International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 3, Issue 12, December 2014

Preemptive scheduling algorithm based object tracking system by using ARM processor

[1] Sukumar.P, [2] Sreenath.M, [12]Asst. Professors, Dept.Of E.C.E., A.I.T.S., Rajampet, Kadapa, A.P., India.

Abstract: *RTOS is an operating system that supports real-time applications and embedded systems by providing logically correct result within the deadline. In this, dealing with the effect of gain time on soft task scheduling in RTOS based application. In Multitasking gain time is a key factor which explicit the difference between the actual time and maximum time for completion of a process. In some real time applications, delay in a particular process may lead to severe effects. In this, delay in a task execution is avoided by using an effective preemptive scheduling and giving importance to high priority interrupts even though if there is any pending low priority interrupts on semaphore.*

As a prototype demonstration implementing the hardware using ARM Processor for an automobile application. An object tracking system will be fixed in the vehicle with vehicle-object distance measurement facility. If an object approaches the vehicle beyond the minimum distance limit, then preemptive scheduler will assign the object interference as a high priority interrupt. As an immediate response the speed of the vehicle will be changed by using a control method PWM. The Pulse width modulating technique will be initiated automatically by the Processor without any manual braking system and a heavy alert will be given to the person to bring concentration in driving.

Keywords– *RTOS, preemptive scheduling, ARM Processor, PWM.*

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 various functions 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. 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 real-time OS, but if it can meet a deadline deterministically it is a hard real-time OS[1].

II. SYSTEM ARCHITECTURE

Switch on the RF transmitter and RF receiver by using power supply switches. The LEDs will glow and indicates that the system is ready to operate. The RF transmitter has 4 switches they are sw1, sw2, sw3, sw4.

If press sw1 the RF receiver receive the from RF transmitter and run in autonomous mode the vehicle run automatically if any obstacle is found by using ultrasonic sensors it calculates the distance if the distance is found to be less than the pre- defined distance then it takes right turn or else forward .It also displays the safe distance measurement in the lcd.

.

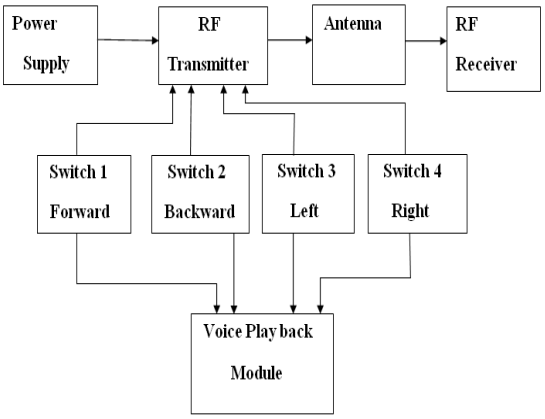

Fig 1: Transmitter Section

Now if press the sw2 then the vehicle runs in manual mode in this mode we can run this vehicle either by left or right by pressing the sw3 and sw4 .If we need to take backward turn then press sw3 and sw2 along this a play back voice is used whenever we press the switches the voice is played according to the switch operation.If press the sw3 we can run the vehicle in left direction and if we press the press sw4 we can run the vehicle in right direction.By pressing the sw2 and sw3 at the same time we can stop the vehicle.

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 real-time[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 pre-emption. Overheads involved with interrupts and tasking will almost inevitably increase, and care must be taken to ensure that time critical data retains its integrity.

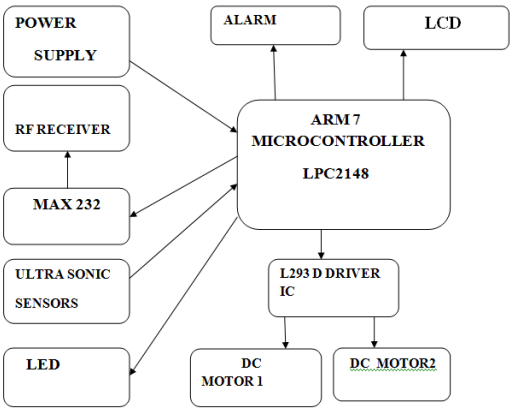

Fig 2: Receiver section

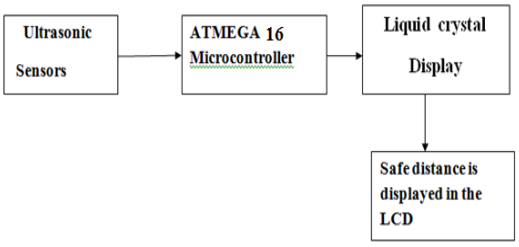

Fig 3: Range Measurement

1. 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.

ATMEGA16:

ATMEGA16 is an 8-bit high performance microcontroller of Atmel's Mega AVR family with low power consumption. Atmega16 is based on enhanced RISC (Reduced Instruction Set Computing, Know more about RISC and CISC Architecture) architecture with 131 powerful instructions. Most of the instructions execute in one machine cycle. Atmega16 can work on a maximum frequency of 16MHz.ATmega16 is a 40 pin microcontroller. There are 32 I/O lines which are divided into four 8-bit ports designated as PORTA, PORTB, PORTC and PORTD.ATmega16 has various in-built peripherals like Analog Comparator, SPI, JTAG , USART, ADC etc. Each I/O pin has an alternative task related to in-built peripherals.

2. VOICE MODULE

In order to save any voice and play the same voice back again we have **AUDIO RECORDER AND PLAY BACK** module. For saving any audio signals i.e., analog in nature we are using a Re-Recording voice IC called ARP9600.The APR9600 device offers true single-chip voice recording, non-volatile storage, and playback capability for 40 to 60 seconds. The device supports both random and sequential access of multiple messages. Sample rates are user-selectable, allowing designers to customize their design for unique quality and storage time needs. Integrated output amplifier, microphone amplifier, and AGC circuits greatly simplify system design. The device is ideal for use in portable voice recorders, toys, and many other consumer and industrial applications. APLUS integrated achieves these high levels of storage capability by using its proprietary Analog/multilevel storage technology implemented in an advanced Flash non-volatile memory process, where each memory cell can store 256 voltage levels. This technology enables the APR9600 device to reproduce voice signals in their natural form. It eliminates the need for encoding and compression, which often introduce distortion.

3. ULTRASONIC SENSORS:

The ultrasonic distance sensor provides pr ecise, non-contact distance measurements from abo ut 0.8 to 120 inches.ultrasonicsensor works by emit ting a short ultrasonic burst of soundWell above hu man 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. 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.

4. 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.

III. 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.

IV. RESULTS

1 AUTONOMOUS MODE:

In the section we have a5v battery, a voice play back module,RF transmitter and four push buttons.the power supply is given to the RF transmitter for the operation of the device.The voice playback isplayed according to the operation of the switches.

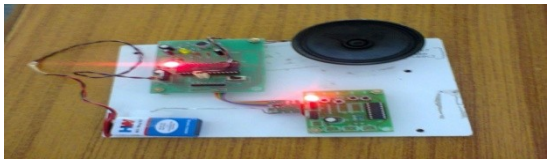

Fig 4: Transmitter Section

The receiver section consist of a RF receiver, ultrasonic sensor LCD, 12v battery, DC motors, a microprocessor, the ultrasonic sensor detects the object if any obstacle is found, the LCD displays whether the vehicle runs in autonomous mode or normal mode.

Fig 5: Receiver Section

This section shows the LCD displaying that the vehicle is autonomous vehicle

Fig 6: LCD Displaying That the Vehicle Is Autonomous Vehicle

The ultrasonic sensors detect the obstacle and send the interrupt signal to the microcontroller and moves in left direction if there is no obstacle in this direction.

Fig 7: Obstacle Is Detected And The Vehicle Is Moving In Left Direction

The ultrasonic sensor detects the obstacle and sends the interrupt signal to the microcontroller and moves in right direction if there is no obstacle in this direction.

Fig 8: obstacle is detected and the vehicle is moving in right direction

The vehicle is stopped because of obstacles in three sides i.e.; in right, left and forward.

Fig 9: The Vehicle is stopped in autonomous mode

2 MANUAL MODE:

The vehicle is moving in right direction and the LCD is displaying the mode and direction.

Fig 10:Right direction.

The vehicle is moving in the left direction and the LCD is displaying the mode and direction.

Fig 11: Left direction

The vehicle is moving in the forward direction and the LCD is displaying the mode and direction.

Fig 12: Forward direction

The vehicle is moving in the backward direction and the LCD is displaying the mode and direction.

Fig 13: Backward direction

V. CONCLUSION

At present the vehicles are being controlled by humans which might lead to accidents sometimes, so we have introduced a system which will work in both autonomous and manual mode by using pre-emptive scheduling which divides the tasks based on priority and functions the high prioritized tasks[5]. This vehicle can detect the obstacles automatically using the ultra-sonic sensors and proceeds in the obstacle free direction accordingly. So we can conclude that man can make mistakes but machines cannot. Non pre emptive scheduling executes the tasks in cyclic order so only one task can run at a time later it moves to next one even though it is a higher priority task. Hence pre-emptive scheduling is chosen.

VI. 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 system by using a GPS system along with the present, we can locate the destinations accurately.

VII. REFERENCES

[1] 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 Real-Time Operating Systems and Software, pp. 23-27, 1994.

[2] 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.

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

[4] 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.

[5] G. Bernat and A. Burns, "New Results on Fixed Priority Aperiodic.

[6] Luis Bu´ rdalo, Andre´s Terrasa, Agustı´n Espinosa, and Ana Garcı´a-Fornes "Analyzing the Effect of Gain Time on Soft-Task Scheduling Policies in Real-Time Systems "IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 38, NO. 6, NOVEMBER/DECEMBER 2012

Sreenath.M, received B.Tech degree in ICE from JNTU Hyderabad and M.Tech degree in Embedded Systems from JNTU Anantapur. Currently working as Assistant Professor in the Department of E.C.E., Annamacharya Institute of Technology and Sciences, Rajampet, Kadapa, AndhraPradesh, India. Areas of interests are embedded Real Time systems, Microprocessors & Microcontrollers and Control Systems.

Sukumar.P, received B.Tech degree from JNTU Hyderabad and M.Tech degree from ANU Guntur. Currently working as Assistant Professor in the Department of E.C.E., Annamacharya Institute of Technology and Sciences, Rajampet, Kadapa, Andhra Pradesh, India. Area of interests are, Microprocessors & Interfacing, embedded systems and Real Time Operating System.

