ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Effect of Gain Time on Soft-Tasking and Scheduling in Real-Time Operating Systems for Collision Aversion Applications

K.Divya Jyothi
M.Tech Student,
Dept of ECE,
Institute of Aeronautical Engineering,
Dundigal, Quthbullapur, Hyderabad, Telangana, India.

Dr.P.Kesavarao
Professor,
Dept of ECE,
Institute of Aeronautical Engineering,
Dundigal, Quthbullapur, Hyderabad, Telangana, India.

Abstract:

The paper deals with Multitask and scheduling policies, 'gain time' is a key factor which explicit the difference between the actual time and maximum time for completion of a process This project deals with the effect of gain time on soft task scheduling in RTOS based application. RTOS is an operating system that supports real-time applications and embedded systems by providing logically correct result in time.In this project work , task will be completed within deadline by avoided delay, and by using an preemptive scheduler which can decide to suspend a task before finishing its execution & restart it later because a higher priority task becomes ready. This will have use in space ship docking, marine collision avoidance, and military and in automobile applications.

As a prototype demonstration we are implementing in the Hardware using ARM7 Processor for an automobile application. An object tracking System based on either object physical condition or sound will be fixed in the vehicle with vehicle-object distance measurement facility. If an object approaches the vehicle beyond the minimum prefixed distance limit, then preemptive scheduler will assign the object interference as a high priority interrupt. As an response the speed of the vehicle will be control by using PWM method. The Pulse width modulating technique will be initiated automatically by the Processor without any manual braking system and a information will be given to the person to bring concentration in driving.

Keywords:

RTOS, Ultrasonic, Preemptive scheduling, µCOS-II.

I.INTRODUCTION:

Embedded technology is now in its prime and the wealth of knowledge available is mind blowing. Embedded technology plays a major role in integrating the variousfunctions associated with it. This needs to tie up the various sources of the Department in a closed loop system. This proposal greatly reduces the manpower, saves time and operates efficiently without human interference. This project puts forth the first step in achieving the desired target. An embedded system is a combination of software and hardware to perform a dedicated task. Some of the main devices used in embedded products are Microprocessors [4] and Microcontrollers. Microprocessors are commonly referred to as general purpose processors as they simply accept the inputs, process it and give the output.

In contrast, a microcontroller not only accepts the data as inputs but also manipulates it, interfaces the data with various devices, controls the data and thus finally gives the result. The introduction of automotive Collision Warning Systems potentially represents the next significant leap in vehicle safety technology. Such systems attempt to actively warn drivers of an impending collision event, allowing the driver adequate time to take appropriate corrective actions to mitigate, or completely avoid, the event. Crash statistics and numerical analysis strongly suggest that such collision warning systems will be effective. Crash data collected by the U.S. National Highway Traffic Safety Administration (NHTSA) show that approximately 88% of rear-end collisions are caused by driver inattention and following too closely. These types of crashes could derive a beneficial influence from such systems.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

In fact, NHTSA countermeasure effectiveness modeling predicts that "head-way detection systems can theoretically prevent 37% to 74% of all police reported rear-end crashes." Clearly, the introduction of collision warning systems could result in the dramatic reduction of crash fatalities, injuries, and property damage. A real-time operating system (RTOS) is an operating system (OS) intended to serve real-time application requests.

It must be able to process data as it comes in, typically without buffering delays. Processing time requirements (including any OS delay) are measured in tenths of seconds or shorter. A key characteristic of an RTOS is the level of its consistency concerning the amount of time it takes to accept and complete an application's task; the variability is jitter]

A hard real-time operating system[3] has less jitter than a soft real-time operating system. The chief design goal is not high throughput, but rather a guarantee of a soft or hard performance category. An RTOS that can usually or generally meet a deadline is a soft realtime OS, but if it can meet a deadline deterministically it is a hard real-time OS[1].Timing is the most important factor in any real time applications where some applications should react in very small amount of time in those situation applications needs a platform where the timing constrain was meet.

In real-time systems, all real-time tasks are differentiated based on their timing, such as sporadic, response time, deadline etc. Real time systems are classified in to two types' hard real-time systems and soft real time systems. Hard real time system means it should complete the task within the deadline period otherwise its computation is useless. The damages caused by the hard real time systems are irreparable.

The system builder's should responsible to choose an operating system that can support and schedule these jobs with respect to their timing criteria so that no deadline will be missed. Soft real time systems require performance assurances from the operating system. Some applications such as video/audio visual gaming require performance assurance and also acceptable jitter in timing these applications comes under soft real-time applications.The general architecture of RTOS is shown in Figure 1.

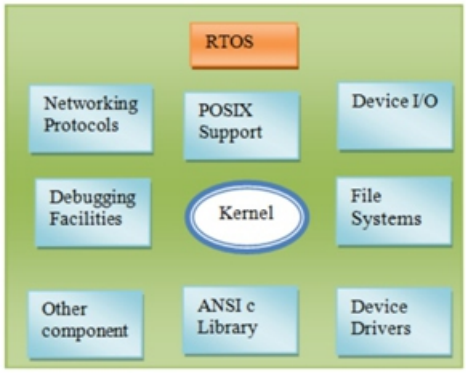
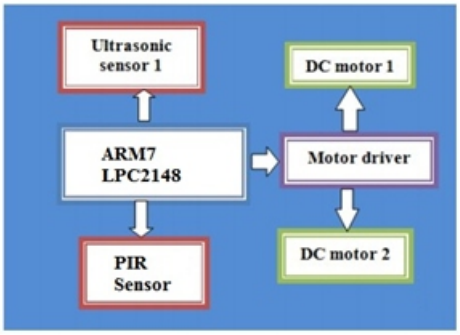


Figure 1. General architecture of RTOS.

Any RTOS basically have the characteristics like Multitasking and Preemptibility, Task Priority, Priority Inheritance, Short Latencies (Task switching latency Interrupt latency, Interrupt dispatch latency), Reliable and Sufficient Inter Task Communication Mechanisms, Control of Memory Management. The general purpose operating system for the proposed research work is based on KEIL flavor ( ). Many researches from the past few years used KEIL as their platform Figure 2 it is inferred that, use of KEIL is increasing in developing countries where technology increases1 .

II . DESIGN AND IMPLEMENTATION:

The proposed system design is implemented on ARM7 LPC2148 consists ARM11 as processor running Real time application consists four main components • ARM7 LPC2148 • Ultrasonic sensor module • DC motor drivers • DC motors. The proposed system with GPOS and with RTOS showed in Figure 3 and 4 respectively. Figure 3 shows ARM7 LPC2148 was booting with GPOS ( ) and in Figure 4 shows ARM7 LPC2148 was booting with RTOS (RT-Preempt patched Raspbian) and same application was executed on both platforms and response time is calculated.

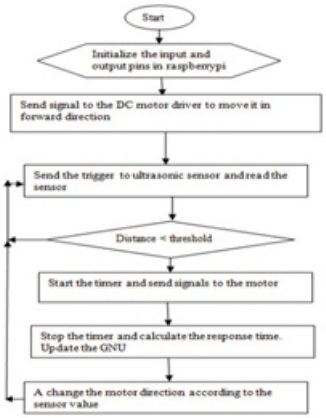


ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The main system of mobile robot is control by the ARM7 LPC2148 in both Methodologies, where the Ultrasonic sensor receive input from the control module and send the acquire data in the form of distance Measurement. The control module will make the decision when obstacle conditions are meeting and move in according to avoid the obstacle. The motor driver circuit is used to receive the signals from the control module in the form of binary logic and it will be supplied to the DC motors with sufficient voltage and current supply. The motor driver circuit will also protect the control module from DC motors when they get damaged.The experimental results were carried out on both platforms using mobile robot in real-time environment. KEIL was booted first and following steps are carried out, after that same was done on RTOS.

• Calculate the response time of the mobile robot in real-time environment.

• Calculate the time taken by mobile robot to move full rotation and half rotation on flat surface.



Flow chat of the system

Pre-Emptive Scheduling: Pre-emptive scheduling [2] retains many of the features described above e.g. tasks, task states / queues / priorities etc. However there is one very important difference. In a co-operative system a task will continue until it explicitly relinquishes control of the CPU. In a pre-emptive model tasks can be forcibly suspended. This is instigated by an interrupt on the CPU. These interrupts may be from external systems as above or possibly from the system clock.The difference here is that the scheduler is invoked following one of these system events. If a sensor detects an alarm condition, the input circuitry can generate an interrupt to the CPU. The Interrupt Service Routine (ISR) will be called immediately and may perform some suitable action such as setting a semaphore. However, instead of the ISR returning to the interrupted task, the scheduler is executed. The highest priority ready task will then be enabled which may or may not be an interrupted task. It can be seen that this allows systems to more rapidly respond to realtime[6] events in applications such as avionics, where any response delay may have serious effects. Additionally, clock interrupts may also invoke rescheduling, for example when a high priority task timer expires. There are some implications when using preemption. Overheads involved with interrupts and tasking will almost inevitably increase, and care must be taken to ensure that time critical data retains its integrity.

MICROCONTROLLERS LPC2148:

The LPC2148 microcontrollers is based on a 16-bit/32-bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combine the microcontroller with embedded high-speed flash memory ranging from 32 kB to 512 kB. A 128-bit wide memory interface and unique accelerator architecture enable 32-bit code execution at the maximum clock rate. For critical code size applications, the alternative 16-bit Thumb mode reduces code by more than 30 % with minimal performance penalty. Due to their tiny size and low power consumption, LPC2148 are ideal for applications where miniaturization is a key requirement, such as access control and point-of-sale. Serial communications interfaces ranging from a USB 2.0 Full-speed device, multiple UARTs, SPI, SSP to I2C-bus and on-chip SRAM of 8 KB up to 40 KB, make these devices very well suited for communication gateways and protocol converters, soft modems, voice recognition and low end imaging, providing both large buffer size and high processing power. Various 32-bit timers, single or dual 10-bit ADC(s), 10-bit DAC, PWM channels and 45 fast GPIO lines with up to nine edge or level sensitive external interrupt pins make these microcontrollers suitable for industrial control and medical systems.

ULTRASONIC SENSORS:

The ultrasonic distance sensor provides precise, non contact distance measurements from about 0.8 to 120 inches. ultrasonic sensor works by emit ting a short ultrasonic burst of sound Well above human hearing range and then “listening” for the echo. The ultrasonic sensor emits short bursts of sound and listens for this sound to echo off of nearby objects.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The frequency of the sound is too high for humans to hear (it is ultrasonic). The ultrasonic sensor measures the time of flight of the sound burst. A user then computes the distance to an object using this time of flight and the speed of sound (1,126 ft/s). This sensor uses ultrasonic sound to measure distance just like bats and dolphins do. Ultrasonic sound has such a high pitch that humans cannot hear it. This particular sensor sends out an ultrasonic sound that has a frequency of about 40 kHz. The sensor has two main parts: A transducer that creates an ultrasonic sound and another listens to its echo. To use this sensor to measure distance, the robot's brain must measure the amount of time it takes for the ultrasonic sound to travel. Sound travels at approximately 340 meters per second. This corresponds to about 29.412us (microseconds) per centimeter. To measure the distance the sound has travelled we use the formula: Distance = (Time x Speed of Sound) / 2.The "2" is in the formula because the sound has to travel back and forth. First the sound travels away from the sensor and then it bounces off of a surface and returns back. The easy way to read the distance as centimeters is use the formula, Centimeters = ((Microseconds / 2) / 29).For example, if it takes 100us (microseconds) for the ultrasonic sound to bounce back, then the distance is ((100 / 2) / 29) centimeters or about 1.7 centimeters.

L293D DRIVER IC:

The L293 and L293D are quadruple high current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600- ma at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive supply applications. In this project we provide two drivers, one for two motors which move the robot forward and backward and another for a motor which controls the arm of the robot.

SOFTWARE: KEIL (IDE):

It is possible to create the source files in a text editor such as Notepad, run the Compiler on each C source file, specifying a list of controls, run the Assembler on each Assembler source file, specifying another list of controls, run either the Library Manager or Linker (again specifying a list of controls) and finally running the Object-HEX Converter to convert the Linker output file to an Intel Hex File. Once that has been completed the Hex File can be downloaded to the target hardware and debugged. Alternatively KEIL can be used to create source files; automatically compile, link and covert using options set with an easy to use user interface and finally simulate or perform debugging on the hardware with access to C variables and memory. Unless you have to use the tolls on the command line, the choice is clear. KEIL Greatly simplifies the process of creating and testing an embedded application. Because of the high degree of flexibility required from the tools, there are many options that can be set to configure the tools to operate in a specific manner. It would be tedious to have to set these options up every time the application is being built; therefore they are stored in a project file. Loading the project file into KEIL informs KEIL which source files are required, where they are, and how to configure the tools in the correct way. KEIL can then execute each tool with the correct options. It is also possible to create new projects in KEIL.

III.Conclusion:

The overall research shows that ultrasonic sensor can detect the obstacle and communicate with control module to avoid the obstacle all this happens in 500ms second in GPOS and it is 250ms in RTOS. The results show that real time operating system have better response time compared to the general purpose operating system. RTOS is reliable even the threshold limit is given as 5cm. The average difference between the response time of GPOS and RTOS is 82µs. This response time is important in many real time applications. They are many direction of future work for this analysis one of it is this can be implemented in a maze and calculate the response of the mobile robot using GPOS and RTOS. This research can be implemented in many industries needs low response reactions, fire rescue operations and advance areas.

IV. Future Scope:

This system is implemented using the pre-emptive scheduling policy to reduce the gain time and handle the tasks based on priority. However, by using the scheduling policies we can accomplish the tasks efficiently but destinations cannot be determined. We can obtain an efficient FUTURE SCOPE This system is implemented using the pre-emptive scheduling policy to reduce the gain time and handle the tasks based on priority.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

However, by using the scheduling policies we can accomplish the tasks efficiently but destinations cannot be determined. We can obtain an efficient.

IV.REFERENCE:

1. Kshetri N. Economics of KEIL adoption in developing countries. IEEE Softwares. 2004 Jan/Feb; 20(4):7 4–81.

2. Marchesin A. Using KEIL for Real Time Applications. IEEE Softwares.2004; 15:18–21.

3. Hamblen JO, van Bekkum GME. An embedded systems laboratory to support rapid prototyping of robotics and the internet of things. Transactions on Education. 2013 Feb; 56(1):121–8.

4. Dash SK, Srikanthan T. Instruction cache tuning for embedded multitasking applications. IET Comput Digit Tech. 2010; 4(6):439–57.

5. Marieska MD, Kistijantoro AI, Subair M. Analysis and benchmarking performance of real time patch KEIL and Xenomai in serving a real time application. IEEE Electrical Engineering and Informatics.2011 Jul; 32(8):21–19.

6. Burdalo L, Terrasa A, Espinosa A, Garcia-Fornes A. Analyzing the effect of gain time on soft-task scheduling policies in real-time system. IEEE on Software Engineering. 2012 Nov/Dec; 38(6):1305–18.

7. Barbalace A, Luchetta A, Manduchi G, Moro M, Soppelsa A, Taliercio C.Performance comparison of VxWorks, KEIL, RTAI and Xenomai in a hard real-time application. IEEE Transactions on Nuclear Science.2008 Feb; 55(1):435–9.

8. ADEOS [Online]. 2015 Mar. Available from: http://www. adeos.org .

9. Weinberg B. Uniting mobile KEIL application platforms. KEIL Pundit. 2008.

10. Liu M, Liu D, Wang Y, Wang M, Shao Z. On improving real-time interrupt latencies of hybrid operating systems with two-level hardware interrupts. IEEE Transactions on Computers.2011 Jul; 60(7):978–91.

11. Regehr J, Stankovic JA. HLS: A framework for composing soft real-time schedulers. The 22nd IEEE Real-Time Systems Symposium (RTSS'01); 2001.

12. N.C. Audsley, R.I. Davis, A. Burns, and A.J. Wellings, "Appropriate Mechanisms for the Support of Optional Processing in Hard Real-Time Systems," Proc. IEEE 11th Workshop RealTime Operating Systems and Software, pp. 23-27, 1994.

13. L. Sha, B. Sprunt, and J. Lehozky, "Aperiodic Task Scheduling forHard Real-Time Systems," The J. Real-Time Systems, vol. 1, no. 1,pp. 27-60, 1989.

14. J.M. Banus, A. Arenas, and J. Labarta, "An Efficient Scheme to Allocate Soft Aperiodic Tasks in Multiprocessor Hard RealTimeSystems," Proc. Int'l Conf. Parallel and Distributed ProcessingTechniques and Applications, vol. 2, pp. 809-815, 2002.

15. J.M. Banus, A. Arenas, and J. Labarta, "Dual Priority Algorithm to Schedule Real-Time Tasks in a Shared Memory Multiprocessor," Proc. Int'l Parallel and Distributed Processing Symxp., 2003.

16. G. Bernat and A. Burns, "New Results on Fixed Priority Aperiodic.

Student Profile:

K.Divya Jyothi Pursuing M.Tech in IARE, Hyderabad, she pursued her B.Tech from ELLENKI College of Engineering for Women Patancheru.. Her research areas of interest are Embedded Systems and Communications.

Dr.P.Kesava Rao is working as professor in IARE, Hyderabad; He received his doctorate from JNTU Hyderabad in Spatial Information Technology. He was with ISRO, NRSC as Group Director. He was a recipient of ASI National award and ISRO team excellence awards. He is specialized in Satellite mission operations, Microwave Radars, Digital image processing and Remote Sensing.