International Journal of Electrical and Electronics Research ISSN 2348-6988 (online)
Vol. 3, Issue 4, pp: (147-155), Month: October - December 2015, Available at: www.researchpublish.com

Implementation of Effective Real-Time Scheduling in Mine Detection Robot

[1]A. Sai Kumar, [2]Mr. G. Harish

[2]Assistant Professor, [1,2] Department of E.C.E, VLIST, VADLAMUDI-522213, Guntur (DT), Andhra Pradesh, India

Abstract: **This paper deals with loading artificial functions to a robot based on scheduling. Here, for implementation used a mine detection robot with 5 tasks of metal detection/bomb detection, obstacle avoidance, sound receiver for identification of blast, automatic lighting system in case of night vision and a communication module (GSM) for user interface. These five tasks are scheduled according to their priorities Based on RTOS scheduling. The semantic time scheduling is done to run all tasks at a time without any time delay. The project involves two sections 1. Robot section and 2.Monitoring section. The robot section deals with the data receiving from sensor nodes without any delay. The Monitoring section runs with RTOS and LPC2148 and acts as master node to which sensors are connected.**

Keywords: **RTOS, GSM, ARMLPC2148, sensor modules.**

I. INTRODUCTION

The purpose of a real-time operating system (RTOS) is to schedule tasks in order to guarantee that inputs are acquired and outputs are produced according to timing constraints [1]. In robotic applications, tasks periodically receive information about the environment through sensors or user interfaces, whereas commands to actuators and other outputs are sent at periodic intervals [2].

SEMANTIC TIME SCHEDULING is an ideal system for issues related to timing integrity, the extra traffic caused by the inter layer interaction in robotics [3]. In the existing prototype, we have noticed that, bulks of messages are transmitted between nodes so there are chances of message collision in transmission. In the proposed system we avoid this problem by optimizing the architecture and enhancing the system resources by implementing Real Time Operating System which manages the shared resources in real time environments, Besides the RTOS this system also provides power efficiency [4]; this Real Time Operating System is developed on micrium ucos-II OS [5]-[10].

The basic system requirements for porting this OS are that the device must have sufficient flash memory, on chip interrupts and timers [11]. This OS file structure is mainly divided into three sections, they are:

1) Processor specific files, where all the library files of the devices are available.

2) Application specific code, where all the header files of the user application code are present,

3) Processor independent code where all the OS concepts and their functions are available, such as OS_MUTEX.c, OS_SEM.c, etc,

The job of the RTOS is to manage the allocation of these resources to users in an orderly and controlled manner. For illustration used a mine detection robot, these kinds of robots are used in battle fields to detect the ground mines. By using this type of robot we can reduce the human damage and provides the information about mine in the border section. This kind of robots could be implemented with simple programming and without any scheduling and inbuilt OS, but those systems are failed in the case of delay parameters. Due to lack of coordination between the tasks there would be delay in the operation of the tasks and damage would be huge. For example let us take the task the robot is moving and the mine is detected, but the robot can't detect the mine without any delay, because after completion of programming section of robot

movement the programming section of mine detection runs, in this delay the mine might be exploded and the robot would crash and there is no use of system. To avoid this problem and to improvise the system performance we implemented the real-time scheduling in robotics.

By using effective real-time scheduling in robotics we can schedule the tasks according to their priorities. and priorities are could be shared between the tasks by using SEMAPHORE sharing process [13]-[16]. In this system we implemented 5 tasks of mine detection, obstacle avoidance, sound detection, communication module, changing light intensity in case of night vision. According to their priorities tasks are allocated as followed

1. 1st priority- mine detection for detecting ground mines
2. 2nd priority- obstacle avoidance and robot movement
3. 3rd priority- sound detection to detect if any ground mine is exploded
4. 4th priority- communication with the user to pass current status by using communication module
5. 5th priority- changing the light intensity according to the day light condition.

These 5 tasks are scheduled by using pre-emptive scheduling and in case of sound detection the priority should be changed from 3rd to 2nd priority of obstacle avoidance for this purpose we used priority inheritance to change the priority and it can run as the 2nd task.

Block Diagram:

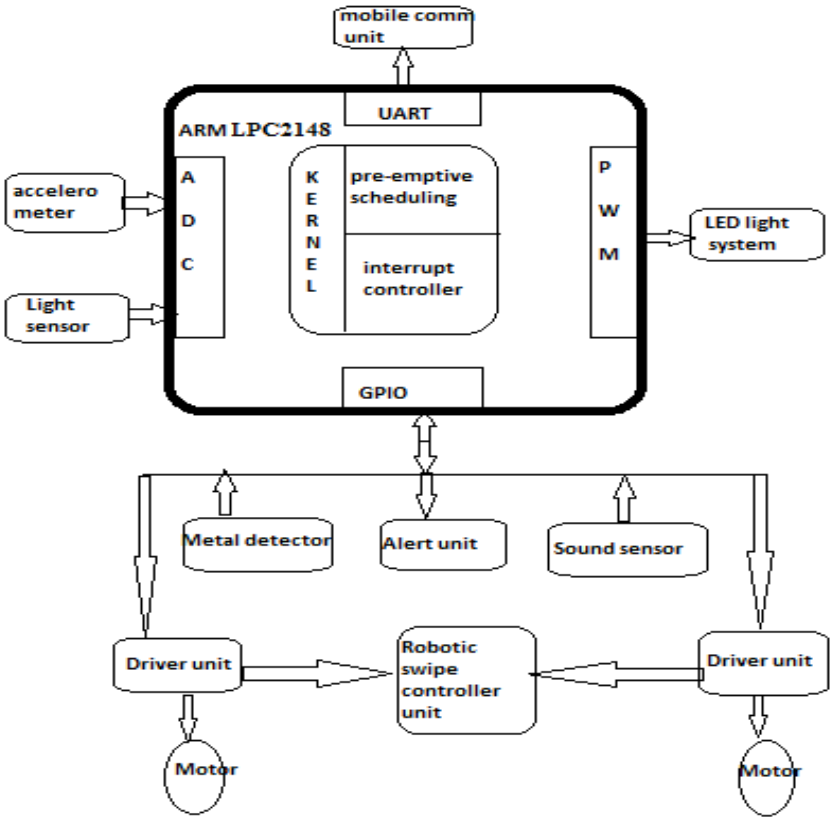

Fig.1. Block diagram of effective task scheduled metal detection robot

Fig.1 shows the block diagram of mine detection robot it consists of two sections

1. Robot module section
2. Monitoring section

The robot module section consist of sensors like metal detecting, LDR, sound sensors, MEMS sensor, one communication unit, one alert unit and robotic swipe control unit to monitor the direction of the robot.

The monitoring section consists of ARM7 processor LPC2148 of in built KERNEL, pre-emptive scheduling and interrupt controller services by installing µCOS-II OS in the processor. ADC, PWM, GPIO, UART are used to control the modules of the robot

II. HARDWARE

RTOS task scheduling and resource allocation are for the real time changing events; here the resources for the external events processed are a sound sensor, metal detector, MEMS accelerometer, and light dependent resistor. Values of these resources change in real time. Initially all of these task functionality are present in task dormant state, but TASK functionalities are not visible to the processor, to get these functions into TASK ready state we have to create a task by using this function OS_TASK_CREATE(), and OS_TASK_CREATE_EXT(), and the function is given below.

```
INTU OSTaskCreate (void (*task) (void *pd)),
Void *pdata,
OS_STK *ptos,
INTU8U prio)
```

Where we have the task name its data, stack memory size and priority of each task. The hardware includes ARM lpc218, GSM, sensor.

LPC2148 Processor:

LPC2148 Microcontroller Architecture: The ARM7TDMI-S is a general purpose 32-bit microprocessor, which offers high performance and very low power consumption. The ARM architecture is based on Reduced Instruction Set Computer (RISC) principles, and the instruction set and related decode mechanism are much simpler than those of micro programmed Complex Instruction Set Computers (CISC). This simplicity results in a high instruction throughput and impressive real-time interrupt response from a small and cost-effective processor core.

Pipeline techniques are employed so that all parts of the processing and memory systems can operate continuously. Typically, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory. The ARM7TDMI-S processor also employs a unique architectural strategy known as Thumb, which makes it ideally suited to high-volume applications with memory restrictions, or applications where code density is an issue.

The key idea behind Thumb is that of a super-reduced instruction set. Essentially, the ARM7TDMI-S processor has two instruction sets:

- The standard 32-bit ARM set.
- A 16-bit Thumb set.

The Thumb set's 16-bit instruction length allows it to approach twice the density of standard ARM code while retaining most of the ARM's performance advantage over a traditional 16-bit processor using 16-bit registers. This is possible because Thumb code operates on the same 32-bit register set as ARM code. Thumb code is able to provide up to 65% of the code size of ARM, and 160% of the performance of an equivalent ARM processor connected to a 16-bit memory system

GSM Overview:

A GSM modem is a wireless modem that works with a GSM wireless network. Global system for mobile communication (GSM) is a globally accepted standard for digital cellular communication. GSM is the name of a standardization group established in 1982 to create a common European mobile telephone standard that would formulate specifications for a pan-European mobile cellular radio system operating at 900 mhz.

GSM modems support an extended set of AT commands. These extended AT commands are defined in the GSM standards

Sending the message

To send the SMS message, type the following command:

AT+CMGS="+31638740161" <ENTER>

Replace the above phone number with your own cell phone number. The modem will respond with:

>

You can now type the message text and send the message using the <CTRL>-<Z> key combination:

TEST GSM! <CTRL-Z>

Here CTRL-Z is keyword for sending an SMS through the mobile device. After some seconds the modem will respond with the message ID of the message, indicating that the message was sent correctly:

+CMGS: 62

LDR sensor:

The system needs to know what the light level is in a particular room so when automating internal lighting it needs to know if the lights should be activated or not. Otherwise it defeats the purpose of energy saving by automating the lights for cost savings. This LDR wires directly into a M1 Zone Input (Any Zone). The Zone need to be programmed as a Analog Zone. The more light the LDR sensor has on it the lower the voltage the zone will read and the lower the light level, the higher the zone voltage [16].

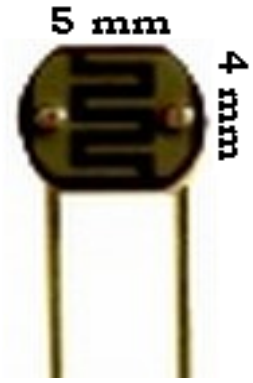

Fig.2. LDR sensor

MEMS sensor:

MEMS accelerometers [17]-[18] are one of the simplest but also most applicable micro-electromechanical systems. They became indispensable in automobile industry, computer and audio-video technology. This seminar presents MEMS technology as a highly developing industry. An accelerometer is an electromechanical device that measures acceleration forces. These forces may be static, like the constant force of gravity pulling at our feet, or they could be dynamic - caused by moving or vibrating the accelerometer.

Sound sensor:

Sound sensor uses the principle of microphone. The Microphone**,** also called a "mic", is a sound transducer that can be classed as a "sound sensor". This is because it produces an electrical analogue output signal which is proportional to the "acoustic" sound wave acting upon its flexible diaphragm. This signal is an "electrical image" representing the characteristics of the acoustic waveform. Generally, the output signal from a microphone is an analogue signal either in the form of a voltage or current which is proportional to the actual sound wave.

Metal sensor:

A metal detector is a portable electronic instrument which detects the presence of metal nearby. Metal detectors are useful for finding metal inclusions hidden within objects, or metal objects buried under ground. The simplest form of a metal detector consists of an oscillator producing an alternating current that passes through a coil producing an alternating magnetic field. If a piece of electrically conductive metal is close to the coil, eddy currents will be induced in the metal, and this produces a magnetic field of its own. If another coil is used to measure the magnetic field (acting as a magnetometer), the change in the magnetic field due to the metallic object can be detected.

III. DESIGN AND IMPLEMENTATION

The whole system coding can be divided into two parts

1. System OS coding
2. System tasks coding

Coding is done in KEIL IDE virsion4 here we used µcos-II as the OS. The compatibility of this OS is we can access the required functionalities by simply calling library files of the OS this library files are can download from authorized OS developer MICRIUM site. The development and compilation of code in KEIL IDE as shown in the Fig4

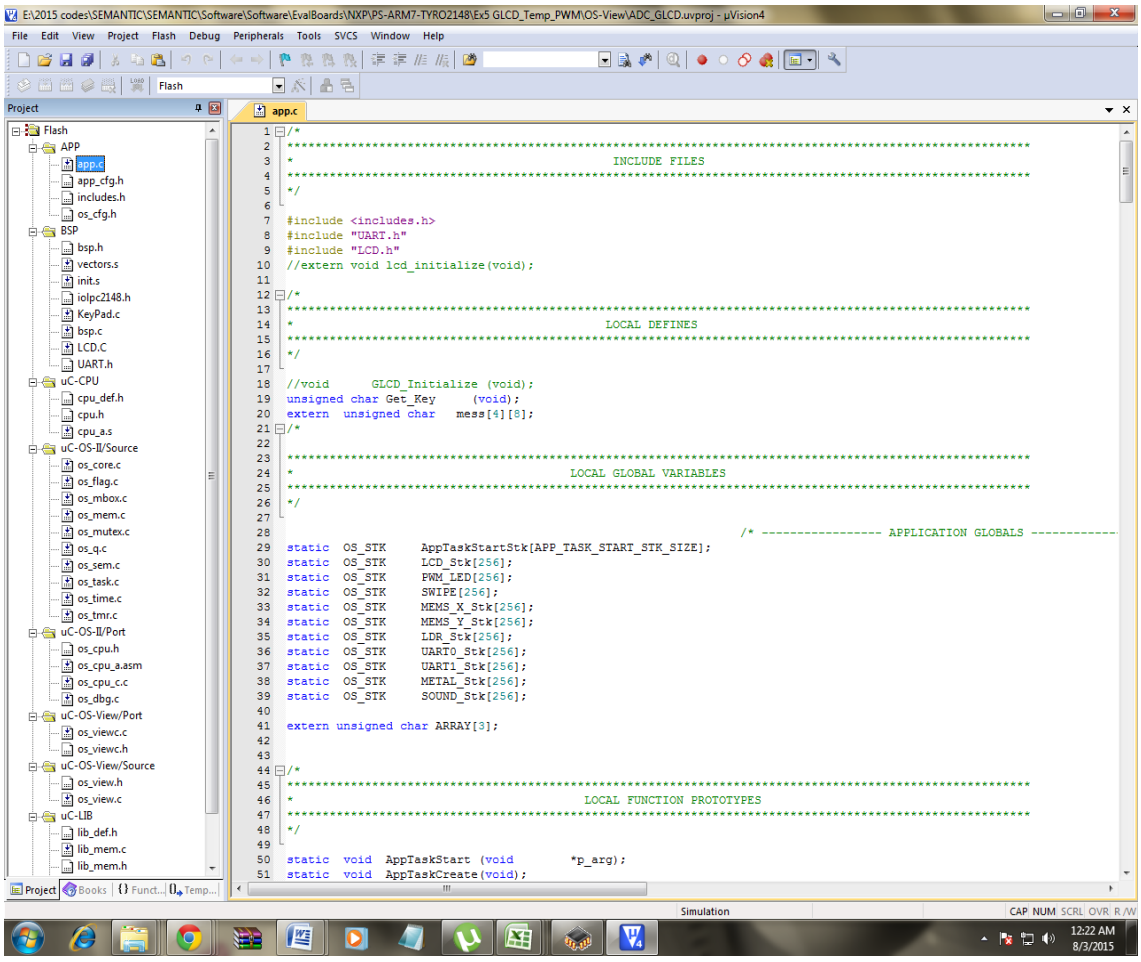

Fig.3. Development and compilation of the code

As shown in the Fig.3 the system OS includes system files like

1. BSP (board support package) in this we implement OS to processor interfaces and configuration of processor.
2. µc-CPU in this we will define all header files of the CPU
3. µc-osII/source in this we will define mutex, semaphore, mail boxes, message queues, time delays etc.
4. µc-osII/port in this will define the all the header files of the OS

As shown in the Fig.3 implementation of the tasks and their priorities allocation could be done under the APP system file. In this we write the coding of all tasks and use mutex, semaphores by simply calling header files of that functions.

Here we use semaphore for priority purpose, here we have metal detector has the highest priority and when it occurs it holds the semaphore by calling the function OS_SEM_start() and shows the status of the semaphore as '0', and after completion of the tasks it releases the semaphore by calling the function OS_SEM_release() and shows the semaphore status as '1'. And after releasing the semaphore next priority tasks like sound detection and obstacle avoidance acquires the semaphore as it required.

The controlling and communication could be done through UART by using GSM communication for this purpose we use message queues

The light sensors having the least priority and it could be run after completion of the all tasks. These five tasks are run in cyclic way with an infinite loop.

Sound detector which is connected to GPIO pins when sound is detected the respective GPIO pins go low and the status of the pin becomes low, this read through IOPIN register. And the condition if any sound is to send an information serially through UART1 which is configured with 9600 BR with 8 bit transfer, one stop bit, and no parity. This task is created through OSTASKCREATEEXT () function by specifying the task name, its relative data, stack memory, and PRIORITY etc., Metal detector which detects any metal explosives comprise of magnet, if any metal is detected a DC voltage is generated, this sensor is connected to GPIO pins of ARM, if any metal is detected the condition is to forward an information, relating to the metal.

The light dependent resistor where the resistance of it changes with the intensity of the light, detects the day/night light intensity, this sensor is connected to ADC0 of ARM to its respective channel and through AD0CR register and for respective value PWM output will be changed. The MEMS accelerometer which detects any vibrations (movements in un predictive direction which is caused due to bomb explosion) the incoming sensor values are in analog values, Through on chip ADC0 we convert these signals into digital through AD0CR register by configuring its channel, frequency, resolution and start condition. If the MEMS value exceeds the threshold value information will be transferred to main loop.

Here each and every task goes into TASK READY state whenever the its TASK create function is called, the OS_SCHED () function automatically schedules these tasks and sends the highest priority task into TASK READY state, in order to do this we have to call OS_START () OS_INIT () FUNCTIONS, where the functionality of the respective tasks starts executing according to their priority. Here each and every task must have at least anyone kind of delay and every TASK must be written in infinite.

Compilation of the code could be done in KEIL IDE and it convert in to execution file and dumps to the processor by using flash magic softwar

IV. RESULTS

The mine detection robot with all sensors setup is as shown in the Fig4. It connects with a PC by using serial communication cable to monitor the robot from remote access by using GSM communication module.

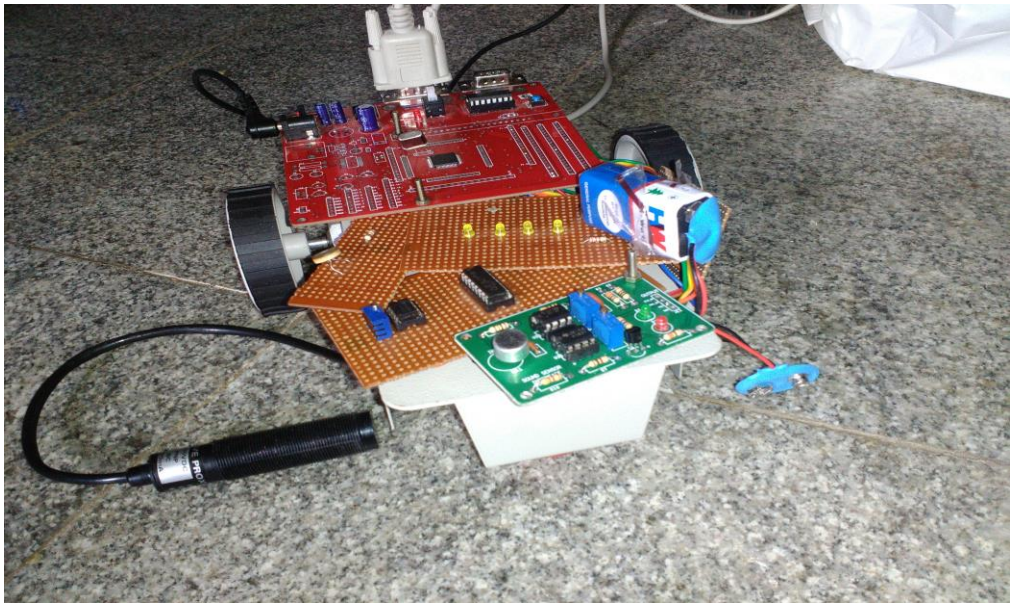

Fig.4. Mine detection robot

The change in tasks priorities and working status of the tasks are could be displayed on HyperTerminal in the PC which is connected to the robot through GSM communication module.

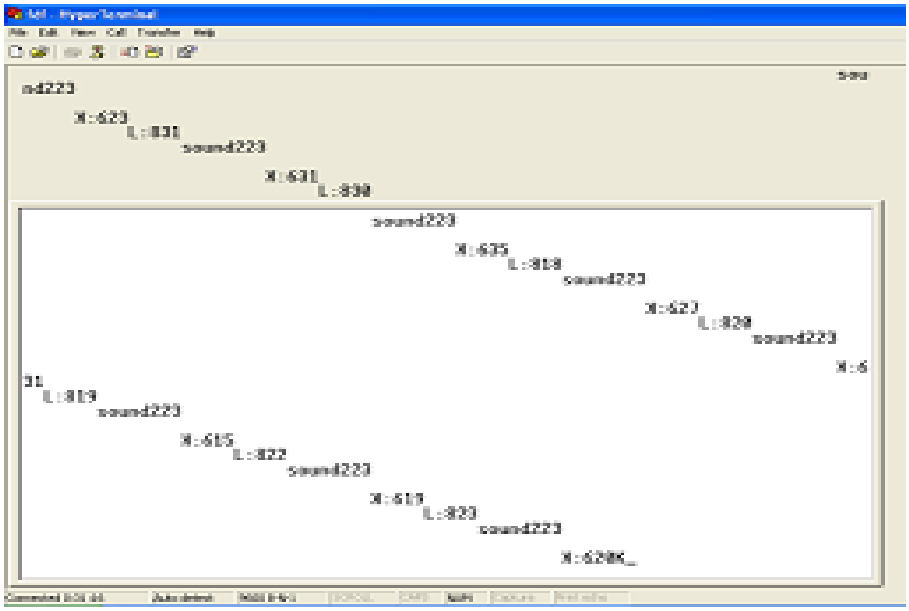

Fig.5. Status of tasks functioning in normal condition

Fig.5 represents the status of tasks which are functioning in normal condition i.e. there is no detection of ground mine or obstacle faced by the robot.

Here X represents the MEMS sensor value it increases when there is any obstacle or ground mine exploded under the robot and information could pass to the user and it would be controlled by the user.

The value L represents the light intensity of the LDR sensor, when there is day light condition the intensity of the LDR sensor is very high and the lighting system of the robot in off condition, if the day light decreases gradually then the intensity of the LDR sensor decreases gradually and the intensity of the lighting system would increases automatically.

The value sound represents the sound sensor if any ground mine exploded then intensity of sound increases and indicates the bomb is exploded.

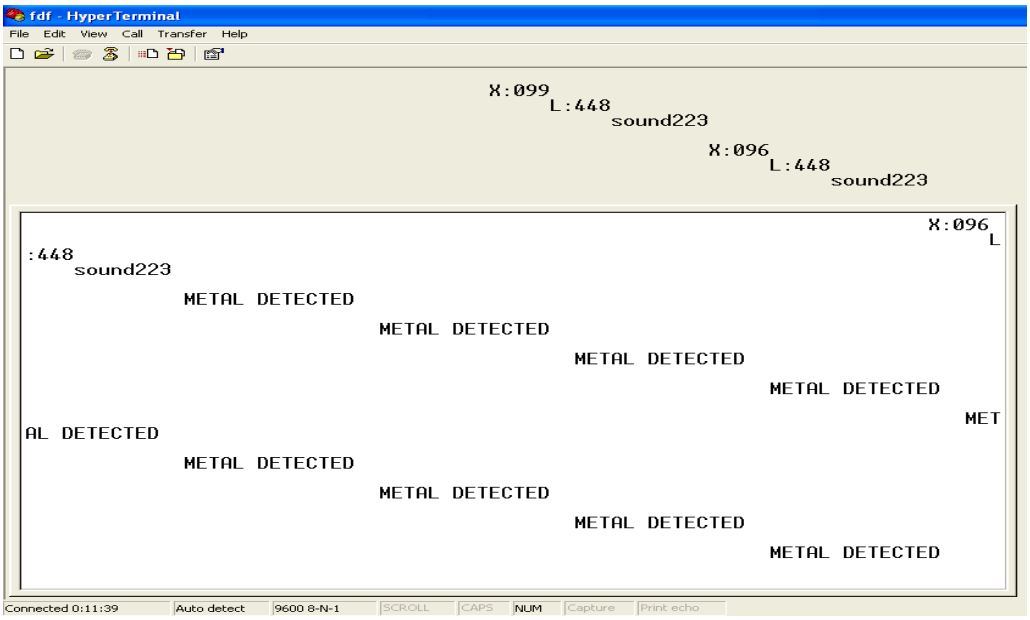

Fig.6. Ground mine detection

Fig.6 represents when ground mine detected, when there is ground mine it consists of metal and metal detector sense the metal and displays on the HyperTerminal as METAL DETECTED as shown in Fig3, as far as metal detector has the highest priority when it occurs all tasks are could be stopped and it runs and give the information about the ground mine without any delay

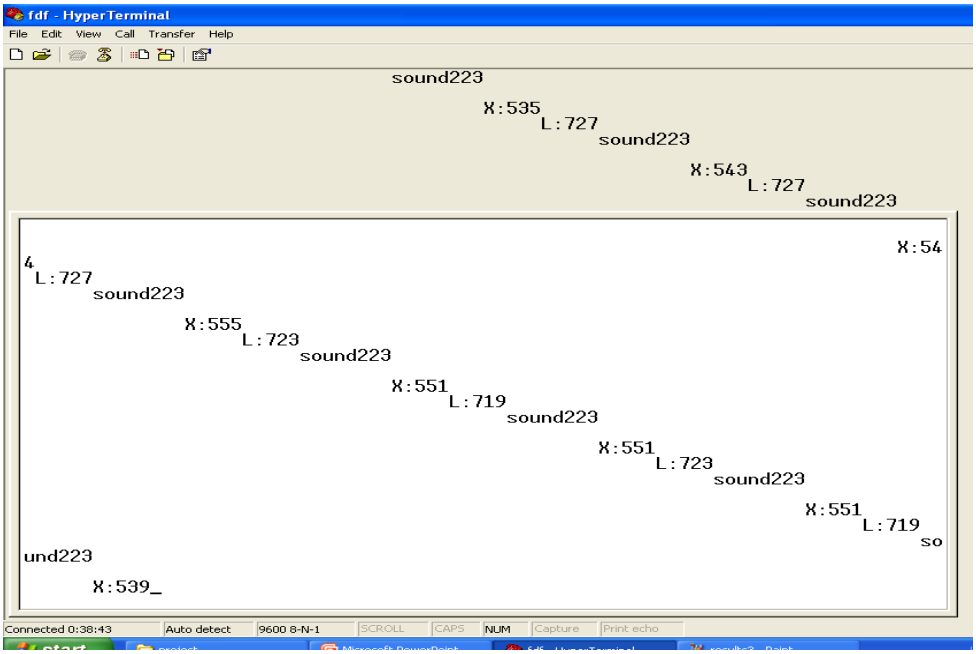

Fig.7. Priority inheritance

Fig.7 shows the priority inheritance. For sound sensor it has 3rd priority and in case of any explosion it jumps to the 2nd priority and gives the information about the explosion and after the completion it returns to its position and the remaining tasks obstacle avoidance and light intensity works according to their priorities

V. CONCLUSION

In this project we are making a self decision make and monitor robot application here, scheduling is used to avoid the delay between the applications. Based on RTOS scheduling has been done and a framework designed to deal with time period programming and scheduling of task sets depending on the present context and on the "semantic content of tasks."

REFERENCES

[1] R. Brennan, M. Fletcher, and D. Norrie, "An agent-based approach to reconfiguration of real-time distributed control systems," *IEEE Trans. Robot. Autom.*, vol. 18, no. 4, pp. 444–451, Aug. 2002.

[2] .I. A. D. Nesnas, A. Wright, M. Bajracharya, R. Simmons, and T. Estlin, "CLARAty and challenges of developing interoperable robotic software," in *Proc. 2003 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2003, pp. 2428–2435.

[3] .A. Brooks, T. Kaupp, A. Makarenko, S. Williams, and A. Oreback, "Towards component-based robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2005, pp. 163–168.

[4] Y. hsin Kuo and B. MacDonald, "A distributed real-time software framework for robotic applications," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2005, pp. 1964–1969.

[5] Shyan-Ming Yuan ; Dept. of Comput. & Inf. Sci., Nat. Chiao Tung Univ., Hsinchu, Taiwan ; Chiao-Jang Wu ; Hsiou-Mien Lien ; I-Neng Chen- "Design and implementation of a distributed semaphore facility" in *Proc. IEEE Int. Conf.* distributed computing systems Proceedings, 1992. pp. 180-184.

[6] Wang Xibo ; Sch. of Inf. Sci. & Eng., Shenyang Univ. of Technol., Shenyang, China ; Liu Tao- "Research on Subsystem Hybrid Scheduling and Priority Inversion Based uCOS-II" in *Proc. IEEE Int. Conf.* ICINIC, 2012 pp. 117-121.

[7] Xu Shuaiqing ; State Key Lab. of Engines, Tianjin Univ., Tianjin, China ; Wang Yang ; Tao Chengjun ; Song Mingzhi "Transplantation of RTOS uC/OS-II on TriCore processor" in *Proc. IEEE Int. Conf.* ICEICE Proceedings, 2011. pp. 5598-5601.

[8] Zhang Jiawen ; Shanghai Univ., Shanghai ; Hong Liang ; Lu Xiaofeng- "An Improved Memory Management Method of uC/OS-II" in *Proc. IEEE Int. Conf.* microwave Proceedings, 2008. pp. 558-560.

[9] Wang Xibo ; Shenyang Univ. of Technol., Shenyang, China ; Li Nan- "Embedded System Memory Management Mechanism Based on uC / OS-II" in *Proc. IEEE Int. Conf.* communication and mobile computing Proceedings, 2010. pp. 258-262.

[10] Mingsong Lv ; Northeastern Univ., Shenyang, China ; Nan Guan ; Yi Zhang ; Rui Chen-WCET "Analysis of the mC/OS-II Real-Time Kernel" in *Proc. IEEE Int. Conf.* computational science and engineering Proceedings, 2009. pp. 270-276.

[11] L. B. Becker and C. E. Pereira, "SIMOO-RT—An object-oriented framework for the development of real-time industrial automation systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2001, pp. 85–90

[12] Zuberi, K.M. Dept. of Electr. Eng. & Comput. Sci., Michigan Univ., Ann Arbor, MI, USA- "An efficient semaphore implementation scheme for small-memory embedded system" in *Proc. IEEE Int. Conf.* Real-Time Technology and Applications Symposium, 1997. pp. 25-34.

[13] Cena, G. Dipartimento di Autom. e Inf., Politecnico di Torino, Italy Valenzano, A.- "Efficient implementation of semaphores in controller area networks" in *Proc. IEEE Int. Conf.* industrial electronics Proceedings, *IEEE/RSJ* 1999. pp. 417-428.

[14] Gomez, E. ; Sch. of Comput. Sci. & Eng., California State Univ., San Bernardino, CA, USA ; Schubert, K.- "Algebra of Synchronization with Application to Deadlock and Semaphores" in *Proc. IEEE Int. Conf.* networking and computing Proceedings, 2010. pp. 202-208.

[15] Tapus, C. ; Hickey, J.- "Distributed synchronization with shared semaphore sets" in *Proc. IEEE Int. Conf.* cluster computing Proceedings, 2005. pp. 921-928.

[16] Shahmohammadi, M. ; Electron. Instrum. Lab., Delft Univ. of Technol., Delft, Netherlands ; Souri, K. ; Makinwa, K.A.A. "A resistor-based temperature sensor for MEMS frequency references" in *Proc. IEEE Int. Conf.* ESSCIRC Proceedings, 2013,. pp. 225-228.

[17] DAVID S. EDDY AND DOUGLAS R. SPARKS-"Application of MEMS Technology in Automotive Sensors and Actuators" in *Proc. IEEE/RSJ.*, 1998, pp. 1747-1755.

[18] Tai-Ran Hsu "RELIABILITY IN MEMS PACKAGING" in *Proc. IEEE Int. Conf.* Reliability Physics Symposium Proceedings, 2006. pp. 398-402.