Journal of Theoretical and Applied Information Technology
10th July 2014. Vol. 65 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ZIGBEE BASED REALTIME PLANT MONITORING SYSTEM USING µC/OSII

[1]M.SUJITHA, [2]Dr. V.KANNAN
[1]Research Scholar, Department of Electronics and Communication Engineering, Dr. M.G.R. Educational and Research Institute University, Chennai.
[2] Principal, Jeppiaar Institute of Technology, Kanchipuram.
E-mail: [1] msujidhas@yahoo.co.in , [2]drvkannan123@gmail.com

ABSTRACT

This paper describes a realtime transformation for an existing non realtime process control system. The proposed system combines realtime scheduling with ZigBee wireless communication and Monitoring. This paper implements the effective handling of multiple tasks using RTOS concepts to suit critical applications and concurrent task handling in realtime environment. Priority based preemptive task scheduling algorithm is used to protect shared data and to achieve task synchronization. Thus the reliability and processing ability of the system are improved greatly by adding the µC/OS-II operating system.

Keywords: *ZigBee, µC/OS-II, Task, Nonlinear Process, Real-Time Transformation, Priority Scheduling*

1. INTRODUCTION

Embedded targets for Nonlinear Real Time Applications cannot handle multiple inputs and multiple outputs, time constraints but perform sequentially. The algorithm needs to be redesigned for concurrency. A real-time operating system (RTOS) is developed for real-time applications. (Example: mobile phones, industrial robots, or scientific research equipment). It is a priority-based pre-emptive real-time multitasking operating system kernel for microprocessors, written mainly in the C programming language. It is intended for use in embedded systems. RTOS services basic include OS functions, priority allocation, memory management, memory allocation, task management, task predictability, scheduling and interrupt latency control, timer and IPC synchronization functions[1]. To increase the realtime performance and control Pre-emptive Scheduling Algorithm is implemented efficiently in µC/OS-II RTOS manages critical tasks.

RTOS provides following features [14]:

- Synchronization: Synchronization is necessary for realtime tasks to share mutually exclusive resources. For multiple threads to communicate among themselves in a timely fashion, predictable inter-task communication and synchronization mechanisms are required.
- Interrupt Handling: Interrupt Service Routine (ISR) is used for interrupt handling. Interrupt latency is defined as the time delay between the occurrence of an interrupt and the running of the corresponding Interrupt Service Routine (ISR).
- Timer and clock: Clock and timer services with adequate resolution are vital part of every real-time operating system.
- Real-Time Priority Levels: A real-time operating system must support real-time priority levels so that when once the programmer assigns a priority value to a task, the operating system does not change it by itself.
- Fast Task Preemption: For successful operation of a real-time application, whenever a high priority critical task arrives, an executing low priority task should be made to instantly yield the CPU to it.
- Memory Management : Real-time operating system for large and medium sized application are expected to provide virtual memory support, not only to meet the memory demands of the heavyweight real-time tasks of an application, but to let the memory demanding non-real-time applications such as text editors, e-mail etc. An RTOS uses small memory size by including only the necessary functionality for an application while discarding the rest.

In Process Control Wireless applications offer benefits for monitoring purposes, however issues and challenges are well known and agreed upon in this area. The system studied comprise of ZigBee Module and has several advantages such as low cost, low power consumption, data Integrity and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

good throughput, etc. In process control, precise liquid level control of storage tanks and reaction vessels is essential in many industrial operations and mainly in chemical engineering systems where the liquids are pumped to the tanks, stored and delivered through discharge valve. It is most commonly used in the area of water purification, chemical and biochemical processing, automatic liquid dispensing, food and beverage processing and pharmaceutical industries.

This paper describes algorithms for transforming a nonlinear control system to realtime. The proposed system consists of a plant, processor with embedded µC/OS-II operating system, sensors and ZigBee based data transmission technologies.

1.1. Network Control System

A networked control system is a control system wherein the control loops are closed through a real-time network. The feature of an NCS is that control and feedback signals are exchanged among the system's components in the form of information packages through a network.

The functionality of a typical NCS is established using four basic elements:

1. Sensors to acquire information.
2. controllers to provide decision and commands
3. Actuators to perform the control commands
4. Communication network to enable exchange of information.

The most important feature of a NCS is that long distance communication, eliminate the wiring, reducing the complexity and the overall cost in designing and implementing the control systems. They can also be easily modified or upgraded by adding sensors, actuators and controllers to them with relatively low cost and no major changes in their structure.

A task can obtain information about itself or other tasks. This information can be used to know what the task is doing at a particular time. The system in which process of scheduling and switching the CPU between several tasks takes place is referred to as a multitasking system. There are 2 types of multi-tasking are there that is given bellow

- Pre-emptive
- Non Pre-emptive

Pre-emptive multitasking is a multitasking operating system, which permits pre-emption of tasks, from a cooperative multitasking system wherein processes or tasks must be explicitly programmed to yield when they do not need system resources. Non-preemptive multitasking is a style of computer multitasking in which the operating system never initiates a context switch from a running process to another process. Each task requires its own stack, µC/OS-II allows each task to have a different stack size. This allows us to reduce the amount of RAM needed in our application. With µC/OS-II's stack-checking feature, we can determine exactly how much stack space each task actually requires. Task Scheduling Algorithms in RTOS are

- Priority-based scheduling algorithm
- Priority-based round-robin scheduling algorithm
- EDF(Earliest-Deadline-First)scheduling Algorithm
- RM (Rate-Monotonic) scheduling algorithm.

Task Information is useful for debugging and monitoring parameters. Task is an active entity which could do some computations. Tasks become "ready" after they are created. Each task is assigned a priority. The lower the priority number, the higher the priority of the task. Always the highest priority task that is ready to run should be executed for realtime.

1.2. Task states

Task state is a state of a task that changes on scheduler directions. A task at an instance can be in one of the states, waiting, ready to run and running that are controlled by the scheduler. The task states are such as

- Dormant
- Ready
- Running
- Waiting
- ISR

The dormant state corresponds to a task that resides in memory but has not been made available to the multitasking kernel. A task is ready when it can execute but its priority is less than the currently running task. A task is running when it has control of the CPU. A task is waiting when it requires the occurrence of an event (for example, waiting for an I/O operation to complete, a shared resource to be available, a Timing pulse to occur, or time to expire). A task is in the ISR state when an interrupt has occurred and the CPU is in the process of servicing the interrupt. State transient diagram is shown in Figure 1

Journal of Theoretical and Applied Information Technology
10th July 2014. Vol. 65 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

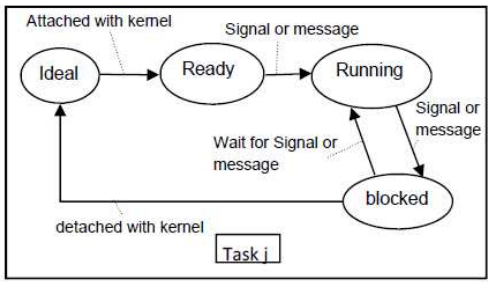


Figure 1 – Process State Diagram

This article is structured in following manner. Section 2 describes about related work, section 3 describes about research motivation, section 4 describes about proposed environment used for this work, section 5 describes about performance metrics, section 6 describes about system architecture and section 6 describes about implementation methodology.

2. RELATED WORK

A. M. Franklin Richard and S. Sudheer Sukumaran Implemented an embedded platform for industrial automated system designed with multiple tasks. In order to manage the various tasks evaluated Priority based Preemptive Task Scheduling algorithm in µC/OS-II Real Time Operating System is used. In his paper Motor task is controlled by PI algorithm. When an interrupt is created, the corresponding task is executed and thus response is obtained in few microseconds the corresponding task output are displayed on the LCD. Consequently, with the sensor value, processing speed and timing constraints of the task scheduled, the automated system is functioning with rapid task execution context under multitasking environment [4].

B. Ishwarya Singh, Rajput and Deepa Guptha implemented priority based round robin CPU scheduling Algorithm for realtime systems where there is more than one task with same priority to share CPU time, so the burden is on the user to proxy out the time slicing code to a high level mechanism of their own design. Allowing multiple tasks to have the same priority by adding a level of integration implies a fundamental redesign of the ready list and scheduling Algorithms and probably the adoption of queue based Scheduler. In its state µC/OS-II is an optimal solution of embedded realtime software engineering problems[13].

C. Zengyapeng and Yandong zhao attempted Realtime monitoring systems for some applications. The system consists of an µC/OS-II operating system and RS232 bus based data transmission technologies. Due to some limitation in system performances, speed and system stability, the realtime scheduling with wireless data transmission technologies is an solution for an embedded realtime systems [6].

3. RESEARCH MOTIVATION

Event based and time based planning, scheduling can be implemented by RTOS stack and timer management, for the efficient functioning of the automated system. Industrial and vehicle automation can be effectively implemented based on these techniques. Performance of time sharing systems can be improved with the priority based round robin algorithm and can also be modified to enhance the performance of realtime system. Due to the µC/OS-II operating system, task scheduling is available, and the system's reliability, real-time capability and stability are greatly improved. However, µC/OS-II has some defects such as. different from time sharing operating systems, such as Linux, µC/OS-II doesn't support time slice scheduling, thus it is called multitask rather than multi-process. That makes it difficult to run multiple tasks with frequent switching. So it's hard to determine the priority of each task and efficiently schedule them in the program.

4. IDENTIFIED ENVIRONMENT DESCRIPTION

A realtime process control application approach is the host-server Approach. A server uses general purpose OS/RTOS. Algorithm is running on host using µc/osII Operating system. In this work for realtime implementation ARM processor is chosen, it has multi parameter Acquisition and multi-level monitoring and supports Networking. STM32 is ARM cortex M3 based Microcontroller Application. The ARM Cortex M3 is a next generation core data system enhancement such as enhanced debug features and high level support block integration. Software coding for the functionality is written in embedded C language using keil software.

4.1. Merits Of Research Work

Merits of research work include
(i) Very Small realtime kernel. (ii) Memory footprint of about 20kB for a fully functional kernel.(iii) Highly Portable, scalable, preemptive realtime, deterministic kernel (iv) Connectivity

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

with µC/GUI Platform and µC/file system. (v) Supports all types of processor from 8 bit to 16 bit. (vi) An embedded TC/IP stack.

4.2. Novel Realtime Control Using Zigbee Modules

ZigBee modules are embedded solutions providing wireless end point connectivity to devices. These modules use the IEEE802.15.4 networking protocol for fast point to multipoint or Peer to Peer networking.

The new knowledge created include:

(i) Algorithms design for high throughput applications requiring low latency and predictable communication timing. (ii) Low cost, low power wireless sensor Networks. (iii) Modules require minimum power and ensure reliable delivery of data between devices. (iv) Operating frequency ISM 2.4GHZ. (v) Long Range data Integrity, Advanced Networking and Security. (vi) ADC and I/O line support easy to use. (vii) Power ratings are (Tx current 45mA, Rx current 50mA@3.3v). (viii) Distance for communication for indoor up to 30m, outdoor line of sight up to 100m. (ix) ZigBee RF Modules are designed to operates in five different modes. (a) Idle Mode (b) Transmit/Receive Mode (c) Sleep Mode (d) Cyclic Sleep Modes (e) Command Mode. IEEE and ZigBee Alliance have been working closely to specify the entire protocol stack. IEEE802.15.4 focuses on the specification of the lower two layers of the protocol (physical and data link layer. On the other hand, ZigBee Alliance aims to provide the upper layers of the protocol stack (from network to the application layer) for data networking, security services and arrange of wireless home and building control solutions. ZigBee Protocol Layer diagram is given in Figure 2. Functionality of the two lower layers defined by IEEE 802.15.4 are:

4.2.1. The Physical Layer (PHY)

The PHY layer provides the basic communication capabilities of the radio and is responsible for the wireless transmission and reception of MAC frames. It performs such functions as radio control, energy detection, clear channel assessment, channel selection, data modulation, signal spreading, and the transmission and reception of bits onto the physical medium. The unit of transmission at this layer is the PHY frame. PHY–MAC Data frame format is given in Figure 3.

- Data Frame Provides up to 104 byte data payload capacity.
- Data sequence numbering to ensure that all packets are tracked.
- Robust frame structure improves reception in difficult conditions
- Frame Check Sequence (FCS) ensures that packets received are without error.

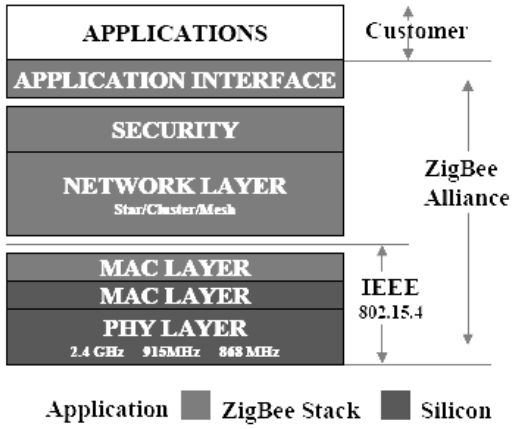


Figure 2 : ZigBee Protocol Layer

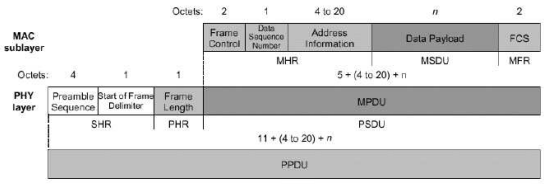


Figure 3 : ZigBee PHY-MAC Data Frame Format

Comparative Analysis of different technologies providing similar services and their tradeoffs is shown in Table 1. ZigBee will increasingly play an important role in the future of computer and communication technology. In terms of protocol stack size, ZigBee's 32 KB is about one-third of the stack size necessary in other wireless technologies (for limited capability end devices, the stack size is as low as 4 KB).

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1 : Wireless Technology Comparison Chart

Feature(s)	Wi-Fi	Bluetooth	ZigBee
Power Profile	Hours	Days	Years
Complexity	Very Complex	Complex	Simple
Nodes/Master	32	7	64000
Linking time	upto3 seconds	10seconds	30ms
Range	100 m	10m	70m-300m
Extension	Dependig on existing work	None	Automatic
Transmission speed	11Mbps	1Mbps	250Kbps
Security	Authentication Service Set ID (SSID)	64 bit, 128 bit	128 bit AES and Application Layer user defined

4.2.2. The Medium Access Control Layer

The MAC layer establishes reliable and secure single-hop communication links between devices. It provides the basic functions of monitoring and accessing the wireless communications medium to coordinate the transmission of data from the higher layers. The MAC layer handles network association and dissociation functions and uses unique 64-bit MAC hardware addresses assigned by the manufacturer.

5. PERFORMANCE METRICS

5.1. Cpu Utilization Time

Increasing CPU utilization, i.e., keeping CPU as busy as possible that increases the responsiveness of the system. CPU utilization time related with number of tasks. As a rule of thumb, designing a system to use 60 to 70 of CPU time is always desirable to meet all hard realtime deadlines.

5.2. Interrupt Latency

The most important specification of a realtime kernel is the amount of time interrupts are disabled. All realtime systems disable interrupts to manipulate critical section of code and reenable interrupts when the critical sections have been executed. The longer interrupt are disabled, higher the interrupt latency. It is given by ,

Maximum amount of time interrupts are disabled + Time to start executing first instruction of ISR

5.3. Task Execution Time

Non RTOS systems usually do not allow user programs to mask interrupts as the user program could control the CPU for as long as it wishes. RTOSs allow application itself to run in the kernel mode and permit the application to have greater control of the OS environment without requiring OS intervention.

5.4. Waiting Time

Waiting time is the total time a process has been waiting in ready queue. The CPU scheduling algorithm does not affect the amount of time during which a process executes or does input-output; it affects only the amount of time that a process spends waiting in ready queue.

6. SYSTEM ARCHITECTURE

In this paper liquid level process is used as an example. The proposed system consists of two different sections, Client and Server Sections. Liquid level process setup is given in Figure 4.

6.1. Client Section

Client section has the following modules
(i) Host (ii) Controller (iii) Sampling device (iv) Control element.

In client section the sensor node gathers the information such as water level, motor status, in flow, out flow etc. The change in level is measured through a Resistive probe. Outputs of the plant are sampled at periodic intervals by the controller, a control algorithm applied to the samples, and the result of the control is produced at the output of the controller generally as a zero order hold signal. The host system has three main functional aspects: Controller Configuration, Monitor and Communication. The experimental set up consisting of liquid level process is shown in Figure 5. The laboratory setup consists of a conical tank, water reservoir pump, an interfacing module etc. Figure 6 shows the board set up of client side liquid level process and it consists of separate modules for zigBee, UART and STM 32 hardware.

Journal of Theoretical and Applied Information Technology
10th July 2014. Vol. 65 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

6.2. Server Section

The server side consists of a status display and ZigBee module. It can display the parameters such as level, inflow, outflow, motor status etc. The set point is entered in server side. RTOS system is used in the server.

6.3. Client Server Communication

ZigBee RF modules interface to the host/ server device through a logic-level asynchronous serial port. Through its serial port, the module can communicate with any logic and voltage compatible UART. Devices that have a UART interface can connect directly to the pins of the RF module as shown in the Figure 7. ZigBee communication setup between server & host is shown in Figure 8.

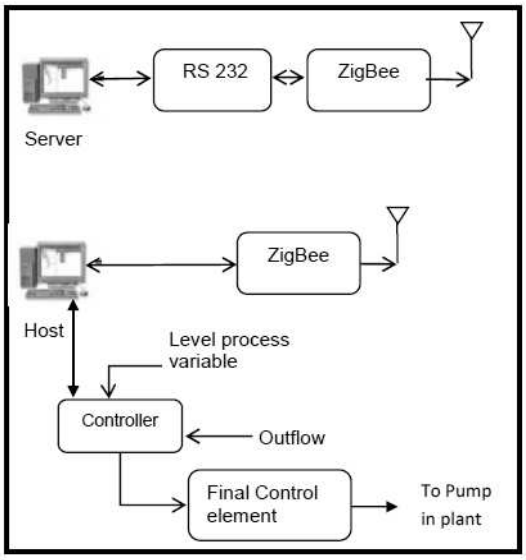
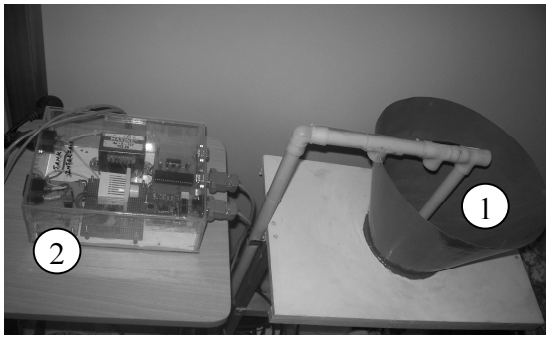
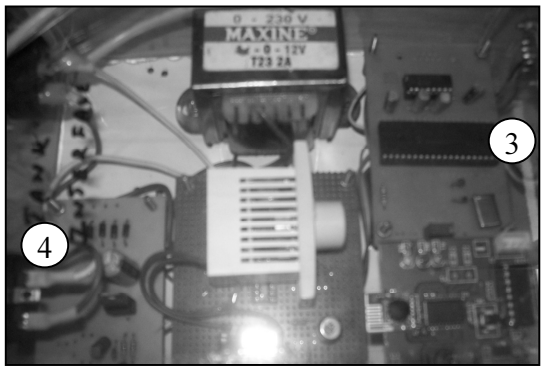


Figure 4 - Liquid Level Process



1. Conical tank 2. ZigBee Remote end

Figure 5 – Real Time Experimental Setup



3. Controller 4. ZigBee Server

Figure 6 - Client Section Board Setup

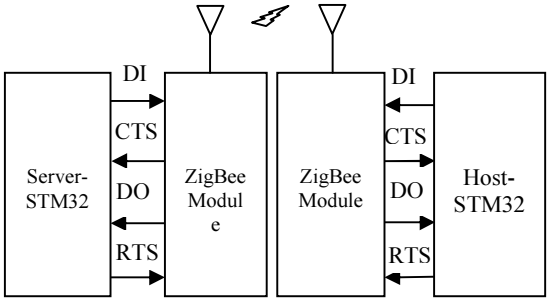


Figure 7 - Host/Server ZigBee Communication

The system setup details are described below. The connection details include

1. Connection between RS232 and STM32 board.
   a. Connect Tx pin of RS232 to PA2 of microcontroller. Connect Rx pin of RS232 to PA3 of microcontroller.
   b. Connect VCC pin and GND pin of RS232 to 5V and to GND.
2. Connect RS232 TTL Converter to XBEE module using UART cable (male to male).
3. Connect USB cable of the ZigBee module to the computer's USB port.
4. Connect the JTAG port of the board to the computer's USB port using ULINK2/JLINK.

6.4. Host Software Design

The µC/OS-II operating system is embedded host system to improve system reliability and stability. µC/OS-II operating systems are designed for embedded application. Most of the program is developed with C language. Because of the simple and well-knit structure, the µC/OS-II operating system becomes one of the most important

Journal of Theoretical and Applied Information Technology
10th July 2014. Vol. 65 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

embedded operating system. Table 2 list the details commands used to configure the ZigBee module.

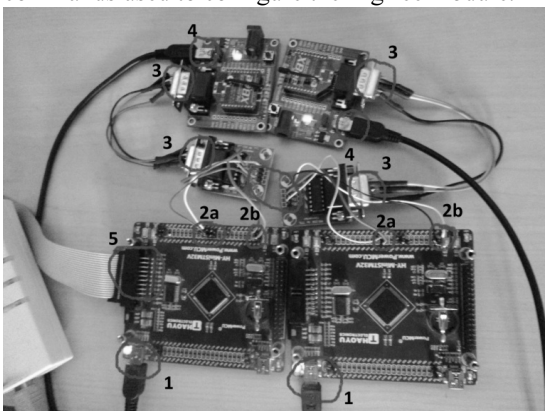


Figure 8 – Host/Server Setup using STM32

Table 2 - ZigBee Commands

Command	Description
'+++'	Enter into command mode.
ATID	The ATID command is used to set and read the PAN (Personal Area Network) ID of the RF.
ATMY	The ATMY command is used to set and read the 16-bit source address of the RF module
ATDL	he ATDL command is used to set and read the lower 32 bits of the RF module's 64-bit destination address
ATBD	The ATBD command is used to set and read the serial interface data rate used between the RF module and host.
ATCN	The ATCN command is used to explicitly exit the RF module from AT Command Mode

6.4.1. Steps to initialize ZigBee module

#defines the following Global variables:

a) MYID 10 (change to 11 for the second board)
b) DSTID 11 (change to 10 for the second board)
BAUD9600 3
c) NETWORKID 1111

Initialize the ZigBee module using AT command set.

a) +++
b) ATID = NETWORKID (network Id)
c) ATMY = MYID (Id of the this module)
d) ATDL = DSTID (Id of destination module)
e) ATBD = BAUD9600 (9600 baud rate)
f) ATCN (Exit Command Mode)

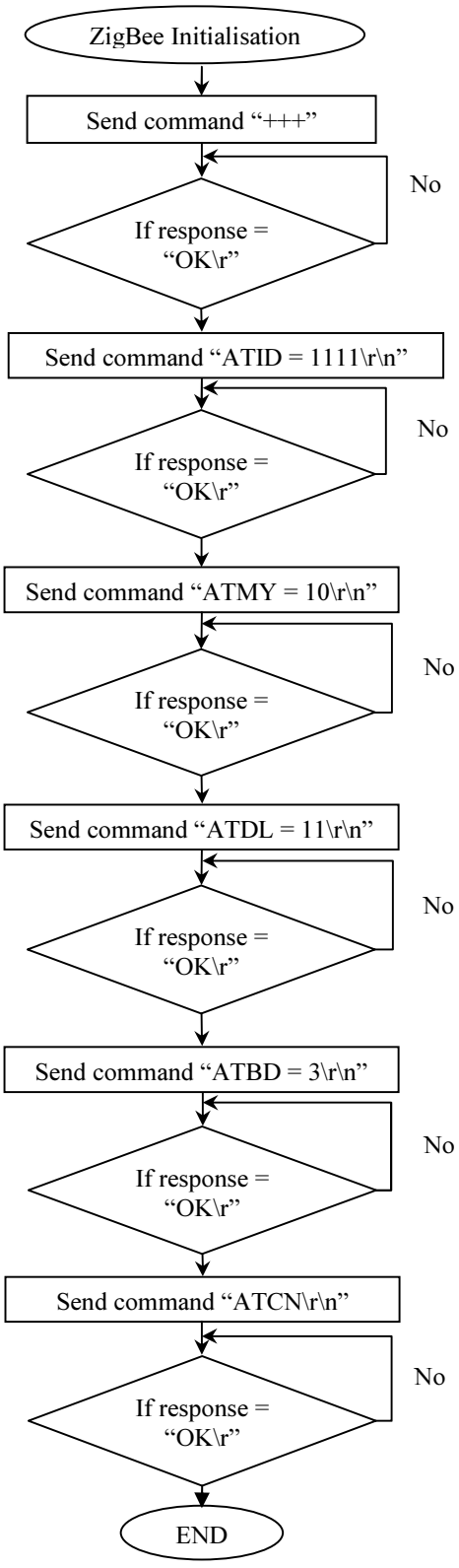


Figure 9 - ZigBee Initialization Flowchart

Journal of Theoretical and Applied Information Technology
10th July 2014. Vol. 65 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Response is "OK\r" for all the above commands. Check the response for each command before continuing with the next command. ZigBee Initialization Flow chart is shown in Figure 9.

6.4.2. Steps to transmit data

1) Send start of file, data and end of file using ZigBee module through USART peripheral.
2) Wait for USART flag USART_FLAG_TC (Transmission Complete flag) to set.

The flowchart fort transmitting zigBee data is given in Figure 10.

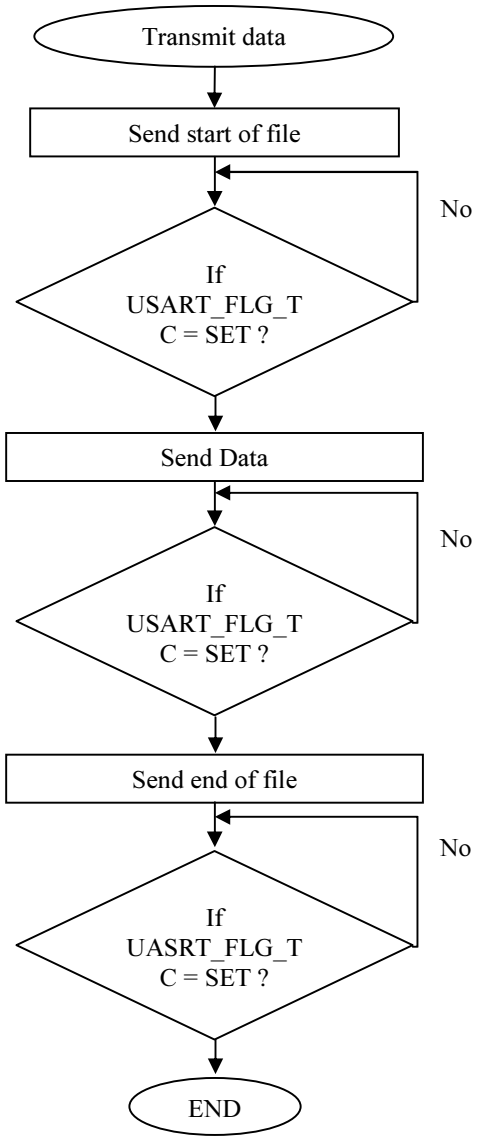


Figure 10 – ZigBee Data Transmission Flowchart

6.4.3. Steps to receive the data

1) Wait for USART flag USART_FLAG_RXNE (Receive data register not empty flag) to set.

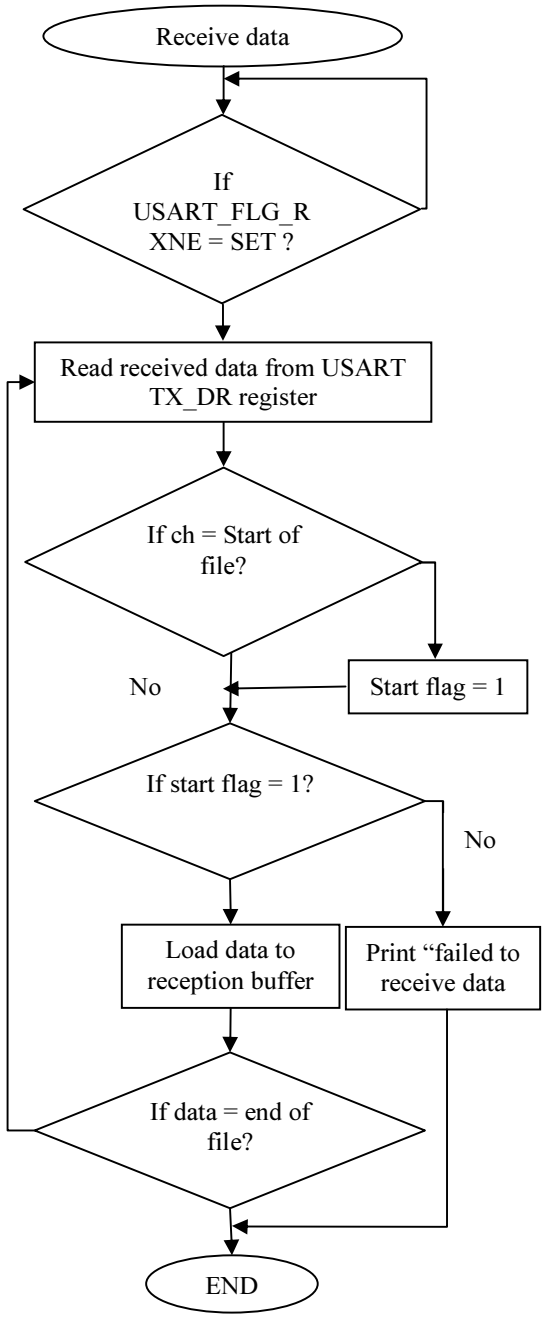


Figure 11 – ZigBee Receive Data Flowchart

Journal of Theoretical and Applied Information Technology
10th July 2014. Vol. 65 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2) Read the most recent received data by USART peripheral i.e., from USAR Tx DR register using ZigBee module.
3) Check for start of file.
4) If start of file is received, set the start flag else print failed to receive data.
5) If startflag is set, load the data in USARTx DR register to reception buffer till the end of file.
6) Wait for USART flag USART_FLAG_RXNE (Receive data register not empty flag) to set.
7) Read the most recent received data by USART peripheral i.e., from USAR Tx DR register using ZigBee module.
8) Check for start of file.
9) If start of file is received, set the start flag else print failed to receive data.
10) If startflag is set, load the data in USARTx DR register to reception buffer till the end of file.

The flowchart for receiving zigBee data is given in Figure 11.

6.4.4. Steps to Send Command

1) Send AT commands to ZigBee module through USART peripheral which is connected to this module with baud rate 9600.
2) Wait for USART flag USART_FLAG_TC (Transmission Complete flag) to set.

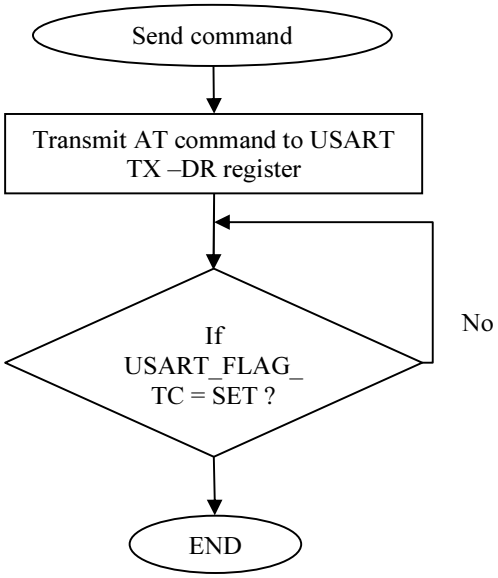


Figure 12 a– ZigBee Sending command Flowchart

6.4.5. Steps to Check Response

1) Wait for USART flag USART_FLAG_RXNE (Receive data register not empty flag) to set.
2) Receive the response from ZigBee module by USART peripheral.
3) Check the received response with "OK\r".

The flowchart for ZigBee commands for sending and checking response is given in Figure 12a and 12 b respectively.

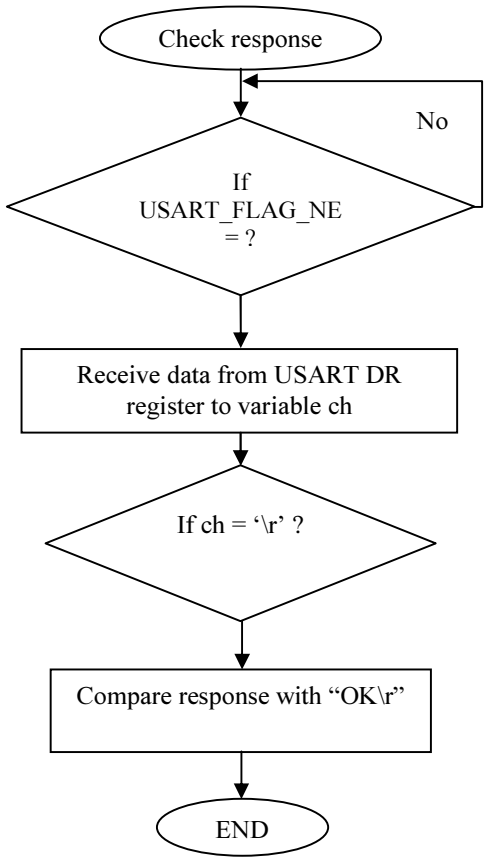


Figure 12 b– ZigBee Sending Response Flowchart

7. IMPLEMENTATION METHODOLOGY

At startup, the system runs initialization functions by itself by using restore system state and set the initial value of each parameter. Initialization task include (i) initializing all data structure (ii) allocating memory space for stack (iii) establishing semaphore message queue of inter-task communication (iv) create tasks and set different priority. The resources are allocated for the tasks defined in the application the scheduler is started then it schedules the tasks in pre-emptive manner.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The system turns fully operational when the initialization is done by using specified OS function.

7.1. Host System Task

In the Host system following tasks are introduced (i) Read Status (Task1) (ii) Display process status in host (Task2) (iii) (v) Transmit data status (task3) (iv) write set point(task4). All the tasks are in pre-emptive round robin scheduling algorithm. Tasks from task1 to task3 are running round robin. So when task1 is running all other tasks are in waiting state. The tasks become active when the scheduler schedule it. Tasks such as writing set point and transmit status data to server are based on the interrupt received from the ZigBee. Write set point message gives the values liquid level and outflow. When host system receives set point through ZigBee, it initiate write liquid level and outflow values to the controller. When host receives status read message from server, it response to server through ZigBee with values of liquid level, outflow and motor ON/OFF status.

8. RESULTS AND DISCUSION

The scheduling time for different tasks such as non-critical tasks Read Status (task1), Display process status in host (task2), Ttransmit status (task3) and Interrupt tasks (writing set points) are shown in Figure 13.

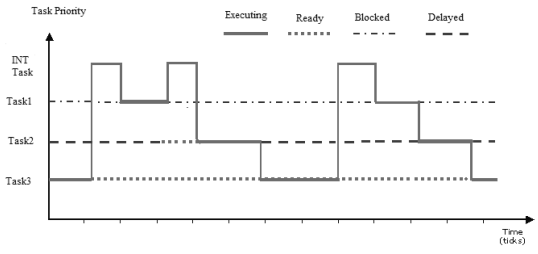


Figure 13 – Tasks scheduling time

From Figure 13 & Figure 14, it is observed that task execution time is reduced for realtime based system comparing to Non realtime based system considerably and improves system performance.

Task execution time, average waiting time, interrupt latency and CPU utilization time are greatly improved in realtime system. The snap shot of the resulst is given in Appendix-I.

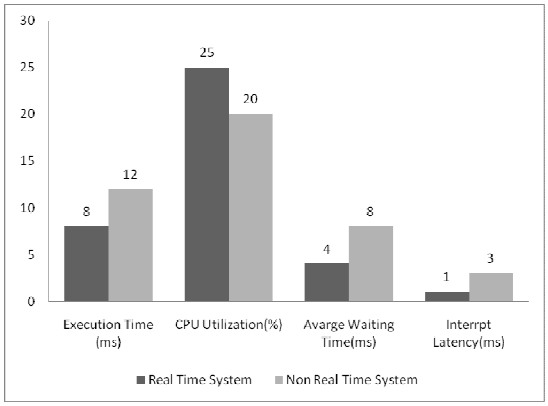


Figure 14 – Performance metrics

9. CONCLUSION

All the non-realtime activities have been effectively transformed into realtime using task based implementation to enhance the speed. From the obtained results it is inferred that realtime approach has shown significant benefits on process control applications that need fast execution time and optimal interrupt response time. The reliability and processing ability of the system are greatly improved. In future the system performance is to be improved further by integrating the proposed system with multitasking, task scheduling and inter task communication.

REFRENCES:

[1] Rajkamal, "Embedded Systems-Architecture, programming and Design 2nd Ed", *McGraw-Hill Education (India) Pvt. Ltd., ISBN-13: 0070667640*, 2008, pp. 296-423.

[2] Jean J.Labrosse, "MicroOS II- The real time kernel.2nd Ed", *CMP books, San Fransisco. ISBN-13: 978-1.57820.103-7, 2007, pp. 44-295.*

[3] Alen rajan Abt and K.Thomas, "ARM based Embedded web servers for industrial applications", *International conference on computing and control Engineering, Coimbatore Inst. ISBN:978-1-4675-2248-9, April 12-13*

[4] M. Franklin Richard, S. Sudheer Sukumaran, "A Real Time Industrial Automation System with Task Scheduling", *International Journal of Computer Sci ence And Technology, ISSN (Print) : 2320–3765, February 2014.*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[5] Pradibaa.S, Srimathi.R, Suganya.S, Sivaranjani.T and Aravind.P, "A Modelling and Analysis of Level Process Using Different Control Techniques", *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, ISSN (Print): 2229-4333, Vol. 3, Iss ue 2, April - June 2012.*

[6] Zengyu Peng and Yandong Zhao, "Real-time Monitoring System for Soil Moisture Content Based on µC/OS- II Operating System", *Computer Science and Information Technology (ICCSIT), 3rd IEEE International Conference, vol. 1, pp. 418 – 421, 2010.*

[7] Debashreet Das, Sibarama Panigrahi and Ashok Bhoi, "Real-time Industrial Automation Process Control using Network – Enabled RFID", *International journal of Computer Applications, vol. 61-No.10, pp. 42 – 46, 2013.*

[8] Ajit Sing, Priyanka Goyal, Sahil batra, "An optimized Round robin algorithm", *International journal of Computer Science and Engineering(IJCSE),vol. 2-No. 7, 2010.*

[9] QingLi and Caroline Yao, "Real-Time Concepts for Embedded Systems", *CMP Books, ISBN: 157820*124-1, *2003.*

[10] D.P.Bovet and M.Cesati, "Understanding the Linux Kernel", *O'Reilly & Associates, 2002.*

[11] C.Naga srikanth, M.Veda chary and M.Sudhakar, "Development of microkernel for multitasking with ARM11", International journal of Engineering Science and innovation technology, vol. 2, Issue-2, January, 2012.

[12] Nirmala R, Kolhari and Nithin I.Bhopale, "Porting & implementation of features of µC/OS II RTOS on ARM7 Controller LPC 2148 with different IPC mechanisms", *International journal of Engineering research & technology (IJERT), vol. 1, Issue-6,* August, *2012.*

[13] Ishwarya Singh Rajput and Deepa Gupta, "A priority based round robin CPU scheduling algorithm for realtime system", *International journal of* innovation in *Engineering & technology, vol. 1, Issue-3,* October, *2012.*

[14] Jane W. S. Liu, "Real-time System", *Person* Education*, ISBN: 9788177585759, September, 2010.*

Journal of Theoretical and Applied Information Technology
10th July 2014. Vol. 65 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

APPENDIX I

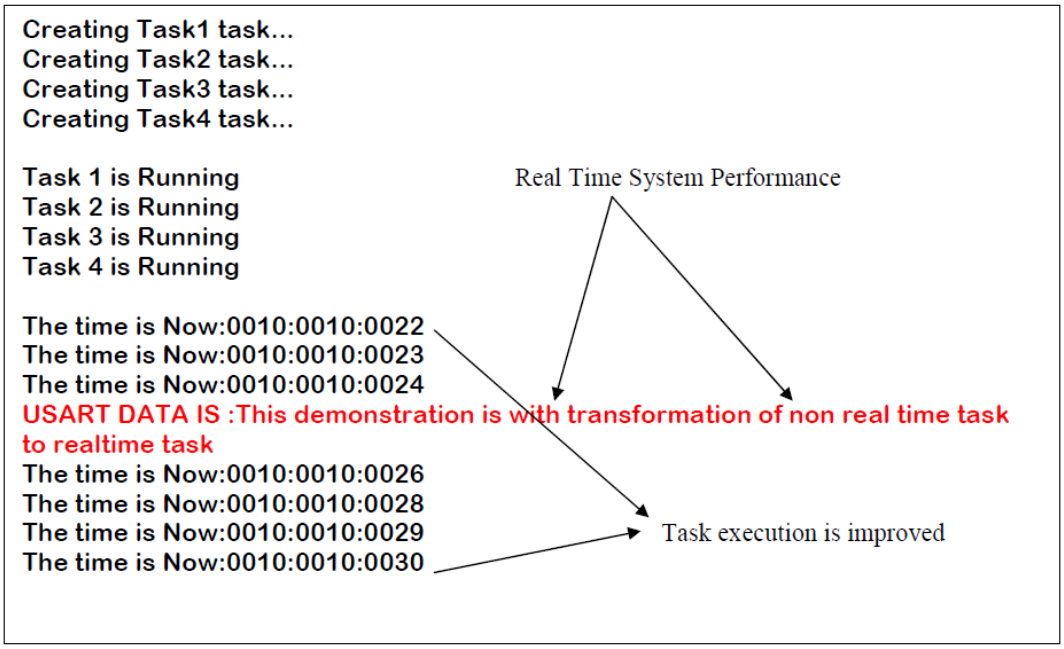


Figure 1 – Real Time System Task Execution

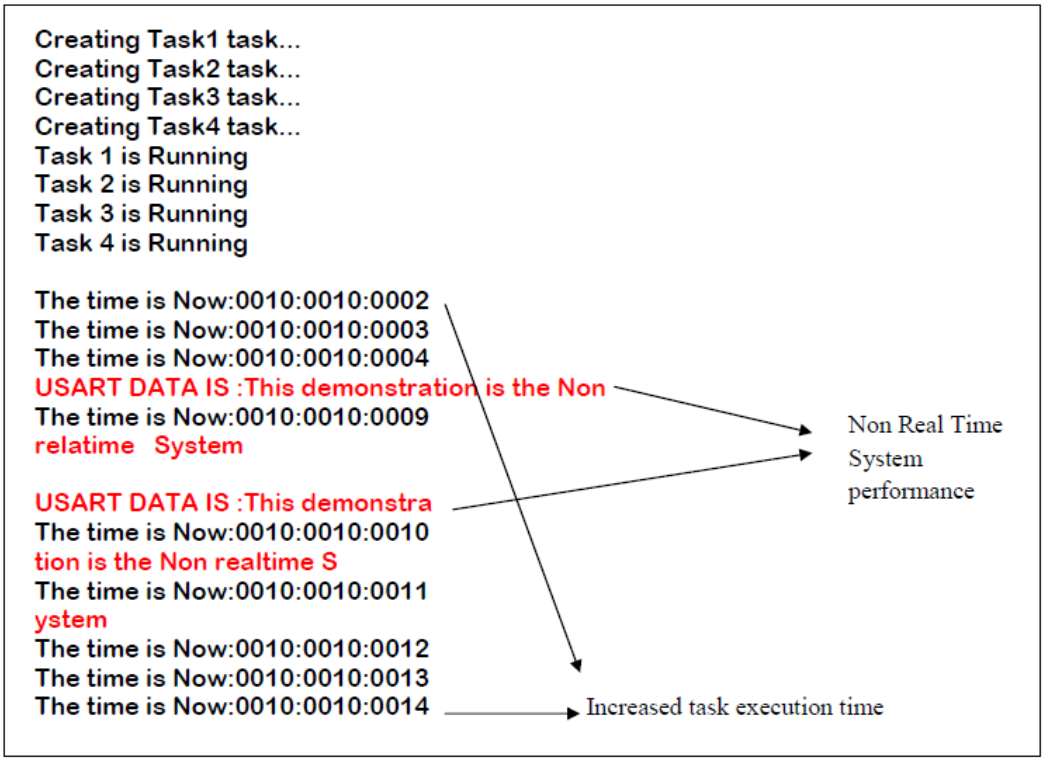


Figure 2 – Non -Real Time System Task execution