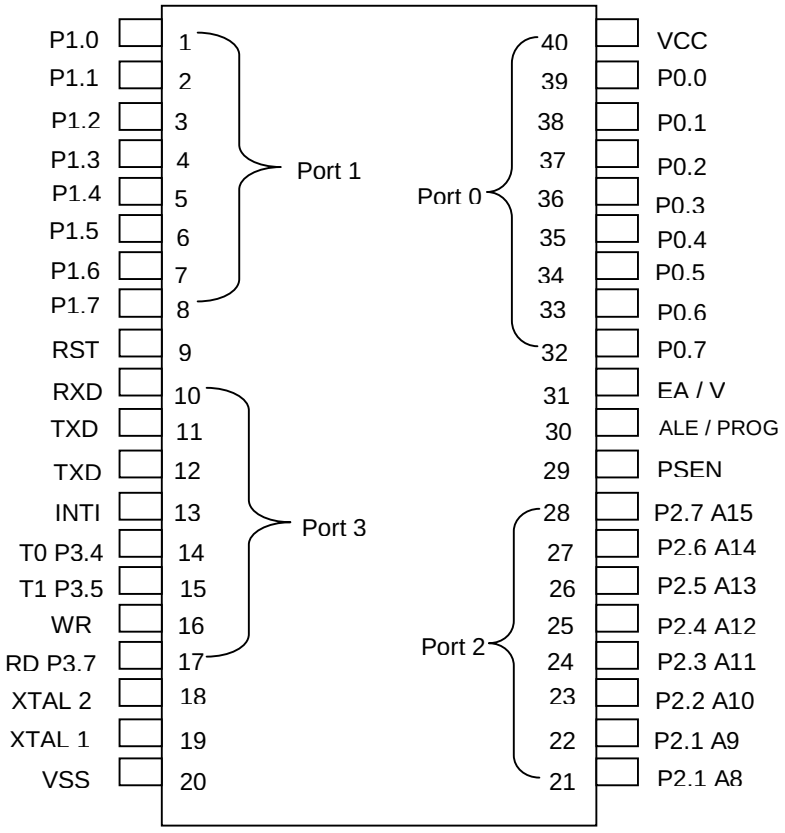


Fig.1; The 89C51 pin configuration

PORT 0 (pin 32 ï 40) is dual purpose serving as either an 8-bit bidirectional I/0 port (p0.0-p0.7) or the low other multiplexed address/data (AD0 ï AD7). AD0-AD7, are used to access external memory. They are activated automatically whenever reference is made to external memory. The AD lines are demultiplexed into A0 ï A7 and D0 ï D7 by using the ALE signal.

PORT 1 (pin 1-8) Port1 is an 8-bit bidirectional I/0 port.

PORT 2 (pin 21-28) Port 2 is a dual-purpose, serving as either an 8-bit bidirectional 1/0 port (P2.0 ï P2.7) or as the high order address bus (A8 ï A5) for access to external memory. The port becomes active as the high order address bus whenever reference to external memory [6] is made.

PORT 3 (pin 10-17) Port 3 is a dual-purpose, serving as an 8-bit bidirectional I/0 port. Its functions listed in table 2-1.

$\overline{\text{PSEN}}$ (pin 29) Program Store Enable is a read strobe for external program memory. It is connected to the Output Enable (OE) of an external ROM or EPROM.

$\overline{\text{EA}}$/VPP (pin 31) External Access (EA) is tied low to enable the $\overline{\text{micro}}$controller to fetch its program code from an external memory IC This pin also receives the 21V programming supply voltage (VPP) for programming the EPROM parts.

XRAL 1, XTAL 2 (pin 18, 19) Connections for a crystal or an external oscillator.

The address spaces of the 89C51 are divided into four distinct areas; internal data memory, external data memory, internal code, memory, and external code memory.

The 89C51 allows for up to 64k of external data memory (RAM) and 64k of external code memory (ROM/EPROM). The only disadvantage of external memory, in addition to the additional circuitry, is that ports 0 and 2 get tied up for the address and data bus. When 89C51 is first reset, the program counter starts at 0000H. if EA is tied low, the CPU, issues a low on PSEN (Program Store Enable), enabling the external code memory ROM/EPROM instead.

With EA $\overline{\text{tied}}$ HIGH, the first 4k of program instruction fetches are made from the internal code memory. Any fetches from code memory above 4k (10000H is FFFFH) will automatically be made from the external code memory. If the application has a need for large amounts of data memory, then external data memory (RAM) can be used. I/0 to external RAM can only be made by using one of the MOVX instructions [5]. When executing MOXV instruction, the 89C51 identifies a reference to external RAM, it thus issues the appropriate WR or RD signal.

3.0 SYSTEM HARDWARE DESIGN DEVELOPMENT

The system developed is based on the infra red technology. The infrared system as shown in figure; 2 and 3 below, provides for an interface between the hardware and any possible software integration. The detection unit of the system realized consists of two components; an infrared emitting diode and an infrared-sensitive photo transistor, with the other unit containing an infrared reflector, as showcased effectively in the figure;4. When positioned in-front of an entrance to a protected area, the two units establish an invisible beam, and any object captured within the area leads to an interruption of the beam path subsequently, an alarm is triggered .

THE BASIC CONCEPT OF MOTION DETECTORS.

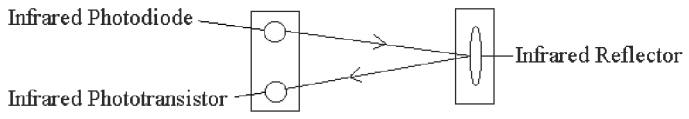


Fig.2. The Infrared Detector Unit.

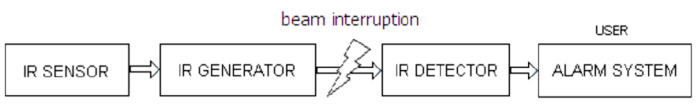


Fig.3: The Block Diagram of Infrared Obstacle Detection Systems, (IODS).

The system hardware consists of four units as shown fig.3, below.

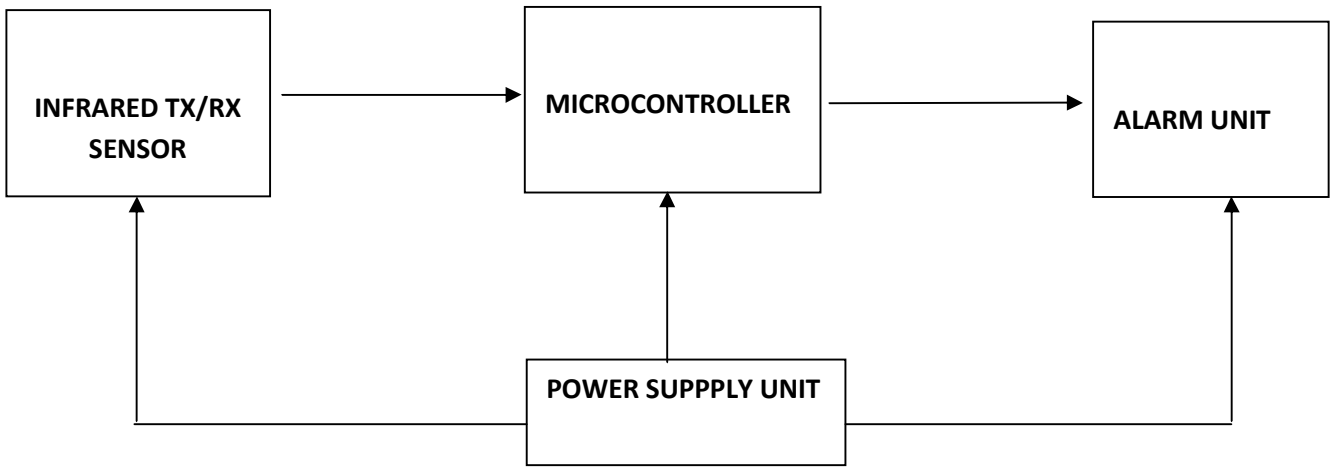


Fig.3, A block diagram of the Intelligent Infrared Obstacle Detector.

In this system, when an object or obstacle is standing in front of a blind person, the infrared receiver-transmitter component is agitated. A 555timer integrated circuit operating in an astable mode injects the required current to the Light Emitting Diode (LEDôs), thus ensuring a long range of operation. There are three such units, each operating at different frequencies and positioned at various directions; North, East, and West

The receiver uses a sharp IR module when the IR beam from the transmitter falls on the IR module; the output is activated or goes ñHighò. But when there is obstruction, it goes ñLowò.

But in this project, we placed the transmitter and receiver module side by side to each other. Meanwhile, the infrared transmitter continues to transmit its 48 KHz signal. Once there is an object in front of the transmitter, the IR beam is reflected back to the receive module. This causes the receive module to go ñHIGHò. The microcontroller uses the high state to turn ON the specific alarm signal dedicated to the direction, thus indicating that an object has been detected or is standing in the way at that direction. The complete circuit diagram is shown below.

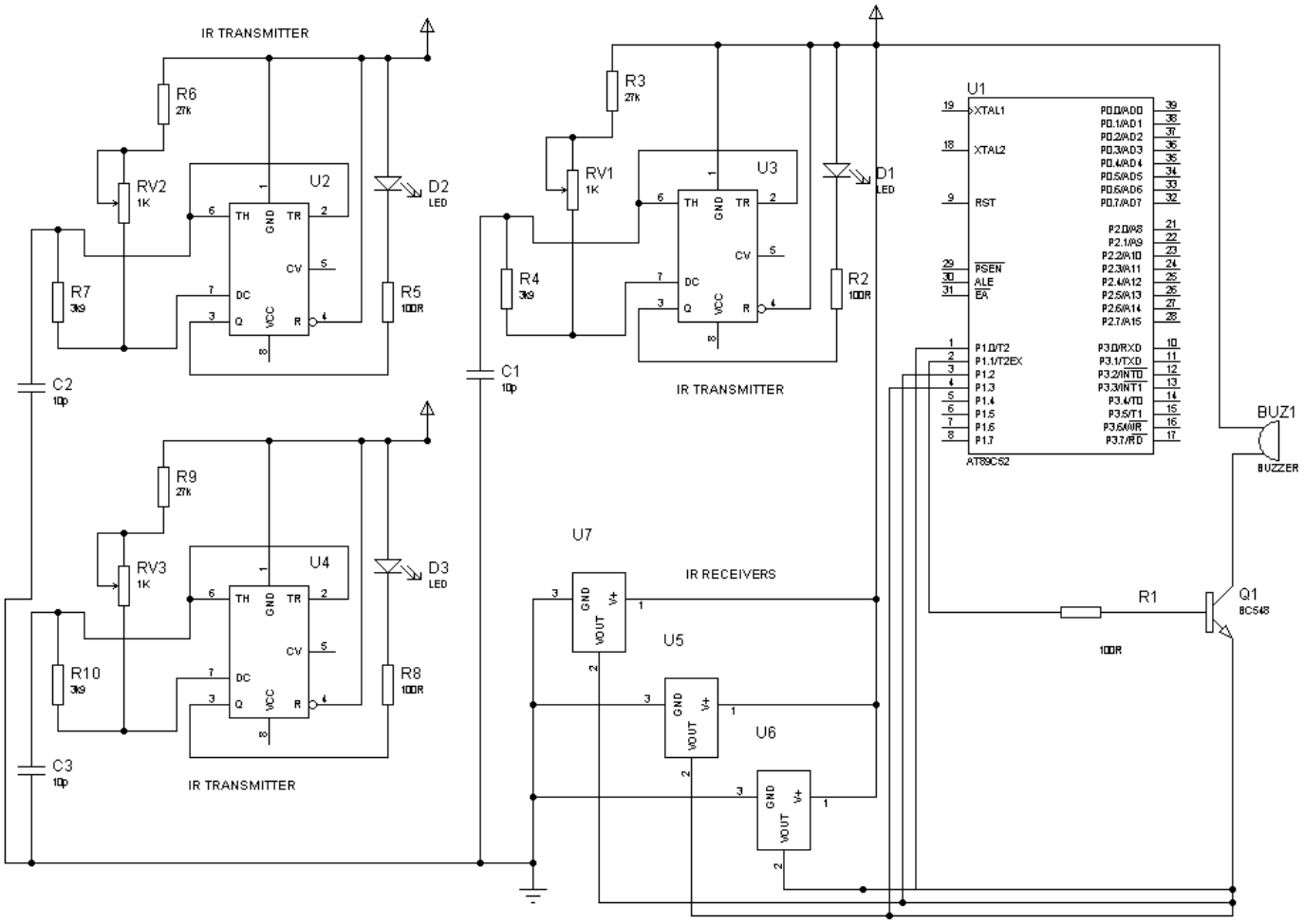


Fig.5. System circuit diagram.

4.0 SYSTEM SOFTWARE DEVELOPMENT

The system flowchart is shown in Fig 6 below. The algorithm continuously monitors all the directions, and generates the appropriate alarm signal when it detects an obstacle in the direction monitored. The software was finally developed using the 89c51 assembly programming language kit.

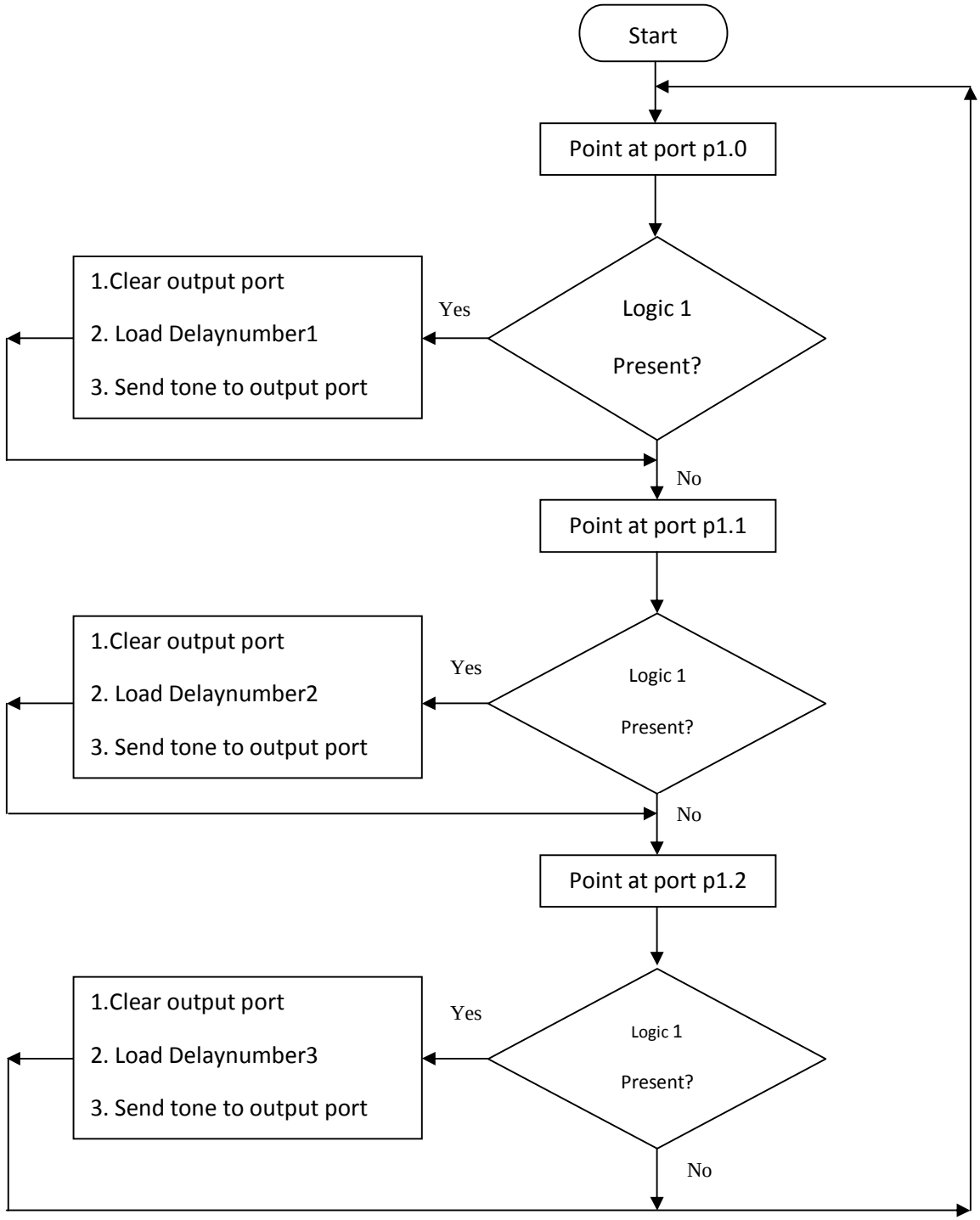


Fig 6 Flowchart for embedded obstacle detector for the visually impaired

5.0 TESTS AND RESULTS

The transmitters were tested for proper signal generation using the oscilloscope. Fig 7 below is the output of transmitter 1 which was developed to produce a 38khz signal at 10% duty cycle.

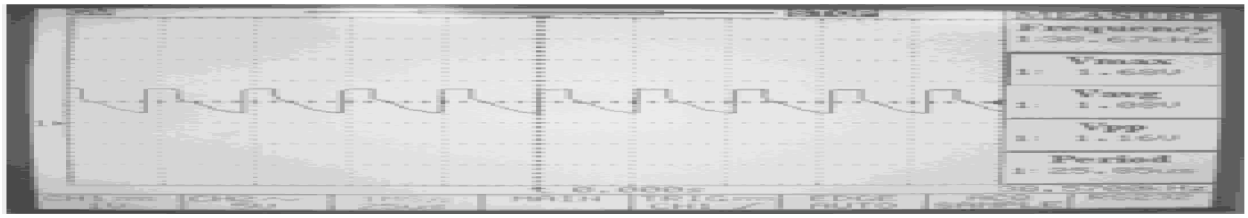


Figure 7. The 38 kHz frequency generated by the emitter circuit.

The hardware was developed and tested with the circuit wizard application software. The 89c51 assembly programming language [7] kit was used in the development, debugging and translation of the system software. The system was finally integrated with an eprom programmer.

6.0 CONCLUSION

This system provides a low cost alternative to the visually impaired persons in their ability to move freely in their homes without being aided. The system achieved an appreciable obstacle detection above knee level and at good distance and with the attendant interface capabilities offered by the use of the microcontroller in its implementation.

REFERENCES.

[1] Alan Clements,2009, ñ The Principles of Computer Hardwareò, Oxford University PressPp 338 ï 355

[2] Anil K Maini, 2006, ñElectronics and Communication Simplifiedò Khana Publisher Pp 1 ï 23

[3] B.S Chalk A.T Carter, R.W Hind, 2006, ñComputer Organisation and Architectureò, Palgrave MacMillan Pearson Prentice Hall, Pp 259 ï 315

[4] Carter J. W., 1996, ñMicroprocessor Architecture and Microprogramming - A State Machine Approach.ò Prentice-Hall, Englewood Cliffs, pp100,120

[5] Gothmann W. H., 1977, ñDigital Electronics - An introduction to Theory and Practice.ò Prentice-Hall, Englewood Cliffs, NJ, USA, pp1-75

[6] Jimmie J Cathery, Syed A Nasar; 2004 ñ Schaumôs Outline Series; Basic Electrical Engineeringò , Tata McGrawHill , Pp 3 ï 4

[7] Dielectric ï Ashutosh, Pramanik , 2006 ñ Electromagnets Theory and Applicationsò, Prentice Hall of India, Pp 68 ï 69

[8] John P, Hayes, 2005, ñDigital System design and Microprocessorò Tata McGrawHill Pp 1 ï 69

[9] Microchip PIC16/17 Microcontroller Data Book. Microchip Technology Inc., 1996.

[10] Microchip Embedded Control Handbook 1994/95 Microchip Technology Inc., 1994.

[11] Microchip PICStart Design Contest Application Brief Notebook Microchip Technology Inc., 1993.

[12] P. B Ram, 2008, ñFundamentals of Microprocessor and Microcomputers,ò Pahupat Rai Publications. Pp 7.1 ï 7. 60

[13] Wilkinson B. and Makki R.,1992, ñDigital System Design.ò Prentice Hall, International (UK) Ltd. Pp 100-120

Developing an Ultra-Low Power Remote Infrastructure Monitoring System

Ross Liederbach[1], Ashley Little[1], Flora Xiao[2], Cornel Zlibut[3], Guoqing Zhou[1], Chris Farnell[1], Brett Sparkman[1], Jingxian Wu[1], and Scott C. Smith[1]

Department of Electrical Engineering, University of Arkansas, Fayetteville, AR[1]
Department of Material Science and Engineering, University of Illinois, Urbana-Champaign, IL[2]
Department of Electrical Engineering, Tennessee State University, Nashville, TN[3]
rliederb@uark.edu, arlittle@uark.edu, ffxiao2@illinois.edu, czlibut@tn.state.edu, gzhou@uark.edu, cfarnell@uark.edu, bsparkma@uark.edu, wuj@uark.edu, and smithsco@uark.edu

***Abstract*— This paper develops an infrastructure monitoring system to wirelessly monitor vibrations that may cause impairment to structures. A wireless sensor network comprised of a fusion center and four sensor nodes allows for data to be transmitted from the sensors to the Fusion Center (FC), stored on an SD card, and later uploaded to a server via Global System for Mobile Communication/ General Packet Radio Service (GSM/GPRS). Another option is to read live data straight to the computer from the FC using a LabVIEW graphical user interface. The FC is controlled by SMS from the user's cellular phone, with the option of manual override.**

Keywords- Wireless Sensor Network, Arduino, Piezoelectric Sensor, XBee Radio, Vibration Monitoring

I. INTRODUCTION

Present day infrastructure such as bridges, buildings, roads and tunnels can use wireless sensor networks (WSNs) to test for structural fatigue caused by high vibrations, fire, flood, ice, etc. It is crucial to monitor these infrastructures for pending dangers. Patterns of these conditions can be monitored by using sensor nodes that transmit information to a fusion center (FC). This allows engineers to analyze structural damage and prevent future destruction. These WSNs will be needed for civilian and military use for extended time periods, requiring the system to use ultra-low power consumption, and to harvest energy from the surrounding environment. Several challenges were presented including a low signal-to-noise ratio and low maintenance. Surrounding environment conditions like noise and interference can lower efficiency of low power signals. The WSN was designed to withstand all forms of environmental conditions and harvest energy from the environment so that the sensors need little to no maintenance. Access to the data collected by the WSN was also included in the design. The power of the XBee has a limited distance transmission as it only allows up to a mile for transmitting data. XBee has mesh-network capabilities used for extending the range of device transmissions. This allows for data harvesting by storing data at a FC and transmitting data from the FC to the computer via GSM.

II. PREVIOUS WORK

Many studies have been conducted on vibrational testing with wireless sensor nodes for infrastructure health monitoring. Factors such as cost and power vary widely for every project. This project reduces cost and uses low power while harvesting energy from the environment. It is important to point out that this is on a much smaller scale than the previous works mentioned below.

The Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks [8] used 64 nodes to monitor the Golden Gate Bridge in San Francisco, CA. Each sensor node cost approximately $600 [8] and was also tied to a high cost of installation and maintenance. However, they each achieved a sampling rate of 1kHz and accuracy of 30 muG. The WSN used mesh networks due to its scale and a laptop was used as the base station. The information was then sent long range through the internet. Towards Wireless Sensor Networks for Railway Infrastructure Monitoring [9] used its own WSN, SENSORAIL, to monitor structural health and environmental security of a railway system. The WSN involved some overhead for the local wireless infrastructures in the form of ZigBee communication protocol. The information was then sent through a Wide Area Network (WAN) such as GSM/GPRS, UMTS/EDGE, and Fiber Optics geographic networks.

III. OVERVIEW OF COMPONENTS

After complete construction of the project, the node costs is approximately $90 and the FC with included components mentioned below equates to approximately $300.

A. Arduino Uno

The Italian-made Arduino Uno [6] is an open-source single-board microcontroller. A standard boot loader and programming language compiler run directly on the board. It is programmed using a language similar to C++ [5]. Fourteen input/output connector pins are exposed as female 0.1 inch headers, allowing the Arduino to interface with accessories, called shields. Some shields may be able to communicate directly over pins while others use the I^2C serial bus. The

Arduino Uno uses an Atmel AVR 8-bit ATMega328P processor [1]. For USB to Serial conversion the Arduino uses an ATMega16U2 chip. The Arduino Uno functions on a 5V operating voltage, with a 7-12V recommended input voltage. Also, the Arduino includes a 16MHz crystal oscillator to provide a stable clock signal.

Along with its processing power, the low power consumption and shield compatibility makes the Arduino Uno an optimal choice for the fusion center. With several shields available for the Arduino, and with a basic programming language, the Arduino Uno is extremely versatile and user-friendly.

B. Series 1 XBee

The Digi International Series 1 XBee [3] uses the IEEE 802.15.4 protocol. XBees allow for easy communication between microcontrollers, computers, or anything with a serial port. These modules have 8 digital I/O pins (with 6 being capable of analog) and a built-in antenna. XBees are programmed using AT commands. They are compatible with X-CTU software, which is used for testing and configuration of XBee devices. The Series 1 XBee is ultra-low power and has long distance reliability allowing it to transmit up to 100' indoor and 300' in outdoor settings. The Radio Frequency (RF) Data rate is up to 250,000 bits per second (bps). The Serial Interface Data rate travels between 1200 bps and 250 kbps. Power requirements are between 2.8 and 3.4 V for supply voltage. The Series 1 XBee modules have point-to-point or point-to-multipoint capabilities.

C. SM5100B GPRS Shield

The RF part of the GPRS shield converts into baseband for receiver chains and translates base band signals into the RF spectrum. This module has 60 connector pins and is connected to the SM5100B evaluation board. The GSM module connects directly to a Quad-band Cellular Duck Antenna. The operational voltage is between 3.3V and 4.2V with a recommended 3.6V power supply. The available frequency bands are EGSM900, GSM850, DCS1800, and PCS1900. In America and for this project the GSM850 was used. It operates using AT Commands for the transmission control protocol (TCP) [4].

D. LabVIEW Graphical User Interface

LabVIEW is system-design software allowing scientists and engineers to program tools on a GUI for measurement and control of systems. LabVIEW allows the user to program graphically in a language called "G". This programming method allows the user to wire graphical icons together while directly compiling to the computer, greatly simplifying the debugging process. G Programming is much more intuitive because it allows scientists and engineers to think and problem solve visually. One benefit of the built-in compiler is the broken run arrow in the toolbar that does not allow the user to run the program if there is an error. LabVIEW is able to connect to the serial port on the Arduino Uno to read in live data. Also, it has the capability to read or write to spreadsheets. Because of the intuitive programming language and applicable measurement tools provided, LabVIEW offered a very straightforward approach to designing a graphical user interface (GUI) system.

E. Li-Po Rider

The Li-Po Rider [7] is a power module that has a Lithium-Polymer (Li-Po) battery charger and a boost circuit with a standard USB output connection. It connects to the Photovoltaic Module (i.e., solar panel) in order to charge the Li-Po battery. The solar panel uses 5.5V at peak power, which is 0.935W. The Li-Po Rider can also charge the battery via the USB port or with a wall charger. The Li-Po Rider can automatically adjust the current to account for the varying charge of the solar panel. This module has automatic recharging capabilities as well as a user output power switch. The batteries used in this project are 3.7V, 3000mA/Hr batteries. This means that they should supply 3000mA of current for one hour. Paired with the Photovoltaic Module, the Li-Po Rider provides an ultra-low power solution for the WSN.

IV. SYSTEM DESIGN

A. Nodal Modules

Nodes are enclosed in 4x2x1" dimensional boxes. As shown in Figure 1, they are packaged to hold an XBee wired to the ATMega328P, the 3000mAh battery and the Li-Po Rider. This gives the nodes their own sources of power, and the ability to harvest energy from the environment. The sensing pin, ground, and voltage controlled by the MOSFET, are all off-board for easy connection to piezoelectric sensors or sensors that require voltage. The MOSFET/microcontroller combination allows for less power consumption by the sensor devices and the XBee. All eight I/O pins of the XBee may be made external to the node for access. An external switch allows the user to manually turn the sensors on and off to conserve power. The ATMega328P analyzes the commands from the FC, which in turn may shut off current entirely to the XBee radio, or start transmissions, thus conserving power. This switching of power to the XBee and sensors is controlled by the MOSFET in conjunction with the microcontroller. Currently, during the sleep cycle, the node draws ~13 mA, and while continuously transmitting, the node draws ~74 mA.

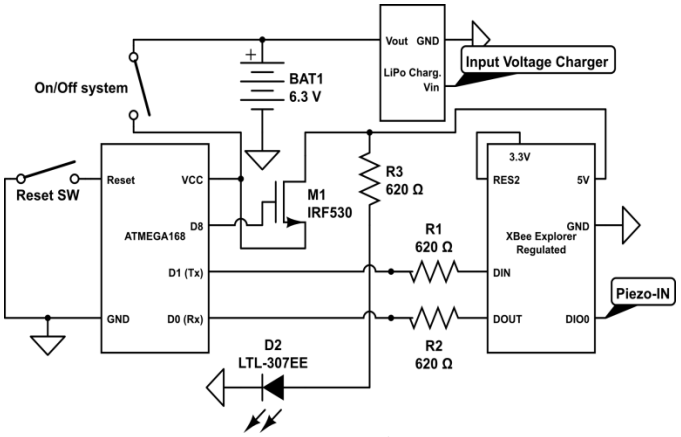


Figure 1: Wiring of Nodes

1) Li-Po Rider

The Li-Po Rider [7] is extremely useful in its ability to utilize external energy, and store that energy in a battery. The Li-Po Rider also allows for switching 5V on and off from the battery to power another system. With the Li-Po Rider board, managing power at each node is simplified, with each node having its own solar panel and 3000mAh battery.

2) ATmega328P

For controlling the XBee radio, with ultra-low power consumption in mind, pre-programmed sleep modes on the XBee did not work. These pre-programmed modes on the XBee radio created problems with the Fusion Center interface; therefore, a microprocessor, the ATMega328P, allows for more specialized and specific communication between the node and the FC. The ATMega328P is first boot-loaded with the Arduino boot loader, which is shown in Figure 2.

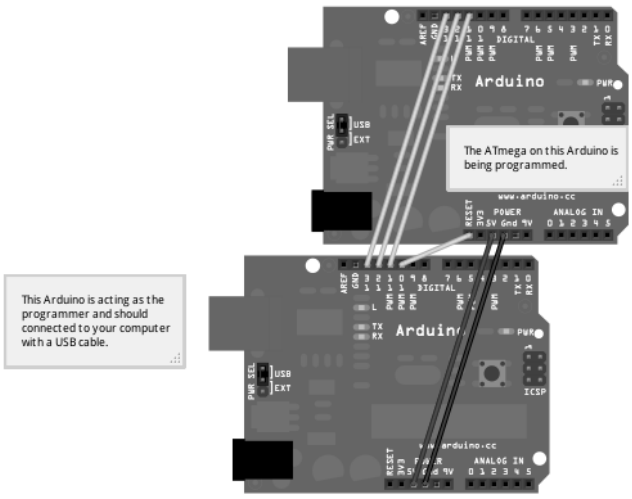


Figure 2. Bootloading ATMega328P

After being bootloaded, this chip may be programmed using the Arduino IDE. Programming the chip involves having a digital pin written high and low based on commands sent from the FC. This digital pin is connected to a PNP mosfet, which in turn powers the XBee on and off.

3) XBee

Original programming of the XBee sets the baud rate, and Personal Area Network ID (PAN ID) to the same as the XBee at the fusion center, sets the destination address to the coordinator XBee's address, and sets the sampling rate and "samples before TX" to A and 0, respectively (A is 10 in hex and represents a 10ms or a 100Hz sample rate). Before plugging the XBee into the system, the I/O pins must be configured as needed to digital, or ADC. These values are stored in the XBee and are not reset when powered off.

For the system of XBee and ATMega328P to work properly, the XBee must turn on periodically to check for a command from the FC. A sleep cycle consisting of "Wake" and "Sleep" stages is implemented to accomplish this. If a command is sent from the FC during the "Wake" stage of the sleep cycle, the ATMega328P allows the XBee to stay on. Once the XBee is on, other commands may be sent, such as a command to start transmitting. In the case that the "Transmit" command is sent, the ATMega328P will enter the XBee command mode and reprogram the "samples before TX" value to "A," which again means 10, and therefore begins transmitting after acquiring 10 samples. This method allows for longer "Sleep" stages, however the longer the sleep stage, the longer it will take for all nodes to wake up. Minimizing the "Wake" stage of the sleep cycle would lower overall power consumption, but this again could affect the time it takes to wake up since the node might not receive the wake command during its "Wake" stage. All XBee AT Commands can be altered as shown in Table I [3].

TABLE I. XBEE AT COMMANDS

Command	Description	Value
ATID	PAN ID	3332
ATDL	Destination Low Address	40492D6A
ATBD	Interface Data Rate	7- 115200bps
ATIR	Sampling Rate	0
ATIT	Samples before TX	A
ATD0-ATD7	I/0 Settings	1-NA 2-ADC 3-DI 4-DO LOW 5-DO HIGH

4) Abilities/Sensor capabilities

The ATMega328P microprocessor chip is the same as in the Arduino Uno, and has a lot of processing power that is unused in this application. Alternate and additional commands may be analyzed and carried out by the ATMega328P, but in this project, only sleep, wake, and transmit commands are used.

Any sensors that are powered by 5V or less, and output between 0 and 3.3V, are capable of being used with the sensor nodes. External circuitry is helpful in scaling voltages of sensors down to what is readable by the XBee radio.

Using Series 1 XBee radios bring the advantage of higher sampling rates, up to 1KHz, and simpler setup than Series 2 radios, which are only capable of sampling rates up to 20Hz. The Series 1 radio may store up to 46 analog samples before transmission in its buffer, which allows for expansion in the number of sensors the node is capable of supporting, and allows for lower transmission rates.

B. Fusion Center

The fusion center is comprised of several elements including: SD Wireless Shield, S1 XBee Radio, SD card, two Arduinos, GPRS Shield, 12V power supply with a voltage regulator, and a multiplexer. All the components are enclosed in an 8x6x4" box, as shown in Figure 3. The box has two

switches controlling system power and a live or SD data collection and a mini USB plug for live data collection.

The FC is the center piece that allows the whole system to cooperate together. The FC system can also be broken into two distinct systems sustained by individual Arduinos. The first Arduino is dealing with XBee and SD communication, while the second Arduino deals with the GPRS communication. Together they allow for a larger variety of integrations and improvements.

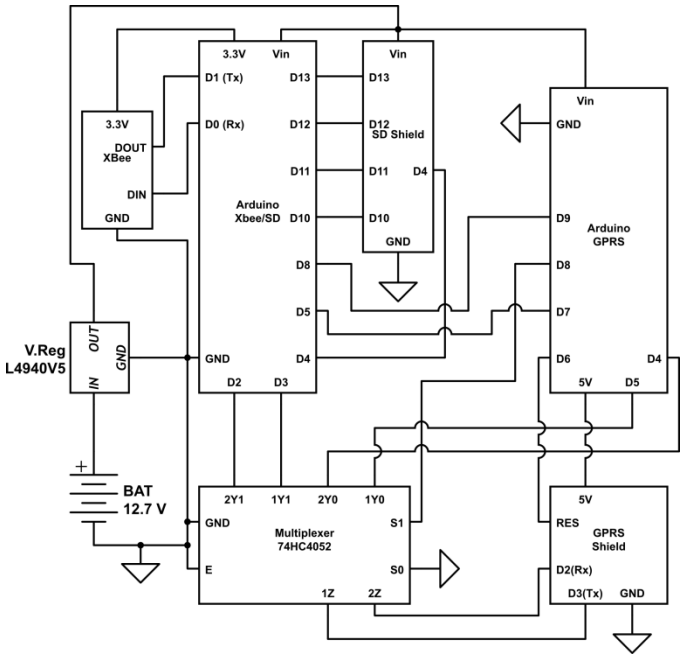


Figure 3: Wiring of Fusion Center

1) System One (Arduino/XBee/SD)

The Arduino Wireless Shield allows the Arduino board to communicate through the XBee with all the sensor nodes. In this system the microcontroller is programmed to deal with the XBee and SD card for collecting the data from all the sensor nodes and saving it to the SD card in separate files depending on the sensor node. If the live data collection switch is "On" then all the data will be output to a new serial port for viewing it in LabVIEW.

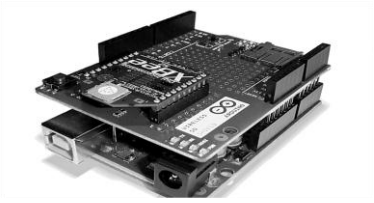


Figure 4: Arduino with the XBee/SD Shield

a) XBee

The XBee at the FC is working as the coordinator for the entire wireless network on the PAN. The XBee is programmed in X-CTU, which includes the networking & security, RF interfacing, serial interfacing, I/O settings, diagnostics, and AT command options. One important difference in the coordinator XBee is that its destination address is required to be set to a broadcast address over the PAN so that each node receives the commands sent from the FC. Commands like wake, sleep, or transmit are all broadcast. The RF interfacing should set the "select/read transmitter output power" value to its highest level for efficient communication between the XBees in different environments. Furthermore, the clear channel assessment threshold is set to 44dBm so that if the modem detects energy above the threshold it does not transmit, which limits the loss of data samples. The serial interfacing settings were modified to 115200 interface data rate (BD) [3]. This option allows adding more sensor nodes to the wireless network without lag or losses in the data received.

b) SD

The SD is formatted with FAT32 file system with one partition on it since Arduino can only support this system. The communication between the microcontroller and the SD card uses SPI, which takes place on digital pins 11, 12, and 13. Additionally, another two pins must be used to select the SD card for input and output of data, which in this case were digital pins 4 and 10. The SD is used to hold the received data until it is being transmitted to the remote server through the GPRS; afterwards the data is removed to prevent duplication or retransmission of the same data.

2) System Two (Arduino/GPRS)

The Arduino GPRS/GSM Shield is a physical add-on to Arduino that allows send/receive SMS and voice calls, but also establishes TCP communication [4] over the broadly spread GPRS network. In this project the GPRS/GSM shield is used for sending and receiving SMS commands and confirmations while connecting to a socket on a remote server to send data. In this system the microcontroller is programmed [4] to communicate with the GPRS/GSM in order to establish socket and SMS connection. It is fully commanded using AT commands [2] over the serial port connection from the microcontroller. The GPRS uses a SIM card from T-mobile which monthly or annually can be recharged with the data plan desired by the user.

The microcontroller on this system also controls a dual 4-channel analog multiplexer/demultiplexer with common select logic for the ability to change between serial ports on the Arduino, and three digital pins for communication between the two Arduino systems.

Figure 5: SM5100B GPRS Shield/Arduino

3) FC Mechanism

The two Arduino systems at the FC should cooperate together in order for the entire project to work properly. When the entire system is turned on, one of the microcontrollers is initializing the SD and XBee while the other microcontroller is setting up the SMS and server socket connection based on the server specified. Once the initialization and socket connection is ready, a SMS confirmation is sent to the specified user. Now the user is able to text a command, as shown in Table II. If the GPRS receives the #ON command, it sends back a confirmation to the user and lets the XBee/SD Arduino know through a digital pin to broadcast the wake command. After all the nodes respond with an acknowledgment of "On", the coordinator gives an order for transmission. Now the microcontroller checks to see if the live data is switched "On". If the live data switch is not "On" then it automatically writes the data to the SD card. If the live data switch is "On" it then can be viewed in the LabVIEW interface. Once the user sends the #OFF command, the GPRS again sends confirmation back to the user and then lets the XBee/SD Arduino know to end the transmission by sending the sleep command. Once the nodes are in sleep mode and SD has all the data, the GPRS lets the user know that data is ready to be transmitted to the computer/TCP server. Therefore, the XBee/SD Arduino takes over the TX/RX pins of the GPRS through a high-speed Si-gate CMOS device for ease of uploading the data from the SD card to the serial port of the GPRS. This data is continually being sent to the computer/TCP server until no more data is left. Once the data is finished, the TX/RX pins of the GPRS are switched back to the previous Arduino and confirmation of the completed data is sent back to the user. The SD files can now be deleted to make room for new files or they can be renamed to be kept for future analysis.

TABLE II. SMS COMMANDS

SMS Command	Command Specification
#ON	Turn Sensor nodes ON
#OFF	Turn Sensor nodes OFF
#OR	Reset GPRS

4) FC Optional Improvements & Capabilities

The FC can also be improved in many ways based on preferences of the user. For example, if different types of sensors are needed for monitoring the infrastructure, they can be integrated into the FC. Sensors like humidity devices and barometers were tested on the FC.

C. Node/FC Interface

Figure 6 shows the basic schematic of the communication between cell phone, computer, fusion center, and nodes. In the fusion center, two Arduino Uno boards are housed, each with their own shield and separate tasks. The Arduino with the GSM/GPRS shield handles all incoming commands from the cell phone and sends back confirmations via SMS. This shield also handles uploading data to a TCP server where it can be accessed through a computer. Because this Arduino receives commands, it passes the commands along to the Arduino coupled with the SD Wireless shield. This second Arduino directly communicates with the sensor nodes, giving the nodes commands to turn on and transmit data, or enter the sleep cycle. Once the command is sent to wake the nodes, the fusion center waits for a confirmation from each node to send the command to begin transmission. After the command to transmit is sent, the processor at each node enters into the XBee command mode and reprograms the "Samples Before TX" value to "A", or 10 samples before transmitting. The transmit command shall only be sent once from the fusion center, and broadcast to all the nodes to prevent interference while the processor is entering command mode. Finally, when the transmission is stopped and the nodes enter the sleep cycle, the Arduino with the GSM/GPRS shield begins forming packets to send over the TCP server, and the Arduino with the SD shield writes the data, packet by packet, to the GSM/GPRS shield. Once all the data is sent, the files on the SD card are deleted, but are accessible through the TCP Server.

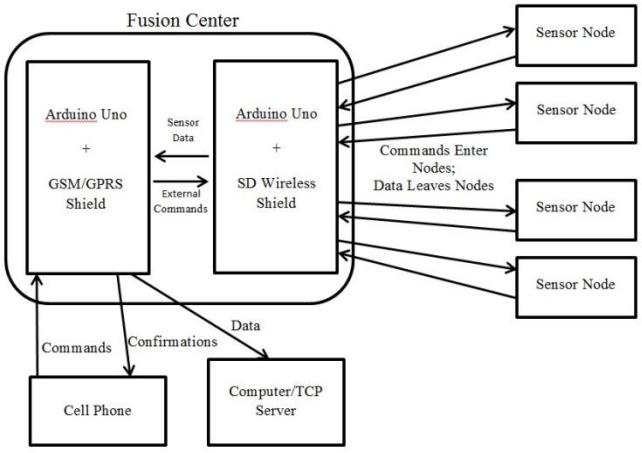


Figure 6. Basic Communication Schematic

D. Server

TCP File is an online open-source program that allows for a socket connection and listens for incoming data. It then takes this data and stores it into a .txt file.

The data is saved to a TXT file, which is then sorted by a MATLAB program. The MATLAB program reads in the file as a string, identifies all occurrences of sensor filenames, and sorts each sensor's data into its own column of a master matrix. The sorted matrix is given a size at the beginning of the program to minimize runtime, and after all the data is sorted into it, all rows of all zeroes are deleted, and the master matrix is written into a .csv file. The LabVIEW GUI reads in the .csv file and displays it in a comprehensible format.

E. LabVIEW Interface

For this project, two graphical user interfaces were used. Since there were two different methods for inputting data, two different interfaces were needed. A GUI was designed for reading in data from a data file, such as from an SD card. The

other interface focused on reading live data from the COM port that the FC connected to.

1) Reading Data Files

Figure 7 shows the front panel of the GUI for reading in data directly from a .csv data file. Detailed below in Table III are the functions for each designated part on Figure 7.

Before the user presses the run button, they must choose a file to read from by clicking the folder represented as number 4. This will pull up a dialog box allowing the user to select what file they would like to import. If the user presses the run button without choosing a file, it will pop up with an error and allow them to try again. Once the interface is running, the sensors will show the last data point collected on the .csv file, shown at number 5 in Figure 7.

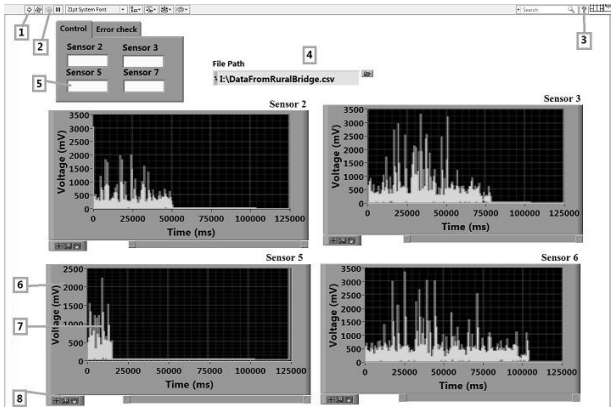


Figure 7. GUI Front Panel

On the waveform chart, number 6 shows the Y-axis is auto-scaled in order to display all the voltage points. The time-voltage chart allows the user to scroll backward and see previous plot points. Number 8 offers the user the option to zoom in on the data values as shown in Figure 8. When the user is finished using the program, they should press the stop button located at point 2 in Figure 7.

TABLE III. GRAPHICAL USER INTERFACE FUNCTIONS

1	Run
2	Stop
3	Help
4	Dialog Box
5	Sensor Data Feed
6	Autoscaled Y-axis
7	Data points
8	Zoom function

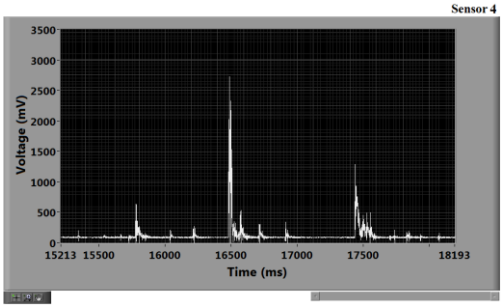


Figure 8. Zoom Feature on GUI

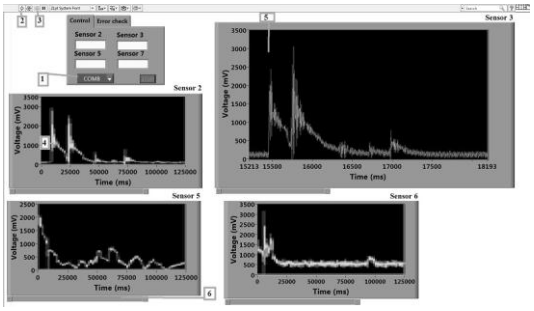


Figure 9. Live GUI Front Panel

TABLE IV. GRAPHICAL USER INTERFACE FUNCTIONS

1	COM Port
2	Run
3	Stop
4	Chart
5	Expansion Function
6	Scrollbar

2) Reading from COMports

In this version of the GUI, shown in Figure 9, the user can see live data from the COM port. The FC will be connected to a USB port in the computer. The user will select the COM port, as shown as number 1 in Figure 9. Once the COM port is selected, the user can then run the program and see the plotted data as it comes in. The chart will scroll with the new points as well as auto-scale the Y-axis to ease the analysis process. The GUI also is programmed to expand into a larger chart to the right of the smaller charts to make it easier to see the graphed points, which can be seen at number 5 in Figure 9.

V. RESULTS AND CONCLUSION

Testing was done in a three part phase. First, one wireless sensor node was built and tested against a wired sensor. Then, more nodes were built and tested as a final system without the GPRS. Finally, the GPRS was connected to the final system.

1) Comparison to Wired Sensor Network

Once the first sensor node was built, an accelerometer was attached to the wireless node so that the waveform could be compared to that of a wired sensor. It was tested on a cantilever beam in a lab at the Engineering Research Center (ENRC) located in Fayetteville, AR. Results above 600mG were successful. When the signal dropped below this level, much noise was picked up resulting in faulty data. After more

testing, it was determined that either the voltage-scaling circuit for the signal caused the noise or the accelerometer was defective at low voltages.

2) Field Test

In order to test the final system without GPRS, a field test was completed at a local bridge in Fayetteville, AR. The sensors were placed underneath the bridge at its joints, which seemed the most optimal location for reading vibrations from the moving vehicles and other outside factors. To begin with, data was collected live to make sure the sensors were working properly. Then data was collected and stored in the SD card for an hour with four piezoelectric sensors connected to the four nodes. The data successfully transmitted to the FC on the ground approximately 15 meters from the sensors. All components worked as expected, and the data stored on the SD card was analyzed later at the lab. One issue is that the .txt file showed different amounts of data for each sensor node in the one hour period. This may be caused by outside interference that corrupts some of the transmissions from the XBees to the FC.

3) GPRS/GSM

Once the GPRS was connected to the FC, all commands were transmitted through SMS communication. After the data was stored to the SD card, it was then transferred through the TCP/IP server using the GPRS. The server waited for the data to store into a single .txt file, which later was converted using MATLAB into a .csv file for LabVIEW. This effectively transmitted the data for further analysis.

4) Future Work

There are several things that can be improved upon in this system. For example, a variety of sensors can be added to the system such as a water sensor to test for flooding, a barometer to test for air pressure, temperature, and altitude, or a humidity sensor. Also, more nodes can be added to the system; however, the program on the FC does not account for additional nodes. Thus, the program could be altered for easier expandability of the network, for more robustness, and for higher efficiency.

REFERENCES

[1] "8-bit AVR® Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash", ATMEL® Corporation, 2009, [Online] Available: http://www.atmel.com/Images/doc8161.pdf

[2] "SM5100B-D AT Command," Shanghai Sendtrue Technologies Co., Ltd., 2008, [Online] Available: http://www.sparkfun.com/datasheets/CellularShield/SM5100B%20AT%20Command%20Set.pdf

[3] *XBee/XBee-Pro OEM RF Modules*,Maxstream, Inc., Lindon, UT, 2007.

[4] *Using AT commands to control TCP/IP stack on SM5100B-D module*, Shaghai Sendtrue Technologies Co., Ltd., 2008, [Online] Available: http://www.sparkfun.com/datasheets/CellularShield/SM5100B%20TCPIP%20App%20Note.pdf

[5] Margolis, Michael, *Arduino Cookbook*, Ed. 1, O'REILLY® Media, Inc., March 2011, pp. 81-213.

[6] "Arduino Uno", ©Arduino., 2006, [Online] Available: http://arduino.cc/en/Main/ArduinoBoardUno

[7] "Li-Po Rider V0.9b," Seeed Studio, November 2011, [Online] Available: http://www.seeedstudio.com/wiki/Lipo_Rider_V0.9b

[8] Sukun Kim, et al., "Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks," Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on , vol., no., pp.254-263, 25-27 April 2007. [Online] Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4379685&isnumber=4379652

[9] Flammini, F., et al., "Towards Wireless Sensor Networks for railway infrastructure monitoring," *Electrical Systems for Aircraft, Railway and Ship Propulsion (ESARS), 2010* , vol., no., pp.1-6, 19-21 Oct. 2010 [Online] Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5665249

A Knapsack Scheduling Algorithm for Soft Real-time Multiprocessor System

Shaohang Cui[1], Douglas Buchanan[2], and Ken Ferens[2]
[1]ERLPhase Power Technologies Ltd.
[2]Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB, Canada
scui@erlphase.com, Douglas.Buchanan@ad.umanitoba.ca, Ken.Ferens@ad.umanitoba.ca

Abstract—Due to the development of commercial multicore architectures, scheduling in Multiprocessor real-time system has attracted huge research interests and various algorithms have been proposed. However, most prior research on real-time scheduling on multiprocessors has focused only on hard real-time systems where no deadline may ever be missed. Due to the increasing need of soft real-time multiprocessor systems where deadline violation is tolerable, scheduling in overloaded soft real-time systems should be studied as well. In this paper, we consider the time/utility functions (TUFs) express the utility of completing a task as a function of that task's completion time. Our objective is to maximize the overall system utility with given computation capability constraints and TUFs of tasks. We first formulate the scheduling problem to an allocation optimization problem. Then, for practical implementation, we propose a knapsack method to achieve the suboptimal solution. Simulation results demonstrate that the proposed knapsack method could achieve about 90% of the optimized performance.

Keywords - Soft-real time scheduling; knapsack greedy scheduling.

I. INTRODUCTION

Real-time multiprocessor systems are now ubiquitous. The need for multiprocessor real-time analysis has been well realized in the past 10 years due to the development of commercial multicore architectures and the research conducted on the design of effective scheduling algorithms. The difficulty with achieving effective scheduling algorithms in multiprocessor systems is that the algorithm must optimally allocate processors to tasks, and then optimally schedule the tasks in each processor, which is a doubly complex endevour [1].

Research on real-time scheduling has largely focused on recurring tasks, i.e., for each task, a potentially infinite sequence of events (or jobs) arrive gradually [2], [3], [4]. These algorithms and systems under consideration could be classified to different categories for various perspectives. From the perspective of processors, multiprocessor systems can be classified into [5]: (1) Heterogeneous where the processors are different; (2) Homogeneous where the processors are identical (3) Uniform where processors operate at different integral multiple of a unit speed. From the perspective of the degree of run-time migration, algorithms could be classified to [6]: (1) no migration, where tasks are partitioned and allocated to different processors before their running (offline), and uniprocessor scheduling algorithm is applied to each processor; (2) full migration, where jobs are allowed to arbitrarily migrate across processors during their execution; and (3) restricted migration, where some form of migration is allowed, e.g., in a job's scale.

However, most prior research on real-time scheduling on multiprocessors has focused only on hard real-time systems where no deadline may ever be missed, which also indicated that the systems under analysis must be under-loaded for a feasible algorithm design. When resource overloads occur, meeting deadlines of all activities is impossible as the demand exceeds the supply. For a soft real-time, violation of the deadline constraints results in degraded quality, but the system can tolerate such violation and continue to operate [7]. Due to the increasing need of soft real-time multiprocessor systems with large computation requirements, such as tracking, signal-processing, and multimedia systems, scheduling in overloaded soft real-time systems should be studied as well.

In a soft real-time system, deadlines by themselves cannot express both urgency and importance, e.g., the most urgent activity can be the least important, or vice versa [8]. In this case, in order to consider the importance and urgency of tasks jointly, we consider the abstraction of time/utility functions (or TUFs) as similar in [9] that express the utility of completing a job as a function of that job's completion time. Utility is a task-specific and time-related measurement of the value to the system. TUF quantifies the utility of completing each task at a given time. We consider a simple monotonically non-increasing functions as TUFs that, for a given task, once a job of this task arrives and is ready to process, the more time used to complete the job, the smaller its value to the system. Our objective is to maximize the

overall system utility with given computation capability constraints and TUFs of tasks. For simplicity, we consider a homogeneous multiprocessor system with full migration capability and worst case demand.

In [10][11], it has been proved that for a homogeneous multiprocessor system with full migration capability, as long as the utilization requirements of all tasks is less than the system computation capability, a scheduling policy could be achieved by the proposed LLREF algorithm. Our idea for the soft real-time system is that, based on the system computation capability, we are going to find an allocation of utilization of processors to each tasks to maximize the overall system utility. Once the allocation problem is solved, a scheduling policy could be achieved based on LLREF algorithm. In this way, we convert TUFs to utilization/utility functions (UUF), and formulate the allocation problem as an optimization problem. Due to practical reasons, we then propose a knapsack method to achieve a suboptimal solution for the allocation problem.

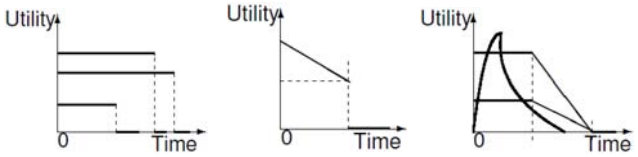
The rest of this paper is organized as follows. In Section 2, we propose the system model and formulate the scheduling problem to an allocation optimization problem. Then, for practical applications, we propose a knapsack method to achieve the suboptimal solution in Section 3. The performance evaluation of the concerned algorithms is presented in Section 4. And Section 5 concludes with a summary of the work presented in this paper.

II. SYSTEM MODEL

Similar to [10], we consider a homogeneous multiprocessor system of M identical processors. The application under consideration consist of a set of tasks, denoted by $T = \{T_1, T_2, ..., T_N\}$. Each task T_i has a number of jobs, and these jobs may be released either periodically or sporadically. All tasks are assumed to be independent, i.e., they do not share any resource or have any precedence. Full migration is allowed, i.e., tasks may be pre-empted at any time. As it is a soft real-time system, it is acceptable for a job of a task to be completed after the arrival of the next job. The constraint is that all tasks could be completed steadily, say, the overall process rate should be no smaller than the overall demands. Worst case demand is considered here, say, there is a given minimum time e_i required to complete a task i in the worst case.

A. TUFs and UUFs

On the one hand, a hard real-time system will result in total system failure if a hard deadline of any job is missed, and there can be no benefit in completing any job later than its hard deadline. A hard real-time system results in such failure when resource demand exceeds system supply, or, in other words, when resource-overload occurs. On the other hand, a soft real-time system may continue to perform tolerably if any job misses its deadline, albeit with degraded performance. In other words, completing an activity at any time in a soft real-time system, even after its deadline, will result in some utility (benefit) to the system, although that benefit (utility) depends on (is proportional to) the activity's completion time. Generally speaking, the amount of benefit of completing a task after its deadline decreases with increasing actual time of completion. Many systems quantify the amount of benefit of completing a task after its original deadline using the so called time-utility function [8], which allows the semantics of soft real-time constraints to be precisely specified. TUF quantifies the utility (benefit) of completing a job as a function of time, including before and after that job's deadline time. Some examples of TUFs are shown as in Fig. 1.



(a) (b) (c)

Fig. 1 Example TUF Time Constraints. (a) Step TUFs; (b) MITRE/TOG AWACS TUF [12]; (c) GD/CMU BM/C2 TUFs [13].

This paper defines optimal scheduling of tasks to processors by the maximization of the so called Utility Accrual (UA) function [8]. The UA function is the accrued benefit of the activities completing their tasks at certain times, given that the utility of the tasks are described using the time-utility (time-benefit) functions. For example, the accrual method may be a simple sum, so that an optimal schedule is obtained when the sum of each activity's attained benefit is maximized. For simplicity, this paper considers monotonically non-increasing time-utility functions, so that, for a given task, once a job of this task arrives and is ready to process, the more time used to complete the job, the smaller is its benefit to the system. Given the system's resource capability, this paper's objective is to maximize the overall system utility, given the task time-utility functions.

In [11], a scheduling algorithm called LRE-TL was proposed for homogeneous hard real-time multiprocessor system with periodic or sporadic arriving jobs, in which it was proved that as long as the overall utilization requirements of all tasks is less than the processing capability, a feasible scheduling policy could be achieved to meet all task deadlines. The utilization of a task is defined as $u_i = \frac{e_i}{\min\ (P_i, D_i)}$, where e_i is the longest execution time of a job of a task i, P_i is the smallest interval between any of task i's two successive jobs, and D_i is task i's deadline.

For the soft real-time multiprocessor system under consideration, if we could find a time interval d_i for each task i, which is the maximum allowed interval between task i's arrival and completeness, we could get a utilization: $u_i = \frac{e_i}{d_i}$. Based on LRE-TL algorithm, as long as $0 \le u_i \le 1$ and $\sum_{i \in N} u_i \le M$, where M is the number of processors, a feasible scheduling policy could be acquired. Thus, the

scheduling problem could be converted to an optimization problem: find d_i to maximize the overall utility given the constraints for u_i. Or TUF could be converted to utilization utility functions (UUFs) which describe the relationship between a task's utility and utilization. In this way, the optimization problem is an allocation problem: how to allocate utilization to different tasks so as to maximize the overall utility, given the overall available utilization (processing ability) and UUFs.

B. Optimization Problem

From the above analysis, we could formulate the scheduling problem in a soft real-time multiprocessor system as an optimization problem for allocation of utilization to tasks as follows:

$$max_{u_i} \sum_i U_i(u_i), \text{s.t.}, \sum_i u_i \le \text{M}, 0 \le u_i \le 1 \quad (1)$$

When $U_i(u_i)$ are convex functions on the constrained set, it is a convex programming problem and could be easily solved by calculating the KKT conditions, i.e., let:

$$\text{L(A, u)} = \sum_i U_i(u_i) + a_0(\sum_i u_i - M) - \sum_i a_i u_i + \sum_i a_{N+i}(u_i - 1), \text{where } A = [a_0, \dots, a_{2N}] \quad (2)$$

From KKT conditions, we have:

$$\frac{\text{dL}}{\text{du}_i} = 0 \quad (3)$$

$$a_0\left(\sum_i u_i - M\right) = 0 \quad (4)$$

$$\sum_i a_i u_i = 0 \quad (5)$$

$$\sum_i a_{N+i}(u_i - 1) = 0 \quad (6)$$

Note that to solve KKT conditions, U_i should be differentiable on the interval [0,1].

III. KNAPSACK GREEDY ALGORITHM

In practice, sometimes it may be not easy to acquire the analytic expression of the TUF and UUF, or it is too complex to solve the convex programming problem, or even U_i may be not be convex nor differentiable on [0,1]. In this case, we may use a Knapsack method to achieve the suboptimal solution.

Intuitively, to maximize the overall utility (benefit), higher utilization should be allocated to tasks which could achieve high utility with small utilization first. However, as the relationship between utilization and utility of a task is determined by its UUF, which is not a constant, we need to find a suitable point which is most profitable, say, how much utilization should be allocated to a task to achieve utility.

Based on this consideration, we divided the UUF to 9 zones as shown in Fig. 2.

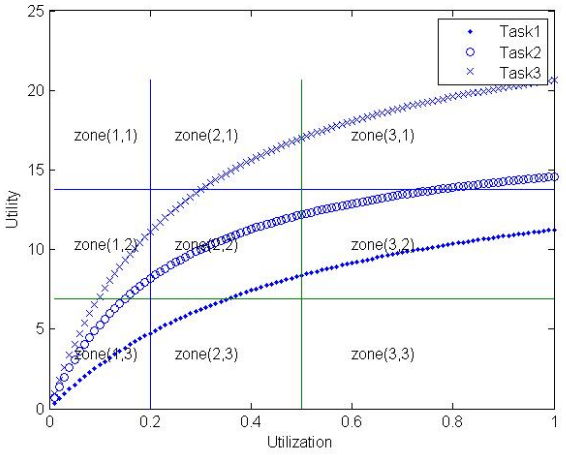


Fig. 2 UUFs and zones.

We select UUF $U_i = A_i - B_i/(u_i + B_i/A_i)$ where A_i and B_i are task specific constant as the example. u=0.2,0.5 and $U = \frac{1}{3}Y, \frac{2}{3}Y$. The Y is the maximum achievable utility of all tasks, given the maximum utilization (utilization = 1). In Fig. 2, Y ≈ 20.64 is a random number.

Each zone is abstracted as represented by its centroid, which is related with a required utilization and an estimated utility, as shown in Table 1.

Table 1 Required utilization and estimated utility for zones.

Zone(i,j)	**i=1**	**2**	**2**
j=1	(0.1, 5/6 Y)	(0.35, 5/6 Y)	(0.75, 5/6 Y)
2	(0.1, 1/2 Y)	(0.35, 1/2 Y)	(0.75, 1/2 Y)
3	(0.1, 1/6 Y)	(0.35, 1/6 Y)	(0.75, 1/6 Y)

For a task passing a zone, by estimation, within this zone, this task could achieve the utility with a utilization determined by the centroid of this zone. Based on its UUF, a task will pass several zones. For a task passing zones in the figure above, from left to right, more utilization will be allocated to a task. From low to high, more utility could be achieved by a task. A task passing several zones means it could be processed with several policies, i.e., requested utilization and achievable utility. In this way, these tasks' UUF have been fuzzed as each task many have several allocation policies. Now, the problem is, given all the tasks and related policies, how to maximize the overall utility.

This problem could be converted to a group Knapsack problem. However, given the requested utilization in each zone (the weight in Knapsack problem) is decimal, it is not efficient to solve it by using Knapsack methods. A greedy method is then introduced, which consists of three steps:

1. For a zone, its profit rate is the estimated utility divided by the required utilization. A task will select a zone with the highest profit rate from all the zones it passes, and use the corresponding allocation policy. The task with the highest profit rate will be allocated with its required utilization first. In this way, Task2 and Task3 will be allocated based on the policy for zone (1,2), and Task1 will be allocated based on the policy for zone (2,2).

2. After all tasks have been allocated with their required utilization to achieve the most profitable utility, if there is still abundant resource, tasks which could increases its utility markedly, should be provided with supplementary utilization, say, which could go from a lower zone to a higher zone by increasing utilization.

3. After tasks cannot achieve marked utility improvement by increasing utilization, if there is still abundant resource, such resource will be provided to all tasks equally, given each task could get utilization 1 mostly.

In this way, based on Table 3, if there is abundant resources remaining, task3 will be allocated first by increasing its utilization to 0.35, and task2 will then be allocated by increasing its utilization to 0.75. Note that based on step 1, if a task is already in zone(i',j), we do not need to take the number of marked changes happen in $i \le i'$. If a task passes several zones for the same i, only a higher zone will be kept.

Table 2 Task related zones zone(i,j).

Task1	Zone (1,3)	Zone (2,2)	Zone (3,2)
Task2	Zone (1,2)	Zone (2,2)	Zone (3,1)
Task3	Zone (1,2)	Zone (2,1)	Zone (3,1)

Table 3 Marked increase as utilization increases.

Task1	Zone (1,3)	Zone (2,2)	Zone (3,2)
Task2	Zone (1,2)	Zone (2,2)	Zone (3,1)
Task3	Zone (1,2)	Zone (2,1)	Zone (3,1)

IV. PERFORMANCE

Since the optimization problem could only be solved by KKT conditions when $U_i(u)$ are differentiable and convex functions on the constrained set $u \in [0,1]$. To compare the performance of the proposed knapsack algorithm to the optimal solution, we select the UUF as $U_i = A_i - B_i/(u_i + B_i/A_i)$, where A_i is a random number uniformly distributed on [5,20], and B_i is a random number generated by normal distribution norm (5,5).

For N=10, increase M for 1 to 10, we acquired the performance of knapsack algorithm and optimal solution as shown in Fig. 3. Note that performance of both algorithms has been normalized by the system demands. From this figure, we could see that the proposed knapsack algorithm could achieve about 90% performance of the optimal solution even when M is small. As M increases and the system becomes less overloaded, the knapsack algorithm could achieve more utility until satisfy all system demands.

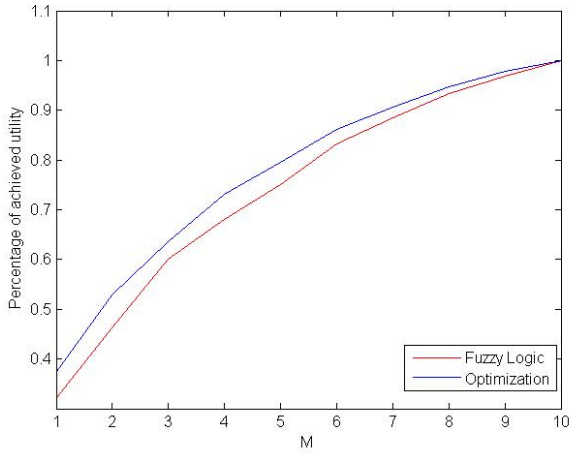
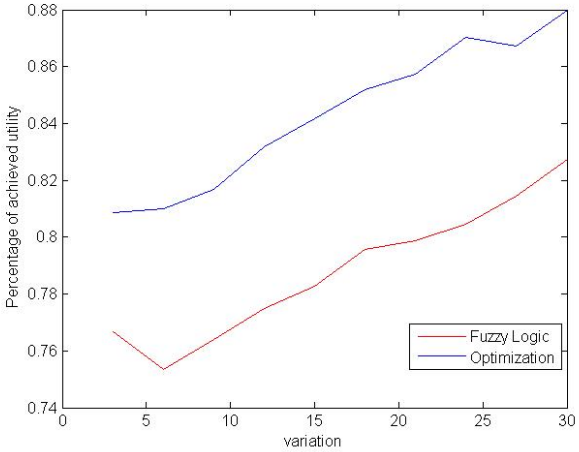
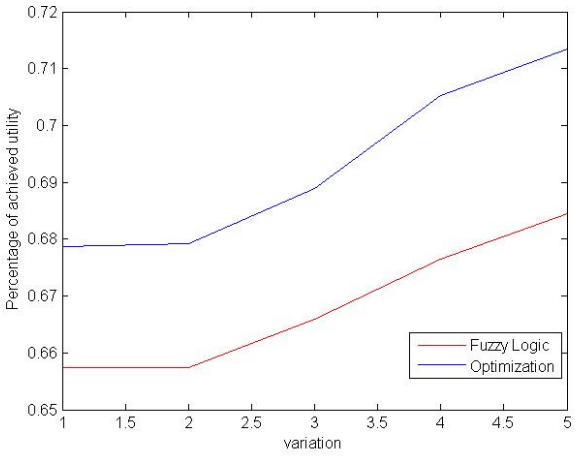


Fig. 3 Performance vs. processors (M).



(a)



(b)

Fig. 4 Performance vs. variation. (a) variation of A_i; (b) variation of B_i.

Select M=3. Set A_i a random number generated by abs(norm(30,v)), and B_i is a random number generated by normal distribution norm(5,5). Increasing v from 3 to 30 we got a performance result as shown in Fig. 4(a). We could see that in general, as the variation, or the difference of tasks increases, performance of both knapsack algorithm and optimization algorithm will increase.

V. CONCLUSIONS

When resource overloads occur, meeting deadlines of all activities is impossible as the demand exceeds the supply, while it is still acceptable in a soft real-time system. In this paper, we consider the time/utility functions (TUFs) express the utility of completing a task as a function of that task's completion time. Our objective is to maximize the overall system utility with given computation capability constraints and TUFs of tasks. We first formulate the scheduling problem to an allocation optimization problem. Then, for practical implementation, we propose a knapsack method to achieve the suboptimal solution. Simulation results demonstrate that the proposed knapsack method could achieve about 90% of the optimized performance.

REFERENCES

[1] O. Zapata and P. Mejia-Alvarez. Analysis of real-time multiprocessors scheduling algorithms. In Proceedings of the Real-Time Systems Symposium (RTSS), 2003.

[2] Baker, T.P., "An analysis of EDF schedulability on a multiprocessor," Parallel and Distributed Systems, IEEE Transactions on , vol.16, no.8, pp.760-768, Aug. 2005.

[3] Anderson, J.H.; Srinivasan, A., "Early-release fair scheduling," Real-Time Systems, 2000. Euromicro RTS 2000. 12th Euromicro Conference on , vol., no., pp.35-43, 2000.

[4] Jinkyu Lee; Easwaran, A.; Insik Shin, "LLF Schedulability Analysis on Multiprocessor Platforms," Real-Time Systems Symposium (RTSS), 2010 IEEE 31st , vol., no., pp.25-36, Nov. 30 2010-Dec. 3 2010.

[5] Davis, Robert I. and Burns, Alan., "A survey of hard real-time scheduling for multiprocessor systems", Journal of ACM Comput. Surv., vol. 43, no. 4, pp35-34, 2011.

[6] Carpenter, J.; Funk, S.; Holman, P.; Srinivasan, A; Anderson, J;Baruah, S., "A categorization of real-time multiprocessor scheduling problems and algorithms," Handbook On Scheduling Algorithms, Methods, And Models, 2004.

[7] Muppala, J.K.; Woolet, S.P.; Trivedi, K.S., "Real-time systems performance in the presence of failures," Computer, vol.24, no.5, pp.37-47, May 1991

[8] Wu, H.; Ravindran, B.; Jensen, E.D.; Peng Li, "Time/utility function decomposition techniques for utility accrual scheduling algorithms in real-time distributed systems," Computers, IEEE Transactions on , vol.54, no.9, pp.1138,1153, Sept. 2005.

[9] Ravindran, B.; Jensen, E.D.; Peng Li, "On recent advances in time/utility function real-time scheduling and resource management," Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on, vol., no., pp.55-60, 18-20 May 2005.

[10] Hyeonjoong Cho; Ravindran, B.; Jensen, E.D., "An Optimal Real-Time Scheduling Algorithm for Multiprocessors," Real-Time Systems Symposium, 2006. RTSS '06. 27th IEEE International, vol., no., pp.101,110, Dec. 2006.

[11] Shelby Funk, "LRE-TL: an optimal multiprocessor algorithm for sporadic task sets with unconstrained deadlines," Real-Time Systems, vol. 46, no. 3, pp 332-359, 2010.

[12] R. Clark, E. D. Jensen, A. Kanevsky, J. Maurer, P. Wallace, T. Wheeler, Y. Zhang, D. Wells, T. Lawrence, and P. Hurley, "An Adaptive, Distributed Airborne Tracking System," in Proceedings of The IEEE Workshop on Parallel and Distributed Systems, ser. LNCS, vol. 1586. Springer-Verlag, pp. 353-362, April 1999.

[13] D. P. Maynard, S. E. Shipman, R. K. Clark, J. D. Northcutt, R. B. Kegley, B. A. Zimmerman, and P. J. Keleher, "An Example Real-Time Command, Control, and Battle Management Application for Alpha," Department of Computer Science, Carnegie Mellon University, Archons Project TR-88121, December 1988.

A Hardware Accelerator Design Process for Speech Recognition Application

Byron Buitrago P.[1], Johnny Aguirre M.[2], Andrés Benavides A.[2], Marcela Rivera P.[3]
[1] Systems Engineering Department, University of Antioquia, Colombia
[2] Electronic Engineering Department, University of Antioquia, Colombia
[3] University of Antioquia, Colombia

Abstract— *This paper shows a methodology to improve time response in a sub-stage inside a complete speech recognition system, and shows how to apply such methodology to other stages inside the same process. This methodology involves hardware and software cooperative work based on soft-core design using MicroBlaze processors and reconfigurable hardware in Xilinx Spartan 3E FPGA. Design process, proposed methodology, software and hardware components, and the migration approach are described and analyzed along with the process results.*

Keywords: Speech recognition, Soft-core processor, MicroBlaze, FPGA, Software to hardware mapping.

I. INTRODUCTION

Optimization of desktop and embedded applications is often performed with respect to the target processor. Time consumption is a very important part of the performance measure in design, because from the final user point of view, a slow time response of the application could be interpreted as a poor processing behavior.

Nowadays, a wide range of applications aimed to human-machine interaction, are available for every type of devices. Fingerprint and face detection in addition to speech recognition can be mentioned, between others [1], being that last our case of study.

Human language processing requires different skills, such as computer science, electrical engineering, mathematics, psychology, linguistic, and others [2]. Due to its implementation complexity, speech recognition applications exhibit a wide range of requirements and limitations which represent important design challenges. From computer science and electrical engineering fields the most important parameters to optimize are: time performance, power consumption, circuit's area and memory read/write operations, between others. Currently, there are several tools for the implementation, analysis, research and development of software-based speech recognizers [3] [4] [5], which are the main trend of speech recognition systems. However, they are not aimed for exploring potential concurrences that can help to improve some aspects of a speech recognition system. In this work, we analyze a technique for improving time response inside a single stage of the speech recognizer, and show how the same technique may be applied to other stages in the same application. For that, we use hardware and software cooperative work through soft processor combined with reconfigurable logic.

This paper starts with a brief description of speech recognition process, continuing with a study case analysis in order to give a contextualization. In section IV, the implemented strategy is described and results are presented and analyzed. Finally, conclusions and future work concerning this strategy are depicted.

II. SPEECH RECOGNITION

Human voice is a mechanic wave generated through vocal cords vibration. This signal is emitted in a frequency range that is between 100 and 3400 Hz approximately. Human voice signal is sampled at 8 or 16 kHz in most applications by an analog to digital converter, as can be seen in Figure 1.

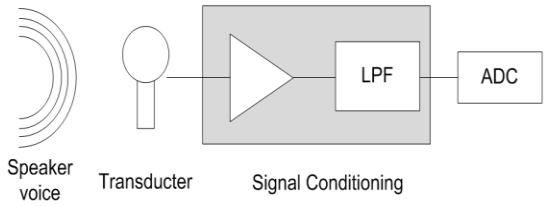


Figure 1: Speech signal acquisition.

Speech recognition can be seen as a process which transforms an acoustic speech signal into a text representation. Two clear differentiable stages are found on this process, as it is illustrated in Figure 2. First a preprocessing and feature extraction phase, common to any recognition system, followed by a decoding and language database phase

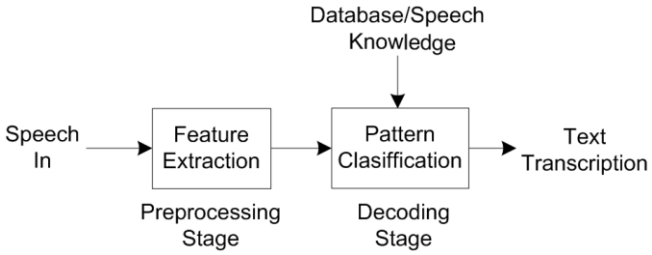


Figure 2: Speech recognition block diagram.

A lot of approximations can be used in the second part of the recognizer. In this particular work, statistical based models are used, this type of techniques implements Hidden Markov Models (HMMs) and Gaussian emission densities for pattern comparison and decision. However, several alternative approaches have been proposed in literature [6].

Preprocessing stage is composed principally by a low pass digital filter for band limitation (LPDF) followed by a fast Fourier transform block (FFTB) which feeds a bank of filters used to calculate Mel Frequency Cepstral Coefficients (MFCCs). All that process generates a compressed form of the original speech signal. Preprocessing stage is shown in Figure 3.

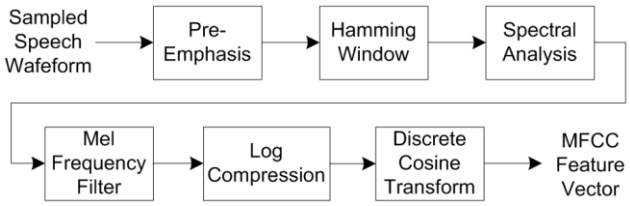


Figure 3: Preprocessing stage sub-blocks

It is extremely important to guarantee a good preprocessing analysis, in order to have properly recognition results. Digital filtering highlights inside this stage. In fact, multiplications and accumulation algorithms, necessary for the digital filtering process, demands a considerable amount of computational load [2].

III. CASE OF STUDY

Commercial speech recognizers can be found as licensed software, their main uses are oriented to servers, desktop computers and, more recently, video-game consoles and smart-phones, among others [7]. Available open source recognizers are supported by academic research centers and some universities around the world. Among them, the most matures are: CMU Sphinx developed by Carnegie Mellon University [5] and HTK developed by Cambridge University [4]. Both of them are written in C language. Their main idea consists in provide speech recognition tools for researchers around the world. Actually, Sphinx stands as the best academic speech recognizer for several reasons, including that its statistical models are more refined and accurate than other tools [8], and it has a totally open user license.

In order to measure some speech recognizer parameters, it was decided to implement the Sphinx software in a MicroBlaze soft-core processor [9] using Xilinx design environment tools. The parameters that were taken into account for the recognizer implementation were: computing time, power dissipation and memory usage.

IV. STRATEGY APPROACH

By taking advantage of the available reconfigurable resources for implementation, the choice of implementing some software routines as equivalent hardware models was developed, by following the next migration approach:

STEP 1 – SOFTWARE MIGRATION: In order to make source code compilable, it should be eventually adapted to particular architectural details in target processor.

STEP 2–SOFTWARE ROUTINES IDENTIFICATION: A routine or set of routines is selected from the original source code. According with optimization criteria, such routines are replaced by hardware modules.

STEP 3–PURE SOFTWARE ROUTINES CHARACTERIZATION: Assessment must follow the routines selections. Measures may include power consumption, execution time, and memory operations, and may provide a baseline for future comparisons.

STEP 4–SOFTWARE TO HARDWARE MAPPING: Once the software routines are selected, they must be translated into digital hardware modules. Some tools may be used in

this step like Cynthesizer ForteDS [9] and Xilinx Vivado HLS [10]. At this point it's highly desirable to implement a functional simulation over generated hardware, in order to provide a guarantee of correct mapping.

STEP 5–WRAPPER GENERATION:Generated hardware must be adapted as a peripheral module for the central processing unit. This step represents hardware and software cooperative work in order to improve application performance in the way selected at Step 2.

STEP 6–COOPERATIVE HARDWARE AND SOFTWARE SCHEME MEASUREMENTS: Data parameters of the new design must be obtained in order to compare it with purely software results of Step 3. If such results are not satisfactory, it will be necessary go back to Step 2.

V. RESULTS AND ANALYSIS

The previously described strategy was applied to preprocessing stage in a Sphinx speech recognizer which was implemented on a MicroBlaze soft-core processor; more precisely low pass filtering routines were addressed.

STEP 1 – SOFTWARE MIGRATION: Original Sphinx C source code was adapted to MicroBlaze, because the compiler threw several errors. Special architectural details of MicroBlaze must be taken in account.

STEP 2–SOFTWARE ROUTINES IDENTIFICATION: Filtering routines were selected for this work by using the reasons exposed in section II. A 16 kHz and 16 bits per sample test signal was used as digital filter input. A four hundred samples vector, corresponding to 25 ms of voice were submitted to the filter, it's a good approximation given that speech recognizer usually needs frames between 10 and 25 ms[11]. Filter frequency characteristics are shown in Figure 4.

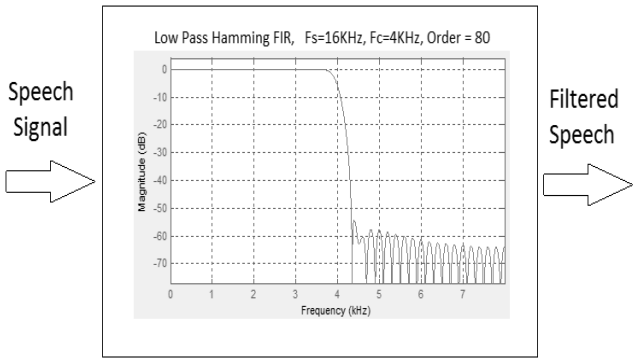


Figure 4: Digital filter frequency response

STEP 3–PURELY SOFTWARE ROUTINES CHARACTERIZATION: Filtering routines were implemented using Embedded Development Kit (EDK) of Xilinx ISE Design Suite in a MicroBlaze processor. Time execution was measured using internal timing function tool. Baseline results found in this step will be depicted in Step 6 of this section.

STEP 4–SOFTWARE TO HARDWARE MAPPING: Using Vivado HLS tool, digital hardware version of filtering software routines were obtained in HDL form.At this step it was necessary make tuning over various synthesis directives as pipeline, unrolling and flattened, between others. After this tuning, generated hardware showed the best performance with pipeline directive applied, as can be seen in table 1:

	BRAM_18K	FF	LUT	MULT18x18	SLICE
Component	-	-	-	-	-
Expression	-	0	83	1	-
FIFO	-	-	-	-	-
Memory	2	-	-	-	-
Multiplexer	-	-	83	-	-
Register	-	132	-	-	-
Total	2	132	166	1	0
Available	20	-	9312	20	4656
Utilization (%)	10	-	1	5	0

Table 1: Generated hardware area characteristics

A test bench of new hardware was implemented, putting special interest in functionality and time consumption. Test bench results are shown in Figure 5.

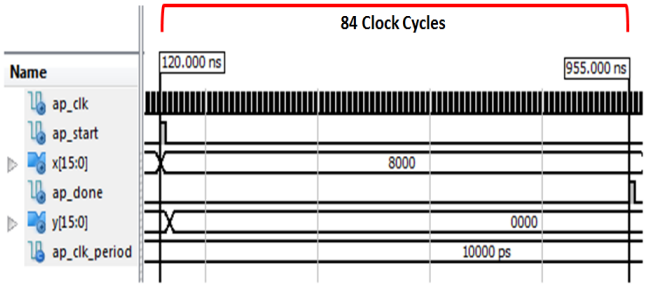


Figure 5: Generated hardware test bench

STEP 5–WRAPPER GENERATION:Xilinx ISE and EDK were used in wrapper generation to adapt generated filtering hardware as an external module connected to

MicroBlaze processor through Fast Simple Link (FSL) bus. Figure 6 illustrate the interconnection.

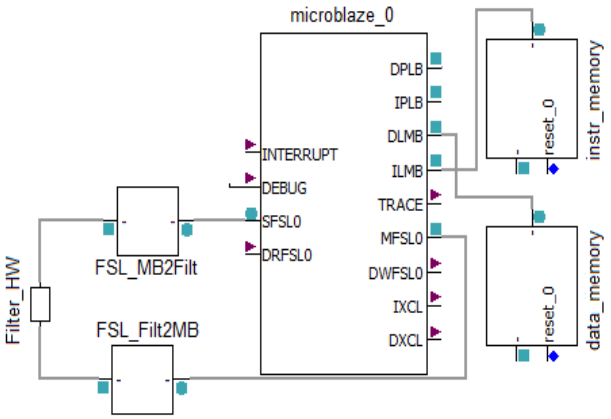
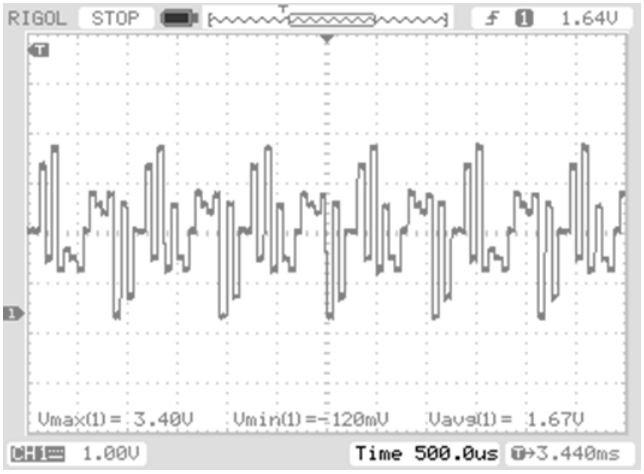
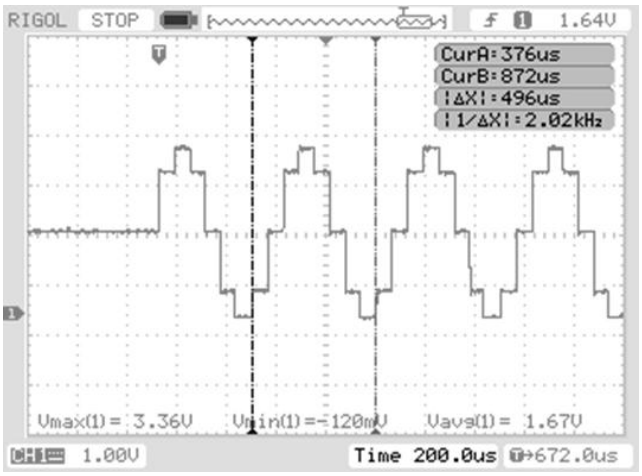


Figure 6: Schematically representation of MicroBlaze and filtering hardware communication

Filter functionality was tested with a linear combination of three sinusoidal waves of 2, 6 and 7 kHz, as can be seen in figure 7(a). Pure software and cooperative scheme responses were tested in MicroBlaze in order to compare its functionality. Results were identical in both cases, 4 kHz low pass filter response is showed in figure 7(b), where 2 kHz tone is the only one present in filter's output.



(a) Test input signal



(b) FIR filter output

Figure 7: Filtering functionality test

STEP 6–COOPERATIVE HARDWARE AND SOFTWARE SCHEME MEASUREMENTS: Generated filtering hardware was connected with a MicroBlaze processor as peripheral, old software routine call was replaced by peripheral activation and data to process was sent to it, waiting for returning results (filtered signal). Both schemes threw the same results, being the cooperative design which shows a considerable reduction in time execution, as could be seen in Table 2.

Parameters \ Solutions		**MB**	**MB+HW**
Latency(us)		9714	414
Area	Logic	3767	4092
	Signals	5095	5565
	BRAMS	16	18
	MULT	7	8
Power		0.128	0.130
Energy(Time*Power)		1243,39	53,82

Table 2: Pure software vs cooperative scheme results.

As it can be seen in Table 2, time delay to filtering related tasks are widely reduced, given a speedup of 23.4. The time execution of this process has been reduced to 4.2% of its original pure software implementation. The table shows a remarkable increase in area, especially in logic blocks usage. Power consumption presents a little increase and it could be expected a grown of it if this strategy is applied to other blocks inside the recognizer. However, energy estimation shows a considerable improving when the time execution is reduced by a factor bigger than power consumption increase.

VI. Conclusions and Future Work

Cooperative hardware and software scheme shows good results as an embedded design strategy. Manufacturers actually provide new and powerful tools which present huge advantages in the design process when an application involves software routines and custom hardware modules. This integration level is very hard to reach using standard hard-core processors.

Specific performed experiment shows an important speedup when both schemes are compared. Applying the same strategy over more Sphinx software routines simultaneously, the application speedup will be increased drastically.

Applications implicit concurrency presents an implementation difficulty doing it an aspect rarely exploited in traditional application development way. This strategy shows an easy form to natural concurrency exploration and exploitation of this feature inside applications.

Circuit area is an important topic inside this strategy, when a software routine is mapped into a hardware module, silicon area is inevitable increased. However, FPGA design is limited by its internal connections structure, which could be optimized in an ASIC design process. Area increase implies bigger hardware power dissipation too, but if we take into account *Power* × *Time* product, we can achieve a better energetic performance compared with pure software scheme.

To reach a better strategy performance, it will be necessary to propose an automatic routines profiling methodology in order to obtain easily the candidates to be submitted to hardware conversion process.

Proposed strategy could be applied to other speech recognition stages, among them, it can be mentioned the Fast Fourier Transforms, Gaussian densities calculation, Viterbi and Baum Welch algorithms [2], [11]. The same strategy may be also extended beyond speech processing applications.

In our interest to extend the scope of words and phrases recognition, the analysis of recognized sounds is proposed. Specifically from the psychology field, analyzing some speech characteristics such as volume, tone, speed, fluency, clarity, coherence, among others [12], we can study possible inferences about psychological aspects related to moods, personality traits and disorders in human thought.

VII. Acknowledgments

The authors would thank to Microelectronics and Control group staffat University of Antioquia, who provided the software and hardware tools during the realization of this project. Special thanks to professor Freddy Bolaños for his contributions in the development of this work.

References

[1] Desney S. Tan, Anton Nijholt, Brain-ComputerInterfaces Applying our Minds to Human-Computer Interaction, Springer, 2010.

[2] X. Huang, A. Acero y H. Hon, Spoken Language Processing, Prentice-Hall, 2001.

[3] Center for Spoken Language Understanding, http://cslu.cse.ogi.edu.

[4] Hidden Markov Model Toolkit – HTK, http://htk.eng.cam.ac.uk/

[5] SPHINX, Carnegie Mellon University,http://cmusphinx.sourceforge.net/

[6] Joseph Keshet, Samy Bengio, Automatic Speech and Speaker Recognition, Large Margin and Kernel Methods, A John Wiley and Sons, Ltd, Publication, 2009.

[7] Dragon NaturallySpeaking,Siri, Nuance, http://www.nuance.com/

[8] Amudravijaya K, Maria Barot "A comparison of Public Domain Software Tools for Speech Recognition", SNDT University of Mumbai, 2003.

[9] Forte Design Systems, http://www.forteds.com/

[10] Xilinx All Programable,. www.xilinx.com

[11] Levinson Stephen E., "Mathematical Models for Speech Technology". John Wiley & Sons Ltd, 2005.

[12] Marco Aurelio Fierro Urresta, "Semiología del psiquismo". ed Marco Fierro, 2008.

SESSION

MICRO-CONTROLLERS AND EMBEDDED SYSTEMS + NETWORK ON CHIP SYSTEMS AND APPLICATIONS

Chair(s)

TBA

Unexploded Ordnance Detection with Cooperative Mobile Robots

Brent A. Bell, Andrew L. Suchanek, Dalesha Cartman, James D. Stuckey,
Chris Farnell, Brett Sparkman, Guoqing Zhou, Jingxian Wu, and Scott C. Smith

***Abstract*—Due to the current cost of detecting unexploded ordnances (UXOs), alternate systems of detection need to be developed. The Department of Defense views UXO contamination as a high priority problem [1]. The current method of UXO detection involves sweeping an area for UXOs manually with handheld metal detectors. An automated system of detection would save time, cost less, and provide a safer alternative to the current system. This paper describes a system that was created composed of autonomous robots that detect UXOs and wirelessly transmit their coordinates to a central hub where they are mapped. These robots are meant to be smaller prototypes that could be implemented as a larger system for real world application. After these robots create a map of the UXOs in the area, time can be saved in the removal process and human interaction with the contaminated area will be kept to a minimum.**

***Index Terms*—Global Positioning System, Microcontrollers, Mobile Robots, System Sensors, Wireless Sensor Network**

I. Introduction

MINES, bombshells, and ammunitions, also known as Unexploded Ordnances (UXOs), pose a serious hazard when dealing with military training sites and battlefields. These UXOs can be both unstable and unpredictable. This increases the danger that humans and wildlife face when in an area contaminated with UXOs; therefore, UXOs should be removed when decontaminating an area for reuse. There have been many cases where civilians have tampered with UXOs resulting in serious injury or fatality. The Department of Defense estimates that there are more than 10 million acres of land that contain UXOs, and billions of dollars would need to be spent in order to decontaminate these areas [1].

The current method of UXO removal involves manual search by personnel with handheld metal detectors; however, this method of UXO discovery is time consuming, expensive, and impractical for large areas. Due to the large area of land contaminated by UXOs, compounded with their unpredictability, there would be a large gain from an automated system of detection for these UXOs. This would make covering larger areas of ground easier while also reducing the risk involved in the removal of the UXOs.

Several different attempts have been made to create better systems of UXO detection, and most of these use mobile robots in order to detect the buried UXOs. A researcher at the Massachusetts Institute of Technology completed such a project by creating a robot that moved using pressure sensors mounted on the robot. The robot had a unique design based on inspiration from an ant, and had five different sensors including a metal detector. However, there were some problems with this implementation: the robots had problems staying within defined boundaries, and the robots would travel out of range [2].

Another similar project was completed at Tennessee State University where a robot was created that could detect the UXO and distinguish it from false positives for objects that weren't UXOs. The best result of this project was 94% accuracy in UXO detection [3].

Many researchers have investigated alternate ways to detect UXOs without a metal detector. Researchers from University College London have begun exploring the use of Multi Frequency Ground Penetrating Radar in order to detect both surface and buried explosives. This was mounted on an Unmanned Aerial Vehicle for contactless remote detection [4].

The method proposed in this paper involves using autonomous robots that will sweep a given area to detect hidden UXOs through the use of a metal detector while also avoiding any objects in their paths. By using autonomous robots, UXOs could be detected, and their location would be wirelessly transmitted back to a central location without any need for human interaction in the detection process. As the coordinates of the UXOs are received, the fusion center will plot the coordinates on a map of the area. With this map, the removal of UXOs would be much easier than the current

Manuscript received May 26, 2013. This work was co-funded by the ASSURE program of the Department of Defense in partnership with the National Science Foundation REU Site program under Grant ECC-1005106.

Brent A Bell, Chris Farnell, Brett Sparkman, Guoqing Zhou, Jingxian Wu, and Scott C. Smith are with the Department of Electrical Engineering, University of Arkansas, Fayetteville, AR 72701 USA (emails: bab010@uark.edu, cfarnell@uark.edu, bsparkma@uark.edu, gzhou@uark.edu, wuj@uark.edu, and smithsco@uark.edu).

Andrew L. Suchanek is with the Department of Computer Science and Computer Engineering, University of Arkansas, Fayetteville, AR 72701 USA (email: asuchanek32@gmail.com).

Dalesha Cartman is with the Department of Mathematics, Computer, and Information Sciences, Mississippi Valley State University, Itta Bena, MS 38941 USA (email: dhcartman@gmail.com).

James D. Stuckey is with the Department of Electrical and Computer Engineering, Missouri University of Science and Technology, Rolla, MO 65401 USA (email: jds2p8@mst.edu).

TABLE I
COMPONENT PRICE LIST

Component	Cost	Quantity	Total
DFRobotShop Rover V2	$94.99	2	$189.98
Compass Module HMC6352	$34.95	2	$69.90
Encoder Pair	$15.99	2	$31.98
GPS LS23060 V1.0	$59.95	3	$179.85
Ultrasonic Sensor HC-SR04	$2.43	4	$9.72
XBee Pro Series 2	$28.00	3	$84.00
Miscellaneous			$144.46
Total Project Cost			**$709.89**

method.

This paper is divided into five major sections. Section II provides an overview of the system. Section III describes the autonomous navigation methods researched. Section IV describes the fusion center including the user interface. Section V provides conclusions and opportunities for future work.

II. OVERVIEW OF SYSTEM COMPONENTS

The system can be split into two basic parts: the fusion center and the robots or rovers. The fusion center is in communication with the computer via a graphical user interface (GUI). Commands can then be sent wirelessly from the fusion center to the rovers, and the rovers will autonomously search an area for UXOs. If any are found, the GPS coordinates are wirelessly transmitted back to the fusion center where they are displayed on a static Google map of the area.

All of the components for both the rovers and the fusion center were chosen based on their quality, compatibility, power consumption, and price. Power consumption was a concern because the rovers run on a battery, and the price was an obstacle because many of the components, especially those with higher accuracy, can be very expensive. An approximate breakdown of cost for the project is included in Table I.

A. Platform

The Arduino Uno development board was chosen as the platform for the project due to its flexibility and compatibility. The Arduino microcontroller is built onto a board that makes programming and using the microcontroller as simple as possible. Another advantage to using the Arduino development system is that "shields" may be added onto the basic board to add different levels of functionality. Each shield adds sensors or allows you to easily attach components to the Arduino.

Arduino also has its own programming language for programming the microcontroller. It is based on C++, so learning to code the Arduino microcontroller is straightforward, since most basic programming courses cover C/C++. The IDE for Arduino streamlines the process of writing, compiling, and uploading the code to the Arduino microcontroller.

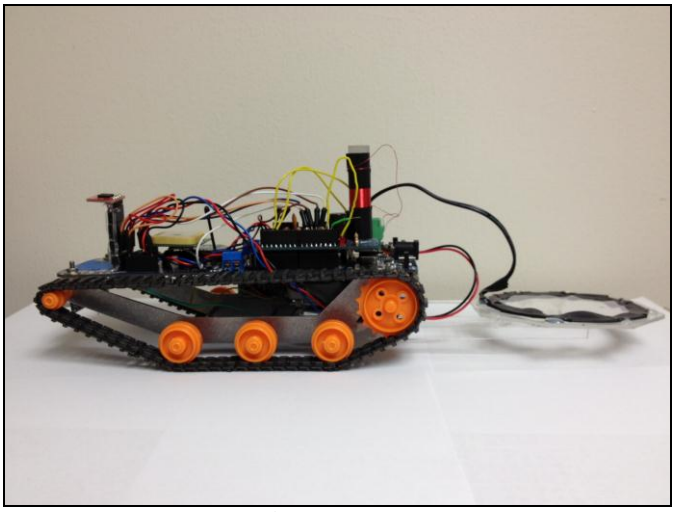


Fig. 1. Final implementation of rover with all components on board.

B. Robot

Several different options of mobile robots were available to choose from. The two main options that were explored were the Rover 5 Robot Platform from Sparkfun Electronics and the DFRobotShop Rover V2 from Robotshop. The Rover V2, shown in Fig. 1, was chosen due to its compatibility with Arduino and the XBee wireless communication modules. The Rover V2 has a built-in Arduino Uno and receptacles to mount the XBee. The Rover 5 lacks these features.

Each rover runs off of a 3.7V lithium-polymer rechargeable battery. There is a built-in charging circuit on the rover that allows these batteries to be charged by the USB port. The gear ratio is set to the lowest setting, 203:1, to improve the off road capabilities of the rover. The rover is relatively small with dimensions of 200mm long x 108mm wide x 58mm high. The rover is also tracked instead of having wheels. The small size of the rover does make it good for prototyping but hinders its off road capabilities; however, the tracks make it more maneuverable off road.

C. Communication

Wireless communication is accomplished through the use of XBee Wireless RF Modules. The XBee modules used were the XBee Pro Series 2 with a wire antenna for longer distance communication. The pro modules also have a longer range than other XBees, and the Series 2 XBees offer more customizability than Series 1. The Series 2 XBees also support mesh networking, and while mesh networking was not used in the current application, it would be a feasible future implementation that would be especially beneficial for systems with several Rovers.

Two communication modes are available to the XBees: transparent mode and API mode. In transparent mode the radio passes information exactly as it receives it on the serial port, and the data is output to the serial port exactly as it is received. API mode allows for more flexibility by creating packets that can be sent to any XBee. In transparent mode, the XBees must be reprogrammed either through commands from the microcontroller or the computer in order to change the destination address of the data it is sending. In API mode, the

TABLE II
METAL DETECTOR TEST RESULTS

Metal Detector	Size	Detection Range
Zircon Metal Finder	3.25" x 1.5"	3"
ZX Metal Detector Module	½" x 1.5"	< ½"
Pulse Induction Type*	4" x 6.5"	Non-functioning
555 Timer Circuit*	2" x 2"	< ½"
Beat-Frequency Oscillation Type*	2.25" x 2.75"	3"

Metal Detectors denoted by an asterisk are circuits built from scratch from a block diagram. The other two metal detectors are commercial metal detectors.

destination address is within the packet being sent, so sending data to different XBees is simple.

The XBee modules for this project are configured as API modules. There is one XBee at the fusion center that is programmed as the coordinator. This is the XBee that sets up the network. The two rovers are configured as routers. This allows them to communicate with the coordinator and each other.

D. UXO Detection

Due to most UXOs being made of metal, the most convenient method of detecting them is by using a metal detector. Many different metal detectors were researched and tested for this project due to the need for a small compact metal detector that could be mounted on the robot. Also, the metal detector needed to have an adequate range of detection. Two commercial metal detectors were tested for their range, and three different circuits found online were adapted, built, and tested. The results of these tests are shown in Table II.

The circuit that was the best for our project was the final circuit that was built from scratch, shown in Fig. 2. This circuit uses beat frequency oscillation to detect metal. This type of metal detector works by comparing two coils acting as frequency oscillators [5]. These two coils are tuned to near the same frequency. When the detection coil moves near metal, its oscillation frequency changes due to the disruption of the magnetic fields surrounding the coil. The output of the circuit then changes accordingly, and the Arduino can detect this change in order to detect nearby metal.

When the rover is turned on, the pulse width of the output of the circuit, which is directly determined by the difference in frequencies of the reference coil and the detection coil, is taken several times by the Arduino and averaged. This value is stored as the base pulse width of the metal detector when no metal is nearby. It can then be compared to the current pulse width throughout the sweep to determine if there is any metal in the area. If the output is significantly different, a UXO must be nearby, and then the rover can read in the GPS coordinates and send them to the fusion center.

In order to provide optimum detection, a mount was built to hold the detection coil of the metal detector, as shown in Fig. 1. It extends off the back of the rover and is made of plexiglass. There were several different designs for this mount that were attempted. The chosen design is fixed approximately two centimeters off the ground, and was selected due to its durability. However, this does create a problem when the rover starts to ascend a steep hill. Due to the fixed height of the

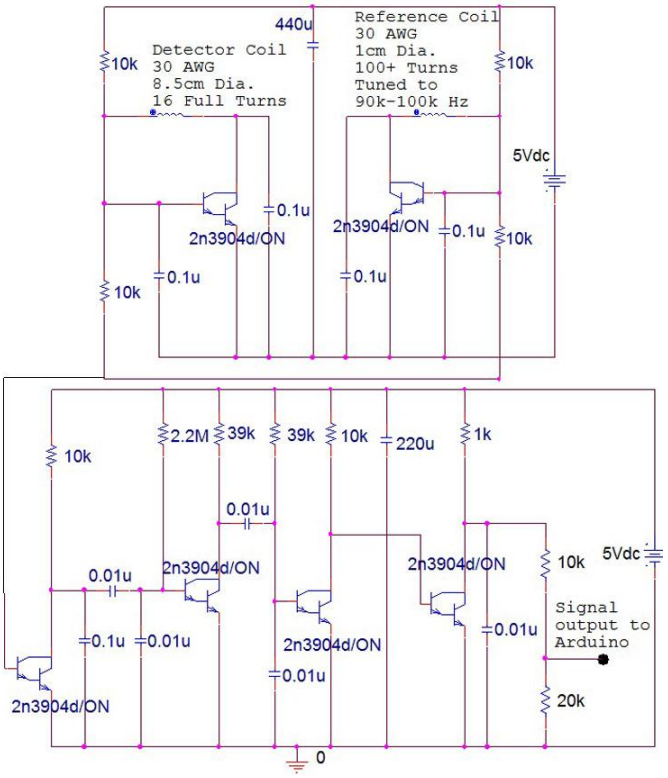


Fig. 2. Metal detector circuit used on rover. This circuit was adapted from a metal detector circuit found online [6].

mount, the plexiglass will drag as the tilt of the rover changes. Another mount that was built fixed this solution by adding hinges and a wheel so the mount could rotate up and down. This design would have fixed the problem of dragging but was significantly less durable, so it was not chosen for our final implementation. Keeping the rover on relatively flat ground was adequate for testing purposes.

E. Other Sensors

There were 4 other sensors used on the rover for navigation and mapping of UXOs.

1) Ultrasonic Ranging Modules

A ranging module was needed for autonomous navigation, so that objects could be detected and avoided. Two different sensors were tested for this purpose: an infrared distance sensor and an ultrasonic ranging module. Both worked well, but the ultrasonic ranging modules provided more accurate results. Output from the infrared sensors tended to be more sporadic. For this reason, the ultrasonic ranging modules were chosen to be the object detecting sensor on the rover. There are two ultrasonic ranging modules mounted on the rover. One of the ultrasonic ranging modules is mounted on the front of the Rover, and the other is mounted on the right side of the rover. Object avoidance could be achieved with one module also, but it would need to be mounted on a servo in order to view different directions.

2) Compass

In order to know which direction the robot was facing, a sensor was needed that could determine its bearing. While it is possible to calculate the bearing from the global positioning system (GPS) when the rover is moving, this would not be an

adequate solution to the problem for two reasons. The GPS is not accurate enough on the small scale used for testing to provide an adequate heading. Also, being able to calculate the bearing only while the rover is moving, places limitations on the rover's navigation, especially when first starting to sweep an area.

3) Global Positioning System

To begin, the GPS was used for two purposes: retrieving the coordinates of the UXOs that were detected by the metal detector, and defining the boundaries of the search area. Due to the smaller scale of the project and the large margin of error, the GPS was inadequate for defining the boundaries of the search area. The GPS lists an accuracy of three meters, but the actual coordinates could have a larger or smaller margin of error. Some points were off by as much as five meters. On the small scale used for testing, this is a huge amount of error introduced into the system. So, an alternative method of determining a robot's position needed to be found. However, the GPS was still used for obtaining the approximate coordinates of any UXOs detected.

4) Encoders

Encoders were added to the rover as an attempt to account for the limitations of the GPS. The encoders could be used to find the distance the rover had traveled in any direction, but they also had limitations. The different techniques used for both the encoders and the GPS in navigation are outlined in the next section.

The encoders use light and a wheel with eight cutouts mounted on the rear axle of the rover to increment a counter 16 times for every single rotation of the wheel. The encoders output an analog value based on how much light is reflected back. Due to the cutouts on the encoders, the amount of light reflected changes, and these changes can be detected by the Arduino in order to keep track of how far the rover has traveled.

III. Autonomous Navigation

Perhaps the largest amount of time spent on this project went into writing and testing the autonomous navigation program. This can be divided into two different phases: the object avoidance function and the area sweep function. First, the rover was designed to go around any objects in its path. Then, two different methods of sweeping an area were tested.

A. Object Avoidance

Object avoidance was achieved through the combination of two ultrasonic ranging modules and timing. The algorithm performed when the path of the rover is obstructed starts when an object is detected at less than three centimeters in front of the rover. The rover will then make a 90° turn to the left. This will put the right side of the rover near the object.

The right sensor will then continuously check the distance between the rover and the object as the rover drives forward. After the sensor detects that there is more than 10 cm between it and anything on the right side, it travels straight for two more seconds. This is to give the rover enough room to make another 90° turn. The rover will then turn 90° to the right and travel until it passes the obstacle at which point it will make another right turn and travel until it is in line with its original path. Finally, the rover will make its final left turn and continue on with its original path. This sequence is shown in Fig. 3.

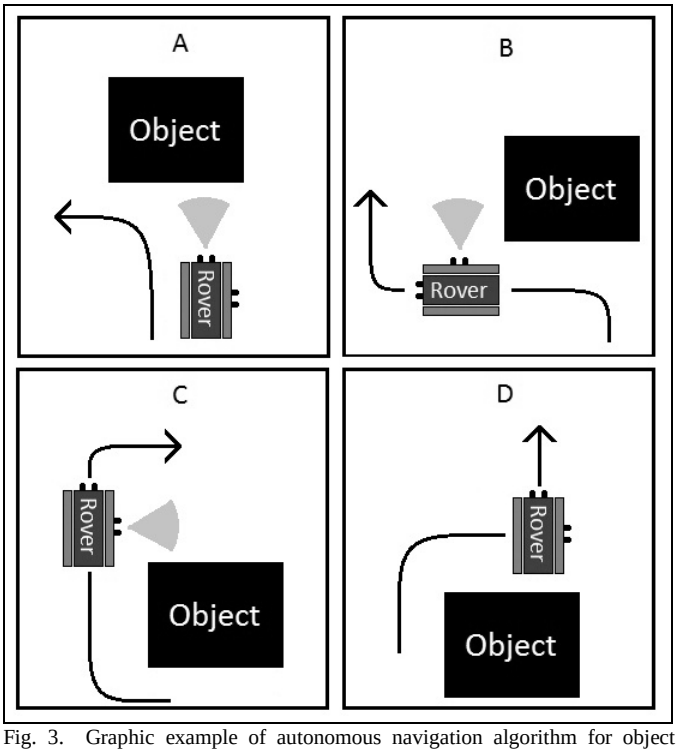


Fig. 3. Graphic example of autonomous navigation algorithm for object avoidance.

If another obstacle is detected during the obstacle algorithm, the rover will act accordingly. The rover will keep track of how many turns it has made in each direction as it continues avoiding any obstacles that are in front of it. It will turn left any time it detects an obstacle in front and turn right any time it has room to travel in that direction. This will continue until the rover is on the other side of all the obstacles in its path.

B. Area Sweep

The second phase of coding for autonomous navigation involved programming the rovers to cover everywhere in a predefined area. There were two main groups of sweep algorithms tested: GPS based navigation and encoder based navigation.

Both of these navigation techniques were built around the same approximate path. The rovers were set to start at the outside corners of an area and to spiral inward until the area was completely covered. This was accomplished through different methods for each type of navigation.

1) GPS Based Navigation

GPS was the first type of navigation tested. The user could enter the coordinates of each corner of the area boundaries into the graphical user interface (GUI). The coordinates would then be wirelessly transmitted to the rover. The rover would then calculate a bearing based on its position and the position of the first point entered into the GUI. When the rover arrives at point one, it calculates the bearing that it needs to follow to get from its current position to the second point. The first point is

then moved about one foot in the direction of the center of the search area. Because each point is moved in as the rover goes around, it slowly spirals inward. Finally, the sweep function will end when all of the points are moved to the center of the search area.

This method of autonomous navigation has several advantages. With this method, the user can enter the coordinates of the boundaries into the GUI. This makes it simple to search a predefined area. With slight modifications to the code, the area could easily be defined with more than 4 points and could be any shape. Also, with GPS based navigation, it is easier to correct for any errors in the direction the rover is heading. If the rover is off the path, it can simply calculate a new bearing based on its current location, and travel in that direction. Finally, with GPS navigation, the rover does not have to be placed at one of the corners of the search area. This is because no matter where the rover is placed, it can calculate the bearing to the first coordinate entered into the GUI.

However, GPS based navigation has limitations that make it infeasible for our prototype. Due to the small scale of the rovers and the small search area in testing, a small error in the GPS coordinates introduces large amounts of error in the rover's path. Because the GPS modules could be off by three meters or more, they were not adequate for navigation exclusively off of the GPS coordinates. So, alternative navigation methods needed to be explored.

2) Encoder Based Navigation

The encoder based navigation algorithm is the one that is used in our final prototype. It works similarly to the GPS navigation algorithm in that it also causes the rover to start at an outside corner of the search area and spiral inward until it reaches the center. This is accomplished by splitting the area up into a coordinate plane where each increment of the encoder is one unit in the plane. The dimensions of the area may be entered into the code before the rover is programmed. The rover must then be placed at one of the corners of the area.

The rover's compass must be calibrated at this point. The user turns the rover in the direction of the first point that the rover should travel to, and the bearing is taken and stored as the rover's positive 'y' direction. The user then turns the rover 90 degrees to the right until the LED indicator on the rover indicates that the positive 'x' value is stored. This process is repeated until all 4 directions are stored in the rover.

Then, once the sweep is started, the rover starts to travel in the positive 'y' direction. Each state change of the encoder will cause the Arduino to increment the 'y' coordinate by one. So, an approximate location of the rover is kept track of based on how far the rover has traveled. When the rover turns 90° in any direction, the appropriate part of the coordinate is incremented or decremented as the rover travels in that direction.

IV. User Interface and Fusion Center

A user interface and fusion center, shown in Fig. 4, was also created in order to provide the user with a way to control the rover's actions wirelessly. The fusion center acts as a bridge for communication between the computer and the rover.

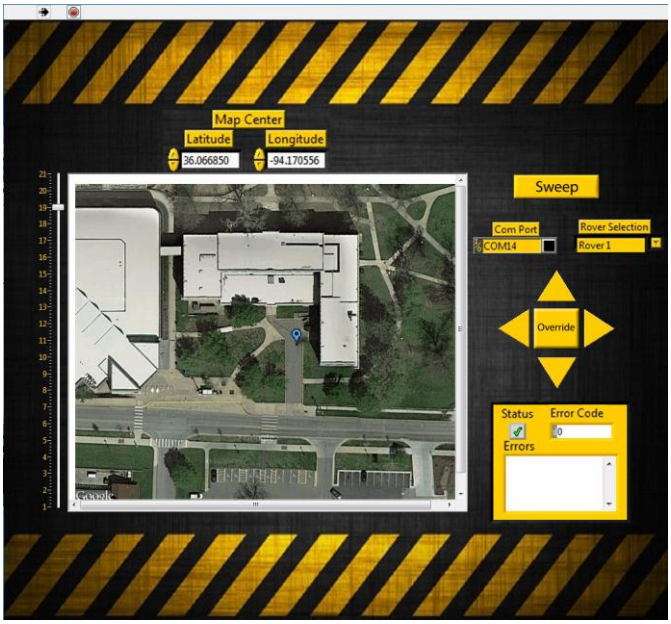


Fig. 4. Graphic user Interface with one GPS coordinate mapped. There are several features: Google static map, zoom bar, map center inputs, six control buttons, com port selection, rover selection, and error output.

A. Fusion Center

The fusion center contains four basic components: an Arduino, XBee, LCD screen, and GPS. The fusion center has two main purposes: communication with the computer and wireless communication with the rovers. It can also output the GPS coordinates to the LCD screen. The Arduino is used to pass on relevant commands from the computer to the XBee and coordinates sent from the rovers to the computer.

When commands are sent from the computer to the fusion center via USB cable, the commands are processed by the Arduino. Depending on which rover the command is for, the Arduino assembles the appropriate packet, which is then sent to the appropriate rover. Upon receipt of the packet, the rover executes the command.

Due to the XBee using the hardware serial port on the Arduino, the Arduino is not able to receive data from both the computer and the XBee normally. To solve this problem, a software serial port was created on the Arduino, and a mini-USB port was connected to these software serial pins. Through this solution, the Arduino can communicate with both the XBee and the computer separately and can pass on the appropriate information.

B. LabVIEW

Two different programs were researched in order to create the GUI: Processing and LabVIEW. Processing was the first considered because it is closely related to the Arduino programming language and therefore is easy to use with the Arduino; however, after also looking at LabVIEW, it was decided that LabVIEW would be better for our application. LabVIEW makes mapping coordinates on a map much easier than in processing, and it provides a more professional looking

GUI, and is easier to use than processing.

The GUI can receive coordinates and plot them on a Google static map. A constant loop checks to see if any bytes are available to read on the serial port. If there are any bytes available, they are read in and the bytes are converted into the float numbers they represent. Those float numbers are then converted into a string, which is appended onto the URL that requests a static map from Google. This map is then displayed on the GUI showing both a map of the area and a balloon for the received coordinates.

The GUI also has ways to send commands that control the movement of the rovers. There are six buttons: Start Sweep, Override, Left, Right, Forward, and Reverse. When the user clicks any of these buttons a one byte hex command is sent to the fusion center. After rover calibration, the start sweep button can be used to begin the autonomous navigation of the robot throughout the area. These commands can then be sent to the appropriate rover and then executed. For the override function, a dropdown menu allows the user to select the rover that is to be overridden. Once in override mode, the directional buttons can be used to control that rover. Clicking the override button again takes the rover out of override mode.

V. Testing

After all of the components were implemented and the rover was communicating with the computer properly, the system was tested for accuracy of automated movement and detection. A total of 8 metal plates of varying sizes were randomly distributed over an area of approximately 50 square feet in order to test the system. Two rovers were programmed to search this area by inputting the values for the dimensions of the area. This test was performed several times with the plates in different random layouts. Most of the tests had similar results.

For a typical test, the rovers detected 7 of the metal plates, and plotted them on the map shown in Fig. 5. It is easy to see in this figure the resolution of the GPS is not very high, since it appears that the plates are arranged in rows. However, on a larger scale with actual landmines, this would not be as much of a problem because the mines would be spread out over a much larger area.

Only one plate went undetected during this test. This was probably due to the fact that the plate was never in the path of the rovers. This problem was due to errors in the autonomous navigation of the rover. This could be corrected with more accurate sensors, and better autonomous navigation. However, overall, the test was very successful with a detection rate of around 88%.

VI. Conclusions and Future Work

Autonomous mine detecting rovers would be a good alternative to the current method of UXO detection, and with proper funding, a system that is adequate for large scale applications could be built. Of the ones tested, the best method

Fig. 5. Results of testing for UXO detection. Eight metal plates of varying sizes were distributed randomly over an area of approximately 50 square feet.

for use in large scale applications would be the GPS based navigation method.

There is also room for future work with this project, especially in the area of autonomous navigation. There are several problems that need to be addressed with the current method of navigation, including errors with the compass and errors with the encoders. All of the errors that come from these two components are due to the inexpensive, lower quality components used, and could therefore be fixed with superior components.

A. Compass Accuracy

The compass does not give accurate headings even when level and away from all sources of metal. Also, each degree of heading change does not directly correspond with one degree of rotation of the compass. Even after recalibration of the compass, the headings are inaccurate and imprecise. In the project, this was corrected by implementing a rover calibration function where the rover takes in compass headings and sets those headings as its bearings. As mentioned before, future solutions to this problem would be to acquire a more accurate compass module.

B. Encoder Problems

Also, since the encoders work based on the amount of light reflected back from the LED on the encoder, they do not work when the rover is placed in sunlight. This is a huge problem due to the rovers needing to be able to search in any environment. This problem was worked around by testing in the mornings when the sunlight was less intense. As mentioned before, this is a problem with the encoders themselves, and these problems could easily be fixed with better encoders.

An alternate solution to this problem would be to create a resistor divider circuit. Because the ADC on the ATMega328p chip only has a 5 mV resolution, it is possible that the reading

in sunlight outputs too high a voltage. There is not much documentation on these encoders, so more research would be necessary to determine if the encoders could be used.

C. GPS as a Solution

The encoders are not necessary when the rover is navigating off of the GPS, so in a large scale application where a better GPS can be acquired for the project, the problem with the encoders would not need to be addressed. Because the system will be used for military purposes, a more accurate GPS could be easily obtained. By using the military tier of GPS, the coordinates would already be more accurate, and there are more expensive devices that can give an accuracy of less than one centimeter [7].

A more accurate GPS would also help to fix the inaccuracies when plotting the location of each UXO. Since the UXOs are plotted on the map based on their GPS coordinates, a more accurate GPS would allow UXO clean up personnel to more easily find the UXOs for removal.

D. Maneuverability

One other problem with the current system is that there are maneuverability problems with the rover. A larger rover would be better for off road use, and therefore large rovers would need to be developed for military application. There are also problems with the metal detector mount hitting objects because the current design does not detect objects beside the metal detector when making a turn. This could be solved by adding code that 'remembers' if an object has been detected at a certain location, and then makes turns based on that. Another solution is to add an ultrasonic ranging module to the rover that is mounted on a servo, so any angle can be viewed to see the distance of any object.

REFERENCES

[1] S. Billings (2009) *Next Generation HeliMag UXO Mapping Technology* [Online]. Available: http://www.serdp.org/Program-Areas/Munitions-Response/Land/Sensors/MR-200741

[2] J. McLurkin. U*sing Cooperative Robots for Explosive Ordnance Disposal* [Online]. Available: http://www.cs.cmu.edu/~biorobotics/papers/sbp_papers/integrated1/mcmcklurk_ants_eod-paper.pdf

[3] R. Haroun, A. Saed, and A. Shirkhodaie (2008). *UXO Detection, Characterization and Remediation Using Intelligent Robotic Systems* [Online]. Available: http://144.206.159.178/ft/CONF/16414685/16414704.pdf

[4] A. Amiri (2012). *Landmine, IED, UXO Detection using Ground Penetrating Radar from an Unmanned Aerial Vehicle* [Online]. Available: <http://www.ucl.ac.uk/secret/2010-phd-projects/amiri-phd>

[5] C. Wessels and T. Palag (2005). *Contsturction of a Beat Frequency Oscillator Metal Detector* [Online]. Available: http://ecee.colorado.edu/~mcleod/teaching/EandM3400/Projects/Examples/metal-detector.pdf

[6] D. Smith (2012). *BFO Metal Detector* [Online]. Available: http://www.easytreasure.co.uk/bfo.htm

[7] E. Edmonds (2012). *Survey Grade, Mapping Grade or Recreational Grade GPS?* [Online]. Available: http://www.geo-jobe.com/blog/2011/03/18/survey-grade-mapping-grade-or-recreational-grade-gps

Design of Multi-Ring On-Chip Network

Chao-Hao Su, Jen-Yu Wang, and Yarsun Hsu
Department of Electrical Engineering, National Tsinghua University, Hsinchu City, 320, Taiwan, R.O.C.

Abstract - *In this study, a multi-ring interconnection network architecture is implemented and examined. This study also proposes a cross-multi-ring topology that allows packets to be transmitted across various rings in the same direction. The performance of the two topologies was then evaluated and compared. The results indicate that a positive correlation exists between the performance and cross-transmission usage rate.*
Keywords: Network on chip, multiprocessor, ring topology, cross-ring topology

1 INTRODUCTION

With the development of semiconductor processes and system-on-chip (SoC) technologies, the number of SoC design cores has increased. According to the Moore law [1], the transistor count increases at a rate of a factor of two per year. This implies that the number of processing units in a SoC design will continue to grow, further increasing the difficulty of controlling the performance, power, and scalability. To overcome these challenges, numerous studies have proposed the network-on-chip (NoC) approach, which adopts the packet switching and routing technique of communication networks. In 2000, Hemani et al. [2] stated that "An NoC is a network of computational storage and I/O resources connected through a network of switches, where resources communicate with each other using addressed data packets routed to their destination by the switch fabric." Subsequently, NoCs have attracted substantial attention. Various NoC architectures exist, including crossbar, mesh, torus, ring, and hybrid architectures [3, 9]; each architecture has differing implementations and limitations.

Among the various interconnection networks, a ring architecture offers the advantages of low power consumption, small chip area, and low implementation complexity, in addition to guaranteeing a certain performance level; thus, it is well suited for multiprocessor system-on-chip (MPSoC) and chip multiprocessor (CMP) designs. Ring architectures are used in numerous commercial systems such as IBM's Cell [4, 5, 6, 7, 8].

This study implements a multi-ring topology, extending it into a cross-multi-ring topology that enables cross-transmissions between rings in the same direction. A performance analysis and discussion regarding the number of rings and the cross-multi-ring topology are also presented.

This paper is organized into sections as follows: In Sections II and III, a four-ring topology and a cross-four-ring topology are implemented, and their internal structures are explained. Section IV provides an analysis and comparison of their performance under various traffic patterns. The study conclusion is presented in Section V.

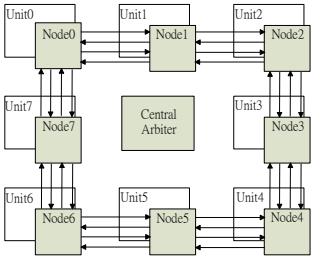


Figure 1. Four-Ring Topology

2 FOUR-RING TOPOLOGY

Because of the limitations of software parallelism, large-scale multicore chips have not been commercialized. Mainstream products have a maximum of 20 cores; in this scale, the ring topology is commonly employed. In this section, a four-ring interconnection network is implemented.

2.1 Four-Ring Topology Framework

Fig. 1 shows the four-ring topology, which comprises two clockwise and two counterclockwise rings. Data units are connected to the rings, and the number of data units can be configured. This study adopted eight for use as an example.

This study partitioned each ring with the number of units into segments, and each segment was maintained by a corresponding node. To produce a transmission, the units send a request with a destination ID to the central arbiter. Upon receiving the request, the central arbiter checks the usage status of all segments, and selects an available transmission path. The central arbiter then sends specific pulses to each node in the path notifying them to alter the status of their control table. Fig. 2 shows the internal architecture of the node corresponding to Unit 0; each node possesses a control table and a set of transfer switches, and each transfer switch is connected to a corresponding ring.

As shown in Fig. 2, Rings 0 and 2 are responsible for clockwise data transmissions and Rings 1 and 3 are responsible for counterclockwise transmissions. Transfer Switch 00 controls the data flow on Ring 0 of this node. Similarly, Transfer Switches 01, 02, and 03 control the data flows on Rings 1, 2, and 3, respectively. The function of a ring transfer switch is similar to that of a highway ramp. When data flows toward the node, it can choose between the options of

on-ramp, off-ramp, or pass-ramp, as decided by the control table. The control table receives a pulse from the central arbiter mentioned previously, and converts the pulse into corresponding control signals; these signals (e.g., din00, cl00, and dout00 shown in Fig. 2) guide the data flow to enter the ring, leave the ring, or pass directly to the next node.

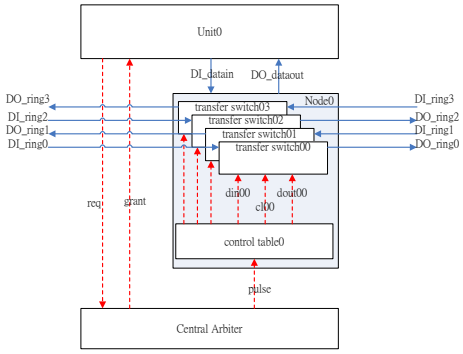


Figure 2. Internal architecture of the node corresponding to Unit 0 (Node 0) in a four-ring topology

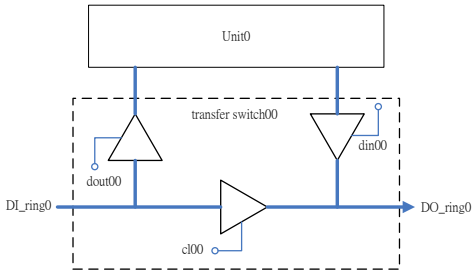


Figure 3. The structure of Transfer Switch 00 in Node 0

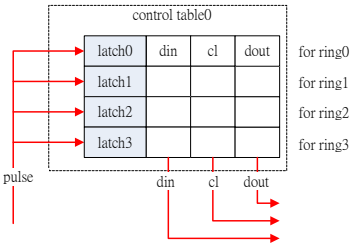


Figure 4. The structure of Control Table 0 in Node 0

2.2 Node Architecture

The structure of Transfer Switch 00 in Node 0 is shown in the dash-lined box in Fig. 3. (The first 0 of Transfer switch 00 indicates that it belongs to the node of Unit 0, and the second 0 indicates that it is connected to Ring 0.) Fig. 3 shows three switches, namely, switch in (Sin), switch out (Sout), and switch pass (Spass), which are controlled by the control signals din, dout, and cl, respectively.

- When din00 is set to high, the node of Unit 0 is in the on-ramp data sending state and can send data to Ring 0.
- When dout00 is set to high, the node of Unit 0 is in the off-ramp data receiving state and can retrieve data from Ring 0.
- When cl00 is set to high, the node of Unit 0 is in the pass-ramp data passing state, in which data from Ring 0 input port DI_ring0 is passed to the Ring 0 output port DO_ring0.
- When dout00 and cl00 are both set to high, data is sent to multiple destinations (i.e., multicast). At this time, the node of Unit 0 receives data from Ring 0 and also passes data to the next node because, although Unit 0 is one of the destinations of this data, it is not the final destination.

The states of control signals din, dout and cl are maintained by the control table inside the node. As shown in Fig. 4, a control table possesses four latches, and each latch maintains a set of din, dout and cl control signals. The control signal sets of Latches 0, 1, 2, and 3 correspond to Transfer Switches 00, 01, 02, and 03, respectively. The control table converts the pulses sent from the central arbiter into corresponding control signals, and maintains them in the latch. The pulses sent from the central arbiter can be classified into four types, and the four pulse types are explained below.

- *SEND*: When a latch receives a SEND pulse, the latch sets din to high, which means that the central arbiter has found an available path for the unit to send data into the ring.
- *PASSTHRU*: When a latch receives a PASSTHRU pulse, the latch sets cl to high, which means that the data transmission will pass through this node to the next node.
- *RECEIVE*: When a latch receives a RECEIVE pulse, the latch sets dout to high, which means that the unit will receive data from the ring.
- *MULTICAST*: When a latch receives a MULTICAST pulse, the latch sets both dout and cl to high, which means that the unit will receive data from the ring as data is passing to the next node.

An example explaining how the system works is provided below (Fig. 5). In the initial state, the values of all control signals are reset to low, and all segments are available. In Fig. 5(a), for Unit 0 to send data to Unit 3 (purple section), it first sends a request with the destination ID to the central arbiter. Because the central arbiter is in the initial state, it can locate a clockwise path in Ring 0 after evaluation. To construct a path, the central arbiter sends a SEND pulse to the source node, a PASSTHRU pulse to Nodes 1 and 2, and a RECEIVE pulse to destination node. The central arbiter also sends a grant to Unit 0, allowing it to transmit data after a few cycles. As shown in Fig. 5(b), Unit 1 then sends a request to the central arbiter (orange section) to transmit data to Unit 6 while the transmission from Unit 0 to Unit 3

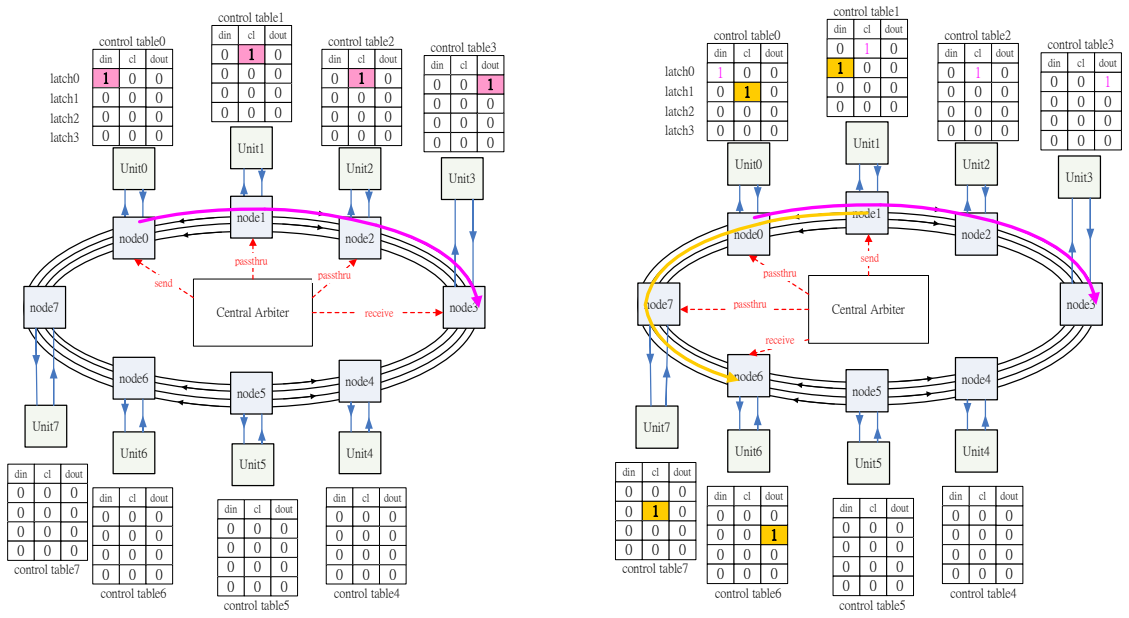


Figure 5. Behavior of the nodes and central arbiter: (a) Unit 0 sends data to Unit 3; (b) Unit 1 sends data to Unit 6

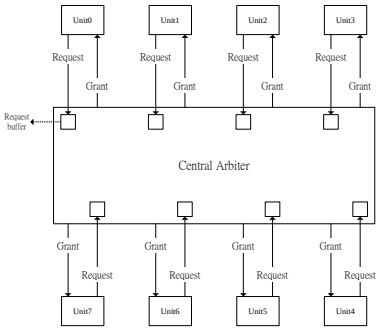


Figure 6. Connections between the central arbiter and units

is still being completed. The central arbiter can determine the paths from the three available rings, which are the counterclockwise paths in Rings 0 and 3, and the clockwise path in Ring 1. Because the paths of Rings 1 and 3 are shorter than the path of Ring 2, the central arbiter chooses the path of Ring 1 and sends SEND, PASSTHRU, PASSTHRU, and RECEIVE pulses to Nodes 1, 0, 7, and 6, respectively.

2.3 Central Arbiter Architecture

Fig. 6 shows the architecture of the central buffer, including eight request buffers that correspond to the eight units. These request buffers are served by the central arbiter in a round-robin manner. If a request buffer has failed three times with the same request, it is allocated the highest priority in the next cycle; this principle was designed to prevent starvation.

Arbitration is conducted by the central arbiter, as shown in Fig. 7. The central arbiter possesses a ring arbiter and four segment arbiters, which correspond to the four rings. Segment arbiters evaluate the availability of corresponding rings, and forward their result to the ring arbiter. After the ring arbiter selects the shortest available path from the results of the segment arbiters, it replies to the segment arbiter with a ring-chosen signal while sending a grant to the source unit. As a segment arbiter receives a ring-chosen signal, the segment arbiter sends pulses to the source node, destination node, and nodes along the path, as shown in Fig. 5. In Fig. 7, Node 0 is connected with four pulse signals: Pulses 00, 01, 02, and 03. These signals are also shown in Fig. 4. When Node 0 receives Pulse 00, it updates Latch 0 of Control Table 0, and Pulse 01 updates Latch 1 of Control Table 1, and so on. In the implementation for this study, each segment arbiter possessed eight pulse signals connected to segment arbiters. However, for clarity, not all signals are shown in Fig. 7.

Fig. 8 shows the flow of a request in the central arbiter. Initially, a request waits in the request buffer for selection by the round-robin module, where it may be moved into the destination busy table. The destination busy table maintains a list of destinations that are reserved by selected requests and data transmissions, and filters out requests to destinations that are busy to avoid destination contention before arbitration. If a request does not have the same destination as the requests already transmitting, it passes through the destination busy table and enters the segment arbiters. Each segment arbiter maintains a segment table that records the usage status of the corresponding ring (Fig. 9). In the segment table, the values of S0 to S7 represent each segment's

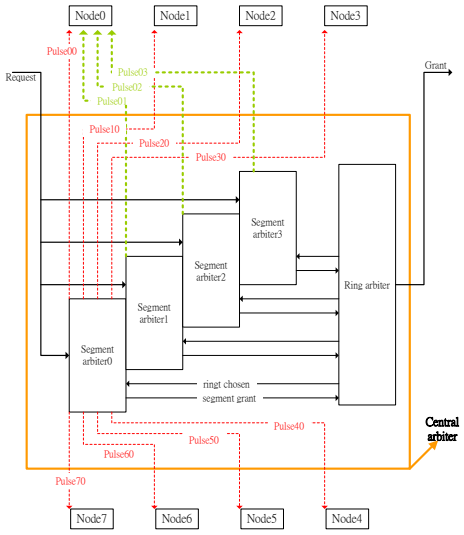


Figure 7. Central arbiter overview

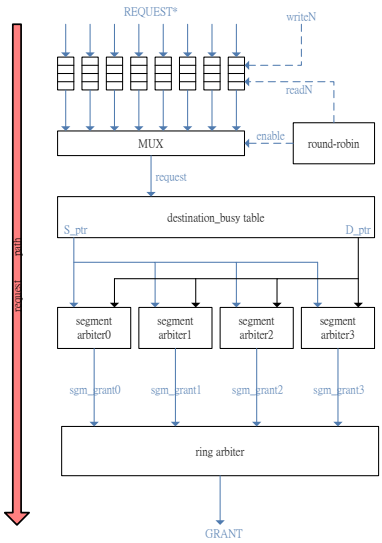


Figure 8. Request flow in the central arbiter

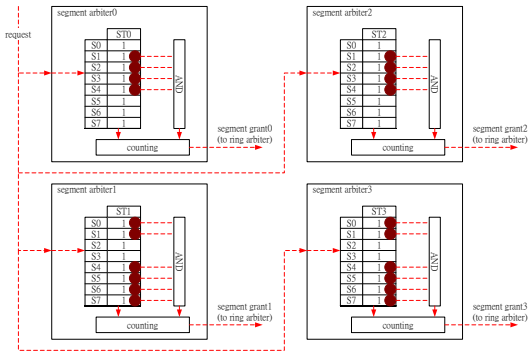


Figure 9. The segment arbiter operations when Unit1 requests a path to Unit 4.

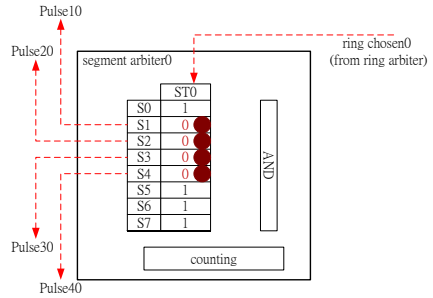


Figure 10. SA0 sends corresponding pulses and updates the segment table after receiving a ring-chosen signal.

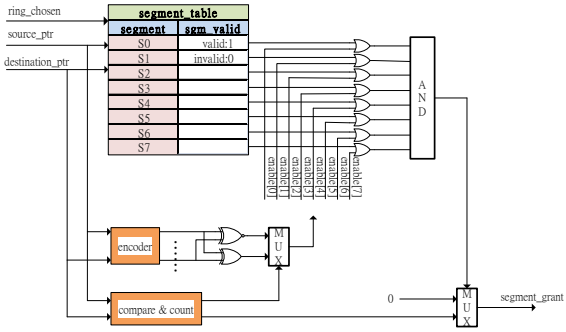


Figure 11. Segment arbiter logical architecture

usage status in the ring. Each counting block in the segment arbiters calculate the path transmission distance, forwarding the results to the ring arbiter.

Fig. 9 shows the process of Unit 1 when it requests to send data to Unit 4. First, all states in all of the segment arbiter tables are reset to 1. When the request from Unit 1 is received, each segment arbiter checks its segment table according to the destination ID in the request. In this case, Segment Arbiter 0 (SA0) checks the values of S1 to S4 in Segment Table 0 (ST0), and Segment Arbiter 1 (SA1) checks the values of S1, S0, S7, S6, S5, and S4, which correspond to the counterclockwise path in Segment Table 1 (ST1). As denoted by the red dots in Fig. 9, the counting block of each segment arbiter counts the number of checked segment table entries, in other words, the path length. If the AND gate outputs a *true* value to the counting block, the counting block determines the path to be available, and thus, forwards the counting result (the segment grant in Fig. 9) to the ring arbiter. The function of the ring arbiter is to compare all received segment grants, select the grant with the smallest value, and send a ring-chosen signal to the chosen segment arbiter. A grant signal is also sent to the source unit simultaneously. In the implementation of this study, if multiple paths possess the same distance, the path with the lowest ring number is selected.

In the situation shown in Fig. 9, the ring arbiter chooses Ring 0 for the transmission, and sends a ring-chosen signal to SA0. Upon receiving the ring-chosen signal, SA0 sends corresponding pulses to the source node, path nodes, and

destination node. As shown in Fig. 10, Pulse 10 is a SEND pulse transmitted to Latch 0 of Node 1. Pulses 20 and 30 are PASSTHRU pulses that change the control signals in Latch 0 of Nodes 2 and 3. Pulse 40, the RECEIVE pulse, is sent to Node 4 to change the control signal in Latch 0. The segment table is simultaneously updated. All segments in the path are set to invalid; thus, subsequent requests can determine that these segments are busy. When the transmission is complete, the entries in the segment table are changed to valid, and pulses are again sent to each node.

In a segment arbiter, the source ID and the destination ID contained in a request are encoded, and the resulting outputs are used to determine which segments will be in the path. By checking the value of these segments in the segment table, the AND gate (Fig. 11) will output *true* if the path is available, and the segment grant signal will forward the counting block result to the ring arbiter.

3 Cross-Four-ring Topology

In this section, a cross-four-ring topology is proposed. Based on the four-ring topology described in Section II, this study implemented an additional function that allows transmission paths to cross rings in the same direction, thereby better using each segment. For example, in Fig. 12(a), where paths from Unit 2 to Unit 5 and from Unit 6 to Unit 1 have been constructed, if Unit 0 requests a path to Unit 3, the central arbiter will be unable to identify an available path for Unit 0. However, by allowing transmission across rings, as shown in Fig. 12(b), the central arbiter can find a path. This path passes through the nodes of Units 0 and 1 in the first ring, and of Units 2 and 3 in the second ring. To implement this function, the architecture of the nodes and the central arbiter were slightly modified, as is explained in the following two sections.

3.1 Node Architecture

This study modified a four-ring topology design into the architecture shown in Fig. 13, which allows cross-ring transmissions in the same direction. Two atomic switches are added before the input ports of the transfer switches, and each atomic switch is responsible for one direction. These atomic switches provide data paths for cross-ring transmissions and are controlled by cross-pulse signals (cr_pulse02, cr_pulse13) sent from the central arbiter. If an atomic switch is set using cross-pulse signals, the crossing status of the atomic switches is retained until the transmission is complete. Once the transmission is finished, the atomic switch returns to its original status while sending a signal to the central arbiter to reset the corresponding table.

3.2 Central Arbiter Architecture

A four-ring topology possesses four segment arbiters, one for each ring (Fig. 7). SA0 and Segment Arbiter 2 (SA2) manage clockwise rings, and the usage status of clockwise rings is maintained in ST0 and Segment Table 2 (ST2). However, in a cross-four-ring topology, this study combined SA0, SA2, and an additional cross arbiter (Cross Arbiter 02; CA2) into Arbiter 02, as shown in Fig. 14. This combination provides CA02 with the usage status of all clockwise rings, which is maintained by ST0 and ST2, enabling it to determine the cross-ring transmission path in these rings. The function of counting blocks in SA0 and SA2 remains the same (i.e., to count the path length and generate segment grants). The counting block of CA02 serves a similar function and also generates cross grants when cross-ring transmissions are possible; the ring arbiter can determine that the transmission path uses multiple rings when receiving a cross-grant. Cross-Table 02 (CT02) inside CA02 maintains the status of the atomic switches, and CT02 sends Cross-Pulse 02 (cr_pulse02) to the atomic switch when a cross-ring transmission is selected by the ring arbiter. This study only describes the clockwise portion because the counter-clockwise portion is organized identically.

Fig. 15 shows the operation of a cross arbiter in the situations presented in Figs. 12(a) and 12(b), in which Unit 0 requests a path to Unit 3. Initially, the request is received by CA02, SA0, and SA2, but only Segment 0 (S0) and Segment1 (S1) are valid in SA0, and Segment 2 (S2) and Segment 3 (S3) are valid in SA2. However, CA02 can use S0 and S1 in Ring 0 and S2 and S3 in Ring 2 to construct a path. For this purpose, CA02 labels S0 and S1 in ST0, S2 and S3 in ST2, and Node 2 in CT02, which is where the cross-transmission occurred (top image in Fig. 15). The counting block then sends Cross Grant 02 to the ring arbiter. If the only available path is the cross transmission path, the ring arbiter will send a ring-chosen signal to CA02. After receiving a ring-chosen signal, CA02 updates the labeled segments to 0, which indicates that the segment is busy (bottom image in Fig. 15). The labeled node in CT02 is also updated to 0, which indicates that the atomic switch in this node is in a cross-transmitting state. Simultaneously, corresponding pulses are sent to the transfer switches in Nodes 0, 1, 2, and 3, similar to that for the four-ring topology. In addition, cr_pulse02 is sent to the atomic switch of Node 2 to set the switch. Thus, a cross-transmission path from Unit 0 to Unit 3 is constructed.

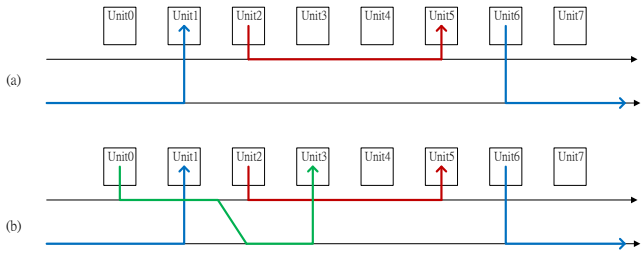


Figure 12. (a) Two data stream paths are constructed from Unit 2 to Unit 5 and from Unit 6 to Unit 1; and (b) a request for a path from Unit 0 to Unit 3 can be provided through cross-ring transmission

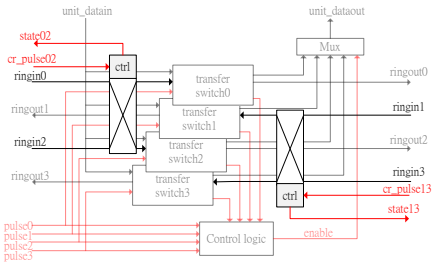


Figure 13. The node architecture of a cross-four-ring topology

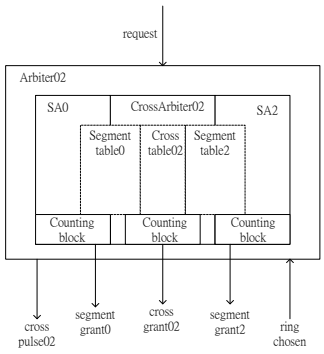


Figure 14. Arbiter architecture to control the clockwise ring in a cross-four-ring network

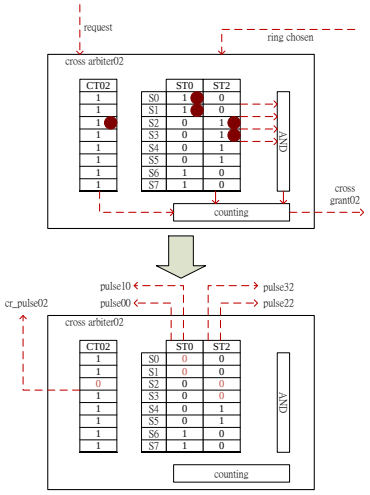


Figure 15. Operation of a cross arbiter when clockwise rings are in the situations shown in Figs. 12(a) and 12(b), i.e., Unit 0 requests a path to Unit 3.

4 Performance Evaluation

In this section, this study evaluates the performance of a two-ring interconnection network, four-ring interconnection network, and cross-four-ring interconnection network under different traffic patterns. The two-ring topology implemented in this study possesses one clockwise ring and one counterclockwise ring. The system architecture and performance analysis was constructed using Verilog HDL.

4.1 Experiment Environment

Each evaluated network had eight units. In the simulation, the first 6000 cycles are the warm-up phase, and packet latency is measured in stable time from Cycle 6000 to Cycle 30 000. The total simulation time was 60 000 cycles. Latency is measured from the time the request is produced by the traffic generator to the time the data traverse through all nodes in the path and includes the wait for arbitration in the injection queue. The journey time for a node is assumed to be one cycle. The injection rate is defined as the average number of flits injected by each node per cycle, and the number of flits in a packet is specified in the data length field.

The following traffic patterns are used for performance evaluations:

- *Random destination*: Each source sends packets to each destination with equal probability.
- *Neighbor node destination*: Each source sends packets to neighboring units; for example, Unit 1 sends packets to Units 0 or 2.
- *Half-ring destination*: Each source sends packets to the farthest unit; for example, Unit 0 sends packets to Unit 4 and Unit 1 sends packets to Unit 5 in an eight-unit ring network.

4.2 Simulation Result

The top image in Fig. 16 shows the simulation results for the two-ring interconnection network (2R), four-ring interconnection network (4R), and cross-four-ring interconnection network (Cross) under random destination traffic. The two-ring topology network exhibited the worst performance because it used the least resources. The results also show that the performance of a cross-four-ring topology network is slightly superior to the performance of a four-ring topology network. In the bottom image of Fig. 16 the percentage of requests that uses cross-functions (cross-rate) is shown. The cross-rate values are approximately 1% to 2% when the injection rate exceeds 0.1; this indicates that cross-functions are infrequently employed with random destination traffic. This is most likely because the random distribution of used segments increases the difficulty of establishing a path, even when cross-functions are enabled.

The top image in Fig. 17 is the simulation result under half-ring destination traffic. The results show that the performance of the cross-four-ring interconnection network was superior to that of the four-ring interconnection network because certain paths can be constructed using cross-functions but cannot be established in a four-ring topology. The results in the bottom image of Fig. 17 show that the maximum cross-rate can reach to 50%.

Fig. 18 shows the simulation results of neighbor destination traffic, in this case, the curves of the four-ring interconnection network and the cross-four-ring interconnection network overlap. Because a unit only communicates to the following or previous nodes in a neighbor destination traffic

pattern, the only reason a request cannot be served is that the request has the same destination as another request, and thus, is filtered out by the destination busy table. Under these circumstances, cross-transmissions never occur; therefore, the cross-rate in such cases is always 0.

Based on these simulation results, theoretically, when the cross-rate increases, the cross-four-ring interconnection network exhibits a superior performance.

5 CONCLUSION

This study implemented a circuit switching four-ring interconnection network using Verilog HDL. To construct a path in this network, a unit sends the central arbiter a request for unused segments and can successively transmit data after the path is constructed. The circuit switching design reduces the transmission latency, ensures the data is continuous and reliable, and prevents conflicts between data flows. However, path construction in the central arbiter requires additional computation time, and a number of the unused segments that cannot be constructed into a path are wasted because existing paths are not shared. To resolve this issue, this study proposed a cross-four-ring topology that allows unused segments to be constructed into a path between various rings in the same direction. This study evaluated the performance of these architectures under differing traffic patterns. The simulation results show that the cross-four-ring interconnection network achieves a comparatively superior performance under half-ring destination traffic. Furthermore, a positive correlation between the performance and cross-rate was observed.

6 REFERENCES

[1] Schaller,R.Moore's law: past, present and future *Spectrum, IEEE, IEEE*, **1997**, *34*, 52-59

[2] Hemani, A.; Jantsch, A.; Kumar, S.; Postula, A.; Oberg, J.; Millberg, M. & Lindqvist, D.Network on chip: An architecture for billion transistor era *Proceeding of the IEEE NorChip Conference*, **2000**, *31*

[3] Dally, W. & Towles, B. Principles and practices of interconnection networks *Morgan Kaufmann*, **2003**

[4] Chen, T.; Raghavan, R.; Dale, J. & Iwata, E. Cell broadband engine architecture and its first implementation—a performance view *IBM Journal of Research and Development, IBM*, **2007**, *51*, 559-572

[5] Kahle, J.; Day, M.; Hofstee, H.; Johns, C.; Maeurer, T. & Shippy, D. Introduction to the Cell multiprocessor *IBM journal of Research and Development, IBM*, **2005**, *49*, 589-604

[6] Kistler, M.; Perrone, M. & Petrini, F. Cell multiprocessor communication network: Built for speed *Micro, IEEE, IEEE*, **2006**, *26*, 10-23

[7] DeveloperWorks, I.Meet the experts: David Krolak on the Cell Broadband Engine EIB bus **2005**

[8] Ainsworth, T. & Pinkston, T. Characterizing the Cell EIB on-chip network *Micro, IEEE, IEEE*, **2007**, *27*, 6-14

[9] Duato, J.; Yalamanchili, S. & Ni, L. Interconnection networks *Morgan Kaufmann*, **2002**

[10] Tala, D. Verilog Tutorial *www. asic-world. com*, **2003**

ACKNOWLEDGEMENT

The authors would like to thank the support from NSC under grants 102-2220-E-007-003 and 102-2219-E-007-002, and also from MOEA under grant 101-EC-17-A-02-S1-202.

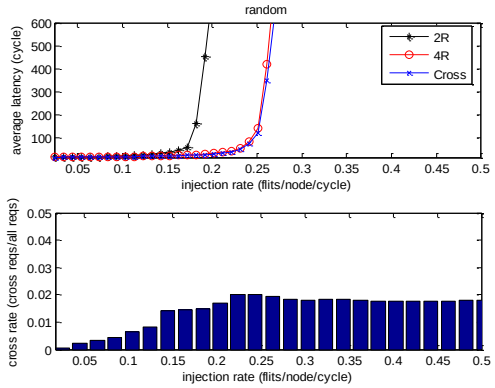


Figure 16. Results of the random destination traffic pattern

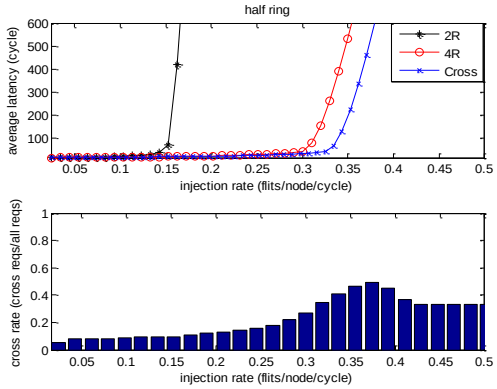


Figure 17. Results of the half-ring destination traffic pattern

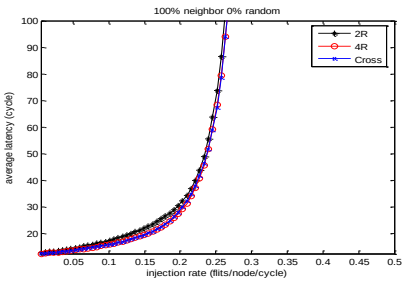


Figure 18. Results of the neighbor destination traffic pattern

Model Based DeSign method for Dynamic Configurable Ecosystem of Embedded System[1]

JungEun Cha[1], **YoungJun Jung**[1], **Chaedeok Lim**[1]
[1] Embedded Software Platform Research Team, Electronics Telecommunication Research Institute, Daejeon , KOREA

Abstract - *We intend to present design methods to construct model-based customized ecosystem that can be naturally built, evolved, and maintained through voluntary participation of various developers and users, in embedded system. To achieve this, we provide a model-based customized ecosystem capable of extending itself at a model level, wherein the course of evolution is decided according to user's feedbacks by steps of: 1) deciding development purposes and scopes of functions required in the development of an ecosystem, 2) classifying the functions 3) suggesting a scenario for building an optimum function architecture and 4) defining a meta model of each function module based on design patterns. Also, we defined the processes reflecting user feedbacks. We expect that our ecosystem is ecosystem is made flexibly adjustable at a model level according to the views of developers and users in various embedded systems.*

Keywords: Ecosystem, Configurable design, Embedded system, Design pattern, Design Process

1 Introduction

Embedded systems come to the front as highlight of IT fusion technologies, but are often troubled by increasing functional complexity, strict safety requirements. Specially, one of the biggest problems in developing and maintaining of embedded system is the variability depending on using devices, developer's purpose, and user's views. We think ecosystem is the way to solve this variability of embedded system1[1].

For example, an ecosystem may be provided to accept feedbacks directly from the users and to make up for drawbacks of distributed works, thereby enlarging the satisfaction of users so as to result in continual and voluntary participation of users and, finally, the technical prosperity. Such an ecosystem means a system that can build and evolve a natural win-win organization where all the related parties can get accomplishments through the voluntary participation of various developers and users. Therefore, the developers should build the ecosystem in order to distribute their works more easily while the users expect to get optimum services therefrom.

However, most of conventional ecosystems are made within a simple user home page level due to their focuses on developer's building easiness, or are provided in the form of an open source software project maintained by user's voluntary participation.

The first one is difficult to actively reflect user's opinion because it provides just a simple distribution function for developers; and the second one has a problem in its building costs resulted from such functions as being related with maintenance of a server or a community, security, verification, and the like. In another view, though the ecosystem is a voluntary system maintained by developers and users, it has a lot of variables such as the kind of works to be spread or the range of users, or the like. Therefore, basic functions of the system, which both developers and users can approve, are fixed by developers in the early development time, and the rest need to be adjusted and confirmed while the ecosystem is being operated[2]. Most of the conventional ecosystems provide only such services as being related with the basic function fixed early by the developers, the utilization of users is too much restricted, and a link to an external system or addition of functions requested by users, which can occur in the future, may be disabled or difficult[3].

Our goal is providing a model-based customized ecosystem capable of guaranteeing evolution without errors, wherein the architecture of the ecosystem is made flexibly adjustable at a model level according to the views of developers and users in various embedded systems, which should be frequently changed or necessarily require user's participation.

So, we provide a common architecture of the model-based customized ecosystem in accordance with the embodiment of the present methods. And suggested a method for designing a model-based customized ecosystem by describing: 1) a process deciding a scope and a scenario consisting of ecosystem functions, 2) steps reflecting user feedbacks and participating by users into existing ecosystem and 3) methods of defining a common meta model and constructing a design template for applying the organized common function architecture of the ecosystem.

[1] This work was supported by the IT R&D program of MKE(Ministry of Knowledge Economy)/KEIT(Korea Evaluation Institute of Industrial Technology), grant NO. 10041332.

2 Related Studies

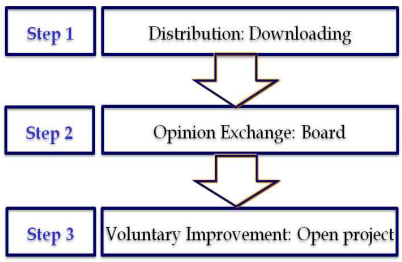
2.1 Function Stages in Ecosystem

Fig. 1 shows a block diagram illustrating the 'function stages'. The function stages support a detailed confirmation for the ecosystem service functions included in each step by mapping the scopes of functions, under the status extended to the related functions, with abstracted function groups.

The function stages sequentially include three stages of distribution, opinion exchange and voluntary improvement. In the distribution stage, easy accessing/downloading functions and user management functions are provided as main services for the purpose of wide-spread of released files. This may be enabled by making up for a function of homepage being currently operated[4].

In the opinion exchange stage, a bulletin board function is a main service to enable the exchange of opinions among developers and users. Herein, basic tools are provided, with which main progresses of projects are posted in view of the developers, various opinions are exchanged about products, technologies, and operating methods and users can make voluntary co-documents. Further, those functions capable of retrieving shared codes already released, providing suitable meta information about them, and managing version thereof are supported. Most of all, it is important to activate a community of sharing each other's knowledge by introducing such an idea of 'a question and answer corner' used in a portal site.

The voluntary improvement stage includes a series of services by which the users can modify, improve, and again share the released files. Therefore, it is important to provide those functions of source code review and version management of the source code so as to upload and share programs developed by users. Further, an issue tracker is necessarily required to track and settle down various issues, and plug-in or open API may be necessary for interworking with related services.



(Figure 1) Function Stages in Ecosystem

2.2 Services in Each Function Stage

Fig. 3 illustrates a table for classifying the main functions of the ecosystem. But these functionalities can change by user feedbacks, features of distributed contents or developer's purpose in ecosystem[5].

The aforementioned functions of the ecosystem are organized and classified into a basic group and an optional group so as to gradually build an ecosystem according to the development purpose. That is to say, the functions are separated into stages 1 to 3 according to the development purpose. Thereafter, the functions are classified as either a basic compulsory function or an optional correctable function in each stage and then those classified functions are combined to organize a single design scenario. For example, the main function of the stage 1 is 'Downloading', which enables users to download work files such as the embedded software platform developed by developers. The main functions of the stage 2 are 'Wiki', 'Notice', 'Board', and 'Community Revitalization'. 'Wiki' is a function for a real time co-documenting tool; 'Notice', for noticing major progresses related with a released work; 'Board', for exchanging of opinions among users about released products, developing technologies, and ecosystem operating methods; and 'Community Revitalization', for sharing knowledge with each other through the leading of an expert group composed of a professional engineer, a program developer, or the like.

(Table 1) Function Categories in Ecosystem

Stage	Basic function	Description
1	Downloading	• Provoding files approved to release after confisuration management
2	WIKI	• Co-documenting tool by voluntary participation of users
	Notice	• Notice of various information related to managed contents and ecosystem operating information
	Board	• User opinion exchanges about released contents, developing technologies, ecosystem operatinf methods, etc.
	Community connectivity	• Learning and sharing technologies by leading od expert groups
2.5	Reusing Service	• Retrieving shared contents(codes, documentation, video, etc.) and providing understand information for contents
	Repository management	• Configuration management of shared assets and management of user information
3	Configuration management	• Storing source codes and managing source code versions
	Code review	• Level 1: reviewing codes uploaded by developers • Level 2: reviewing codes for cooperating among developers
	Issue tracker	• Tracking processes of modification and improvement resulted from various user opinipns, especially, for released contents
	Open API	• Supporting link to various external servies
	Plug-in	• Supporting addition od new functions or interworking with other tools

The main functions of the stage 2.5 are 'Reusing Service' and 'Repository Management', wherein the first one means a function of enabling users to retrieve shared codes and provide meta information and the second one means a function for shared asset configuration management and user management.

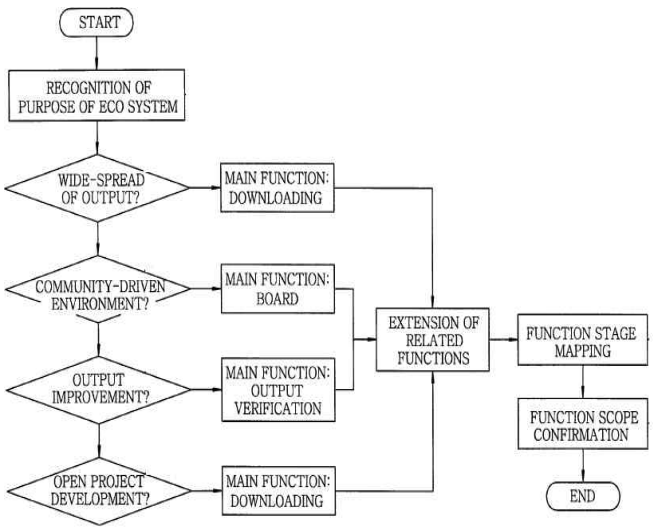
The main functions of stage 3 are 'Source Code Management', 'Code Review', 'Issue Tracker', 'Open API', and 'Plug-in'. 'Source Code Management' is a function for storing source codes and managing versions thereof; 'Code Review', for reviewing codes of developer-uploading programs or those resulted from the co-works among developers; 'Issue Tracker', for managements such as tracking, retrieval, and person-in-charge allocation for bugs of the released works and user requests, or for opinion exchanges in association with the bulletin board; 'Open API', for linking to various external services; and 'Plug-in', for adding new functions or interworking with other tools.

3 Model-based Customized design method

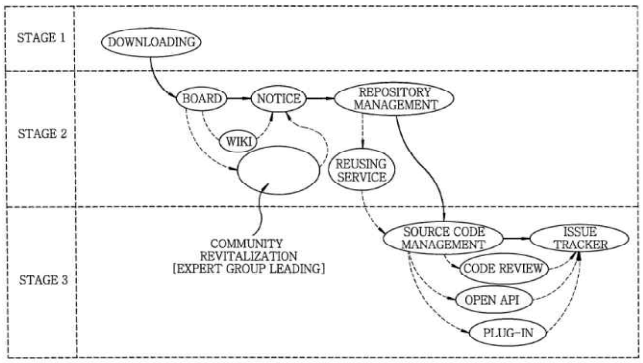
3.1 Process of deciding a scope of functions of an ecosystem

The scope of functions of the ecosystem is determined according to building purposes of the ecosystem by developers in the early development time, and may be gradually changed according to the intentions of participating users and operation purpose of the ecosystem.

Figure 1 is shown the process for defining function scope by considerations happened from purpose and expending strategy of ecosystem. The purpose of the ecosystem should be recognized before anything else to build the initial stage of the ecosystem. In this embodiment, there are four purposes exemplified, including a wide-spread of output, a community-driven environment, an improvement of output, and an open project development (or voluntary improvement).



(Figure 2) Process for defining function scope



(Figure 3) Function lineup scenario for building the ecosystem

Based on each purpose, a main function is defined as 'Downloading' for both purposes of the wide-spread of output and the open project development, 'Board' for the purpose of the community-driven environment, and 'Output Verification' for the purpose of the improvement of output (the developed works). Then, related functions accompanied by the main functions are extended, and final scopes of functions are confirmed through a function stage mapping.

Especially, "functions stage mapping" in figure 2 are connected to function steps in figure 1, then mapped function stage is embodied into ecosystem including concrete functions described in table 1.

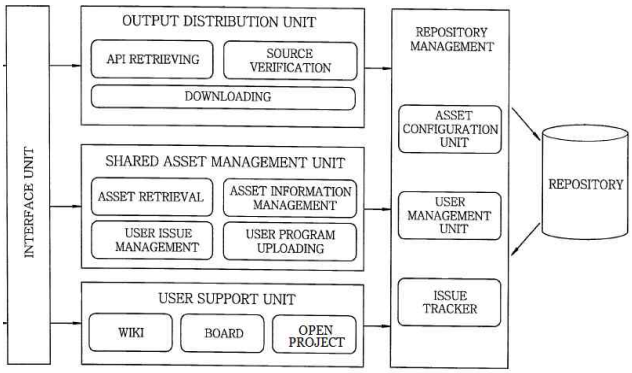
3.2 Scenario for Function Definition of Ecosystem

Figure 3 sets out a conceptual view illustrating a function lineup scenario for building the ecosystem in accordance with the embodiment of the present invention, wherein solid lines show the connections of basic functions while dotted lines show those of optional functions[6][7].

In case of building an ecosystem of stage 3, developers must make the system include: the basic function 'Downloading' of stage 1; the basic functions 'Board', 'Notice', and 'Repository Management' of stage 2; and the basic functions 'Source Code Management' and 'Issue Tracker' of stage 3.

The ecosystem may additionally include optional functions in stage 2, i.e., 'Wiki' as a user co-documenting function, 'Community Revitalization' as a knowledge sharing function by experts, or 'Reusing Service' for the shared asset (or common source) managed by the ecosystem, according to the range of users and the utilization methods of the ecosystem. Also, the ecosystem may additionally include optional functions in stage 3, i.e., 'Code Review' for the developer-uploading programs or the co-works among developers, 'Open API' for linking to various external services, and 'Plug-in' for adding new functions or interworking with another tools.

Based on the aforementioned function architecture of the ecosystem, a common meta model can be provided. To this end, first, a common architecture of the ecosystem will be defined with reference to Fig. 3.



(Figure 4) Common Architecture of Ecosystem

4 Ecosystem Design

4.1 Common Architecture

Fig. 4 provides a view illustrating a common architecture of the model-based customized ecosystem in accordance with the embodiment of the present invention.

Referring to Fig. 4, the ecosystem includes: a work distribution for distributing developed works such as embedded software platforms to users; a shared asset management for performing a voluntary reuse of the developed works; a user support unit 508 for gathering and reflecting user's opinions about the distributed works; and a repository management for managing information on the developed works.

The work distribution enables a plurality of users, who access the ecosystem via a network such as Internet, to download the developed works such as the embedded software platforms through an interface. Further, it performs API test or source verification for the developed works. The shared asset management unit stores, retrieves, and manages the works constituting various embedded software platforms, the information thereof, and users' access information with respect to the embedded software platforms. Further, the shared asset management enables the users to upload a modified version of the developed works. The user support provides such functions as Wiki, Board, and OPEN Project that enable exchanges of opinions among users, so that a plurality of users who download the developed embedded software platforms can express their opinions about the platform and those opinions can be reflected for the improvement of the platform. Open Project is one of the most important modules to support for evolving shared contents into various configurations outputs by voluntary users. The repository management stores all the works and information managed by the ecosystem in a repository, updates them, and maintains openness thereof. An asset configuration unit manages source codes and version information of the developed works such as embedded software platforms. A user management unit manages information on users who download the developed work or offer opinions thereabout. The issue tracker reflects improvements of the developed work.

In detail, the issue tracker collects opinions of various users about the developed works such as embedded software platforms from the user support unit, catches bugs and requests for modification of the developed works, and make the bugs and requests be reflected when a corresponding developed work is modified. Further, it makes the embedded software platforms be linked with an external service by applying an open API thereto, or enable the platforms to be added by new functions or be interworked with another tools by applying plug-in.

4.2 Reflecting Process of User Feedbacks

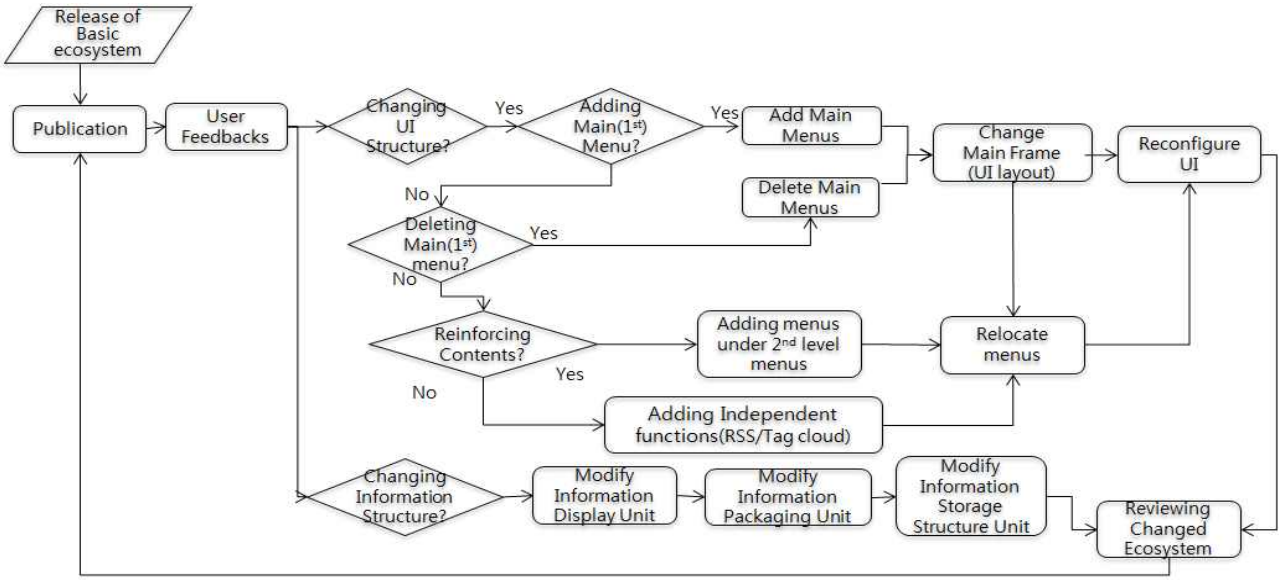
After opening basic ecosystem constructed for distributing initial outputs, users are used to suggesting their opinion related on ecosystem. Then, UI structures or data processing structures of ecosystem are changed by the scope of user feedbacks.

As like figure 5, if users want to change to user interface structures, activities for menus addition and deletion in level of main menu are required, or menus under second level are modified (addition or deletion) for reinforcing shared contents, unless change UI structure. In the case of no changes for UI structures, independent menus are adding as like RSS, tweets and tag cloud, etc. And ecosystem UI acquires requisite space for UI changing by relocating UI menus, and then it's main frame (UI layout) modified and overall UI structure is reconfigured. If user feedbacks are required to change information structure, information display units of ecosystem is modified and these units are repackaged into similar information categories. Lastly, changed ecosystem by reflecting user feedbacks is re-leaded and performed continuous feedback reflection cycles.

4.3 Dynamic Design Template

Figure 6 illustrates design models of defining a common architecture and constructing a design template for applying the organized common function architecture of the ecosystem. We intended these design model can help to easy maintaining ecosystem regardless of changed objects(classes) by applying user feedbacks into UI abstract layer[8][9].

This model is divided into the UI, storage management and main function part of ecosystem. For UI part is deep connected to user feedbacks, so we tried to define the design model to supporting easy changing and dependency minimizing among related modules by separated implementation. We adopted design pattern Abstract Factory from GoF patterns, this pattern including 3 classes (IndependentUIModifying, FirstMenuModifying, SecondMenuModifying) is used to compose different UI by 3 classes when request for UI changing is different each other. Also for performing menu addition and deletion without any conditions, we applied Iterator pattern. This pattern makes class MenuInstance to be independent.



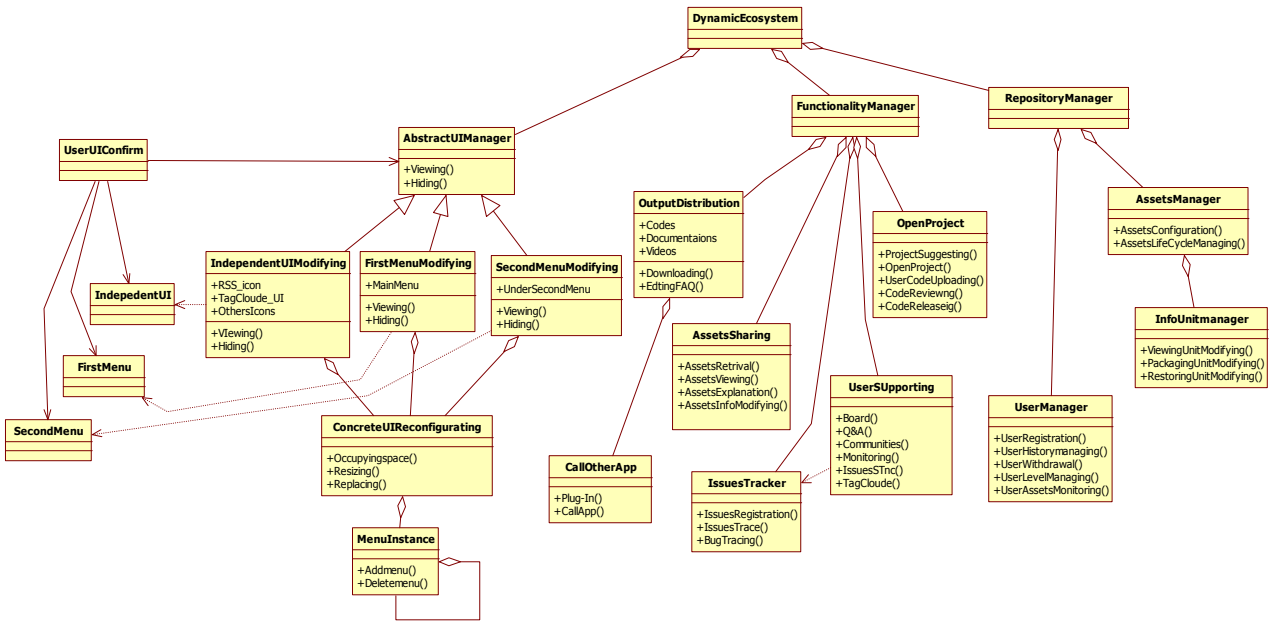
(Figure 5) Process for Reflection of User Feedbacks

In our model, overall classes are structured by Aggregation relationship to support for selecting detail implementation methods according to user feedback's type and adding or deleting the operations in same class. Therefore, because higher level classes include the lower level classes to add the detail implementation, functions of implemented ecosystem depend on selected one of classes with aggregated relationship

At this point, an UML-based meta model can build a flexible ecosystem by adjusting core attributes in each function module as well as selecting each function module.

The model-based customized ecosystem and design method thereof have an advantage in enhancing the efficiency of a system development, by organizing an optimum ecosystem suitable for the purposes and strategies of developers using a common design template, which is composed of meta models that define a function classification depending on the purposes of the ecosystem, a function design scenario, and function modules.

Further, it has another advantages in maximizing efficiency and flexibility of the ecosystem as well as the spread and improvement of the distributed works via the ecosystem,



(Figure 6) Dynamic Design Model for Ecosystem

because it can be voluntarily evolved to have versatile functions according to patterns of accessing the ecosystem and varying requests of users, by using the meta models That is to say, the architecture of the ecosystem is made flexibly adjustable at a model level according to the views of developers and users in various embedded systems, which should be frequently changed or necessarily require user's participation, thereby guaranteeing efficient development, and errorless extension and link of the ecosystem.

6 References

[1] Bosch, Jan., From Software Product Lines to Software Ecosystems. Accepted for SPLC 2009 (13th International Software Product Line Conference), August 2009 [http://www.software-ecosystems.com/Software_Ecosystems/Ecosystems.html]

[2] Jim Highsmith, Agile Software Development Ecosystems, Addison-Wesley Professional, 2022

[3] Leffingwell, Dean, Scaling Software Agility Best Practices for Large Enterprises, 2007, Addison-Wesley

[4] Karl Popp, Ralf Meyer, Profit from Software Ecosystems: Business Models, Ecosystems and Partnerships in the Software Industry, 2010, Synomic GimH

[5] David G. Messerschmitt, Clemens Szyperski, Software Ecosystem: Understanding an Indispensable Technology and Industry, 2005, MIT Press

[6] Jorma Taramaal, Munish Khurana2., and Pasi Kuvaja2, "Product-Based Software Process Improvement for Embedded Systems", EUROMICRO '98, vol. 2, pp.20905, Vol. 2 August 25~ 27, 1998 http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00708120, [accessed June 14, 2009]

[7] Bernd Hardung, Thorsten Kölzow, and Andreas Krüger, "Reuse of Software in Distributed Embedded Automotive Systems ," EMSOFT'04, Pisa, Italy, vol. A247, pp. 203 - 210, September 2004

[8] P. Clemens and L. Northop. Software Product Lines - Practices and Patterns. Addison-Wesley, Boston, 2002.

[9] Erich Gamma, Ralph Johnson, Grady Booch, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 2006

LUCKY THIRTEENS: A NEW BREED OF MICROPROCESSOR BOARDS

William A. Stapleton
Ingram School of Engineering, Texas State University – San Marcos
601 University Drive, 5202 Roy F. Mitte Building, San Marcos, TX 78666 USA
wstapleton@txstate.edu, Voice: (512) 245-8746, FAX: (512) 245-7771
The 2013 International Conference on Embedded Systems and Applications

ABSTRACT

Within the last few months a new breed of small microprocessor trainer boards had emerged onto the marketplace. Boards such as the Freescale Freedom, Texas Instruments Stellaris Launchpad, and ST Micro Discovery are each entry-level ARM-based boards available at a common price point: $13. These boards offer an introduction to a modern 32-bit architecture with state-of-art tools with little barrier to entry for the novice. Consequently, these boards are offering a tremendous opportunity for adoption into a wealth of Embedded Systems curricula.

Keywords: embedded systems, engineering education

1. INTRODUCTION

Selecting a microprocessor architecture for teaching embedded systems fundamentals has long offered a difficult set of compromises. Selecting an architecture such as Intel's latest 80x86 variants, the i3, i5, and i7 cores, certainly offers students to opportunity to learn one of the most widely used architectures on the market. Unfortunately, for a novice student, learning a sufficient portion of Intel's architecture within one, or perhaps two, semesters is daunting to the point of impossible. Many programs seek the other end of the spectrum with architectures such as MIPS which are very simple and straightforward but exist primarily in simulation. Another simple architecture which has become popular is the Arduino microprocessor based around the Atmel Atmega architecture. This is a simple architecture but the boards are generally presented through a simplified set of programming tools. The author has experience with programs which selected the Freescale MC9S12 architecture as an exemplar with the intent of balancing student experience to present a commonly used architecture which is not excessively complex to learn [1-7]. Unfortunately, the MC9S12 is little changed in the last two decades. Many programs have preferred ARM-based architectures for having a clean RISC-type architecture which has been steadily gaining market share. Until recently, the primary limitation of the ARM architecture was finding a suitable low-cost trainer. Within the last few months, at least three companies have released entry-level ARM-based boards at roughly the same price point: $13. With the introduction of these boards there is little remaining limitation to adopting a modern microprocessor architecture.

2. THREE EXAMPLE BOARDS

The first exemplar of the "lucky thirteens" is the Freescale Freedom [8]. This board is based around a member of Freescale's Kinetis line of ARM® Cortex™-M0+ processors. Freescale offers a special edition of the CodeWarrior software optimized specifically for this board for free on their website [9]. The Freescale Freedom is depicted in Figure 1.

The second exemplar of the "lucky thirteens" is the Texas Instruments Stellaris Launchpad [10]. This board is based around a member of Texas Instruments' Stellaris line of ARM® Cortex™-M4F processors. Texas Instruments offers a special edition of the Stellarisware and Code Composer Studio software optimized specifically for this board for free on their website [11]. The Texas Instruments Stellaris Launchpad is depicted in Figure 2.

The third exemplar of the "lucky thirteens" is the ST Micro Discovery [12]. This board is based around a member of ST Micro's line of ARM® Cortex™-M4 processors. St Micro offers sample firmware of applications on their main product page but leaves the choice of development tool to the user. The ST Micro Discovery is depicted in Figure 3.

Figure 1: Freescale Freedom

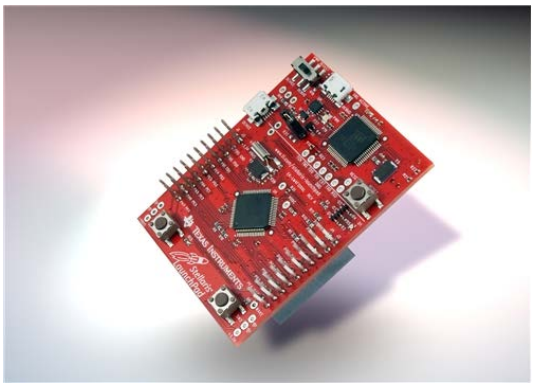


Figure 2: Texas Instruments Stellaris Launchpad

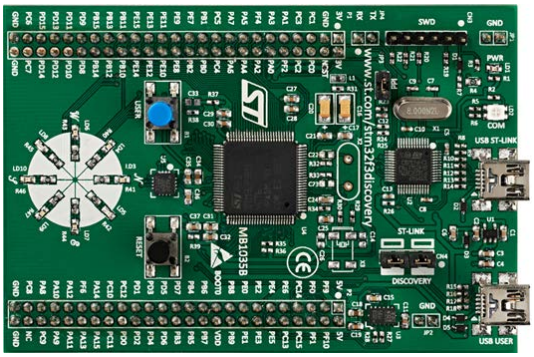


Figure 3: ST Micro STM32F3DISCOVERY

Each of these three exemplar boards offers an excellent set of capabilities. Any one which is selected should provide an excellent platform for teaching microprocessor architecture and embedded systems applications.

The author has considerable past experience with the Freescale MC9S12 architecture and embedded software development in an academic setting [1-7]. What many consider the premier tool for MC9S12 development is the CodeWarrior software platform. Freescale made CodeWarrior extremely desirable for this task within academia when they released the free "special edition" for MC9S12 some years ago. When Freescale released the ARM-based Freedom platform along with a new free special edition for the Kinetis ARM microprocessor, they positioned themselves nicely to give academia a clean upgrade path. Further, the physical layout of the Freedom board mirrors the Arduino Uno, one of the best-selling microprocessor boards of the last decade. Specifically, the position of the I/O headers on the Freedom boards allows it to immediately connect to any of the hundreds of Arduino-compatible "shield" peripherals [13]. Because the author intends on adopting the Freescale Freedom as a replacement for the MC9S12 platform, all examples in the following sections will utilize the Freedom when specifics are necessary but any of the boards should be capable of operating in any of the examples. Any exceptions will be noted.

3. PROGRAMMING THE BOARDS

All three of the exemplar boards are equipped with two USB ports. One port is directly tied to the microprocessor, either to native USB or through a hardware USB-to-RS232 connection to native RS232. The second USB port is connected to some form of on-board background debugger. In the case of the Freedom, the background debugging functions are handled by a second, smaller ARM CPU adhering to the OpenSDA standard.

Programs for the boards are written in the language of choice for the user. The author has verified both C and assembly options for these boards. Once programs are written, binary files are created and

subsequently uploaded into the boards via the background debugging facilities. In the case of the Freescale boards, the binary files are in the S-record format. CodeWarrior can directly program the boards with these S-record files or users can upload them manually.

4. BUILDING SOFTWARE FOR EXISTING HARDWARE

The Arduino microprocessor boards have become fantastically popular by providing a very simple software platform to create simple applications. These are a good choice for novice hobbyists but have a limited top end of their capability. Their popularity has spurred the creation of a large number of peripheral devices all using a common physical interface, the "shield" [13]. Because the Arduino is based on a simple, older architecture with limited I/O, most of the shield peripherals were developed with either a standard serial interface or direct control via a small number of I/O pins. On the Arduino, hardware support for these interfaces is limited and even simple serial standards such as RS232 may be implemented in software via "bit banging".

Any of the three "lucky thirteen" boards provides considerably more native I/O capability than an Arduino including hardware support for standard serial interfaces such as RS232, SPI, and IIC; hardware pulse-width modulation; and analog-to-digital and/or digital-to-analog. What these boards lack, at this point, is a library of software functions to help the novice user readily utilize this superior hardware for the same types of applications that the simple software library of the Arduino provides.

The author previously developed and published a series of assembly language libraries for the MC9S12 aimed at typical embedded systems applications [1]. The equivalent of these libraries will be developed as the author adopts the Freedom board. Newer portions of the libraries are planned to allow simple access to popular shield peripherals. The next section describes some examples of items that the author believes will become common applications as teaching examples with these new microprocessor boards.

5. COMMON APPLICATIONS SUITABLE FOR TEACHING

When utilizing any microprocessor for training purposes, one of the first tasks which students must learn to program is a technique for input and output of text based data. Since these microprocessor ttrainers have no hardware "console" as such, most use a host PC as a terminal, typically either via RS232 or virtual RS232 over USB. Writing routines for console-like user I/O is often inspired by the C/C++-language routines such as *getc*, *putc*, *scanf*, *printf*, *cin*, or *cout*. A significant number of peripherals can be controlled using only an RS232 port. Each of the three "lucky thirteen" boards offers at least three independent hardware RS232 channels. Many older architectures such as the Atmel ATMega on the Arduino or the MC9012 offer only one or two hardware RS232 ports and additional ports must be simulated in software.

The Inter-Integrated Circuit (IIC) and Serial Peripheral Interconnect (SPI) standards also each allow connection to a wide variety of peripherals. RS232 is a two-device peer-to-peer standard. Both IIC and SPI can connect multiple devices on the same bus. RS232 is asynchronous while both IIC and SPI are synchronous. Including lessons on one or both of these communications standards opens a large number of options to the student. All three of the ARM-based boards have hardware support for one or more of each of these ports.

One of the most popular problem realms for both novice hobbyists and for teaching students is utilizing microprocessor trainers for simple robotics. In addition to the more general user I/O tasks, robotics requires the control of various motor types for actions and the input of a variety of data from numerous sensor types for managing control. On the sensor input side of this equation, in addition to serially-interfaced sensors, mot other sensors produce analog outputs which must be

loaded into the microprocessor via some analog-to-digital path. All of the three exemplar boards contain multiple internal analog-to-digital channels. In fact, all three offer a larger number of channels with higher sampling rates and greater bit depth than older architectures such as the ATMega of the Arduino or Freescale's MC9S12. However, the boards do not include internal digital-to-analog, relying instead on external drivers.

Two common motor types, servo mtors and dc motors, both utilize pulse-width modulation (PWM) for control. Servo motor position is controlled by a 50 Hz pulse train with duty cycle between 5% and 10%. DC motor direction is controlled by an H-bridge and motor speed is controlled by PWM. Some inexpensive platforms like the Arduino implement PWM in software. This has the disadvantage of having the timing of the PWM signal be dependent on the overall amount of processor loading. In all of the "lucky thirteen" boards, PWM is supported directly in hardware such that the user can "set and forget" a PWM signal and have an accurate output regardless of other processor loading.

6. CONCLUSIONS

The new generation of inexpensive ARM-based microprocessor trainer boards offers a tremendous opportunity for updating a wide variety of microprocessor curricula. The "lucky thirteen" boards offer both better performance ad better price than their predecessors. Even more importantly, these boards offer greater I/O capacity and full hardware support for multiple serial ports and PWM that their predecessors had to implement in software. These boards are supported with free, professional-level development tools. In short, a strong case can be made for this class of microprocessor boards becoming the "go to" solution for all entry-level microprocessor needs.

7. REFERENCES

[1] Stapleton, William, "Development Of A Library For Teaching And Implementing Resource-Limited Embedded Systems," Proceedings of The 2011 International Conference on Embedded Systems and Applications, July 2011.

[2] Stapleton, William A., "Embedded Assessment of Microcomputer Fundamentals for Embedded Systems Education", The 2008 International Conference on Frontiers in Education: Computer Science and Computer Engineering, Las Vegas, NV, July 2008.

[3] Ricks, K.G., Jackson, D.J., Stapleton, W.A., "An Embedded Systems Curriculum Based on the IEEE/ACM Model Curriculum," IEEE Transactions on Education, Vol. 51, Issue 2, pp. 2262-2270, (2008) ***– selected IEEE Education Society 2009 Best Transactions Paper***.

[4] Ricks, Kenneth G., David J Jackson, William A. Stapleton, "Incorporating Embedded Programming Skills within an ECE Curriculum", SIGBED Review, Volume 4, No. 1, pp. 17-26 (2007).

[5] Ricks, K. G., Stapleton, W. A., Jackson, D. J., "A Focused Curriculum for Embedded Systems", Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA), Madison, Wisconsin, June 2005.

[6] Stapleton, William, Kenneth Ricks, Jeff Jackson, "Implementation of an Embedded Systems Curriculum", Proceedings of the ISCA 20th International Conference on Computers and Their Applications (CATA'05), New Orleans, LA, March 2005.

[7] Stapleton, William, "Zero to Two Hundred in Two Years: Launching a New Program", The 2010 International Conference on Frontiers in Education: Computer Science and Computer Engineering, Las Vegas, NV, July 12-15, 2010.

[8] URL: http://www.freescale.com/webapp/sps/site/overview.jsp?code=FREDEVPLA&fsrch=1&sr=1

[9] URL: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=FRDM-KL25Z&fpsp=1&tab=Design_Tools_Tab

[10] URL: https://estore.ti.com/Stellaris-LaunchPad.aspx
[11] URL: http://www.ti.com/tool/sw-ek-lm4f120xl
[12] URL: http://www.st.com/web/catalog/tools/FM116/
SC959/SS1532/PF254044?s_searchtype=partnumber
[13] URL: http://www.shieldlist.org/

Roto Disp-Pattern Generation using Microcontroller- based Hardware

Jyothi R, Dr.V.K. Agrawal, and Pradeep A.S.
Computer Science Department, PESIT, Bangalore, Karnataka, India
Information Science Department, PESIT, Bangalore, Karnataka, India
Siemens Information Systems Ltd., Electronic City,Bangalore, Karnataka, India

Abstract-*Pattern generation is computer-aided design for the display of different patterns on the wheels of the car to attract attention of prospective Customer to a business or itsa Product or services. In This palper web are présentions a system of pattern generation for entertainment and business promotion purposes with name RD-PatGen. This is an embedded microcontroller- based motorized LED graphical displaying device. The proposed software runs on microcontroller- based hardware named as Roto-Disp. A GUI is used for creation of patterns. The pattern can be generated in two modes namely, graphical mode and text mode. Graphical patterns are generated in graphical mode and ASCII characters like text patterns can be generated using text mode. In both the modes, previously stored patterns can be loaded automatically. The proposed system generates the pattern in the background using GUI. In this GUI Preview, option has been given to view generated patterns and also COM settings options for serial data transmission. As car starts moving, different patterns will be displayed on the wheels of the car.*

Keywords: Roto-Disp pattern generation, Microcontroller, LED graphical displaying device, Graphical patterns and Text patterns.

1 Introduction

In the post- global era of business process, embedded system has become a backbone for the development of many of the present- day electronic products [1]. Embedded systems find much usefulness in the automobile field. These are being primarily used for functioning of an automobile. Embedded systems are also in value- added services of automobile, like advertisement. In order to attract customers for purchase of cars, companies are using attractive pattern display on the wheels of cars. Such patterns can be easily developed by embedded system using microcontroller. We propose to use microcontroller- based system for the generation of patterns. Pattern generation software generates different patterns on the wheels of the car. We are using visual basic for the creation of graphical user interface (GUI). By making use of this GUI, customers can generate patterns of their own interest. The generated pattern is transferred to hardware by the serial interface. Once the power is on, the controller generates different prefixed patterns on the wheel of the car as shown in figure 1. This approach of advertisement is likely to increase consumption of products and services. It also leads to creation and reinforcement of brand image and brand loyalty. In this paper, we present both hardware and software aspects of the pattern generation scheme. In this scheme, we are using a microcontroller- based hardware and it is named as Roto-Disp. We call this proposed pattern generation system as RD-PatGen. With this system, different attractive patterns can be generated on the wheels of the car. Images on the wheels change automatically at different time intervals. RD-PatGen System can also be used for entertainment and other business promotion purposes.

The Section 2 provides the system description, including hardware block diagram. Section 3 discusses the software design aspects, which describes the set of data flow diagrams, context level diagrams and also the operational description for RD-PatGen. Section 4 shows the snapshots for the RD-Patgen. Section 5 presents conclusions and points out some future work.

2 System Description

Pattern generation system consists of three major subsystems. Block schematic of this system is shown in Fig.1. The first subsystem is a personal computer, also named as source unit. It is used for generating different patterns which are to be displayed on wheels. This is an off-line system i.e. the system is used during development phase. After the patterns are generated and transferred to the next subsystem, a microcontroller- based system, the role of personal computer is terminated unless the patterns are required to be modified again. The microcontroller transfers these stored patterns during run time and interfaces with display unit on tyres for the pattern display via slip ring arrangement. Design for RD-PatGen system comprises of software for user maintenance i.e. a GUI for creation and storage of patterns, development of microcontroller-based hardware and firmware for the controller. Different patterns are generated using GUI; these patterns are transferred to the microcontroller- based hardware

(Roto-Disp).

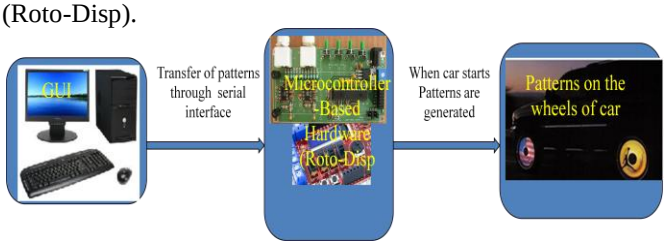


Figure 1. RD-PatGen System components.

More description of PATGEN system is given in Figure 2. As described above, pattern generation system consists of three major components namely, Source Unit, Processing Unit and Display Unit. The Source Unit is having adequate secondary memory for storing the patterns. The stored patterns are transferred offline to the Processing unit through serial interface. Processing Unit is designed based on simple microcontroller and provides interfaces to Source Unit and Display Unit. It communicates Source Unit to receive the stored data using RS 232C interface, and stores it into the EEPROM. This controller controls all the elements within the processing unit. After the required processing, the patterns are mapped to the display format and then transferred to the display unit. The transfer interface is based on slip ring arrangement. When the car starts, controller senses the rotation of wheels through proximity sensor and transfers patterns from the memory to wheels of car for display.

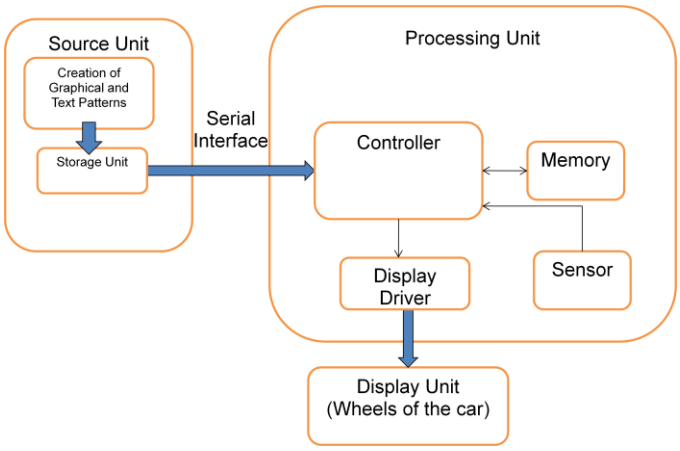


Figure 2. Block diagram for RD_PATGEN System

Length

The maximum allowed number of pages is seven for Regular Research Papers (RRP) and Regular Research Reports (RRR); four for Short Research Papers (SRP); and two for Posters (PST).

2.1 Hardware Realization

As shown in figure 3. Roto-Disp consists of a rotating blade on which arrays of 16LEDs are mounted; these LEDs are connected to a microcontroller processing unit with driving circuitry. This driving circuit is mounted on the microcontroller. An ac motor is used to simulate the rotating wheel. A magnetic proximity sensor is placed on the blade and this sensor is used to sense the reference point. Power to the circuit is provided using slip ring arrangement. Slip ring also provides the signal connectivity between processing unit and display unit. When Roto-Disp unit and motor receive power the motor starts rotating and hence the blade also starts rotating. As soon as the controller senses the rotation through the magnetic sensor, it starts controlling the 16 LEDs. This controller transfers different pre-fixed LED patterns for every 1.4degree movement of the blade. Hence for a rotation of 360 degrees approximately, 256 different LED patterns are generated. If the LED pattern sequence is selected, we can see, a distinct pattern being generated due to the property of perception of the eye. The image may consist of simple graphics or alphanumerical messages. Using this arrangement attractive pattern of images can be created on the wheels of different automobiles.

Newer patterns can be created on the PC using GUI of Visual Basic [2, 3, 4]. This software can be used at a toll to create various images and patterns on the PC. These patterns can be downloaded on the memory chip through serial port. Once the images are created, the software converts this image into hex values, which correspond to LED on/off patterns. When RD receives power, the controller transfers different pre-fixed LED patterns and displays these patterns in a timed sequential manner, and it can also store the direction of the display either in clock wise or anti-clock wise direction.

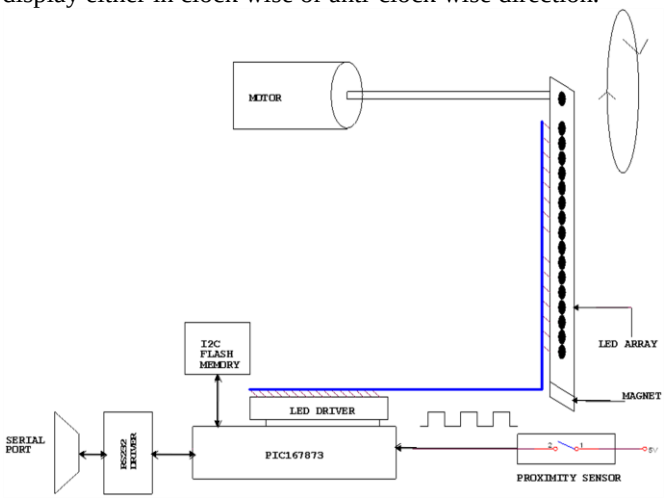


Figure 3. Block diagram of hardware display device

The whole hardware is built around very popular microcontroller PIC16F876 [5], as in figure 3. The firmware for this is developed in 'C' language using Hitech cross compiler. An array of 16 Light Emitting diodes (LED) is mounted on a rectangular strip. A power magnetic is fixed at the exterior end of the LED strip. Whenever the magnet comes near the proximity sensor, it generates a corresponding pulse. This pulse is sensed by the PIC and is recognized as

the reference point. The LED pattern information is stored in the flash memory (24LC256) interfaced externally to the PIC using I2C bus (Inter IC Communication protocol). PIC has two lines, RX (receive) and TX (transmit) dedicated for serial communication. More description about the circuit diagram which is built around the hardware device is given in Appendix A.

3. Software Design for RD-PatGen

Software design of Roto-Disp Pattern generation system gives the GUI for generation of graphical as well as textual patterns. GUI has 256 X16 pattern generation cells and 8 individual frames. The patterns can be generated in two modes, graphic mode and text mode. In graphic mode, different graphical patterns are generated based on the user requirements. In text mode, options have been given to generate text patterns, like ASCII characters. GUI provides an option for new character generation, so that user should be able to create a new pattern with a character of size 16 into 5, and eight different patterns can be generated using eight frames. Here, options are provided for generating eight such images concurrently and also options for switching between images. Images created will represent the pattern to be generated on the device. Once the patterns are created either in graphical mode or text mode, options are given to save the patterns. The saved files are loaded, edited or transferred to the hardware at any time. Options are also given to open an existing pattern. With the Edit option, user can perform changes on the GUI screen. A provision of time for generation of pattern for each image has been given in 'seconds'. Once the different patterns are generated, options are given to preview the generated pattern. Here graphical image to hexadecimal word generation algorithm have been implemented to convert the images into hex values. When power is switched on, the patterns generated by GUI are transferred to the hardware by serial interface for display and Roto-Disp has been used for the display of patterns on the wheels of the motor car.

The user switches between two modes of pattern generation during design time, depending on the convenience. User can create graphical images with a matrix 16 by 256. LEDs to be glown are shown in red and off condition are represented as black. User can select the LED condition as either red or black by clicking on the corresponding cell. In graphical mode, individual cell can be controlled with the mouse click. In text mode, previously stored character pattern can be selected by typing the corresponding character. Options are given to user to select the time for display of each pattern. The display on the grid is in a rectangular manner, but in reality, the patterns will be in a circular shape. Hence a circular simulation screen has been added; this function will read the previously generated pattern and give a corresponding circular display of the pattern. Provision has been given to user to detect serial connection. User should be able to configure the baud rate for serial configuration & also to select different available COM ports.

3.1 Data Flow Diagrams

The flow of data in RD-PATGEN is described in detail with appropriate data flow diagrams.

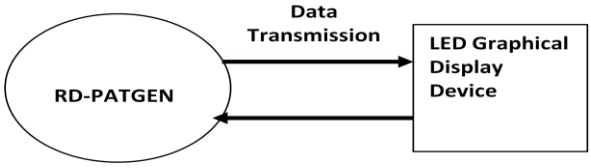


Figure 4. Data Flow Diagram for RD-PATGEN

Figure 4. Shows the data transmission to hardware device for displaying the result.

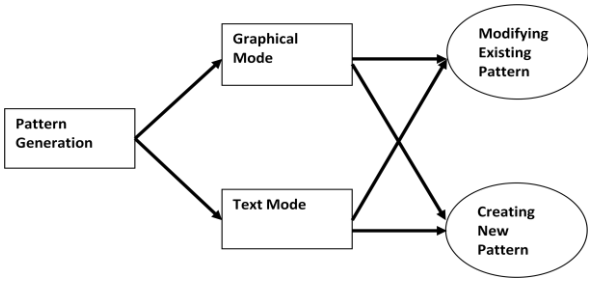


Figure 5. Data Flow Diagram of pattern generation for RD-PATGEN.

Figure 5. Shows Data flow diagram for generating Patterns, There are two modes in which different patterns can be generated. Graphic mode and Text mode. RD-PATGEN can generate eight such types of different patterns, a time delay also is provided.

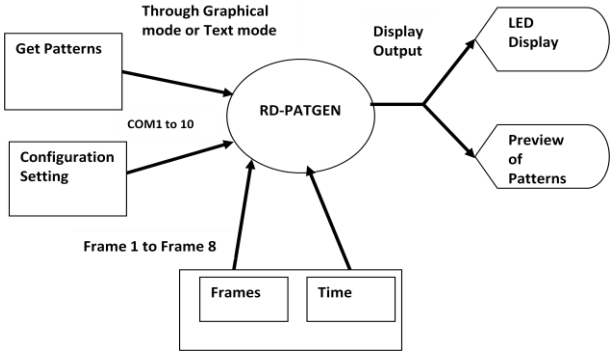


Figure 6. Context level diagram for RD-PATGEN

Figure 6. Shows complete data flow for the display device. GetPatterns will get the corresponding patterns either in graphical mode or in text mode. A set of settings will be required before data transmission e.g. Port settings, options given to select com ports from COM 1 to COM4 can be used. Eight different frames are used at different Time delay to display different patterns. The data transmission from RD-

PATGEN to the Display device will be sent through COM ports using MS Communication. LED Display shows the different patterns on the Image Active Area of the graphical user interface. Preview of Patterns shows the exact display of patterns, in the same ways which are displayed on the hardware device.

3.2 Operational description for RD-PATGEN

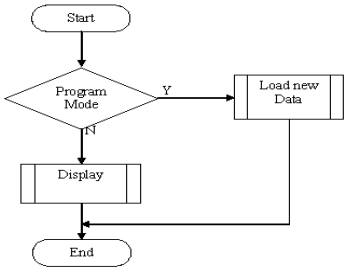


Figure 7. Flowchart for RD-PATGEN

Figure 7. Shows the flow chart for RD-PatGen system. Once the RD-PatGen has been started if it is in program mode i.e., the running mode of RD-PatGen, then it initiates Load new Data method for loading existing patterns. If the RD-PatGen is not in Program Mode, a Display method is called. This Display method provides a Preview option to view the patterns.

4. Snapshots for the RD-PatGen

4.1 Flow of RD-PatGen

Above figure shows the splash screen when RD-PATGEN is initiated.

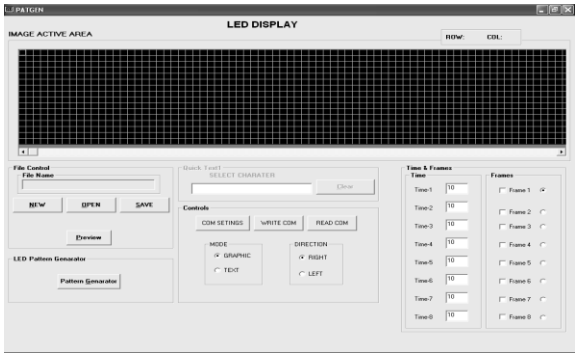


Figure 10. Snapshot for LED DISPLAY.

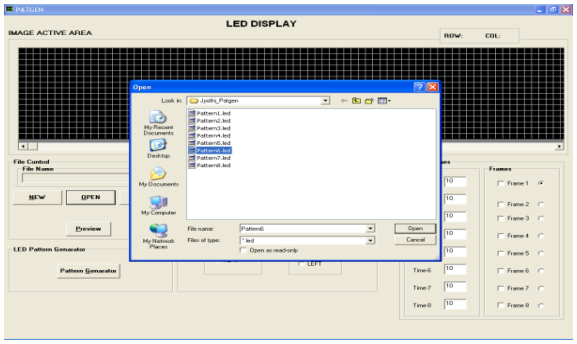


Figure 11. Snapshot for selecting existing patterns

Here user can create a new pattern; open existing patterns (Fig.10) and can do modifications for the existing components and save that pattern. Under the mode option, the user can select either graphic mode or text mode. Once the user presses the pattern generator under LED pattern Generator, the following screen will be displayed.

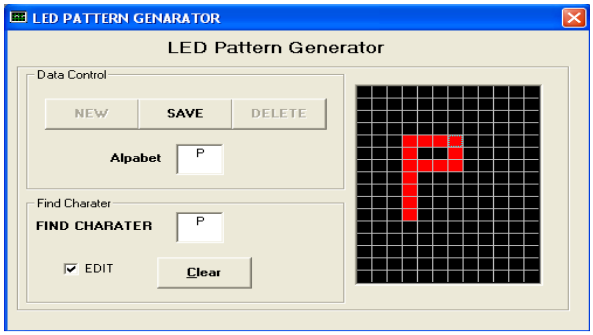


Figure 12. Snapshot for LED Pattern Generator

Here the user can select the text pattern by typing the particular character in the box given, so that the corresponding pattern will be generated. Clear option is given to clear an existing pattern, so that user can type a new character.

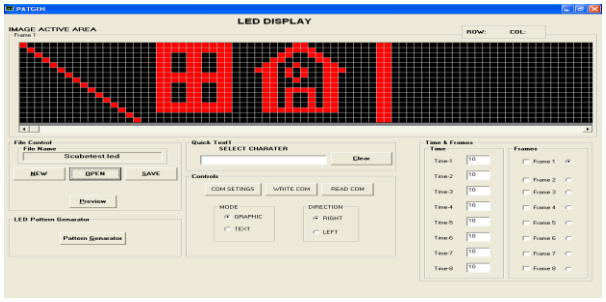


Figure 13. Snapshot for Graphic mode

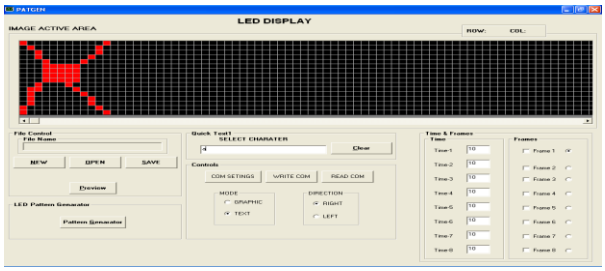


Figure 14. Snapshot for Text mode

Above figures show the Graphic mode and Text mode for the selected pattern, and modifications can be done for the pattern of interest.

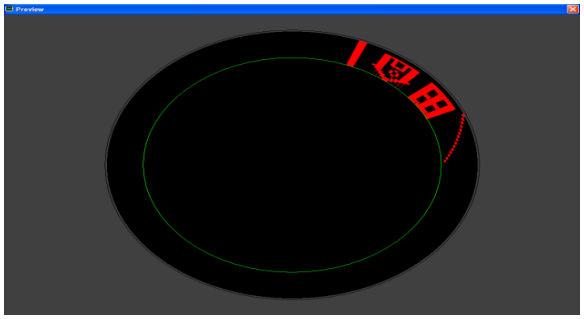


Figure 15. Preview of Graphic mode

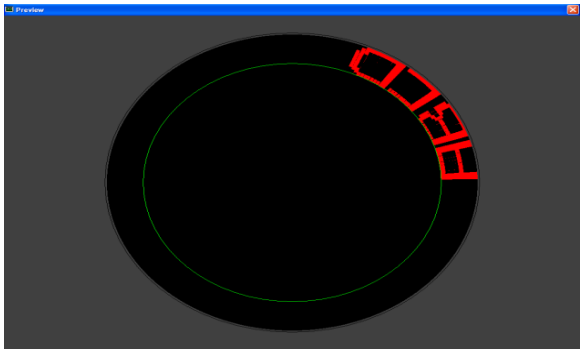


Figure 16. Preview of Text mode

The selection of the Preview button of LED DISPLAY leads to the preview of patterns of interest. The above figures show Graphic mode and Text mode for the patterns created for display on the wheel of motor car.

5. Conclusions

The "RD-PatGen" system has been developed and it has been tested with all the test cases and found to be effective in conjunction with associated hardware. RD-PatGen computer-aided design is very extensively used. In this, many features have been provided to display multiple patterns. Preview is provided to view-generated patterns. Using different COM settings data transmission is made easier. In our future work, number of rows and columns can be increased. Cyclic redundancy check can be made available for error checking., Preview can be made online, network and USB options can be added, Image pattern can be extended for multiple colors and Compact flash card, Plug and play option can also be provided.

Appendix A

Circuit Diagram for hardware display device

PIC16F876 is the heart of the circuit; it is very simple and straight forward [6]. 16 LEDs are connected to the controller. First 8 LEDs are connected to PORTB pins B0-B7. The next 8 LEDs are connected to PORTC pins C0-C7. A 330 Ω resistor is connected in series with each LED to limit the current to about 20ma. Logic '0' on the pic pin will turn off the corresponding LED and logic '1' will turn on the LED. C2 and C3 lines of the PIC will function as SCL (Serial clock) and SDA ((Serial Data) lines during pattern loading mode. SCL and SDA lines are standard I2C communication lines. The same SDA and SCL lines are connected to corresponding flash memory pins. The usage of I2C device 24LC256 reduces the interface lines to only 2. This device can store 256K bytes of information. As a standard requirement, the SCL and SDA lines are pulled-up to 5V through individual 2.2K Ω resistors. C6 and C7 lines of PIC have been designated as RX and a TX lines respectively for serial communications. These lines are connected to Max232 serial port interface chip. This chip converts the TTL level (0 & 5V) signals to RS232 level (+12V & -12V) signal levels and vice versa. Proximity sensor is connected to A0 line of the PIC. This sensor will give a pulse when the magnet fixed on the LED strip comes near during rotation. The PIC will poll for this signal and will synchronize the patterns accordingly.

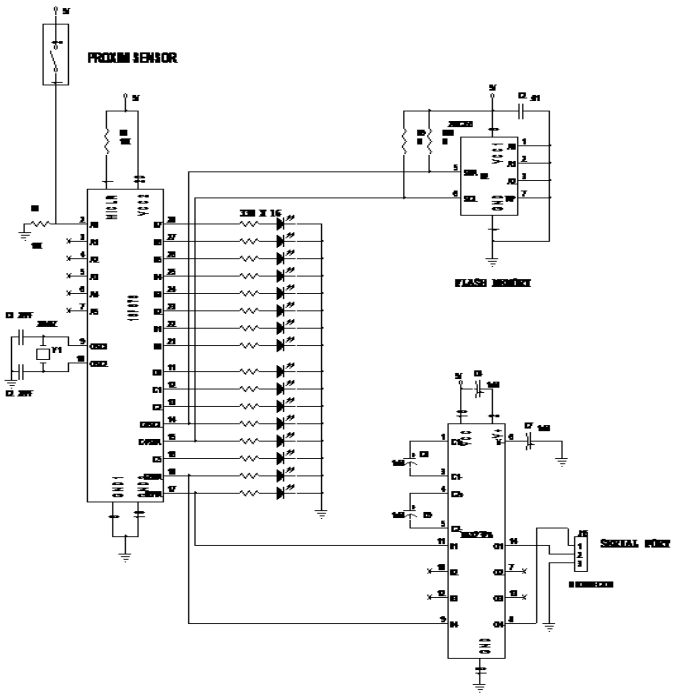


Figure 17. Circuit diagram for hardware device

Acknowledgement

We gratefully acknowledge the financial support we have received from the PES Institute of Technology. Our sincere thanks to Prof. K N B Murthy, Principal, Prof. Nitin V Pujari, HOD, Department of Computer Science and Engineering for their constant encouragement to carry out my research work. I would also like to thank Mr. Madhusudan.S, Technical Manager, IBM, Bangalore for his kind support.

6. References

[1] MINI 11 - microcontroller development board for SCL approach Aziz, H.A. Yusoff, N.M.K.N. Sapien, M.Z.B.M. Fac. of Electr. & Electron. Eng., Univ. Malaysia Pahang, Pekan, Malaysia, Research and Development (SCOReD), 2010 IEEE Student Conference.

[2] Programming Distributed with COM and Microsoft Visual Basic 6.0, Ted Pattison, Microsoft Press,

[3] Real Visual Basic - A Practical Approach to Enterprise Ed. Dan Petit, -- Pearson Education—2000

[4] Visual Basic 6 Database Programming Bible, Freeze IDG Books, India—2000

[5] The Use of PIC Microcontrollers in Multiple DC Motors Controls By Dr. Steve C. Hsiung, the Official Electronic Publication of the National Association of Industrial Technology, 2007

[6] Bates, M. (2004). PIC Microcontrollers: An Introduction to Microelectronic (2nd Ed.). Burlington, Massachusetts: Elsevier: Newness.

Design and Development of ARM based Embedded system Laboratory

Aruna. Kommu and Raghavendra Rao. Kanchi
VLSI and Embedded System Laboratory
Department of Physics, Sri Krishnadevaraya University.
Anantapur, 515003, Andhra Pradesh, India
* kanchiraghavendrarao@gmail.com

Abstract— **In this paper we present a platform consisting of a series of laboratory exercises using ARM microcontroller-LPC2148 (ARM&TDMI). The open hardware system in the present scenario gives an opportunity to perform the experiments even at home. Stress is given to the simple representation in terms of block diagrams instead of the use of high level tools (C or Assembly). The system described supports the transfer of developed program code from Laptop to the Microcontroller, and evaluation of its execution on rapid prototyping hardware. Further it allows the students to add new exercises to the already existing list. They can also enhance their skills by developing software which is suitable for hardware interfacing and data acquisition in the laboratory. The series of experiments described in the present paper will be a point of take off to start a journey on working in the area of embedded system. It gives an opportunity to think individually and to develop the creativity towards the fast growth of technology about an embedded system. Suggestions are given in the design so that incorrect wiring and testing will be avoided.**

Keywords— **ARM7TDMI, LPC2148, On-chip Peripherals, Real-time Interfacing, Hands –on experiences.**

I. INTRODUCTION

In the present scenario, topics like: Application Specific Integrated Circuit (ASIC), System- on Chip (SOC) have replaced Computer Architecture, Organization and design in the field of Computer Engineering. This shift has called for want of knowledge on Embedded System to be a pre-requisite. Majority of universities in India offers courses on microcontrollers like PIC, ATMEL, MOTOROLA, INTEL etc. Few universities in India have taught ARM microcontroller in their curriculum. There are several papers dealing with development of laboratory experiments using different platforms and different microcontrollers [1, 2]. Paper dealing with development of laboratory exercises using ARM-based microcontroller are very less. On the other hand there is a vast development of technology with high performance microcontroller like ARM and on software side the operating system Windows 8 can run on ARM microcontroller has been developed.ARM microcontroller is dominant in the mobile and auto mobile electronics market and is one of the industry leading standard microcontroller. This is compatible with all four major platform operating systems Symbian os, Palm os, Linux os and windows CE. Keeping the above facts in view point, we thought that it is worthwhile to describe a platform on which a series of experiments can be performed that gives a hands –on experience and bringing in a take home concept. Further, the series of experiments described in this paper can be introduced as a part of curriculum for the senior graduate students.

A.ARM7TDMI

ARM (Advanced RISC Machine) architecture is the 32 bit RISC instruction set architecture which is most widely used in a number of embedded systems. ARM is a leading intellectual property provider of high performance, low cast, and power efficient RISC processor microcontroller [3]. ARM processor has some characteristics that made them suitable for low power application such as intelligent voltage management and cache micro architecture. The most prominent future that makes ARM differ from other microcontrollers is [4]:

- Interrupt system
- cache controller design
- voltage management
- Instruction set.

ARM CPU has a von Newman architecture and it uses two buses i.e. AHB (advanced high performance bus) and VPB (VLSI peripheral bus) which increases the speed of execution [5] compared to other commonly used microcontroller such as the INTEL 8051 and the Microchip PIC family. ARM family provide wide range of performance from 100 MIPS to 1000 MIPS. This increase in performance leads to two important factors and advances in a new process technology.

- New pipeline mechanism.
- Implementation of Harvard bus architecture in ARM9 family

An efficient and optimized processor design is an art. This is because the instruction set should cater the need of the programmer in an efficient way. i.e. keeping the future implementation also plays pivotal role in the processor design .This is because it has to bridge the gap between HLL construct and machine language. Thus the compiler design which in turn depends on the type of instruction set. The whole ARM family has the same instruction set of 32 bit ARM instruction set and 16 bit thumb instruction set .The thumb instruction set was introduced in the fourth version of the ARM architecture in order to achieve the high code density for embedded applications.

NXPs ARM mini kit was purchased from NSK electronics which cost about 30$ [6]. Mini ARM kit consists of LPC2148 ARM7TDMI as its core and two types of memory, Flash memory of 512kb with In system program and In Application programming, static RAM of 40kb [7]. LPC2148 CPU has a simple programmer model and it has sixteen 32 bit general registers and two special registers named as current program status register (CPSR), saved program status register (SPSR). It has a vectored interrupt controller, two 10 bit ADCs with 14 channels, one DAC.USB full speed device controller, two UARTS one with full modem interface, two 32 bit timers, watch dog timer, PWM unit, REAL time clock with optional battery backup and 45 general purpose i/o pins, CPU clocks up to 60MHz on chip crystal oscillator and on chip PLL.

There is open source KEIL software that provides a best development tool and technical support [8]. It also offers numerous ways from the technical support that one needs to complete embedded projects. KEIL supports simulation using the target systems and a debugger interface. Micro vision includes traditional features like simple and complex breakpoints, watch windows and execution controls as well as sophisticated features like trace capture, execution profiler, code coverage and logic analyser. Features such as free development tools and compilers plus the ability to integrate a low cost in circuit debugger in to the ARM CPU module made the ARM an excellent choice for the new advances that have been made in new process technologies. ARM CPUs accounts for approximatly75% of all embedded 32-bit RISC CPU making it the worlds largest selling 32-bit architecture. This allows the students to learn popular microcontroller architecture in their gradual level which is widely used in the electronics industry. When the ARM CPUs are designed ARM7TDMI from ARMv5 was the most popular device used in Apple iPod and Newton PDA. TheARM7 microcontroller also provides a wide range of internal peripherals. All these features have made the ARM7 an ideal choice for lab experimental kit for the beginners in to ARM microcontroller. Some of the embedded control application typically requires a processor with cache and memory protection to utilize the real time operating systems. The vast majority of users need security and desktop/web productivity applications. For that purpose ARM has developed a vertical expansion of CPUs to match the user requirements and provides a unique, configurable amount of cache. Recent development includes a new pipeline mechanism and implementation of Harvard architecture in ARM9 family, DSP and java extensions to some of the new architecture enables rich applications to benefit from the high performance and low power consumption intrinsic to ARM processor cores. BY considering the above all facts ARM architecture is compatible, flexible and encompass the full range of embedded requirements, we have chosen to popularise ARM microcontroller by developing a series of experiments useful for graduate level laboratory. The following sections give an account of experiments performed in the laboratory.

II. EXPERIMENTAL DESCRIPTION

In our department embedded laboratory consisted of a series of experiments using PIC, Atmel and Intel microcontrollers. For the reason that ARM microcontrollers have become very popular in mobile technology, automobile and electronic industry, it is felt to introduce ARM based training in hardware and software for the first time in the curriculum. In this direction, a series of experiments useful to both undergraduates and graduate laboratories of engineering and science have been designed and developed. One of the advantages of the described experiments is that they can be implemented even for take-home labs since the hardware components are inexpensive. The experiments are provided with the relevant documentation. Brief description of these experiments is given in the following. Fig 1 shows the mini ARM kit connected to laptop.

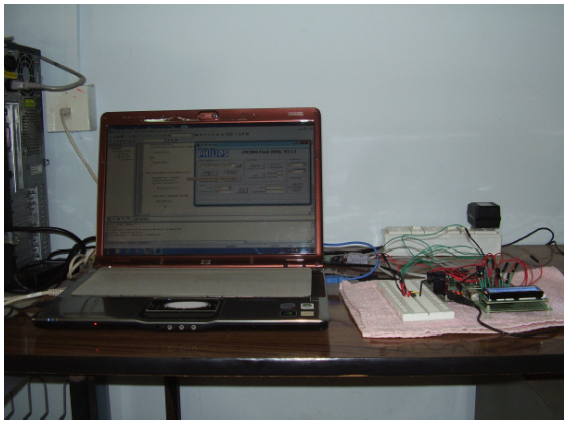


Fig1: The mini ARM kit connected to the laptop and program uploading window

A. *LOGIC CONTROLLER*

The first experiment is to get acquainted with the programming of a port of the Lpc2148 as input/output. This needs the careful study of the I/O pins and selection of their function. The logic controller experiment comprises of interfacing LEDs with output port and connecting a pushbutton(s) to an input port.

Output port: LEDs are connected to PORT of LPC2148 and software is developed to switch alternate LEDs on and off, and different variants of this exercise.

Input port: A single push-button (PB) is connected to one of the pins of port 0 and that bit which PB is connected is programmed as input port and software is developed to display the number of times the push button is pressed on

LCD (interfacing LCD with LPC2148) is explained in exercise B

B. *INTERFACING LCD*

This experiment gives the hardware and software description to interfacing LCD (16charactersx2line) with an output port. In fact, the LCD can be interfaced in two modes: 8-bit mode and 4-bit economic mode. In the present experiment the second technique is explained since this presents a mode of optimal utilization of I/O port pins. Following fig 2 shows the LCD module interfaced with ARM CPU.

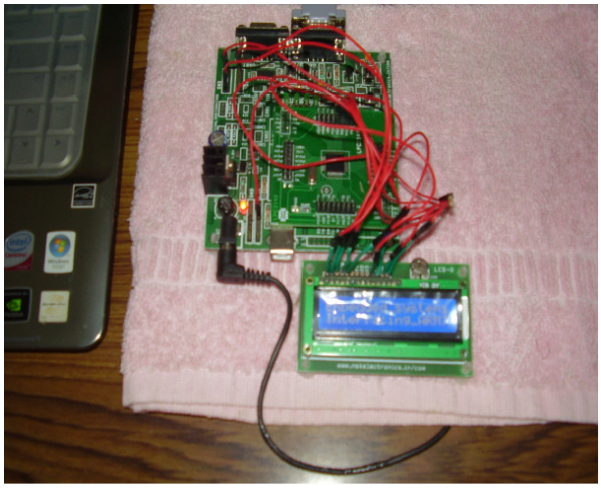


Fig 2: LCD module interfaced with ARM CPU

C. INTERFACING SEVEN-SEGMENT DISPLAY MODULE

A 4-digit LCD display module is interfaced with LPC 2148. The display is driven by to ULN 2003 (Darlington Pair IC) and multiplexed display concept is implemented by software. This ensures low power consumption by the LED display module. The outcome of the experiment is: To understand the concept of multiplexed display.

D. INTERFACING STEPPER MOTOR

A stepper motor collected from a junked 51/4 floppy drive is used. This experiment demonstrates the use of stepper motor for different applications by rotating to motor of different angles and different speeds. The stepper motor is driven by an output via power driven IC ULN 2003.This experiment paves path for applications of stepper motor rotation for different applications.

E. INTERFACING DC MOTOR

A DC motor unlike stepper motor rotates continuously on the application of power to it. The speed of rotation depends on the amount of power applied to the motor. This can be varied by the technique: pulse width modulation. LPC 2148 provides a multiplexed pin as PWM output. Hardware is developed in the form of a power amplifier with H-bridge that provides an opportunity to rotate the motor clock-wise and anti clock-wise directions. Software for obtaining variable duty cycle and hence motor to rotate at different speeds is realized.

F. INTERFACING RELAYS AND SOLENOIDS

Interfacing of relays and solenoids with the output ports of lpc2148 throws light on control applications such as temperature control, flow control etc. These relays solenoids are interfaced via buffer/power amplifier stage. Software controls the on/off time in a programmed way.

G. ANALOG VOLTAGE MEASUREMENT USING ON-CHIP ANALOG-TO-DIGITAL CONVERTER (ADC):

Lpc2148 is fabricated with a 10 bit on chip ADC. It can be used in single ended/differential mode of operations. In the present exit intensity measurement is carried out by interfacing a photodiode with the analog input of ADC. LED intensity is varied by an automated potentiometer driven by a potentiometer. The acquired data is plotted by the computer using origin package.

H. GENERATION OF ANALOG VOLTAGE USING ON-CHIP DIGITAL-TO-ANALOG CONVERTER (DAC):

The on chip DAC of LPC2148 can be used for various applications such as waveform generation, displaying alpha-numeric and graphics, generating different values of analog output voltages etc .In the present work different analog output voltages are measured by giving different digital inputs at the appropriate digital pins that are selected using the pin functions corresponding to DAC

I. REAL TIME CLOCK

The on chip RTC is useful in serial applications such as to count the happening of events, ultra low power design to support battery powered systems. In the present experiment software is developed to display Date, month, year and time on the LCD which is interfaced in the experiment B. The experimental arrangement using LPC2148 is shown in fig 3.

J.UNIVERSALASYNCHRONOUSRECEIVER/ TRANSMITTER (UART):

LPC2148 has two on chip UARTs. One of these UARTs is used. Software is developed to send characters to laptop under HyperTerminal. This experiment gives an insight in to the serial interface protocol-RS232, Baud rate selection etc.

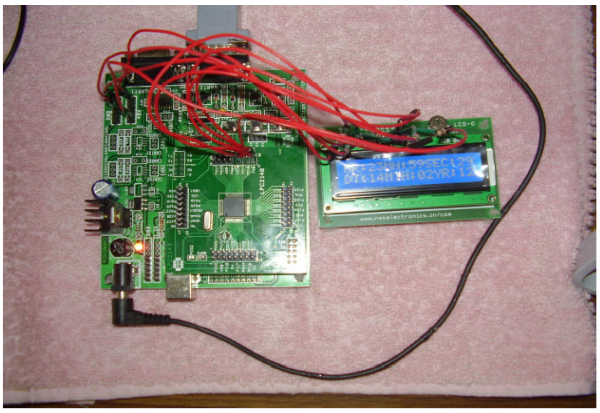


Fig: 3 Displaying real time clock on LCD module

H. SERIAL PERIPHERAL INTERFACE (SPI):

The on chip SPI of lpc2148 uses MOSI, MISO, SCLK, and SSEL. Pin functions are selected to interface a 12-bit ADC (MCP3204), which supports SPI protocol. The digital output equivalent to analog input voltage is displayed on LCD screen. As shown in fig 4.

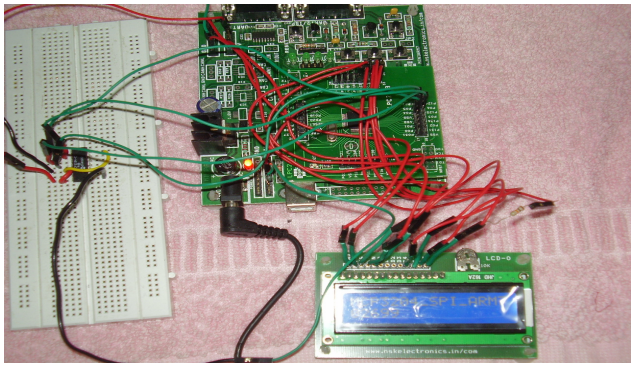


Figure 4 Interfacing MCP3204 (ADC) with ARM Module under SPI Protocol

III. CONCLUSION

A series of experiments were designed using ARM platform. They can be incorporated in the engineering and science discipline. The low cast and fully open structure of the ARM is considered to make it a good embedded platform, especially in the higher education curriculum. With this basic and fundamental knowledge of experimental lab using ARM, students can learn the real engineering challenges by experiencing hardware design, hardware prototyping and sensor connectivity in addition being able to control various signals in a single programming environment. It also allows the students to improve their programming knowledge by engaging them in real experiments. Using this approach it also provides a fast learning process since it gives the opportunity to work with the experimental modules whenever needed.

REFERENCES

[1] Yao Li "Teaching Embedded Systems using a modular Approach Microcontrller Training Kit", World Transactions on Engineering and Technology Education. Vol.6, No.1, PP-135-138, 2007.

[2] Feisel. L and Rosa A "The role of the Laboratory in undergraduate Engineering Education' Journal of Engineering Education . 94,121,005.

[3] Oliver J.P and Haim. F "Lab at Home Hardware Kits for a Digital Design Lab" IEEE Tranactions on Education 52, PP46, 2009.

[4] Andrew N Sloss "ARM Systems Developers Guide: Design and Optimising System software, Morgan Kaffman Publishers, An Import of Elsevier, 2005.

[5] Stew Furbur " ARM System –on-chip Architecture, Pearson Education, 2005.

[6] LPC2148 User Manual NXP Semiconductors-UM10139

[7] http://www.nskelectronics.com

[8] http://www.keil.com

[9] K. Raghavendra Rao and G. Nagamani "Design and Development of Hardware and software to Test the Potentiometer Linearity Using PC's Printer port, World Academy of Science Engineering and Technology Vol.64, PP121-127, 2010.

SESSION

POSTERS

Chair(s)

TBA

Development of Digital Signal Processing Module for Level Measurement Radar

Yeonghwan Ju, Sang-Dong Kim, Jonghun Lee*(Corresponding author)
Advanced Radar Technology Laboratory, Robotics Research Division, Daegu Geongbuk Institute of Science & Technology, Daegu , Korea
Email : {yhju, kimsd728, jhlee}@dgist.ac.kr

Abstract - *This paper presents level measurement FMCW radar. We developed digital signal processing module to measure distance with high accuracy for FMCW radar. We implemented the level measurement FMCW radar algorithm on the FPGA and DSP. For verifying the level measurement radar system, test bed is made to measure the distance for every 1 mm. The experimental results show that the distance accuracy is about 4.5 cm distance between 357 cm and 557 cm.*

Keywords: level measurement radar, FMCW radar, radar distance measurement

1 Introduction

FMCW (frequency modulated continuous wave) radar systems have become a trend for level gauging over the last few years since the digital level measurement radar is more flexible and when the environment changes, the program can be updated accordingly. FMCW radar performs well under harsh environmental conditions, achieve precision in the mm-range, and offer competitive pricing. As a non-contact sensor, level measurement radar requires the high accuracy, the high linearity and the wide bandwidth of transmit signal in order to measure precise range of a liquid level [1].

In this paper, we developed digital signal processing module based on the FPGA (field programmable gate array) and DSP (digital signal processor) for a liquid surface level measurement. Developed digital signal processing module is connected to the 24GHz RF module and then developed level measurement radar system is verified in anechoic chamber of high frequency band.

2 Basic Principle of FMCW Radar

The working principle of a FMCW radar level gauge is shown in Fig.1. The microwave signal with linear frequency modulation, generated by VCO (voltage controlled oscillator), goes through antenna. The emitted signal by antenna is reflected by product surface and received by the antenna again. The transmitted signal and received signal are down-mixed to obtain beat signal.

The FMCW radar transmits a frequency-modulated continuous wave to measure the range of the target.

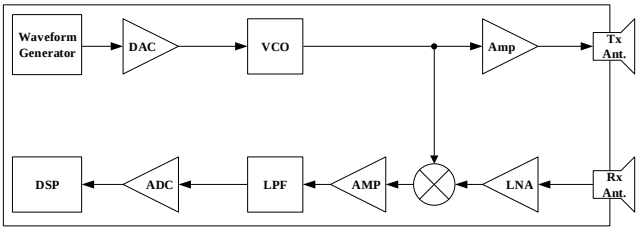


Fig. 1. Block diagram of the level measurement FMCW radar system

Figure 2 shows frequencies as a function of time in the transmitted signals and received signals for a stationary target. The range beat frequency f_b can be obtained by signal processing, and then the range of the target can be estimated as Equation (1).

$$R = (C \cdot PRI \cdot f_b)/(2 \cdot B) \quad (1)$$

Where, C is light speed, B is the modulation bandwidth, and t_d is the delay time between transmitted and received signals. The PRI is Pulse Repetition Interval which is a chirp period.

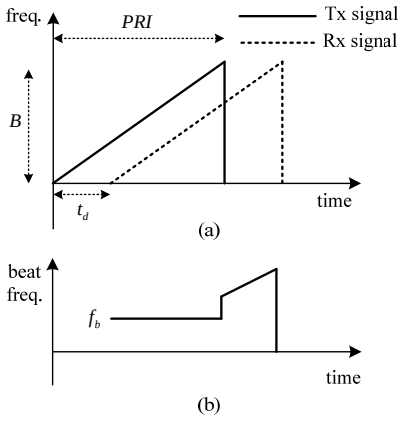


Fig. 2. FMCW radar signal (a) the transmitted signal and received signal (b) beat signal.

3 Digital Signal Processing Module

The digital signal processing module of the developed FMCW radar is shown in Fig. 3. The digital signal processing module consists of the ADC, the FPGA, and the DSP. In this paper, we implemented the level measurement FMCW radar algorithm on the ALTERA Cyclone for FPGA and TI TMS320C6455 DSP, which supports the Gigabit Ethernet, the internal memory, and the high processing clock.

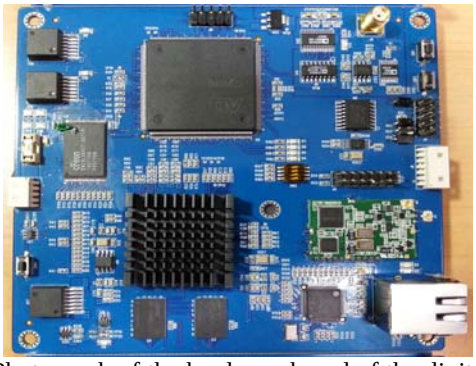


Fig. 3. Photograph of the hardware board of the digital signal processing module.

The level measurement FMCW radar used in this paper is composed of the RF transceiver module including the antenna, the radar user interface, and the digital signal processing module.

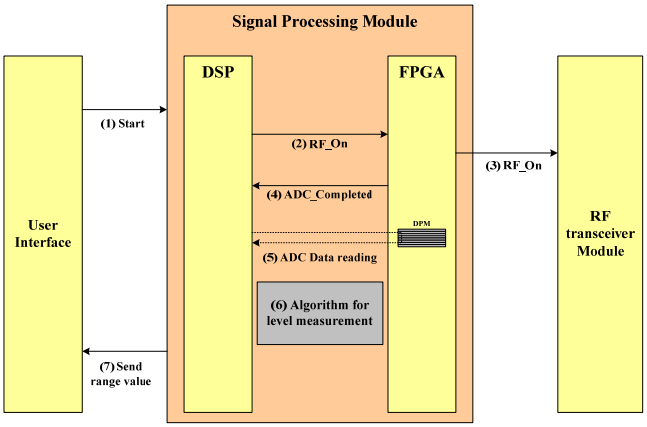


Fig. 4. Level measurement radar algorithm Procedure.

Fig. 4 shows the working procedure of the implemented level measurement radar. After power on, user interface of the radar send start signal and then the DSP send RF_On signal to the FPGA and the RF transceiver receive RF_On through FPGA. At that time, the RF transceiver module generates fast rapid ramps. If FPGA finished to store beat frequency signal in FIFO the complete status is reported to DSP. The DSP start reading the detection results from FPGA, and finally transmit the range information to the radar user interface.

4 Experimental Results

We evaluate the performance of the level measurement radar by combining a 24 GHz RF transceiver. The center frequency and bandwidth of the developed radar system are 24 GHz and 2GHz, respectively. We experimented in anechoic chamber with frequency band from 8GHz to 110GHz to obtain the accurate measurement results.

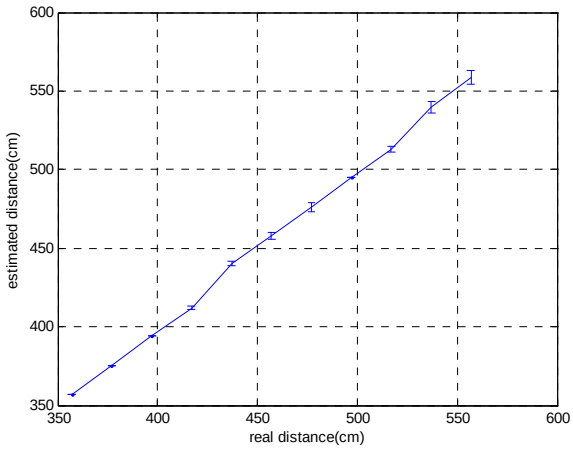


Fig. 5. The result of distance measurement between 357 cm and 557 cm

Fig.5 shows the measurement results of the distance accuracy. The experimental results show that the distance accuracy is about 4.5 cm at distance between 357 cm and 557 cm.

5 Conclusions

In this paper, we developed the level measurement FMCW radar to obtain high accuracy level gauging. The level measurement FMCW radar algorithm is implemented on the FPGA and DSP. We verified the radar system connected with a 24 GHz RF transceiver to obtain accurate measurement results in anechoic chamber. Experimental results showed that the accuracy of range estimation is less than 5 cm.

6 Acknowledgments

This work was supported by the DGIST R&D Program of the Ministry of Education, Science and Technology of Korea (13-RS-02).

7 References

[1] Armbrecht, G., Zietz, C., Denicke, E., & Rolfes, I., "Antenna Impact on the Gauging Accuracy of Industrial Radar Level Measurements", Microwave Theory and Techniques, IEEE Transactions on 59.10 (2011): 2554-2562.

SESSION

EMBEDDED SYSTEMS APPLICATIONS, PROTOCOLS, OPTIMIZATION METHODS, AND MICRO-CONTROLLERS

Chair(s)

Prof. Hamid Arabnia
University of Georgia

Wireless Heart Rate and Blood Pressure Monitoring Network

Hicham H. Hallal[1], May Haidar[1], Maya Fakhriddine[2], Insaf Kamal[1], Angham Salem[1]
[1]Fahad Bin Sultan University, Tabuk, Saudi Arabia P.O.Box 15700 , Tabuk 71454
[2]San Diego State University, 5225 Campanile Dr, San Diego, CA 92115
{hhallal, mhaidar}@fbsu.edu.sa, mfakhrriddine@hotmail.com, e.sofy@hotmail.com, spring_a_sa@hotmail.com

Abstract

We present a networked application to monitor vital signs in patients without affecting their daily routines. The proposed application, which provides a monitoring and management environment, is practical especially in the case of elderly and young patients. We describe the implementation of the application, which targets reading the heart rate (HR) and blood pressure (BP) signs of patients remotely, in a context where multiple patients are considered concurrently. The application includes a control unit that analyzes remote readings, prioritizes them following their urgency, alerts health specialists for alarming readings, and feeds a server application where a database of all monitored patients stores history of the readings for each patient.

Keywords: Embedded systems, Applications of Embedded Systems, Health, wireless health monitor, Wireless sensors.

1. Introduction

Health care is an essential concern for everyone. Individuals are becoming increasingly aware of the need to evaluate their conditions; and governments are equally concerned about providing sufficient and adequate care to their citizens. In many countries, attempts exist to develop platforms to allow the use of available technologies to serve the advancement of health care services. The Internet, networked applications, embedded systems, and nanotechnology are all examples of technologies that are being increasingly used in developing solutions to various problems in health care. In the last few decades, continuous technological advances have brought hope to solve many health related problems including diagnosis, testing, and even surgery. Technologies like embedded systems, nano-electronics, robotics, and the Internet have made microsurgery, remote surgery, and implantation based diagnosis possible and practical solutions in many contexts.

In this paper, we consider the problems of people who suffer from high or low blood pressure and irregular heart rate and who do not have full awareness about the consequences; thus, are unsure of the proper actions to take when symptoms appear. While some people assume that taking medication for a given period of time can resolve a high-pressure problem, others ignore regular checking of blood pressure as long as they do not complain of any clear symptoms. In fact, it cannot be emphasized enough that hypertension patients must consult a doctor regularly to measure pressure and determine a treatment plan. Statistics conducted worldwide show that most failures in hypertension therapy result from the lack of cooperation between patients and physicians. In many cases, the required cooperation is simply taking regular periodic readings of the vital signs of the patient. The main problem is that the current methods for reading blood pressure and heart rate are still intrusive and patient limiting; they require direct interface with the patient's body, and they often require his/her presence for extended periods at a health facility. Even portable devices, which can be worn by the patients to monitor their vitals for extended periods of time away from the clinic or the doctor's office do not provide real time feedback about the changes in someone's situation. For these reasons, researchers in both academia and industry became more interested in developing technologies and tools that can provide readings of the vitals of a patient, including the blood pressure and heart rate, without much disruption to his/her daily routines. This is made possible with the continuous advances in embedded system technologies, telecommunication networks, and sensor technology. Needless to say is that any achievement in the field can simplify the task of the physician and provide the patient with better care. In addition, the benefits of any developed technology for this purpose are especially apparent in the case of elderly and young children patients who are difficult to monitor and control.

The work in this paper addresses the main questions: Is it possible to automate the process of monitoring the blood pressure and heart rate of patients using existing technologies and infrastructure? Can such automation be scaled up to cover multiple patients in an efficient way?

In this paper, we develop a networked system for monitoring blood pressure (BP) and heart rate (HR) to be used in hospitals or with patients at home. The system consists of:

A. Embedded system monitor: bracelet with sensors for BP and HR. The embedded system can be programmed to take readings at specific times or open for remote control.
B. Control unit to log patient data and analyze readings from different monitors. The unit is also responsible for dispatching emergency calls for health specialists when alarming readings are received.
C. A control application that runs at the control unit or at a server system. This application has the task to manage the accounts of different patients, to handle the readings received from different monitors, and to build and manage the history files of each patient. The intended application consists of two main parts:
   a. A database to store patient data and records and to produce reports on the status of each patient.
   b. An interface program to handle the communications with the embedded monitor. The interface program is also responsible for the management of the database of the application (adding new patient records, deleting and updating records, generating reports, etc.). In addition, The interface allows the user, usually a health care professional, to control the embedded device remotely by sending messages to operate or to stop the monitor at designated or chosen times.
D. Wireless connection between the monitor and the control unit. In the proposed setting, a hospital or a care giving center, the communication between the embedded monitor and the control unit can be achieved via Radio Frequency (RF) communication, which is inexpensive and easy to implement.
E. Communication between control unit and third party (parent, doctor, etc.). This communication can be carried either over the Internet or over the cellular networks (GSM, 3G, etc).

This paper is organized as follows. In Section 2, we review the work and discuss similarity with existing solutions. In Section 3, we describe the specification of the proposed application and the main components involved in the implementation. In sections 4, 5, and 6; we describe, in details, the components that make up the proposed application. Then, in Section 7, we conclude the paper and list the potential future extensions.

2. Related Work

Reading the vitals of patients is a major task that cannot be overlooked in any health care process. Improving the techniques and tools used to read blood pressure (BP) and heart rate (HR) has been the focus of many researchers in both academia and industry. We review the main related works that deal with the development of advanced BP and HR monitors. In [2], the authors report the design of an electrocardiogram (ECG)/BP telemonitor using photoplethysmographic (PPG) measurement of BP, microcontroller technology, wireless communication, and long term memory devices (24 hours). The work in [3] describes a monitoring system that uses implantable CMOS based sensor to provide 24/7 monitoring of the BP. The data collected by the sensor is read telemetrically using an external unit that should be carried close to the skin (at about 8 cm from the sensor). Readings in the external unit can be transmitted to a central unit for display, analysis, or storage. iHealth [9] have developed a Wireless Blood Pressure Wrist Monitor that attaches to the hand of the patient with a cuff. This wrist monitor is linked through a wireless connection to a mobile application that can be used on a smart phone to track display instantaneous readings and history of readings. The wrist monitor is also equipped with a motion sensor system to guarantee accurate readings when different positions of the user are assumed. In other words, the device can be seen as a BP monitor that uses a smart phone as its display. The work reported in [4, 5] mentions the development of a monitor for BP and HR of patients to be recorded and displayed either locally or remotely by a wireless connection. The interesting concept is the use of non-intrusive monitoring techniques to read the signs. In addition to the classical oscillometric method, the author discusses the optical method (photoplethysmographic or PPG for short) that relies on one of two techniques. Transmission performed by shining a light through the skin of the patient from

an LED installed at one side of a body part. The light is then collected on the other side of the body part using a photodiode; which allows analysis of the characteristics of the light that passes through the skin. Reflection, on the other hand, is done by collecting the reflection of the light that is sent through the skin. In this case, both the LED and the photodiode are placed on the same side of the body part. Characteristics of the light wave are derived from its reflection (what was not absorbed in the skin). Ideal places of use are the forehead and chest. Finally, [1] describes a monitoring device to record the vitals of a patient over a wireless connection. A patch is fixed to the patient's head along with a processing unit. Optical and electrical waveforms generated by the patch determine the patient's vitals. The proposed solution uses an optical system to measure BP by determining the variation in the optical waveform generated by the patch after it is reflected in the body. This variation is then translated to an electrical waveform that is transmitted to the monitor central unit for analysis.

In this work, we base our approach on results reached in the existing works and develop an integrated framework for the monitoring of BP and HR of multiple patients concurrently. It is a networked application that is useful in different settings including hospitals, long-term care centers, and baby sections in hospitals. The application can also be deployed for single users, and is not limited in terms of choosing the type of sensors or meters. The main advantage is the possibility of analyzing patient data automatically for prioritization of the readings and creating notifications and facilitating the work of health professionals and caregivers. Furthermore, the application, accessible over the Internet, will easily integrate in any E-Health framework. This is actually an ongoing discussion with researchers working on evaluating the deployment of an E-Health application in the Kingdom of Saudi Arabia.

3. Specification of the Monitoring System

Figure 1 shows a high level representation of the BP and HR monitoring system.

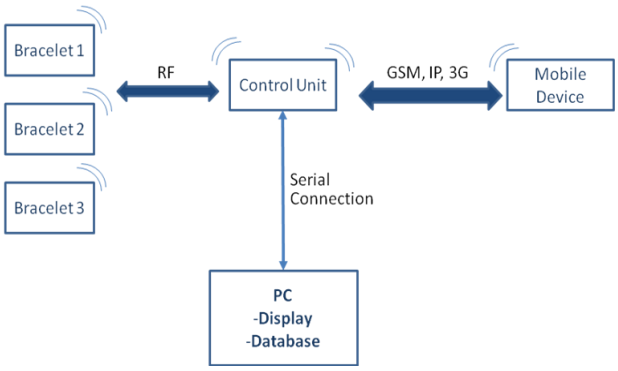


Figure 1. Architecture of the monitoring application.

The figure shows the three main parts in the system along with the interconnection among them:

1. On the patient side, an embedded BP and HR monitor has the responsibility of providing the reading and filtering of data while receiving control commands and executing them (mainly to operate the monitor or stopping it).
2. The control unit application has the task to receive the readings from the monitor and process the information in two ways:
   a. If the readings are not alarming, the application passes the reading with the ID of the monitor to a server system that hosts a database of all patients. The data is stored in the databases for history.
   b. If the readings are alarming, the application sends a message to the number associated with the source monitor. As in the first case, the application passes the information for storage in the database.

   In both cases the information received from the monitors are processed for a graphical or numerical display on the server system that is directly connected to the control unit.
3. The communication between the modules of the system is carried out as follows:
   a. Between the monitor and the control unit, the communication is over RF waves.
   b. Between the control unit and the server system, the communication is direct through serial connection.
   c. Between the control unit and the destination smart or mobile phone, the communication is over GSM (3G) networks. This connection can be modified to be between the server and the smart phone. In the latter case, the information sent to the smart phone will be more detailed. However, for the time being

we consider that the control unit will send the information to the phone because that will be faster, especially in very urgent cases. An extension of the whole application is possible in way that the user can access the files remotely on the server system to check the history of any patient he selects.

In the following sections we detail the description of each of the above mentioned components in the system along with our plan for implementation.

4. Portable BP and HR Monitoring Device

The BP and HR monitoring device is a simple embedded system that can be implemented using a microcontroller or simple logic circuit. The main task of the proposed device is to perform the readings and transmit the collected information to the control unit via RF. For this reason the monitoring device can be simplified to the degree that it can implement either of the following two workflows. In both case, the device can be implemented using simple logic circuit without the need for a microcontroller. This is essential since it contributes to reducing the cost of the device so it is affordable for anyone. In Figure 2, the first workflow (a), the device does not perform any check on the collected information. Instead it is passed directly for the control unit which will be responsible for the analysis. Although this workflow pauses a higher cost on communication (based on the scheduling of transmissions), it might be needed in specific cases where every reading is needed for the diagnosis. On the other hand, the workflow in (b), permits the device itself to decide on which readings to send by performing a comparison against a predefined threshold. This workflow is useful to save on transmission power. Our intention is to implement the monitoring device using the two workflows with an option to preset the device to follow either of the two before it is given to the patient. An extension to this proposition is to make it possible to select the workflow from the device itself, but we need to evaluate how much risk this presents in terms of the accuracy of readings and safety of the patient.

5. Analysis

The main component of the proposed system is the control unit, which has the major task of compiling the readings gathered by the different monitoring devices in the network. Consequently, the unit decides on the action to be performed on the compiled data. If the data is alarming, the reading is sent directly to the designated phone number of the person in charge (this might be a doctor, a care giver, or a relative). In addition, the reading itself is sent to the server machine to be added to the database of the patients. In both cases, the server system displays the readings in a user friendly interface for the person running the application. Such interface would facilitate the task of the user to monitor the different devices registered in the application.
The flowchart shown in Figure 3 shows the main task of the control unit. In the flow, the unit evaluates the readings received from the monitoring devices and decides on sending alarms or not based on the thresholds that are preset for the specific device.

The control unit is composed of the following elements:

1. Microcontroller: Two options are available for the microcontroller of the E-WATER system, PIC16F876A and PIC 16F877A [8], which both are high performance RISC CPU's. While both microcontrollers can do the job of running the embedded system, PIC16F877A has the advantage of containing an analog to digital converter. The main characteristics of the PIC microcontrollers include:
   a. Only 35 single-word instructions to learn
   b. All single-cycle instructions except for program branches, which are two-cycle
   c. Operating speed: DC – 20 MHz clock input
   d. DC – 200 ns instruction cycle
   e. Up to 8K x 14 words of Flash Program Memory,
   f. Up to 368 x 8 bytes of Data Memory (RAM) in the form of internal registers

   In addition, the PIC microcontrollers can be programmed using the C language, which is easier to learn and work with than machine specific Assembly Language. Translation into PIC instructions can be done using compilers like PIC C [8].

2. Memory Unit: we use an EEPROM -Electrically Erasable Programmable Read Only Memory- 24LC16B, which is a 16k bit memory unit. The device is organized as eight blocks of 256 * 8 bit memory with a 2-wire serial interface. Low voltage design permits operation down to 2.5 volts with standby and active currents of only

5µA and 1 mA respectively. The 24LC16B also has a page-write capability for up to 16 bytes of data. The memory unit will be used mainly to store the thresholds for different devices, the messages to be sent for the phones, and the phone numbers associated with each monitoring device.

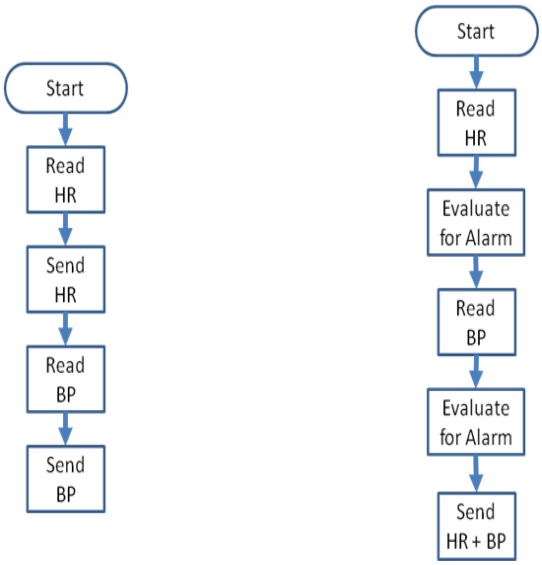


Figure 2. Left: simple workflow and Right: workflow with check in the device.

3. Signal Converter: we use the MAX 232 which converts signals from an RS-232 serial port to signals suitable for use in the microcontroller (TTL logic). The MAX232 is a dual driver/receiver and typically converts the receive/transmit (RX and TX) signals, as well as the Clear To Send and Ready To Send (CTS and RTS) signals.

4. GSM Modem: We use the F1003 GSM MODEM, which supports SMS and CSD functions. The modem is inexpensive and has already been used in industrial settings. The F1003 modem supports the RS232 interface, which makes it possible to connect to the Microcontroller through the MAX232 converter.

6. Server Application

The main role of this module is to process the readings of the monitoring devices for different clients and prepare reports on the medical situation of patients. For this, a database is designed to host the records for the different patients. We first discuss the design of the database and then we describe how the application processes reports. The main objective of the database is to provide storage of the data collected by the monitoring devices, where different records of patients are defined and easily accessed. In the future, the proposed database can be extended to include variety of data that is useful to different types of reports like the medical background of the patients and the recommendations for each type of readings for the patients. This becomes useful when integrating the application with E-Health systems is discussed. Currently, the proposed database consists of four main tables:

1. Patient: This table holds data on the patients who are using the monitoring devices. The table includes fields like Name, Patient_ID, Address Telephone Number, and Device_ID. The latter field is used to identify the system that is installed at the client's site. The table also includes two fields that are common with other tables: the Patient_ID and the Device_ID. The former, which is the primary key of the table, is common with the Report table and the latter is common with the Device table, which is not shown in the above diagram.
2. Report: This table holds the information of an Report, which includes the Patient_ID, the Report_ID, and the Report_Type (could be daily, monthly, or urgent).
3. Reading: This table includes the fields Type (monthly, weekly, or hourly), the Date and Time, the Device_ID and the Patient_ID.
4. Thresholds: This table defines the values limits that should be monitored for the specific patient.

The proposed database structure remains of manageable size and allows for ease of interface since the number of patients usually monitored in the application is usually limited.

The main task of the server application is to interface the control unit to the database of patients. The application has to update the records of each patient based on the readings received from the control unit and to generate reports based on those records. In addition to that, the application should generate reports about the data in the records. For example, reports on the yearly history, differences in readings between different patients and categories of patients based on gender, types of sickness, or age for example.

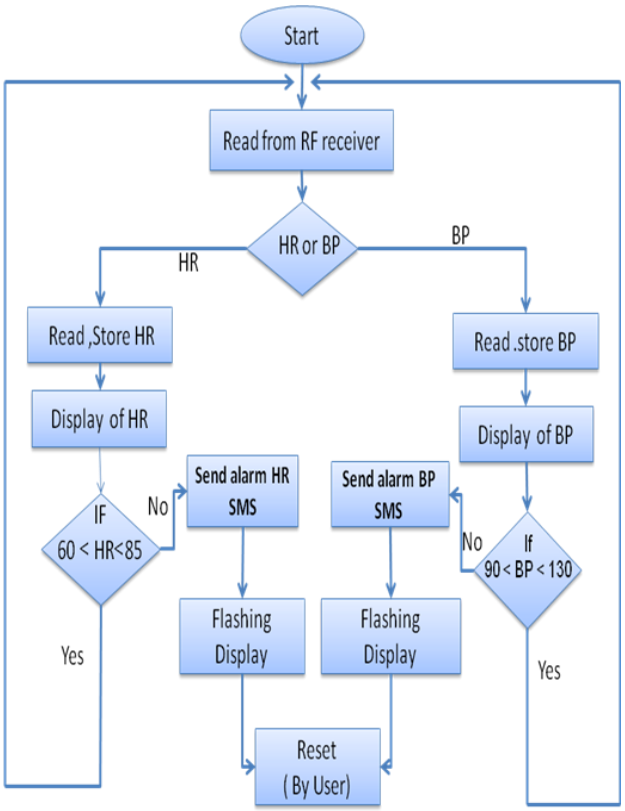


Figure 3. Simplified workflow of the control unit.

7. Conclusion and Future Work

We presented a framework for monitoring blood pressure and heart rate of patients to be used in order to automate the process monitoring the vital signs of patients in hospitals and care centers. At the same time, the proposed application can also be used on individual level in a home for a patient who needs regular reading of the vital signs. The characteristics of the proposed application include:

1. A networked application to improve monitoring blood pressure and heart rate.
2. Useful in hospitals and at home.
3. Inexpensive implementation of the control unit and the monitoring device.

An extension that is under consideration of the application is to implement a framework for remote control of the monitoring devices. The user of the server application, mainly a health professional, can control the monitoring device remotely with some predefined commands that should be used only in specific situations. These commands include two main sets:

1. Turning it on or off based on the patient record or the doctor's request.
2. Changing the frequency of readings; for example, from hourly to daily.

On another level, discussions are already underway with a current Saudi E-Health project coordinators to evaluate the proper means to integrate the proposed monitoring network application in the E-health framework that is still in its early stages.

References

[1] H. Asada. Wearable blood pressure sensor offers 24/7 continuous monitoring. In MIT news, April 2009.

[2] M. Bolanos, H. Nazeran, I. Gonzalez, R. Parra, C. Martinez. *A PDA-based electrocardiogram/blood pressure telemonitor for telemedicine*. Proceedings of IEEE Conference on Engineering in Medicine and Biology Society. 2004;3:2169-72.

[3] M. Gräfe, T. Göttsche, P. Osypka, M. Görtz, K. Trieu, H. Fassbender, W. Mokwa, U. Urban, T. Schmitz-Rode, T. Hilbel, R. Becker, B.Bender, W. Coenen, M. Fähnle, R. Glocker. *A Fully Implantable Blood Pressure Sensor for Hypertensive Patients*. Proceedings of IEEE Conference on Sensors, 2008; PP. 1226 – 1229.

[4] L. Hong, J. Prohaska, A.R. Nara, *Fiber-optic transducer for blood pressure measurements*. Proceedings of the Annual International Conference of the IEEE on Engineering in Medicine and Biology Society,1988; PP. 810-811.

[5] D. Parekh. Designing Heart Rate, Blood Pressure and Body Temperature Sensors for Mobile On-Call System. Department of Electrical and Computer Engineering, McMaster University, Canada, 2010.

[6] http://www.healthworksglobal.net/Pages/OBrien%20Collection/history.html

[7] http://en.wikipedia.org/wiki/Heart_rate_monitor

[8] http://ww1.microchip.com/downloads/en/DeviceDoc/39762f.pdf

[9] http://www.ihealthlabs.com/wireless-blood-pressure-wrist-monitor-feature_33.htm

Solar Array Positioning Controller

B. R. Shaer, N. M. Eubanks, C. A. Hayes, V. S. Ilin, and A. F. Potter
Electrical and Computer Engineering Department
University of West Florida
Shalimar, Florida, USA

Abstract: *Traditional fixed solar panel installation fails to optimize solar energy collection when the sun is not at its peak position. Maintaining a perpendicular angle with the sun optimizes the energy collected throughout the day thereby increasing overall energy collection or reducing the number of solar panels required for any given installation. This project uses sensors, motors, various printed circuit boards, and microcontrollers to maintain a perpendicular angle with the sun and determine the additional solar energy collection gained by tracking the sun's position throughout the day. Additionally, this design continuously measures the wind speed and lowers the solar panel to a flat, safe stow position if dangerous wind levels are present.*

Keywords: *Control, dynamic, efficiency, energy collection, Finite State Machine, optimization, solar*

1. INTRODUCTION

While once a rare sight, solar panels are now commonly found on roadside signs, homes, businesses, and industrial settings. Major improvements in solar panel manufacturing, lower materials cost, and government incentives have made solar technology a viable energy solution for consumers and businesses seeking to reduce their carbon footprint. Most solar panel installations however utilize sub-optimal static positions; the panel is aimed due south (in the Northern Hemisphere) with a few degrees of elevation. While the static mounting technique is generally acceptable, a dynamic installation where the solar panel tracks the sun increases total energy collection. This project controls a dual-axis tracking system that periodically adjusts the azimuth and elevation of the panel to maximize energy collection throughout the day, increasing energy production over traditional, fixed solar photovoltaic panel installation.

This system is intended to operate with two axis for daily movements with a third axis used strictly for emergency stow operations. Various implementation strategies may be used, provided motor movements account for azimuth (left-right), altitude (up-down), and emergency stow operations. This implementation expects only a single mechanical stow movement after the panel is in an appropriate azimuth-altitude position. The system should provide at least 15% energy collection improvement over fixed position installation.

2. SYSTEM COMPONENTS

Microcontroller: This project uses an Arduino Mega 2560 based on the ATmega2560 microcontroller due to flexibility and ease of implementation. The system uses digital and analog input and output pins to read from each sensor and control each mechanical component. The system requires additional hardware to control power conversion, motor control, anemometer input, and other system components.

Photo resistors: The system determines the relative position of the sun via four photo resistors placed in a diamond configuration that is mounted to the top of the solar panel. The photo resistors are divided by a small wall which casts shadows on the resistors that are furthest from the light source. Each photo resistor outputs a voltage between 0 and 5 based on the amount of available light. The system maintains a perpendicular angle with the sun by periodically moving the array to ensure each photo resistor reads the same amount of light. The Printed Circuit Board (PCB) and system-ready board used for this project are shown in Figure 1.

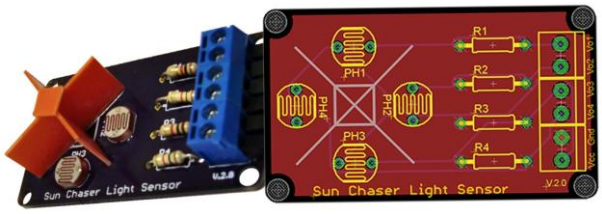


Figure 1: Photo Resistor PCB Configuration and Fully Functional, Device-Ready Board

Clock: A battery-backed crystalline Real Time Clock (RTC) is used to remove timekeeping functions from the microprocessor, which enables the use of system sleep modes for energy conservation. The Arduino microcontroller reads the time directly from the RTC when time functions are required.

Anemometer: An anemometer is used to detect dangerous wind conditions. The system periodically reads the analog anemometer output and moves to a stow preparation state if wind speed exceeds the predetermined maximum safe wind speed. signals the Arduino microcontroller via interrupt pin when excess wind loads are detected. Error checking and speed verification occur prior to signaling an unsafe wind condition to prevent false alarms. The function that reads the wind speed also sets the emergency condition detected flag, goToStow, when the appropriate conditions exist. The anemometer, PCB, and completed analog-to-TTL circuit board are shown in Figure 2.

Figure 2: Anemometer, PCB, and Completed Analog-to-TTL Circuit Board

Power System: A charge controller and battery charger was constructed to condition the power collected by the photovoltaic panel and maintain the system battery. The battery charger is a dual level float charger based on the Texas Instruments UC2906. The charge controller is based on the SolarMagic product line from Texas Instruments. The battery charger PCB and assembled circuit board are shown in Figure 3.

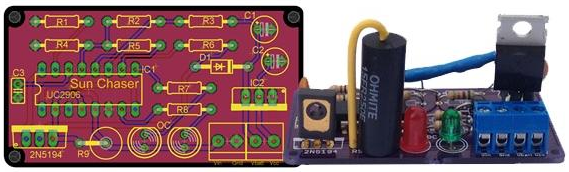


Figure 3: System Battery Charger PCB and Assembled Circuit Board

Charge Controller: The system uses a TI UC2906 based VRLA Battery Charger which provides overcharge protection while ensuring adequate current and voltage for energy collection and system use by working in two distinct modes, trickle charge and bulk charge. In trickle charge mode, the float charger outputs a low current voltage to slowly charge a depleted battery. The low current input will not cause internal heating of the battery should the battery voltage be low due to failed cells. Once the battery voltage has reached the threshold voltage the float charger switches to a bulk charge mode, generating a current of 1A. The bulk charge current continues until the battery reaches the Over Charge voltage threshold when the charge current begins to taper until approximately 0.1A. Once the bulk charge current has dropped to 0.1A, the float charger maintains the battery at the float voltage until a load is placed across the battery, dropping the voltage below the bulk charge threshold.

Battery Charger: The system charge controller used a synchronous buck-boost regulator and four MOSFET switches for low losses as compared to diodes. The SM72442 is an MPPT controller with four PWM outputs to control a four-switch Buck-Boost converter. The SM72442 uses a perturb-and-observe tracking algorithm to maintain the optimum power point of the solar panel. The SM72442 features several analog inputs to sense input current and voltage, output current and voltage, and configuration setting pins. The PWM outputs of the SM72442 are routed to the SM72295 Full-Bridge Driver. This device features internal bootstrap circuitry to drive the high-side gates, two current amplifiers for the input and output currents, and an over voltage sense input. The charge controller PCB and finished assembled circuit board are shown in Figure 4.

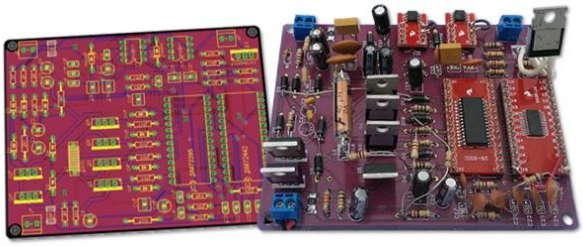


Figure 4: System Charge Controller PCB and Finished Assembled Circuit Board

Motor Control Circuits: Two optional motor controllers are available to maximize implementation flexibility. Either a stepper motors or a linear actuator may be used for each of the three motors as configured by the user at initial setup.

Bluetooth System Control: The system is fully automated, however may be controlled via a Bluetooth enabled Android device with the appropriate application installed.

3. DATA LOGGING

This system records various system data, including date and time, wind speed, system current (Amps), the brightness of light measured in each sensor, actuator arm positions, and two error codes depending on system events using a 2-dimensional array. This system records available data at predetermined intervals and upon detecting errors. A separate data logging function is called to record the following data elements. The calling function passes the error codes to this function; normal operations data logging calls pass zeros, error conditions determines and passes the appropriate values

at function call. Each data point in the array uses 16-bit signed integers allowing for data ranges from -32,768 to 32,767. In order to minimize space, data compression is utilized on date and time fields. The system records year, month, day, hour, minute and second but is capturing this data on three integers:

- Int 0, yearMonth: the first three decimal places record the year as offset from the year 2000; the last two decimal places record the two-digit month. This data logging allows for time tracking through the end of year 2327 AD.
- Int 1, dayHour: the first decimal place is used as a normal operations state bit; 0 indicates daytime state; 1 indicates end of day state; 2 indicates start of day state; and 3 indicates nighttime state. The next two most significant decimal places capture the day of the month; the last two decimal places capture the hour from 0 to 23.
- Int 2, minuteSecond: the first decimal place is not currently used but is available for future use if necessary; 0 is always written to this decimal place. The next two most significant decimal places record the minutes from 0 to 59; the last two decimal places record the seconds from 0 to 59.
- Int 3, mph: this integer records the wind speed as identified by the anemometer using TTL. The two most significant decimal places are currently unused and always record 0; the may be available for additional use in the future.
- Int 4, unnamed: records the current generated by the solar panel from 0 to 1,023.
- Int 5 through int 8, photo resistor readings: These ints record the brightness measured by the photo resistors from 0 to 1,023.
- Int 9 through int 11, actuator arm positions: These ints record the resistance measured by the actuator arm position from 0 to 1,023.
- Int 12 through int 13, error bits: The system currently allows for two error codes. Both bits 0 if no error condition exists, e.g., standard data logging at scheduled daytime intervals or start of day or end of day.
- The first error code is a sequential number indicating the exact error condition. Various potential error conditions can occur throughout the Finite State Machine (FSM); for example, if in Normal Operations mode, only one of the four normal operations condition flags should be set. If all four are cleared, then the system does not know what the current state is. A separate state correcting function exists to correct this condition; if the condition occurs, a data logging function is called and the appropriate error code logged. The error codes are unique to the state and function that initiated the data logging event minimizing troubleshooting time.
- The second error code has a variable function. In most cases, it records the values of all state flags as a binary number stored as an int. This helps troubleshooters know the exact condition flags when the error occurred.

4. SYSTEM CONTROL

This system is controlled by a Finite State Machine (FSM) shown in Figure 5 which tracks the current state and moves to subsequent steps depending on various system readings and surrounding conditions depending on the state of the system flags. The system tests all available measurements and then follows this FSM according to the conditions of the various flags as depicted in the FSM. Excluding stow conditions, the system determines the appropriate state, executes any required movements to achieve the desired position, then goes to a sleep mode. The process of measuring, calculating, moving, verifying, and adjusting as needed generally only requires 5 to 15 seconds; the system then sleeps for 5 to 30 minutes depending on the user's desired configuration. The wind setting is triggered via interrupt pin to minimize system power consumption.

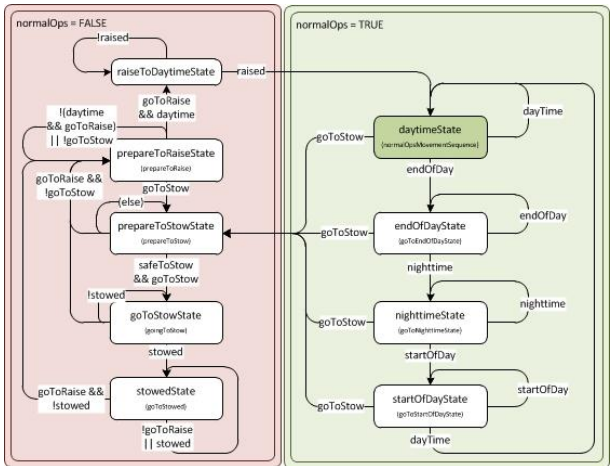


Figure 5: System Finite State Machine Diagram

4.1 SYSTEM STATES

This system is controlled via a single main loop that controls the sequence of events, special conditions, and system sleep states. The main loop calls a function that controls all the sensor functions; each sensor function subroutine sets the system flags that determine what actions need to be executed. The main loop then calls the FSM function which reads all system variables and determines which functions need to be called in predefined sequences. The system uses ten global

control and condition bits used to control the system state:

- normalOperations: This bit is initialized to TRUE at system startup; it is set during routine operations or return from stow conditions and is cleared whenever unsafe wind conditions are detected. The FSM is divided into two sub-state machines controlled by the status of this bit. The FSM function receives a local copy of this variable to ensure data integrity.
- daytime: This bit is initialized to TRUE at system startup; it is set when adequate daylight exists for solar energy collection and cleared when light fades at the end of the day.
- endOfDay: This bit is initialized to FALSE at system startup; it is set when light fades at the end of the day and cleared upon transition to nighttime state. This bit and the related state are expressly used for data logging.
- nighttime: This bit is initialized to FALSE at system startup; it is set upon transition from the end of day state and cleared upon detection of adequate light to collect solar energy. The related state checks for adequate lighting, wind speed, and then puts the system in a sleep mode.
- startOfDay: This bit is initialized to FALSE at system startup; it is set upon detection of adequate light and cleared upon transition to daytime state. This bit and the related state are expressly used for data logging.
- goToStow: This bit is initialized to FALSE at system startup; it is set and cleared by the function that calls wind speed measurement and can therefore be changed in any system state. If the system is in normal operations, the normalOperations flag is cleared as goToStow is set.
- safeToStow: This bit is initialized to FALSE at system startup; it is set when the solar panel has completed movement to a safe to stow position and cleared upon initial movement out of this position in the raiseToDaytimeState. The purpose of this bit is to ensure the solar panel is protected from potential damage by the stow motor movement. The solar panel remains at stow position until the unsafe wind conditions pass and the daytime bit is set, preventing energy use before adequate sunlight exists for energy collection.
- stowed: This bit is initialized to FALSE at system startup; it is set when the solar panel reaches the full stow position and cleared immediately upon movement from the stow position.
- goToRaise: This bit is initialized to FALSE at system startup; it is set upon the departure of unsafe wind conditions and cleared upon the detection of unsafe wind conditions.
- raised: This bit is initialized to FALSE at system startup; it is only set when the solar panel has made full recovery from stow conditions and is cleared after detection of unsafe wind conditions.

The system is divided into two major states, one for normal operations and another for emergency stow conditions when wind load may potentially damage the solar panels. The state function controls the system based on the logic in the flowchart depicting the FSM as shown in Figure 1.

4.2 NORMAL OPERATIONS

The system spends the vast majority of its time in the Normal Operations state. The system transitions between these various states based on the time of day and available light.

- Daytime State: The daytime state operations (right side of the FSM diagram) periodically sample light readings in the four photo resistors to determine if the panel needs to move. The panel then adjusts position until the panel is perpendicular to the sun, controlling all of the solar tracking operations. The system checks time from the external clock at each iteration through the program execution and initiates the data logging function at user-designated predetermined intervals. Global variables control the sampling frequency and can be adjusted by the user as needed.
- End of Day State: When the system detects inadequate lighting at dusk, the machine transitions to an end of day state where the data logging function is called before transitioning to nighttime, sleep sate.
- Nighttime State: The nighttime state is a system sleep state. It periodically samples the wind speed and measures any available sunlight. The system transitions to a start-of-day state when the photo resistors detect adequate lighting.
- Start of Day State: Before the system transitions back to a full daytime state, it calls the data logging function to record system information at dawn.

4.3 EMERGENCY STOW OPERATIONS

Upon detection of unsafe wind speeds, the goToStow bit is set and the normalOperations bit is cleared. If wind speeds fall, the system clears the goToStow bit; this triggers the return from stow sequence. The normalOperations bit remains cleared until the system has fully recovered from stow

conditions, including returning the panel to a pre-stow position. the state diagram was designed to ensure the preservation of the panel as the solar panel is mounted on an adjustable platform and can be damaged if the panel is not in the correct position when the stow motor is activated. The system calls the data logging state whenever the system transitions to or from any normal operations states maximizing end users understanding of system operations during unsafe conditions.

- Prepare To Stow State: The system immediately moves to the prepare to stow state when the goToStow bit is set. This state is designed to move the solar panel to a safe position prior to calling the stow motor function. Since an actuator motor was used for vertical movement, the position of the actuator arm can be routinely measured. This state operates in a loop checking the position of the arm at each loop iteration; once the arm exceeds a minimum extension and stops moving, the function controlling the actuator sets the safeToStow bit. The loop terminates and returns to the main function. The next iteration through the FSM function changes the state to Go To Stow State.
- Go To Stow State: This state controls the stow motor. Triggers are present at the fully raised and fully lowered positions; this state continues to engage the stow motor until the stowed trigger is hit. Additionally, the system checks for premature stow switch activation and recalls the stow function if the switch is disengaged. Once the switch is engaged, the stowed bit is set. The system moves to the Stowed State upon its next iteration through the FSM function.
- Stowed State: This state calls the wind measurement function to monitor the wind speed. The wind measurement function triggers a system return from stow when the unsafe wind condition passes, clearing the goToStow bit and setting the goToRaise bit. Redundancy is used with these two bits to increase system safety. If unsafe wind speeds persist, this function activates a sleep mode for a designated period of time.
- Prepare To Raise State: This state maintains a stow position until the daytime bit is set; this prevents the system from returning to stow when the system will not collect any energy at nighttime. This prevents wasted energy should unsafe wind conditions occur, leave, and reoccur during the night. Once the daytime bit is set and the goToRaise bit is set, the system transitions to the Raise To Daytime State where the panels return to a pre-stow position.
- Raise to Daytime State: This state ensures the system recovers from a full-stow position prior to moving the panel's up-down and left-right motors by activating the stow motor in reverse direction and monitoring the stow recovery switch. The motor stops once this switch is triggered; this state then reads the pre-stow positions, returning the panel to this position. Once the panels return to the defined position, this state sets the Normal Operations bit.

The wind speed measurement function triggers a system recovery from stow when unsafe wind conditions pass; this may occur during Prepare To Stow State, Go To Stow State, or Stowed State. If unsafe wind conditions reoccur while in this state, the wind measurement function will reset the goToStow bit and the system will return to the Prepare To Stow State. Race conditions are a potential risk associated with these states, so the wind speed measurement function utilizes redundancy to prevent false readings.

All system components worked as expected; the system is stable and consistent per design intent. The finished system is shown in Figure 6.

Figure 6: Finished Solar Tracker System

5. RESULTS

Testing centered around current measurement and average voltage readings with the panel at three positions:

- Flat, "tabletop" position
- Traditional fixed-roof position at 30° from level facing due south
- Dynamic, solar-tracked position

The system average voltage and current readings at fixed points in time are compared to determine the energy collection improvement using a dynamic position over traditional rooftop positions. The tabletop position is included as an arbitrary reference point. Motor activation and electronics power consumption are deducted from the dynamic position readings and excluded from stationary position readings. Testing occurred on a single, sunny day. The panel was setup in a well-exposed yard and current measurements were periodically collected at each of the three different positions throughout the day. The resulting measurements were plotted using MS Excel and are shown in Figure 7.

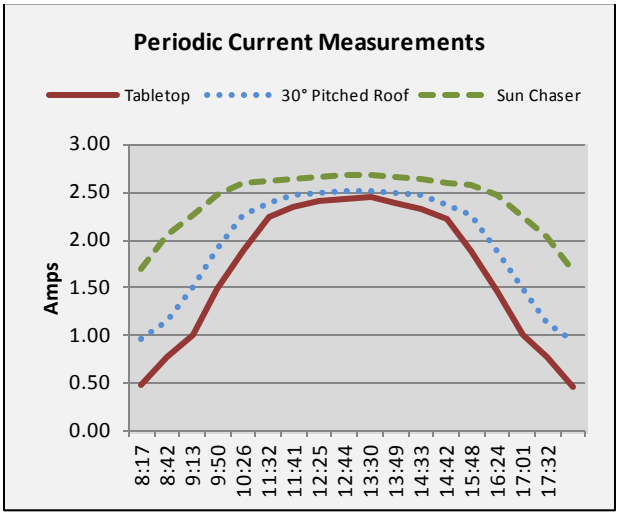


Figure 7: System Test Current Readings at Three Different Positions

Stationary data points which were used to generate the graph in Figure 7 are shown in Table 1.

Table 1: Power Collection and Comparison

Time	Tabletop	30° Pitched Roof	Sun Chaser	% Immediate Improvement (vs. Tabletop)	% Immediate Improvement (vs. 30° Pitched Roof)
8:17	0.48	0.96	1.70	71.76%	43.53%
8:42	0.78	1.15	2.05	61.89%	43.90%
9:13	1.02	1.50	2.27	55.26%	33.92%
9:50	1.48	1.91	2.47	39.90%	22.51%
10:26	1.88	2.27	2.60	27.82%	12.78%
11:32	2.24	2.38	2.62	14.50%	9.05%
11:41	2.34	2.47	2.64	11.36%	6.44%
12:25	2.40	2.50	2.66	9.65%	5.88%
12:44	2.43	2.52	2.68	9.33%	5.97%
13:30	2.44	2.51	2.69	9.29%	6.69%
13:49	2.38	2.49	2.66	10.29%	6.26%
14:33	2.32	2.48	2.63	11.79%	5.70%
14:42	2.23	2.36	2.60	14.29%	9.15%
15:48	1.88	2.26	2.58	27.33%	12.40%
16:24	1.46	1.88	2.46	40.67%	23.61%
17:01	1.01	1.48	2.25	55.03%	33.91%
17:32	0.77	1.13	2.03	62.09%	44.23%
17:57	0.46	0.94	1.68	72.61%	44.19%

The available energy was also collected while the motors were running, resulting in residual power at runtime comparable to the tabletop position. The team conservatively estimated total motor and system daily runtime and calculated total expected daily Sun Chaser power consumption; the results are shown in Table 2. The net energy collection improvement is only slightly less than the gross energy gain due to the extremely limited motor runtime and copious use of system sleep states.

Table 2: Power Consumption Measurements

Component	Power consump-tion (hours)	Amp rating	System Voltage	Max Watts Consump-tion	Watt Hr	Total (Runtime = 9 hr 40 min)
Actuator Motor (0.6 Amp/hr, 15 sec/hr)	0.00417	0.60	12.93	7.76	0.03	0.31256
Stepper Motor (1 Amp/hr, 1 min/hr)	0.01667	1.00	12.93	12.93	0.22	2.08370
Other system power (measured, 1.5 min/hr)	0.02500	1.17	12.93	15.11	0.38	3.65065
Total Power Consumption					0.63	6.05

The dynamic solar tracking system successfully completed its initial proving phase showing 16.96% energy collection improvement over the traditional fixed rooftop position (30° sloped roof) and 29.24% improvement over flat, "tabletop" panel position as shown in Table 3 below.

Table 3: Power Collection and Comparison

	Avg Amps	Avg Volts	Gross Watts	Power Consumption (Watts)	Net Watts	% Difference
Flat, fixed	1.67	12.93	208.41	N/A	208.41	29.24%
30° roof, fixed	1.96	12.93	244.59	N/A	244.59	16.96%
Dynamic Position	2.40	12.93	300.44	6.05	294.39	N/A

This promising improvement to existing solar technology will reduce the installation return on investment period by increasing power returns or reducing the number of required solar panels required for the desired amount of energy collection. The concept is now ready for further development and optimization in preparation for product deployment.

6. CONCLUSIONS AND RECOMMENDATIONS

The work in this paper successfully demonstrated expected energy collection improvement. Fixed position solar panels fail to optimize solar energy

collection; dynamic positioning control corrects for this by maintaining a perpendicular angle between the solar panel or array and the sun.

Additional potential system refinements include:

- Solar array position detection: Incorporating an accelerometer/magnetometer may increase system control over the solar panel position, adding a layer of redundancy to further protect the solar panel from potential damage in stow conditions.
- Data points: The current implementation utilizes a large data element to compress date, time, and time of day state information. Dynamic memory allocation may be more suitable long-term solution. For example, most analog inputs read values between 0 and 1,023, so a 10-bit unsigned integer would suffice in those scenarios. This is not practical since data is stored in Byte-sized elements; 32-bit integers however could be used to store up to three 10-bit unsigned integers. Conversely, the 10-bit integers could be reduced to 8-bit unsigned integers by performing a 2-bit shift right, effectively dividing the reading by 4. Reading from these fields would require a 2-bit shift left resulting with a slight loss in data granularity.
- The current implementation uses flash memory; writing to a data file on a SD card would allow for larger data storage ranges.
- Additional data compression algorithms would further improve data storage and slightly reduce power consumption at data logging events.
- Optional data monitoring via integrated Bluetooth application: The data storage can be transmitted to Android-based mobile platforms since the system includes Bluetooth technology.
- Optional Wi-Fi connectivity: This convenience option would allow consumers to read system data and various conditions from a desktop or laptop computer without directly accessing the system. Power consumption increase would need to be studied prior to implementing this potential feature.
- Optional LCD readout for data retrieval: This convenience option would primarily be used at system setup and initialization but may be beneficial in some applications.

7. References

1. Anderson, A. E., et al. Design of the Support Structure, Drive Pedestal, and Controls for a Solar Concentrator Electronic Resource]. Washington, D.C., Oak Ridge, Tenn, United States. Dept. of Energy; distributed by the Office of Scientific and Technical Information, U.S. Dept. of Energy, 1991. Print.
2. Cook, G., et al. Shining on Electronic Resource]: A Primer on Solar Radiation Data. Washington, D.C., Oak Ridge, Tenn, United States. Dept. of Energy; distributed by the Office of Scientific and Technical Information, U.S. Dept. of Energy, 1992. Print.
3. Hammons, Burrell E., and United States Dept of Energy Office of Scientific and Technical Information. Solar Tracking Apparatus Electronic Resource]. Oak Ridge, Tenn., Oak Ridge, Tenn, United States. Dept. of Energy. Office of Scientific and Technical Information; distributed by the Office of Scientific and Technical Information, U.S. Dept. of Energy, 1980. Print.
4. Howell, B., et al. Solar Kinetics` Photovoltaic Concentrator Module and Tracker Development Electronic Resource]. Washington, D.C., Oak Ridge, Tenn, United States. Dept. of Energy; distributed by the Office of Scientific and Technical Information, U.S. Dept. of Energy, 1995. Print.
5. Jones, S. A., et al. Analysis of Solar Two Heliostat Tracking Error Sources Electronic Resource]. Washington, D.C., Oak Ridge, Tenn: United States. Dept. of Energy; distributed by the Office of Scientific and Technical Information, U.S. Dept. of Energy, 1999. Print.
6. ---. Analysis of Strategies to Improve Heliostat Tracking at Solar Two Electronic Resource]. Washington, D.C., Oak Ridge, Tenn, United States. Dept. of Energy; distributed by the Office of Scientific and Technical Information, U.S. Dept. of Energy, 1999. Print.
7. Seme, S., G. Štumberger, and J. Voršič. "Maximum Efficiency Trajectories of a Two-Axis Sun Tracking System Determined Considering Tracking System Consumption." Power Electronics, IEEE Transactions on 26.4 (2011): 1280-90. Print.
8. Shingleton, J., et al. One-Axis Trackers -- Improved Reliability, Durability, Performance, and Cost Reduction; Final Subcontract Technical Status Report, 2 may 2006 - 31 August 2007 Electronic Resource]. Washington, D.C : Oak Ridge, Tenn, United States. Dept. of Energy ; distributed by the Office of Scientific and Technical Information, U.S. Dept. of Energy, 2008. Print.

Implementation of IEEE 1588v2 PTP for Precision Clock Synchronization of Ethernet Network

Seongjin KIM[1], Kwangman KO[1], and Kyoungyoung SO[2]
[1]School of Computer Information Engineering, SANGJI University, KOREA
[2]Department of Software Engineering, Chonbuk National University, KOREA

Abstract - *A common sense of time among all the elements of a distributed measurement and control system allows the use of new techniques for the solution of problems with complex synchronization requirements or arising from the interaction of many sensors and actuators. Such a common sense of time may be accomplished using IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems (IEEE 1588-2008) to synchronize real-time clocks incorporated within each component of the system. In this paper, we implemented the IEEE 1588v2 PTP emulator on BlueScope BL6000A using delay request-response mechanism to estimate a clock synchronization.*

Keywords: IEEE 1588, Precision Time Protocol, Time Synchronization, Delay request-response mechanism

1 Introduction

The technologies using the measurement and control system are essential to network-based control system to synchronize the distributed devices on distributed computing environment[1]. The clock synchronization method using NTP [2] and SNTP [3] is most commonly utilized in managing/controlling a system using network communication in a distributed system environment. The clock synchronization method based on signal and location using GPS satellite is different from network-based synchronization method such as NTP and SNTP because it can perform synchronization more accurate than network-based synchronization method; however, it has a disadvantage that more cost is required to provide with devices such as receiver and antenna to receive GPS signal[4]. Therefore, a technology is required for more accurate clock synchronization, which can synchronize existing device at low cost but still secures accuracy with error-range of sub-microseconds and uses existing bandwidth without building a new network configuration or adding a new device. In this paper, we implemented a function that can emulate IEEE 1588v2 PTP [5] using delay request-response mechanism in the measurement of accurate clock synchronization of packet based network using PTP. For this, a function, which divides emulator into Master(Requester) and Slave(Responder) nodes, creates detail PTP messages for clock synchronization and transmits them, and a function, which provides with information and statistical values on the received PTP messages, are implemented on BlueScope BL6000A.

2 Baseded Works

The precision time protocol is a protocol for measuring high-ly precise clock synchronization using packet-based network. The IEEE 1588 standard which define precision time protocol have been published in 2002, IEEE 1588v1 [6], and in 2008, IEEE 1588v2. IEEE 1588v2 is composed of master and slave nodes and exchange messages based on timestamp to determine the offset and path delay between two clocks.

2.1 Delay Request-Response Mechanisms

The delay request-response mechanism measures the <mean-PathDelay> between a pair of PTP ports and uses the messages Sync, Delay_Req, Delay_Resp and possibly Follow_Up as shown in the timing diagram of Figure 1. This mechanism is required to be executed independently in each supported domain of the two clocks.

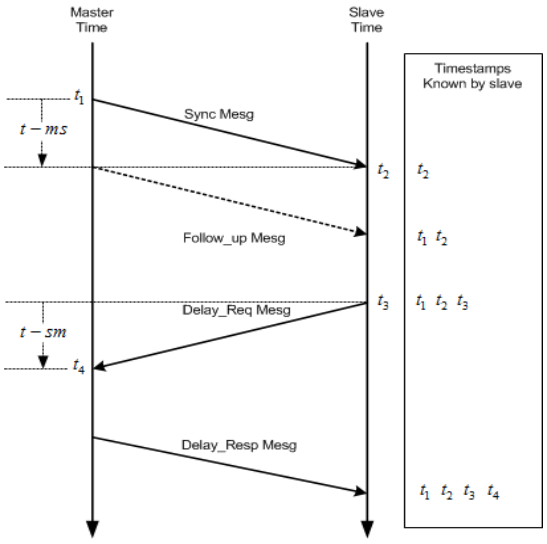


Figure 1. Delay Request-Response path length measurement

The actual value of the path Delay is measured and computed as follows for each instance of a delay request-response measurement:

a) The master node prepares and issues a Sync message. If the node is a two-step clock, it also prepares and issues a Follow_Up message
b) The slave node is required to :
- Generate timestamp t2 upon receipt of Sync message from master node.
- Send the Delay_Req message and generate and save timestamp t3.
- If the received Sync message indicated that a Fol-low_Up message will be received, save timestamp t1 from Follow_Up message.
- If the received Sync message indicated that a Fol-low_Up message will not be received, save timestamp t1 from Sync message.
c) The master node is required to :
- Generate timestamp t4 upon receipt of the Delay_Req message from slave.
- Set timestamp field of Delay_Resp message to the timestamp t4.
- Issue the Delay_Resp message.
d) Upon receipt of the Delay_Resp message by the slave, <meanPathDelay> and <offsetFromMaster> are computed as following (1),

$$meanPahtDelay = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \qquad (1)$$

$$offsetFromMaster = \frac{(t_2 - t_1) - (t_4 - t_3)}{2}$$

2.2 Peer Delay Mechanism

In the two-step mode, propagation time between two nodes is measured through the exchange of three messages (Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up) and its process as shown in Figure 2.

a) Delay Requestor Node-A sends Pdelay_Req message, and creates and saves time stamp.
b) Delay Responder Node-B that has received Pdelay_Req message undergoes the following process.
- Immediately upon receiving Pdelay_Req message from the Delay Requestor Node-A, time stamp is created.
- Copies the correctionField of Pdelay_Req message to the correctionField of Pdelay_Resp_Follow_Up message, and sets the correctionField of Pdelay_Resp as 0.
- Copies the sequenceId field of Pdelay_Req to the sequenceId field of Pdelay_Resp message and Pdelay_Resp_Follow_Up message.
- Copies the sourcePortIdentity field of Pdelay_Req to the requestingPortIdentity field of Pdelay_Req message and Pdelay_Resp_Follow_Up message.
- Copies the domainNumber field of Pdelay_Req to the domainNumber field of Pdelay_Req message and Pdelay_Resp_Follow_Up message.
- Sets the requestReceiptTimestamp field of Pdelay_Resp message as 0.
- Sends Pdelay_Resp message to the Delay Requestor Node-A, and creates and saves time stamp .
- Sets the responseOriginTimestamp field of Pdelay_Resp_Follow_Up message as 0, and after adding turnaround time to the correctionField field, it is sent to Delay Requestor Node-A.
c) Delay Requestor Node-A creates and saves time stamp immediately upon receiving PDelay_Resp message.

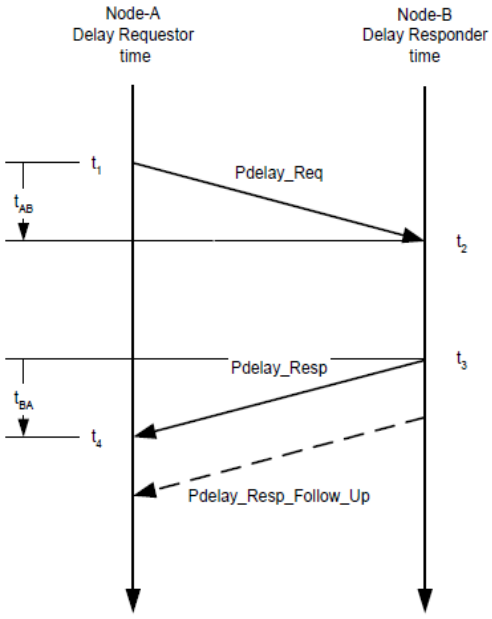


Figure 2. Peer Delay Mechanism

After exchanging messages, Delay Requestor Node-A will have two time stamps and two message field values. With these time stamps and field values that have been saved, the Mean Path Delay time between communication ports can be calculated by applying Formula (2).

$$t_4 - t_1 \cdots ①$$

$$responseOriginTimestamp - requestReceiptTimestamp \cdots ②$$

$$correctionField\ of\ Pdelay\_Resp - correctionField\ of\ Pdelay\_Resp\_Follow\_Up \cdots ③$$

$$meanPahtDelay = \frac{① - ② - ③}{2} \qquad (2)$$

In one-step mode, Pdelay_Resp_Follow_up message is not transmitted, as shown in [Figure 2]. Accordingly, turnaround time that was delivered through Pdelay_Resp_Follow_up message is saved to the correctionField of Pdelay_Resp message for transmission.

a) Delay Requestor Node-A sends Pdelay_Req message, and creates and saves time stamp .

b) Delay Responder Node-B that has received Pdelay_Req message undergoes the following process.

- Immediately upon receiving Pdelay_Req message from the Delay Requestor Node-A, time stamp is created.
- Copies the sequenceId field of Pdelay_Req message to the sequenceId of Pdelay_Resp message.
- Copies the sourcePortIdentity field of Pdelay_Req to the requestingPortIdentity field of Pdelay_Req message.
- Copies the domainNumber field of Pdelay_Req to the domainNumber field of Pdelay_Req message.
- Copies the correctionField of Pdelay_Req to the correctionField of Pdelay_Resp.
- Sets the requestReceiptTimestamp field of Pdelay_Resp message as 0.
- Sends Pdelay_Resp message to the Delay Requestor Node-A, and creates and saves time stamp .

After exchanging messages, Delay Requestor Node-A will have two time stamps and correctionField value of Pdelay_Resp message. With these time stamps and field value that have been saved, the Mean Path Delay time between communication ports can be calculated by applying Formula (3).

$$meanPahtDelay = \frac{(t_4 - t_1) - correctionField\ of\ Pdelay\_Resp}{2} \quad (3)$$

3 Implementations and Test of IEEE 1588v2 PTP

3.1 BlueScope BL6000A

BlueScope BL6000A is a portable all-in-one type measur-ing device which can test 1Gbit/10Gbit Ethernet, SDH/SONE-T, OTN and fiber channel. As seen in Figure 3, only one, either the Master or the Slave, can be emulated in the implemented IEEE 1588v2. When emulation starts, the PTP port state is determined by the receipt of an Announce message. All PTP messages transmitted/received by TX/RX are analyzed in detail. Then the offset for statistical data and average path delay time are calculated and calibrated.

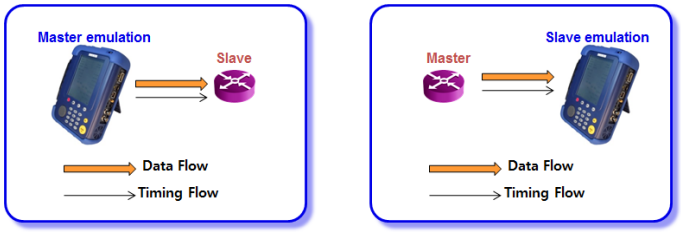


Figure 3. Emulation Environment using BlueScope BL6000A

Figure 4 is TX screen to set up and transmit PTP messages at Master and Slave. The TX PTP step is used to determine the mode of clock synchronization. It is possible to select two-step or one-step. TX PTP message provides with function which enables the setting of Announce, Sync, Follow_up, De-lay_Resp messages at Master and Delay_Req message at Slave. It also provides with function, by which user can direct-ly set up editable data, if necessary. TX PTP Interval of Mas-ter can select transmission cycle of PTP message to Slave and can set up within the range of 125ms~16secs. RX Announce Interval of Slave can select transmission cycle of the announce message from Master. If an announce message would not be received within the RX announce interval, the PTP port state is changed.

Figure 4. Screenshot of TX PTP Configuration

The TX screen to set & transmit the PTP message of Delay Requestor or Delay Responder is as shown in Figure 5. Peer Delay Step is used to select operation mode and it supports two-step mode and one-step mode. TX PTP Interval is to select the transmission interval of PTP message that will be sent to Delay Requestor and it can be set within the range of . It also provides a function of setting Domain Number that is the header information of PTP message.

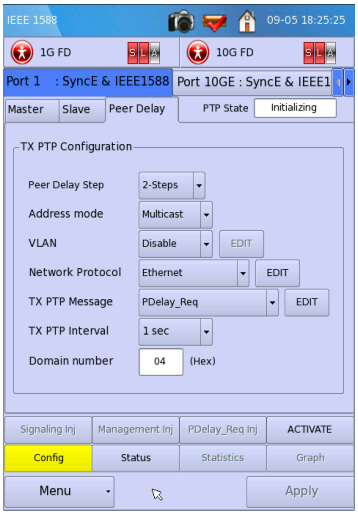


Figure 5. Screenshot of TX PTP Configuration for Peer Delay

Additionally, it also provides a function of setting Address Mode, VLAN, Network Protocol, and a function of allowing user to directly set data that can be edited, as well as showing information by classifying each message into MAC, VLAN, IP/UDP, Header & Body, as shown in Figure 6.

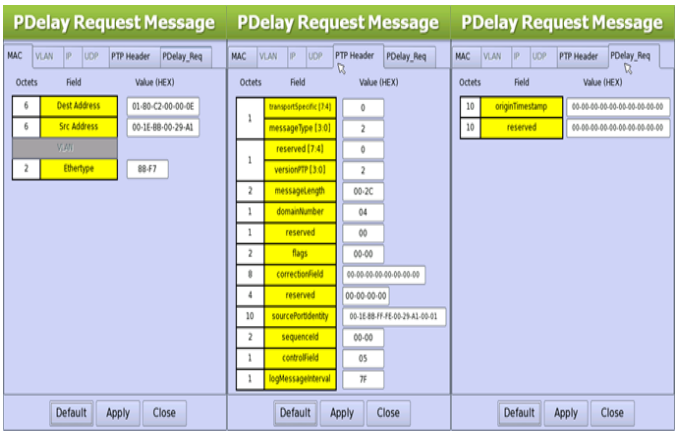


Figure 6. Screenshpt of TX PTP Message (PDelay_Req)

As seen in Figure 7 and Figure 8 the state information of all PTP messages transmitted/received for clock synchronization verification is provided separately for TX and RX. TX PTP and RX PTP show the numbers and ratios of transmitted PTP messages and received PTP messages. The Domain Mismatch compares the domain number in the header of received PTP message. If the domain number is different, it increases the number of domain mismatch and the received PTP messages are ignored.

Figure 7. Screenshot of Master and Slave Status

In the Peer Delay Status information, it shows the statistics on the Mean Path Delay Time between Delay Requestor and Delay Responder, in addition to TX/RX PTP and Domain Mismatch information.

3.2 Experimental Results of Mechanism Test

In order to build an emulation environment to test clock synchronization between Master and Slave, two BlueScope BL6000As and one BlueScope BL1400A, which is to confirm the accuracy of PTP messages, are used. BlueScope BL1400A

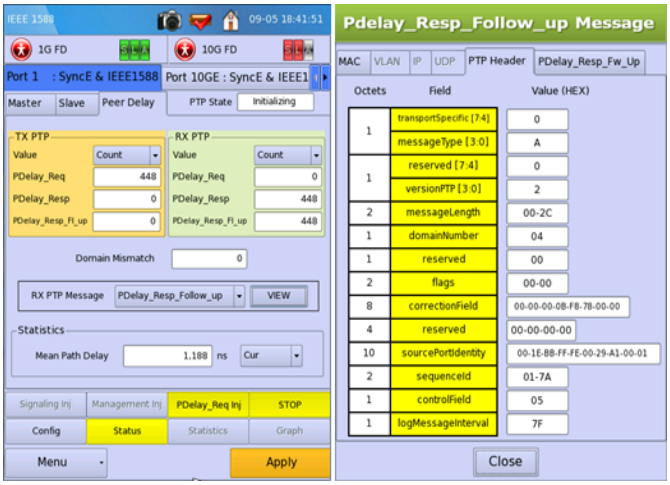


Figure 8. Screenshot of Peer delay Status and RX PTP Message Viewer

is connected by optic 1G using SFP module (wave length: 1310nm, vendor: FINISAR Corp.) between two BlueScope BL6000As as following Figure 9.

BlueScope BL1400 is a portable all-in-one measuring device, which can test 1Gbit/10Gbit Ethernet and it provides with functions to test throughput/throughput32, IP tools, monitoring, RFC2544 and PBB. In a test, it is used to capture the received packets by its monitoring function and analyze the packet information such as size, decode and Hex code.

Figure 9. Back-to-back Test Configuration

3.2.1 Delay Request-Response Mechanism Tests

Master and Slave are arranged for test as shown in Table 1 and the Master is activated for PTP message transmission.

Table 1. TX Configuration Master and Slave

	Configuration	Value
Master TX	PTP Step	Two-steps
	PTP Interval	1 second
	Domain number	04 (Hex)
Slave TX	Announce Interval	1 second
	Domain number	04 (Hex)

The slave receives PTP messages transmitted from Master, store timestamps and analyze them. Then it begins message exchange for clock synchronization. All timestamps used are provided by the hardware in the measuring device for accurate clock synchronization [7].

As seen in Figure 10, when synchronization begins, the Slave shows schematized graphs together with various statistical data using timestamps. In the statistics screen, timestamp t1~t4 time information, Master to Slave Delay, Slave to Master Delay, meanpathDelay, Offset from Master, Sync PDV(Packet Delay Variation)/IPDV(Inter-Packet Delay Variation) [8] and Delay_Req PDV/IPDV are shown in nanoseconds.

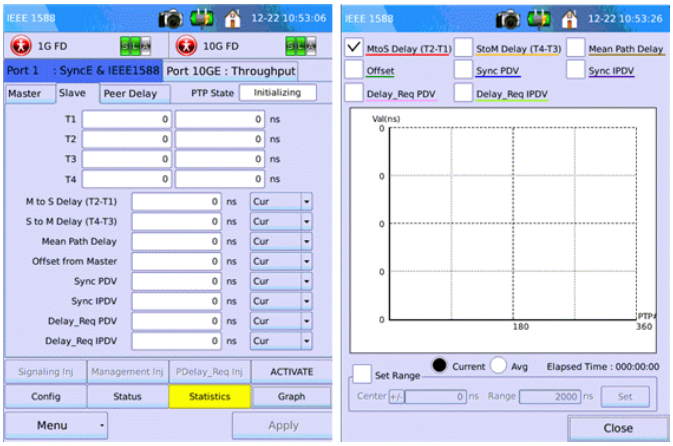


Figure 10. Slave Side of Statistics and Graph Screenshot

In the graphic screen, it is possible to see elapsed times and multiple graphs in one screen by selecting multiple statistical data. Designation of indicating range is also possible.

We performed clock synchronization to test delay request-response mechanism by exchanging PTP messages 1100 times on two-step and one-step mode respectively. In the result, as follows Figure 11 and Figure 12, master and slave have been synchronized after exchanging PTP messages approxi-mately 30 times and then kept to synchronization with error tolerance of -50ns~+50ns.

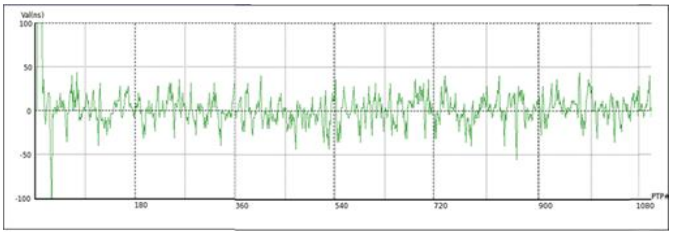


Figure 11. Offset from master of Two-step

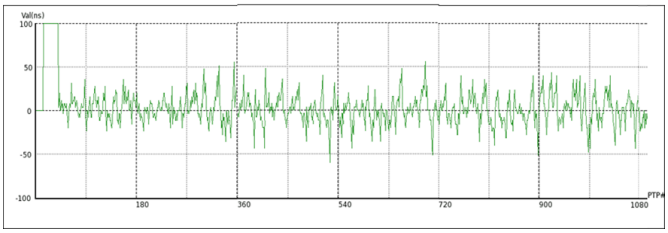


Figure 12. Offset from master of One-step

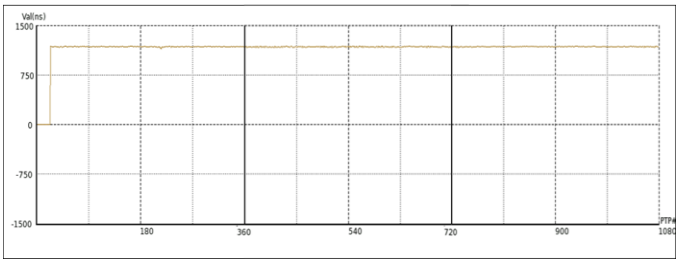
3.2.2 Peer Delay Mechanism Tests

We are arranged for Peer-to-peer test as shown in Table 2 and activated for PTP message transmission.

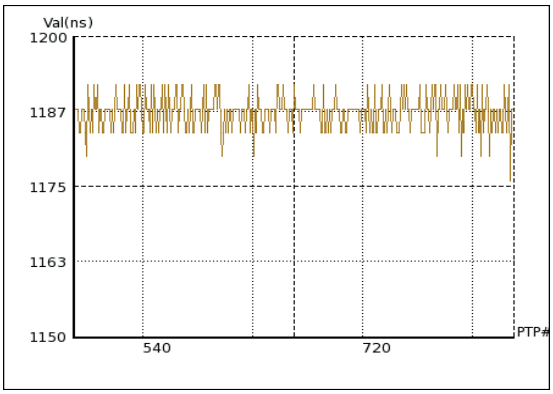
Table 2. TX Configuration of Requestor

	Configuration	Value
Requestor	Peer Delay Step	Two-step/One-step
	Address mode	Multicast
	VLAN	N/A
	Network Protocol	Ethernet
	TX PTP Interval	1 second
	Domain number	04 (Hex)

For Peer Delay Mechanism test, a test was conducted on the propagation time between communication ports by exchanging PTP messages 1,080 times per mode, and the result is as shown in Figure 13 and Figure 14. Figure 13 and Figure 14 show the graphs of Mean Path Time measured respectively in two-step mode and one-step mode. In addition, (b) of Figure 13 and (b) of Figure 14 are the graphs that show the Mean Path Delay Time in certain intervals in expansion. It was confirmed through the test result that Mean Path Delay Time was maintained within a certain range after PTP message was exchanged between Delay Requestor and Delay Responder.

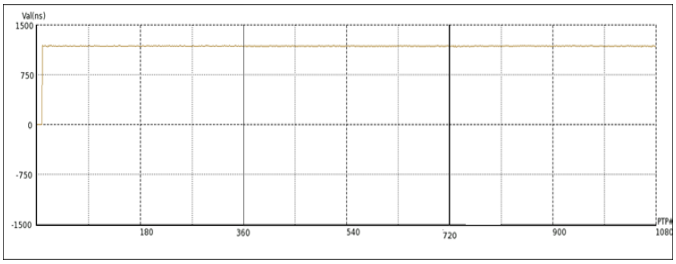


(a) Average Mean Path Delay Graph

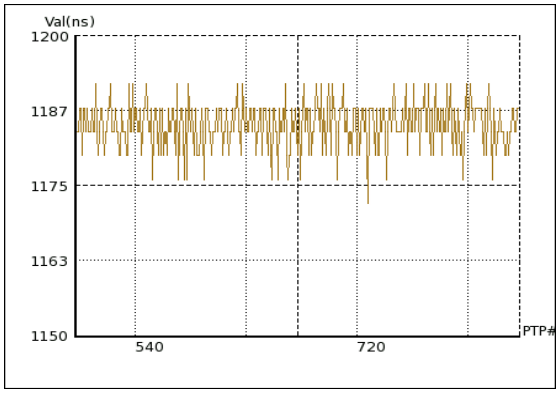


(b) Detail Graph

Figure 13. Average Mean Path Delay of Two-stop



(a) Average Mean Path Delay Graph



(b) Detail Graph

Figure 14. Average Mean Path Delay of One-stop

4 Conclusion and Future Study

This study implemented a function that can emulate IEEE 1588v2 PTP using delay request-response mechanism for accurate clock synchronization of packet based network using PTP. For this, a function, which divides emulator into Master (requester) and Slave (responder), creates detail PTP messages for clock synchronization and transmit them, and a function, which provides with information and statistical values on the received PTP messages, are implemented on BlueScope BL6000A, which is a measuring device of Bluelight Technol-ogy. This study enabled more accurate clock synchronization with improved degree of precision by providing all timestamps at the hardware step.

In the future, in order to support the function of IEEE 1588v2 in detail, a Peer delay mechanism function, which can measure peer-to-peer propagation time by way of Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_up messages exchange, will be provided. Also, a function, which can set up VLAN, UDP over IPv4 and UDP over IPv6 network protocol, will be provided in the future in order to expand the PTP packet by multicast and unicast address.

5 Reference

[1] J. C. Eidson, "Measurement, Control, and Communication Using IEEE1588," New York: Springer-Verlag, Apr 2006.

[2] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification," RFC 5905, Jun 2010.

[3] Mills, D., "Simple Network Time Protocol(SNTP) Version 4 for IPv4, IPv6 and OSI," RFC 2030, Oct 1996.

[4] Lewandowski W., Azoubib J., Klepczynski W.J., "GPS: Primary Tool for Time Transfer," Proceedings of the IEEE, Vol. 87, No. 1, pp. 163 – 172, Jan 1999.

[5] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," IEEE Std 1588-2008, pp. c1-269, Jul 2008.

[6] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," IEEE Std 1588-2002, pp. 1-144, 2002.

[7] K. Correll, N. Barendt, and M. Branicky., "Design Considerations for Software Only Implementations of the IEEE 1588 Precision Time Protocol," Conference on IEEE 1588, 2005.

[8] Morton, A. and B. Claise, "Packet Delay Variation Applicability Statement," RFC 5481, Mar 2009.

VLC System Using PSoC Microcontroller

*Kalla Harish, **Prof. M.V. Lakshmaih, ***Prof. Pendem Suresh Kumar , ****Prof. Thimmaih, *****J Avesh Gopal

*, **, ****Srikrishna Devaraya University, Ananthapur, India.

***, *****Adama Science & Technology University, Adama, Ethiopia.

Emails: anu.harishh@gmail.com, drmvl2009@gmail.com, pendemsuresh@gmail.com, drptm2008@gmail.com, aveshgopal@gmail.com

Abstract:
Present-day wireless communication has many limitations like bandwidth depletion, security etc. Now-a-days the research on VLC (Visible Light Communications) system is offering new possibilities in overcoming the problems associated with wireless communications system. In the present study a model has been developed for addressing the above issues via Visible Light Communications system for power line communication through an 8-bit PSoC microcontroller. The exclusive PLC chip CY8CPLC10, an 8-bit PSoC3 microcontroller, high intensity pi5 light emitting diodes (LEDs) and the LX1972 visible light sensor were used for the transmitter and receiver. The analysis done using software and hardware components have given voltage fluctuations were evaluated as a function of distance from 10-50cm and given good communication between two computers with minimum loss due to deescalating.

Keywords: VLC (Visible Light Communication), 8-bit PSoC3 microcontroller, Power line communication, Visible light communication.

1. INTRODUCTION

Visible light communications technology uses visible light (380-780nm) to deliver information without the effects of electromagnetic waves, keeping pace with current wireless communications. Lights of general fluorescent lamps or the visible light of light emitting diodes (LEDs) are flickered at visible speeds to send information. Visible light communication does not have any possibility of leaking out when the light is isolated, which offers better security than wireless LAN, and does not suffer performance losses even when a variety of computers are connected at once. This technology is also notable because it uses eco friendly IT green technology rather than electromagnetic waves, which can cause harm to the human body [1].

In addition, due to the decreasing prices of LEDs and improvements in their light emitting efficiency, LEDs are now used in special lighting applications such as mobile device displays, cars, traffic signs, and advertisement panels, as well as in the general lighting market, such as current fluorescent lamps or incandescent lights. Specifically, the emitting efficiency of the white LED has already surpassed that of the fluorescent lamp, and more outstanding products are appearing in the current market. Recently, due to RF bandwidth frequency depletion, confusion possibilities, increasing security requirements, and super-speed ubiquitous communication environments, radio frequency technology and mutually complementary optical wireless communication technologies are being developed at a large number of companies and institutions worldwide [2].

When developing more complex projects, there is often a need for additional peripheral units, such as operational and instrument amplifiers, filters, timers, digital logic circuits, AD and DA

convertors, etc. As a general rule, implementation of the extra peripherals brings in additional difficulties: new components take space, require additional attention during production of a printed circuit board, increase power consumption. All of these factors can significantly affect the price and development cycle of the project. The introduction of PSoC microcontrollers has made many engineers' dream come true of having all their project needs covered in one chip.

PSoC: Programmable System on Chip

PSoC represents a whole new concept in microcontroller development. In addition to all the standard elements of 8-bitmicrocontrollers, PSoC chips feature digital and analog programmable blocks, which themselves allow implementation of large number of peripherals.

PSOC3 element is inexpensive and provides the PSOC Creator and it is highly flexible custom microcontroller that allows the user to configure the part for the current task at hand. In fact, the PSoC is considered to be a "blank slate" without any stock peripherals. Instead, the user must specify which peripherals are required for the project. The user may place any combination of available blocks to create the resources required. For instance, if the project requires 5 UARTS for serial communication, then the user can place 5 UARTS inside the PSoC. This flexibility allows for extremely efficient designs but requires significant upfront planning by the user.

This study will confirm the possibility of applying this technology for the next generation wireless network by creating a visible light communication transmitter and receiver for power line communication (PLC) using an PSOC. The CY8CPLC10 is designed for systems that require a communication interface over commercial high voltage (HV) or low voltage (LV) powerlines. Typically, these systems consist of a microcontroller or processor along with other electronic components that implement the host application functionality. The PLC interface is provided by integrating the CY8CPLC10 with a powerline coupling circuit [3, 4].

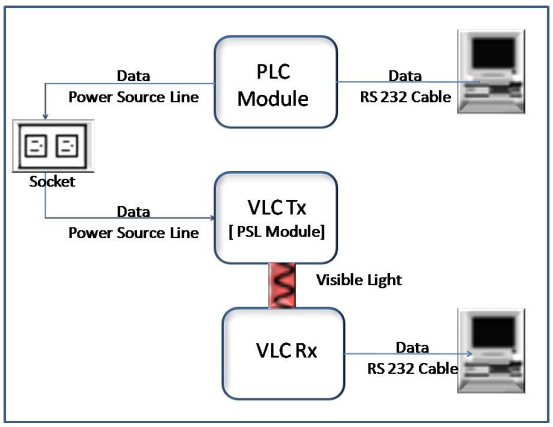


Fig. 1. System architecture.

2. EXPERIMENTS

Visible light communication technology, which has gotten notice as a next generation communication technology, is particularly attractive for home networks. Among the technologies, the visible light communication system is designed and brought into a network using PLC to study application of the LED system that is necessarily relevant for living.

The visible light communication system based on PLC uses a commercial alternating current electric power source 220 V/60 Hz power line and an RS-232C cable as a communication medium, and consists of visible light communication parts including PLC receiving and transmitting parts. The system developed in this study transmits the input data from a computer to a PLC transmitter through an RS-232C port. This signal is converted into the transistor-transistor logic (TTL) signal level through DS276 chip, and is transmitted to the PLC receiver through a power line cable after the data is transmitted to an exclusive PLC chip through an 8-bit microcontroller. Data that is transmitted after these processes is received by an exclusive PLC chip that is attached to the PLC receiver, and this signal passes through the 8-bit microcontroller and is transmitted to the visible light communication light emitting parts. After the data is received from the visible light

receiving sensor through the LED of the visible light communication light emitting parts, the lowered data signal is amplified by the OP-amp circuit, and the RS-232C cable is used through the Microcontroller and the DS276 chip to send the data to other computers [7].

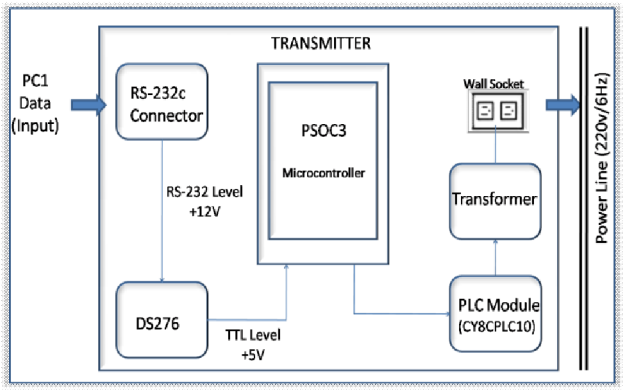


Fig. 2. Block diagram of the power line communication (PLC) transmitter.

2.1 Composition of the system

The design in this study uses a commercial alternating current electric power source, a 220 V/60 Hz power line as the communication medium, and the system shown in Fig. 1 to transmit the information. The distance between the light emitting parts and the receiving parts of the visible light communication should be between 10-50 cm, and each LED should be positioned horizontally. Each transmitting and receiving part has its own LED in order to verify the information transmitting and receiving process so that the conditions and power conditions can be checked in real time.

Real time checking of the transmitted and received data is achieved by using the hyper terminal window using Embedded C programming. The visible light communication data transmitting and receiving program uses letters so that the data transferred between computers can be checked by eye. In this manner, the performance of the processes can be monitored and evaluated. In this process, the transmitting and receiving port drivers of both computers should be well connected in order to communicate.

2.2 PLC receiver

Figure 2 is the circuit image of the PLC receiver used in this study. The PLC receiver mainly consists of the exclusive CY8CPLC10 PLC chip, microcontroller PSoC3 chip, and an DS276 chip for the TTL signal level change. The DS276 chip balances the voltage gap for the serial communication between the PC and the microcontroller. When data is transmitted using the RS-232C cable in PC1, the data is transmitted to the power line through the exclusive PLC chip and the PSoC3 microcontroller.

2.3 PLC receiver and visible light communication transmitter

This part of the system has a similar composition to the PLC receiver, except that 3 of the 5pi LEDs are added to the light emitting parts design. The signal that comes through the power line is received through the exclusive CY8CPLC10 PLC chip, and is transmitted to the LED through the PSoC3 microcontroller chip. Figure 3 shows the transmitter circuit image of the visible light communication system.

2.4 Visible light communication receiver

The composition of the visible light communication receiver is different than that of the PLC transmitter or visible light communication receiver, as shown in Fig. 4. The visible light communication receiver does not have an exclusive PLC chip, so it only has a visible light receiving sensor and PSoC3 microcontroller to receive the information. The data is received in PC2 through PSoC3 microcontroller, which is an 8-bit microcontroller, and OP-Amp LM324N after the data is received from the LED through the LX1972 visible light receiving sensor [8,9].

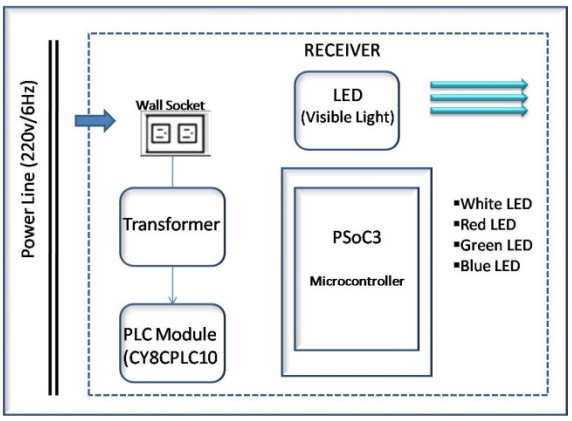


Fig. 3 Block diagram of the power line communication (PLC) receiver and visible light communication transmitter.

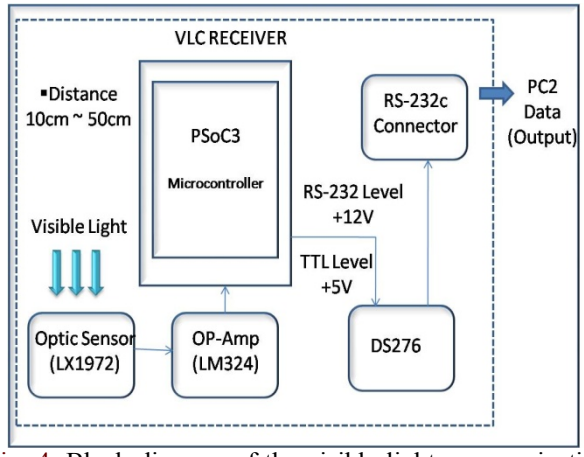


Fig. 4. Block diagram of the visible light communication receiver.

3. RESULTS AND DISCUSSION

Figure 6 shows the process in which the letters are transmitted and received. The transmitter and receiver of the materialized system are connected directly to the RS-232C port of the computer, and the output value is calculated, which is detected by the visible light receiving sensor using an oscilloscope. The transmitting and receiving waveforms can be checked and verified as shown in Fig. 6. Figure 7 shows the values obtained from voltage change through visible light receiving sensor from a voltage that 5 V is approved by distance using an oscilloscope.

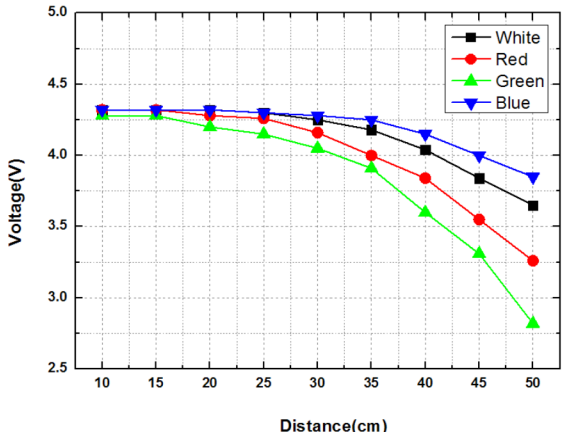


Fig. 5. Voltage change value depending on light emitting diode color of communication distance.

In the case of the white LED, the voltage at a distance from 10-35 cm was constant, but the voltage for a distance over 35 cm showed a dramatic voltage decrease. A voltage loss of 0.67 V was confirmed, with a voltage of 4.32 V at 10 cm to a voltage of 3.65 V at 50 cm. Also, in the case of the red LED, the voltage from 10-30 cm was constant, but there was a dramatic voltage decrease at distances over 30 cm. At a distance of 50 cm, the voltage was 3.26 V, so that 1.06 V was confirmed as the voltage loss. The voltage loss of the green LED is 1.46 V, and the voltage loss of the blue LED is 0.47 V, which shows the best performance among the LEDs.

Figure 8 shows the Visual C++ program that general users can easily take and adapt by adding letters to verify successful data transmissions between the computers. In this manner, "HELLO!!" was used as the input and was transmitted to the other computer. The results were checked, and the blue LED showed the best performance when evaluated using an oscilloscope.

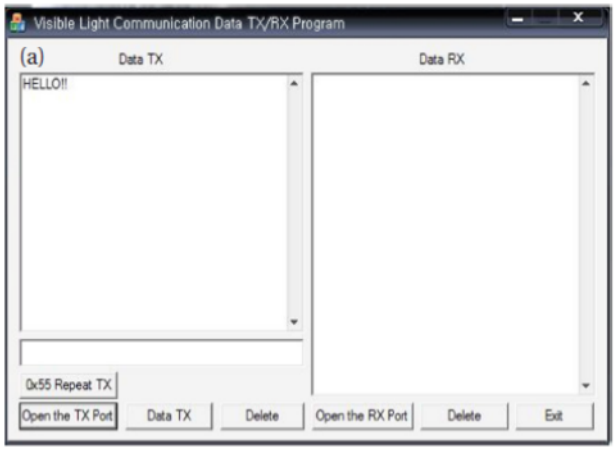


Fig .6. Monitoring screen using program visible light communication data transmitter

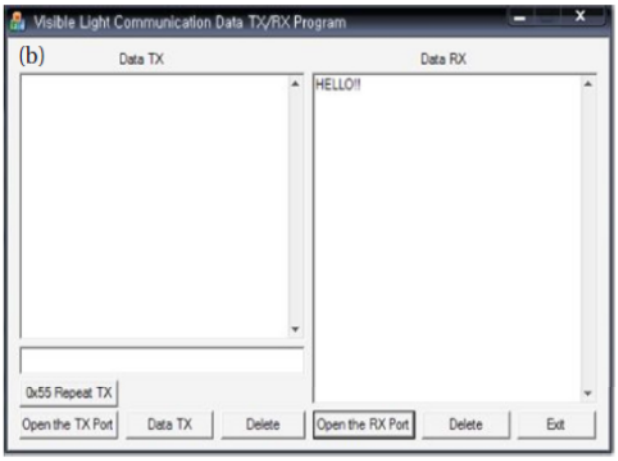


Fig. 7. Monitoring screen using program visible light communication data receiver.

4. CONCLUSION

In this study, a visible light communication system for PLC was created, and a simple experiment was created to verify the system performance using a Embedded C program and an oscilloscope.

The performance analysis was done by color at communication distances from 10-50 cm with 5V, and showed that voltage loss can be observed. As a result, the blue LED showed the best performance at distances from 10-50 cm, and there were no problems in the Embedded program while the letters were transmitted and received. As the distance increased, however, the processing speed decreased due to the weak signal treatment and the background lights, which made it hard to receive precise data. Also, new issues were found by using the estimated values.

In this study, the performance under changing conditions was evaluated, and the efficiency of the light emitting part and the receiving sensor of the visible light were studied so that better communication conditions can be achieved in the future. Continuous study and improvement are required. This study also confirmed the possibility of applying this technology for the next generation network.

REFERENCES

[1] T. G. Kang, K. H. Lee, D. H. Kim, and S. G. Lim, KSII Trans. **10**, 85 (2009).

[2] H. J. Kang, J. Digit. Contents Soc. **8**, 425 (2007).

[3] Aditya Yadav, Designing an External Host Application for Cypress's Powerline Communication, AN55427, http://www.cypress.com/?rID=38366

[4] Remote High Brightness LED Control Using PowerPSoC and Powerline Communication (PLC),AN60934,http://www.cypress.com/?rID=37956

[5] J. M. Jung, J. M. Hwang, M. K. Kang, J. W. Lee, and I. K. Kim, KSII Trans. **8**, 557 (2007).

[6] D. W. Kim and S. W. Park, KIEE Trans. **55**, 161 (2006).

[7] T. Komine and M. Nakagawa, IEEE Trans. Consum. Electron. **49**, 71 (2003) [DOI: 10.1109/TCE.2003.1205458].

[8] S. Haruyama, IEICE Trans. Fund. Electron. Comm. Comput. Sci. **86A**, 1284 (2003).

[9] T. Komine and M. Nakagawa, IEEE Trans. Consum. Electron.