International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Multi-Processor Based Intelligent Industrial Monitoring and Control System Based on µCOS-II and Wireless Sensor Networks

Tinotenda Zwavashe[1], Dr. D. Vasumathi[2]

[1]M.Tech Student, ECE Department, Jawaharlal Nehru Technological University Hyderabad, AP, India
[2]Professor, CSE Department, Jawaharlal Nehru Technological University Hyderabad, AP, India

Abstract: ***This paper is based on my M Tech project which shares the same title as this paper. It presents a model which illustrates how we can incorporate a Real Time Operating System (RTOS) into an Industrial setup whereby a sensor node resides in the field where processing is carried out and a Master or Control station resides in a control room and the two communicate using a wireless protocol. The RTOS is meant to provide predictability, faster time response and high performance among other wide provisions. The RTOS used to implement this model is µCOS-II (Microcontroller Operating System) from J Labrosse. It is a pre-emptive kernel where the highest priority task in the ready queue is executed first. Thus faster processing of control commands and real time logging of data channeled by sensor node to control station is expected as a result of incorporating the operating system rather than use of a super loop.***

Keywords: Monitor and Control, Zigbee, uCOS-II RTOS, LPC2148, PIC16F877A, Task Priority, Touch screen, Graphical LCD, kernel objects

1. Introduction

As the complexity in industrial structures varies from very small to quite complex systems, many methods are coming into play by which controller systems can be implemented. Also there has been a huge advancement in the control process as some systems comprise quite a large number of tasks to be centrally controlled from a common point. In such a scenario it is common that some tasks might need immediate attention once they enter a stage where CPU attention is needed. But remote monitoring means that data has to be captured and relayed through some external port of the controller, be it serial, parallel, USB or any other port. Thus response to incoming data/commands or outgoing data/commands can be accomplished by use of interrupts. However not all tasks and events are externally generated and can be easily responded to using interrupts. In some cases some internal tasks might need prompt action once they request execution time. In such situations Real time Operating Systems (RTOS) come into play. An RTOS enables the assignment of priorities, priority inheritance, priority conversions and use of many kernel objects to ensure that tasks of high priority are executed in a timely manner. For serial data response some of the options are polling and interrupts but they do not provide deterministic behavior to the system as the number of tasks requiring CPU time increases. An analysis on the efficiency of priority assignment was done in [10] for the cases of polling, interrupts and uCOS-II. This design is a follow up to the analysis in [10] and aims at porting uCOS-II into an ARM based microcontroller and come up with a system with a high response and with a provision for easy communication between tasks which subsequently increases system efficiency and response.

2. Existing Designs

Several designs have been carried out in line with wireless sensor networks, Real Time Operating Systems and Environmental Monitoring and /or Control. Briefly I shall look at some of these designs, see how they can be expanded and modified or demonstrated from another point of view, for example by use of different hardware, software, operating systems and design techniques. Then I will try to illustrate the uniqueness of the design which I am going to carry out in relation to the previously carried out pieces of work.

Monitoring and Control has been implemented but for an Agricultural Environment. [1]. In this system although it has been considered to provide real time monitoring and control, it however lacks a real time operating system to manage the various tasks. The system implements a PC as the main control center for visual monitoring and to input control commands. There are several nodes which pass data to a master node so that the master can pass data to Ethernet for online control. It can be seen that all the nodes including master node have implemented the ARM microcontroller (LPC2148). The system also uses Zigbee protocol for inter node communication.

Implementation of touchscreen based zigbee wireless network for microcontroller to microcontroller communication has been implemented without the use of an RTOS [4] since a single task has to be executed. The implementation involves one way communication and no complex processing is required.

A real time operating system has been implemented in [2]. The implementation has incorporated RTLinux as the RTOS and it is ported into the coordinator node. The coordinator node has been designed using the ARM 9 (AT91RM9200) microcontroller whilst all sensor nodes use ARM 7

(LPC2129) microcontrollers. Sensor nodes communicate with the coordinator through a Zigbee protocol and the coordinator is connected to a PC for user interface. No other user interface peripherals are connected at the coordinator. The system has managed to reduce power consumption by providing a QUERY and RESPOND scenario such that there is no continuous transmission of data from sensor nodes to coordinator /master node but data is provided on demand. Thus sensor nodes are provided with control capabilities so as to monitor deviations from set values. Faults at the sensor nodes may take longer to detect since user monitoring is done on demand and the coordinator may take longer to detect a faulty node. However, the system has proved to be versatile in wireless sensor situations where power preservation is of prime importance.

Research challenges in Wireless Sensor and Actuator Networks targeting industrial automation have been outlined [3]. Amongst the several challenges are the issues of scalability and Latency/ Timely processing. Real time processing requires that data be processed quickly since its validity is of limited duration. Scalability dictates that the system should support various network sizes without compromising on system performance. Thus the research which is to follow will try to cater for this by use of µCOS-II which is a scalable Real Time OS which can accommodate varying number of tasks and which can ensure real time processing of data by task prioritization and synchronization.

Zigbee based communication in industrial systems is also implemented but with the main focus being towards miniaturization of the sensor node [5]. The node should be wearable, washable and economic and detects ultra violet radiation and dust. The microcontroller and radio are integrated into a transceiver which communicates via zigbee with a gateway which subsequently sends data to server via internet.

3. Proposed System

The proposed system comprises a Master node controlled by the LPC2148 microcontroller. This is a controller having an ARM7TDMI based processor. The RTOS is ported into this microcontroller and control commands can be input from this node. Real time temperature values, voltage levels at sensor node and intrusions which are relayed wirelessly from sensor node are, also displayed on a Graphical LCD (GLCD). The master node is equipped with a touch screen module which acts as the input device for user input. Selection between various parameters such as display of temperature, voltage, switching ON/OFF of motor on sensor node can be done using this touch screen. The master node communicates with sensor node using Xbee modules which provide a Zigbee communication protocol. The Xbee modules are connected to UART1 of the LPC2148 and thus, data is relayed to and from the microcontroller serially. At the other end of the communication Architecture lies the sensor node. The node is controlled by the PIC16F877A microcontroller which is an 8 bit microcontroller from Microchip. The node consists of temperature sensor, intrusion detector, buzzer (alarm), Motor actuator, and voltage level detector. The temperature values and voltage levels are captured and detected through the Analogue to Digital Converter respectively and send to the Master node via Zigbee. For PIC, the Xbee modules are connected to the USART port which is again a serial communications port. The author has borrowed concepts on zigbee wireless communication from some earlier work in [7] and [9]. Figure 1 shows the block architecture of the Master Node while Figure 2 illustrates the Sensor Node.

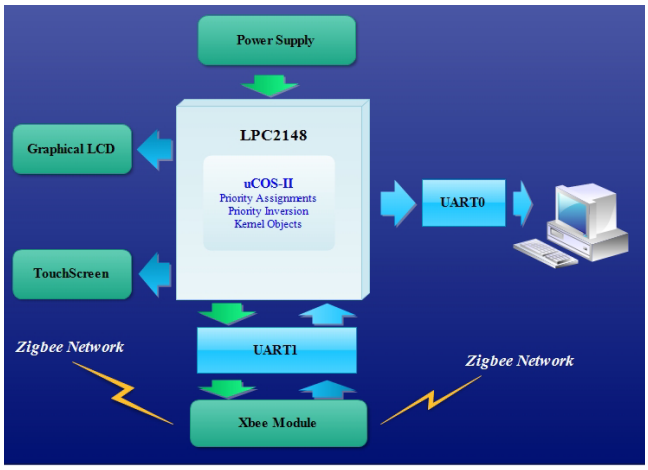

Figure 1: Master node and associated hardware peripherals

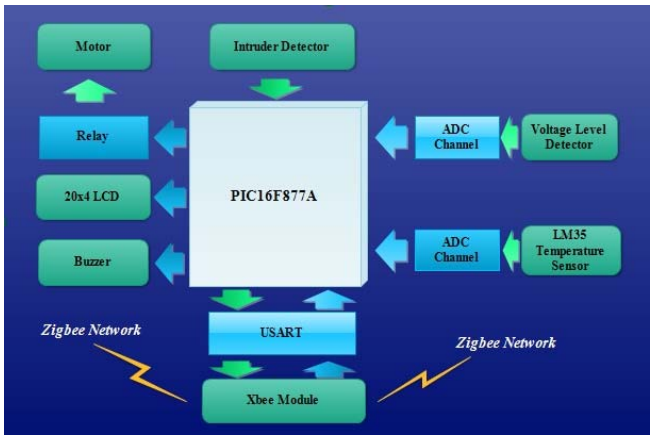

Figure 2: Sensor node and associated hardware peripherals

4. System Functionality

The master node contains a Real Time Operating System for task control. Sensor Node transmits real time temperature and voltage values to the sensor node over the wireless network. Also warning signals are send for example in cases of intruder detections. At the master node all data send is displayed on a GLCD. The user can choose the data to display, e.g., voltage or temperature or check max and min temperatures by making inputs through the touch screen module. These values, in case of voltage and temperature, are displayed in the form of a line graph such that recent and existing fluctuations in these values can be easily viewed. The system has included a supplementary visual display unit in the form of a PC where data values are also relayed for display and from where control commands to the sensor node can also be entered.

Thus it can be seen that quite a number of tasks are to be executed by the master node. Some of these tasks are:

- Serial data receive from sensor node task
- Serial data receive from PC task.
- Task to display temperature values
- Task to display voltage values
- Task to check touch on touch screen.
- Tasks to send commands to serial ports

These tasks are to be managed by the real time operating system and priority assignments has been implemented and kernel objects used for task synchronization and task communication. At the sensor node a provision of a display has also been implemented such that operators in the vicinity can view the system behavior and state. Real time temperature and voltage values are displayed and in cases of too low voltages or temperatures warning messages are also displayed. If temperature exceeds some set point the motor is switched ON. Motor switch ON/OFF can also be done remotely from the master node. Voltage level detector is essential in cases where the sensor node might be powered from a battery source such that there is depletion of power over time.

5. Real Time Operating System- µCOS-II

As stated earlier on the system employs uCOS-II as the RTOS at the master node.

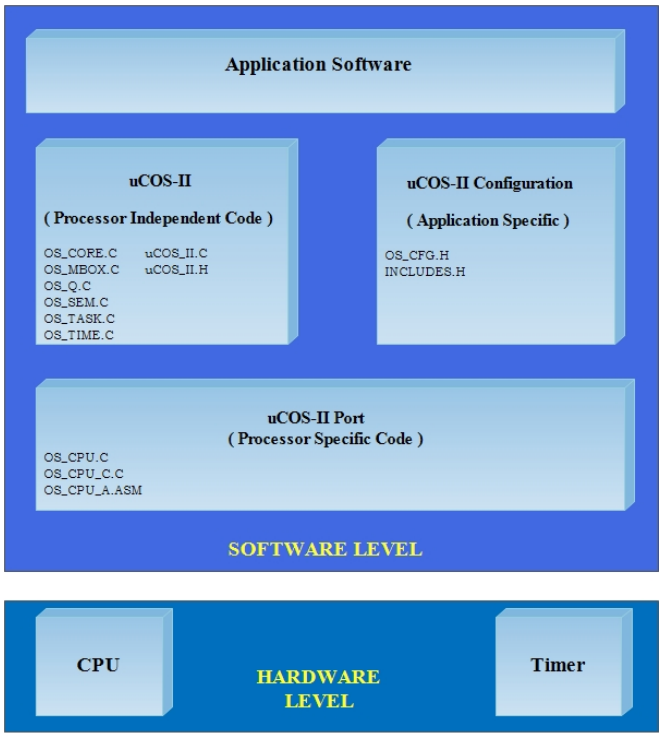

Figure 3: µCOS-II Software and Hardware setup

The diagram of Figure 3 shows the software and hardware architecture of uCOS-II. The user is responsible for the development of the application software and this is dependent on the system being developed. The user also declares the configuration settings and these are dependent on number of tasks, desired stack sizes, priority levels, and operating system objects being used and so on. Depending on the processor being used, source code for processor specific code and processor independent code can be obtained from the micrium website. Thus to port uCOS-II into hardware the port can be found on the website and if not available for a specific processor then one has to develop it. This port is modified to suit the needs of the application being developed.

5.1 Tasks and Task States

The general concept and goal of real time operating systems is on finding out how we can make tasks execute in the desired time frame. Thus it is important that we briefly look at what a task is and the states in which a task can be in a system. A task is also called a thread. It is a program which assumes that it has the CPU all by itself. A maximum of 56 user tasks can be created in uCOS-II. Each task is assigned a unique priority ID. Thus priority numbers can be used to identify tasks. On top of being assigned its own priority, each task is also assigned its own set of registers and its own stack area (memory). When creating a task it should be an infinite loop and it can be in any one of the 5 states namely; waiting, dormant, ready, running or interrupt service routine (ISR). These task states are illustrated in Figure 4.

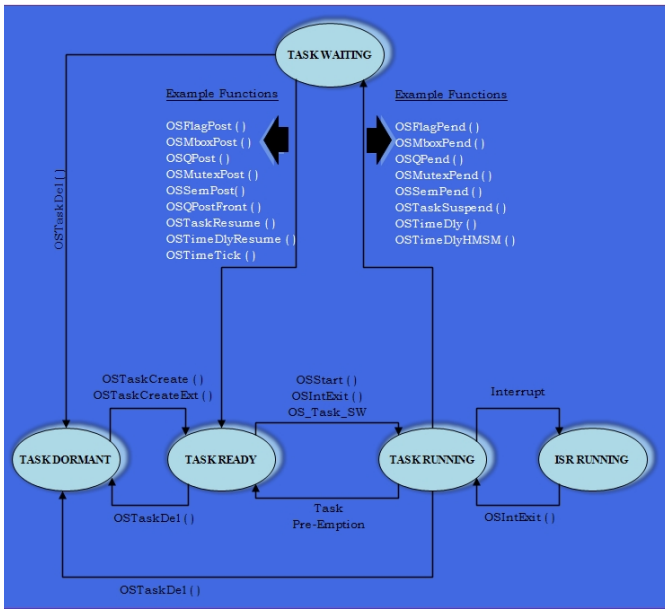

Figure 4: Task States and transitions from one state to another

Ready: A task is said to be ready when it can execute but is waiting for a higher priority task which is already executing to finish. The queue can have many tasks and of these a higher priority one will execute first.

Dormant: this state corresponds to a task which is residing in memory and has not been made available to the multi-tasking kernel. An example is a task deleted using *OSTaskDel ( ).*

Running: A running task is the one which is under execution, that is, the one which has control of the CPU at that particular time.

Waiting: A task is in a waiting state when it is waiting for a certain event to happen. An example is a task which has called a delay. It goes to wait state until that delay expires (unless if the delay is cut short) before going to the ready

queue. A task can also be waiting for a shared resource to be available.

ISR: A task is in ISR state when it is stopped so that the CPU can service an interrupt. After the interrupt has been serviced, that same task continues execution if it is still the highest priority task. Else, the highest priority task in the queue will run.
A task can be created using two functions. These are **OSTaskCreate (***arguments***)** or **OSTaskCreateExt (***arguments***).**

5.2 Task Synchronization and task communication

Synchronizing of tasks and communication between tasks is essential since some tasks are dependent on the outcomes of other tasks and also some resources may be shared between tasks. Thus an efficient way for tasks to signal each other and use shared resources should be available. In uCOS-II this is accomplished by use of various kernel objects. In this design this has been accomplished using semaphores and mailboxes.

Semaphores: Two types of semaphores exist namely binary and counting semaphores. The former can have a value of 1 or 0 and the later any value from 0 onwards. To illustrate their use we shall consider a scenario whereby the display tasks will run only if data has been received serially. Thus the UART receive task should signal the other tasks to run only is it has received some data. In uCOS-II a semaphore is considered as an OS event and should be declared and created before being used. **T**he model in Figure 5 shows the communication and synchronization of the three tasks; UART receive, LDC Display and PC Display.

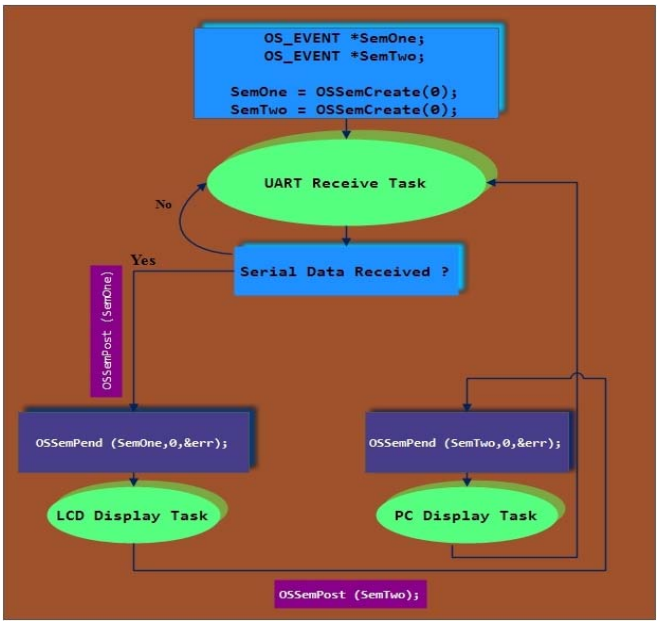

Figure 5: Task Synchronization and communication using semaphores

Mailbox: A mailbox is another kernel object. Tasks can send pointer variables to each other and these will point to some application data structures which contain a message. Just like semaphores, mailboxes need to be created before being used. Five services are available in uCOS-II to access mailboxes and these are:

OSMboxCreate ( )
OSMboxPend ( )
OSMboxPost ( )
OSMboxAccept ( ) and
OSMboxQuery ( )

Mailboxes have been used in this application to send messages to run certain tasks and send acknowledgements that some specific operation has to be or been performed. In the display of temperature and voltage values upon screen touch on touch screen, mailboxes have been used in the implementation. Figure 6 illustrates mailbox operations.

Mailbox Operations – An Example

```
Declaring the mailbox: OS_EVENT  *TemperatureMailbox;

Creating the mailbox: TemperatureMailbox = OSMboxCreate ((void *) 0);

/* Initial value of the pointer is set to null */

If (READ_TOUCH_BUTTON (sense_x, BUTTON1_X1, BUTTON1_X2, sense_y, BUTTON1_Y1, BUTTON1_Y2))
                    {
                          DISPLAY = TEMPERATURE;
                          TEMP_PLOT_CHIP = 0;
                          TEMP_PLOT_COLUMN = 18;
                          TEMP_PLOT_PAGE = 0;
                          GLCD_DRAW (GRAPH_GEN);
                          OSMboxPost (TemperatureMailbox, (void *) &DISPLAY);
                    }

/* Posting a message to a mail box so that temperature can be displayed*/

Waiting for msg to mailbox: OSMboxPend (DISPLAY,0,&err);

/* where 1st argument is the message name, 2nd argument signifies the time out period.
3rd argument is a pointer to a variable which is used to point to error code. */
```

Figure 6: Code snippets showing mailbox operations

6. Testing and Results

This paper is based on the design of the system which employs a real time operating system for the coordination and control of tasks in a multi-tasking system. As stated earlier on the strengths of the RTOS has been looked at through hardware simulations in [10]. As such, the result section will focus on exhibiting the prototype on which the system was implemented.

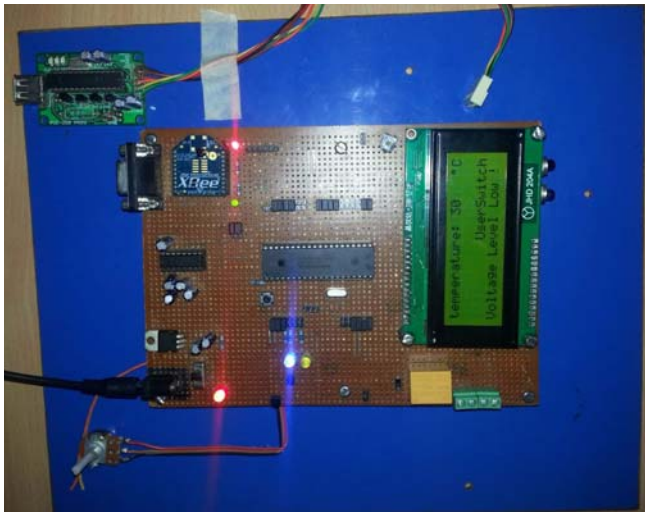

Figure 7: Sensor and Actuator Node using PIC16F877A

Figure 7 shows the prototype version of the sensor node and its associated hardware connections and the Xbee module used to provide wireless communication between the sensor node and the master node. Figure 8 and 9 shows the master node with the LPC2148 board connected to the GLCD and

touch screen module for display and input devices respectively.

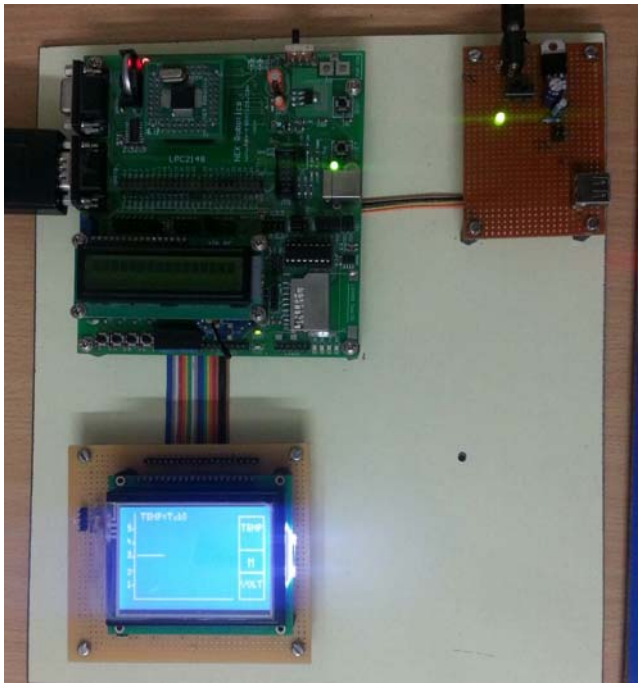

Figure 8: Master Node containing the RTOS

The diagram in Figure 9 is a closer view showing the display of temperature on the GLCD in the form of a line graph and the small fluctuations in temperature values where it is fluctuating between 30 and 33oC.

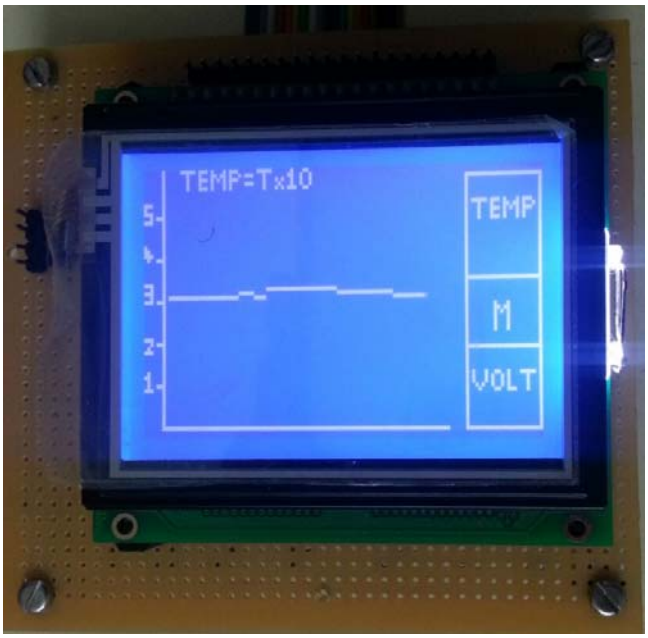

Figure 9: Temperature Display on GLCD and Touch screen Panel for selection.

7. Conclusions and Future Work

The design and implementation was successfully carried out. The prototype gave appropriate responses which illustrates that the tasks communicated and synchronized correctly and that priority assignments made sure that tasks requiring immediate CPU time got that opportunity. As a basis for future expansion we can see that the system can be applicable in a variety of applications where some of these applications may be non-industrial. In some of these cases we find that the sensor node might need powering from a battery source and methods to optimize power consumption might be essential. The system can also be expanded to include several sensor nodes to fully utilize the power of the RTOS since uCOS-II can accept up to 56 application tasks. The analysis and design can also be done on a non-pre-emptive kernel and observe the complexity in the design process and the efficiency of non-pre-emptive compared of pre-emptive kernel.

References

[1] G.M. Kumari and Dr V Devi, “Real Time Automation and Monitoring System for Modernized Agriculture”, International Journal of Review & Research in Applied Sciences and Engineering vol.3, March 2013.
[2] Manoj Kolam & S. R. B. Shree Shree, “Zigbee Wireless Sensor Network for Better Interactive Industrial Automation”, IEEE, 2011.
[3] J. Akerberg et al, “Future Research Challenges in Wireless Sensor and Actuator Networks targeting Industrial Automation “, IEEE, 2011.
[4] M. R. Reddy et al. “Touchscreen and Zigbee based Wireless Communication Assistant”, International Journal of Combined Research and Development vol.1, Issue 4, August 2013.
[5] Elisa Pievanelli et al, “Dynamic Wireless Sensor Networks for Real Time Safeguard of workers exposed to physical agents in construction sites”, IEEE, 2013.
[6] M. R. Reddy et al, “Touchscreen and Zigbee based Wireless Communication Assistant”, International Journal of Combined Research and Development vol.1, Issue 4, August 2013.
[7] T. Zwavashe, “A Zigbee Based Inter-Processor Communication Architecture for the Management of Bedchambers for the Physically Challenged”, International Journal of Innovative Research in Computer and Communication Engineering, vol2, issue4, April 2014.
[8] MaxStream Inc, “XbeeTM Series OEM RF Modules Product Manual v1.x.1x-Zigbee Protocol”, 2007.
[9] T. Zwavashe and R. Duri, “Integrating GSM and Zigbee Wireless Networks for Smart A2 farming Enterprises in Zimbabwe”, International Journal of Science and Research, Vol3, Issue6, June 2014
[10] T. Zwavashe, “Polling, Interrupts & µCOS-II: A Comparative Timing Response Simulation Model for Wireless Processor-to-Processor Communication”, International Journal of Science and Research, Issue3 vol7, July 2014.

Author Profile

Tinotenda Zwavashe: Attained his B.Eng. Degree in ECE from NUST, Zimbabwe in 2010. Currently he is studying towards M. Tech Embedded Systems at JNTUH, India. His research interests are in the area of

Microcontroller Design, Wireless and Sensor networks, RTOSes and SCADA systems.

Dr D. Vasumathi: Ph.D from JNTU Hyderabad. She is currently working as Professor in Department of CSE, JNTUCEH. Her research interests include Data Mining, Computer Networks, Web Mining, Data Warehousing, Wireless Sensor networks and Microcontroller Design.