International Journal of Science Engineering and Advance Technology

A New Approach To Semantic Aware Real - Time Scheduling In Robotics

[1]V. Srinivas Hanumantha Rao, [2]V.V.Subhash
M.Tech Scholar[1], Associate professor[2] in Dept of ECE,
Kakinada Institute of Engineering & Technology-II, Korangi.

Abstract
This work deals with the data receiving from sensor nodes without any delay. The data receiving time is increased with the mobile communication. The section runs with RTOS and LPC2148 as master node to which sensors are connected. Communications between the military section and robot section through mobile communication technique.

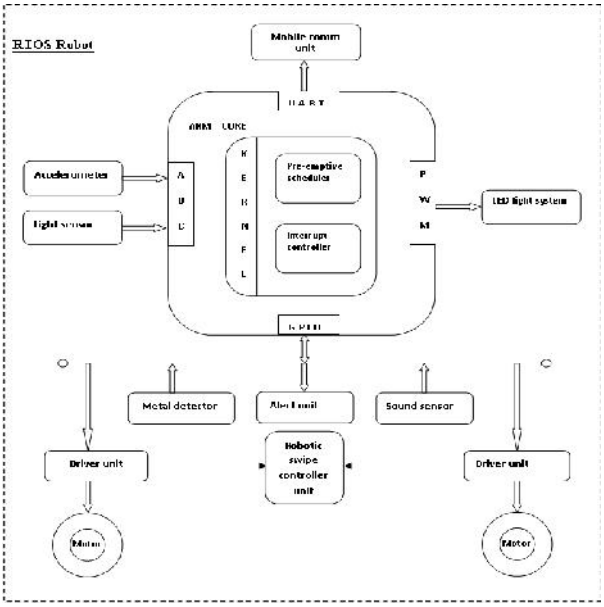
This is an RTOS based architecture designed for mine detection I uses scheduling to avoid the delay between one application with another. We mobile communication is used to receive the condition of the border level .This is done all at a time without any time delay.

Introduction
A system is something that maintains its existence and functions as a whole through the interaction of its parts. E.g. Body, Mankind, Access Control, etc A system is a part of the world that a person or group of persons during some time interval and for some purpose choose to regard as a whole, consisting of interrelated components, each component characterized by properties that are selected as being relevant to the purpose.

- Embedded System is a combination of hardware and software used to achieve a single specific task.
- Embedded systems are computer systems that monitor, respond to, or control an external environment.
- Environment connected to systems through sensors, actuators and other I/O interfaces.
- Embedded system must meet timing & other constraints imposed on it by environment.

An embedded system is a microcontroller-based, software driven, reliable, real-time control system, autonomous, or human or network interactive, operating on diverse physical variables and in diverse environments and sold into a competitive and cost conscious market.

An embedded system is not a computer system that is used primarily for processing, not a software system on PC or UNIX, not a traditional business or scientific application. High-end embedded & lower end embedded systems. High-end embedded system - Generally 32, 64 Bit Controllers used with OS. Examples Personal Digital Assistant and Mobile phones etc .Lower end embedded systems - Generally 8,16 Bit Controllers used with an minimal operating systems and hardware layout designed for the specific purpose. Examples Small controllers and devices in our every day life like Washing Machine, Microwave Ovens, where they are embedded in.



Design Of Embedded System:

Like every other system development design cycle embedded system too have a design cycle. The flow of the system will be like as given below. For any design cycle these will be the implementation steps. From the initial state of the project to the final fabrication the design considerations will be taken like the software consideration and the hardware components, sensor, input and output. The electronics usually uses either a microprocessor or a microcontroller. Some large or old systems use general-purpose mainframe computers or minicomputers.

RF Module Operation

Serial Communication:

The XBee OEM RF Modules interface to a host device through a logic-level asynchronous serial port. Through its serial port, the module can communicate with any logic and voltage compatible UART; or through a level translator to any serial device.

UART Data Flow

Devices that have a UART interface can connect directly to the pins of the RF module as shown inthe figure below.

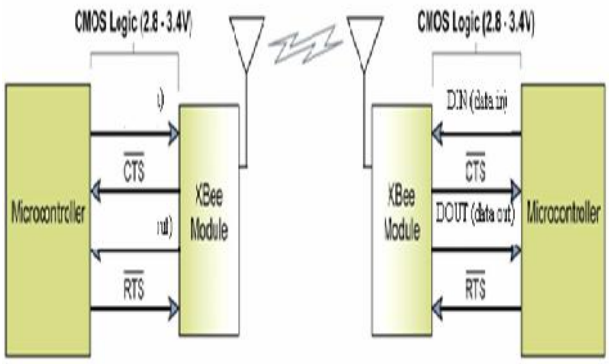


Figure 2.7.4 Zigbee UART Dataflow

Serial Data

Data enters the module UART through the DIN (pin 3) as an asynchronous serial signal. The signal should idle high when no data is being transmitted. Each data byte consists of a start bit (low), 8 data bits (least significant bit first) and a stop bit (high). The following figure illustrates the serial bit pattern of data passing through the module.

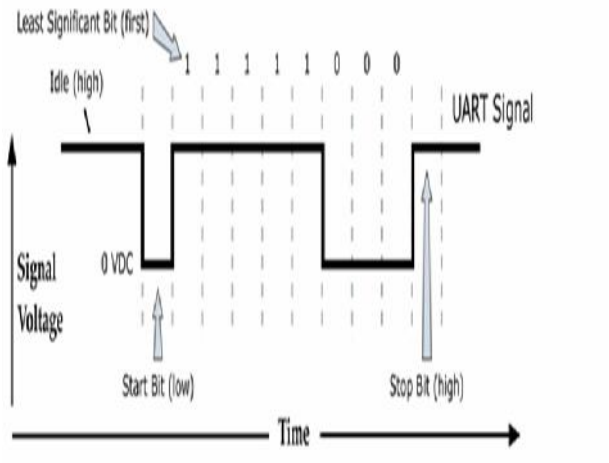


Figure 2.7.5 Serial Data Format

The module UART performs tasks, such as timing and parity checking, that are needed for data communications. Serial communications depend on the two UARTs to be configured with compatible settings (baud rate, parity, start bits, stop bits, data bits).

Serial Buffers:

The XBee modules maintain small buffers to collect received serial and RF data, which is illustrated in the figure below. The serial receive buffer collects incoming serial characters and holds them until they can be processed. The serial transmit buffer collects data that is received via the RF link that will be transmitted out the UART.

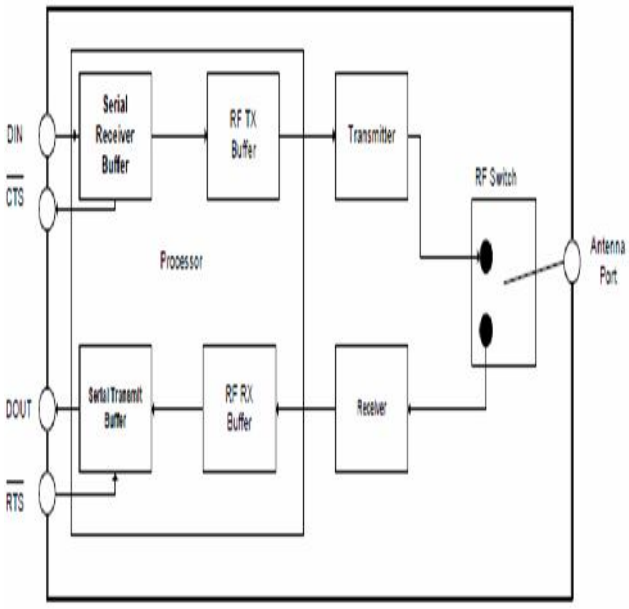


Figure 2.7.6 Internal Dataflow

Serial Receive Buffer

When serial data enters the RF module through the DIN Pin (pin 3), the data is stored in the serial receive buffer until it can be processed. Under certain conditions, the module may not be able to process data in the serial receive buffer immediately. If large amounts of serial data are sent to the module, CTS flow control may be required to avoid overflowing the serial receive buffer.

Cases in which the serial receive buffer may become full and possibly overflow:

- If the module is receiving a continuous stream of RF data, the data in the serial receive buffer will not be transmitted until the module is no longer receiving RF data.
- If the module is transmitting an RF data packet, the module may need to discover the destination address or establish a route to the destination.

After transmitting the data, the module may need to retransmit the data if an acknowledgment is not received, or if the transmission is a broadcast. These issues could delay the processing of data in the serial receive buffer.

Serial Transmit Buffer:

When RF data is received, the data is moved into the serial transmit buffer and sent out the UART. If the serial transmit buffer becomes full enough such that all data in a received RF packet won't fit in the serial transmit buffer, the entire RF data packet is dropped.

Cases in which the serial transmit buffer may become full resulting in dropped RF packets:

- 1. If the RF data rate is set higher than the interface data rate of the module, the module could receive data faster than it can send the data to the host.
- 2. If the host does not allow the module to transmit data out from the serial transmit buffer because of being held off by hardware flow control.

Serial Flow Control

The RTS and CTS module pins can be used to provide RTS and/or CTS flow control. CTS flow control provides an indication to the host to stop sending serial data to the module. RTS flow control allows the host to signal the module to not send data in the serial transmit buffer out the uart. RTS and CTS flow control are enabled using the D6 and D7 commands.

CTS Flow Control

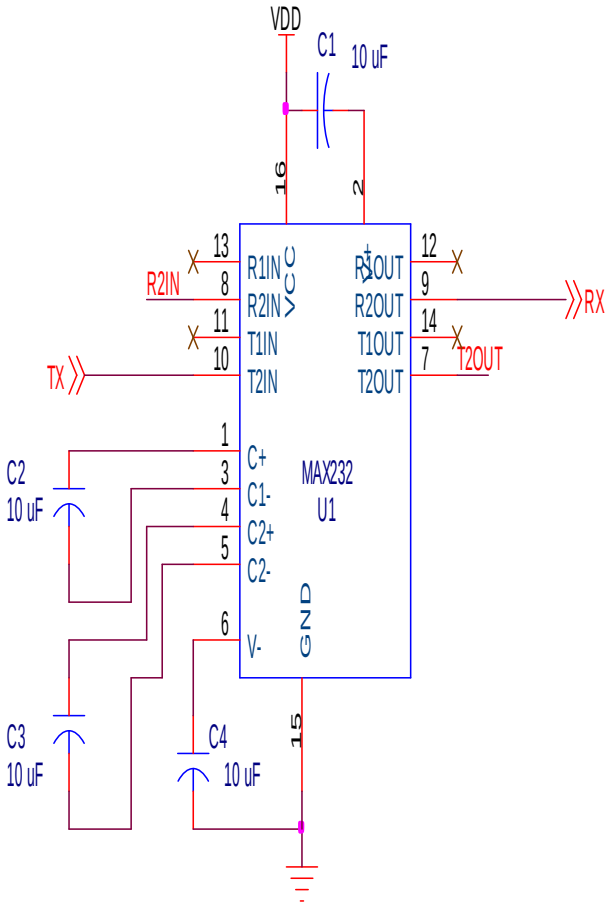
If CTS flow control is enabled (D7 command), when the serial receive buffer is 17 bytes away from being full, the module de-asserts CTS (sets it high) to signal to the host device to stop sending serial data. CTS is re-asserted after the serial receive buffer has 34 bytes of space.

RTS Flow Control

If RTS flow control is enabled (D6 command), data in the serial transmit buffer will not be sent out the DOUT pin as long as RTS is de-asserted (set high). The host device should not de-assert RTS for long periods of time to avoid filling the serial transmit buffer. If an RF data packet is received, and the serial transmit buffer does not have enough space for all of the data bytes, the entire RF data packet will be discarded.

2.8 RS232 Communication RS232*:*

Information being transferred between data processing equipment and peripherals is in the form of digital data which is transmitted in either a serial or parallel mode. Parallel communications are used mainly for connections between test instruments or computers and printers, while serial is often used between computers and other peripherals. Serial transmission involves the sending of data one bit at a time, over a single communications line.



In contrast, parallel communications require at least as many lines as there are bits in a word being transmitted (for an8-bit word, minimum of 8 lines are needed).Serial transmission is beneficial for long distance communications, whereas parallel is designed for short distances or when very high transmission rates are required.

Conclusion :

This paper introduced UART a framework designed to deal with time period programming and reconfiguration of task sets depending on the present context and on the semantic content of tasks.‖ this is often a haul that's typically left within the background by researchers within the field of intelligent robotic systems. Here, the matter has been formally outlined, the answer implemented by UART has been delineating intimately, and its theoretical properties are mentioned.

References :

[1] M. Piaggio, A. Sgorbissa, and R. Zaccaria, —Pre-emptive versus non-preemptive real time scheduling in intelligentmobile robotics,‖ J. Exp. Theor. Artif. Intell., vol. 12, no. 2, pp. 235–245, 2000.

[2] H. Bruyninckx, —Open robot control software: The OROCOS project,‖ in Proc. IEEE Int. Conf. Robot. Autom., Seoul, Korea, May 2001, vol. 3, pp. 2523–2528.

[3] L. B. Becker and C. E. Pereira, —SIMOO-RT—An objectoriented framework for the development of real-time industrial automation systems,‖ IEEE Trans. Robot., vol. 18, no. 4, pp. 421–430, Aug. 2002.

[4] 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.

[5] 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.V.SRINIVAS HANUMANTHA RAO, V.V. SUBHASH International Journal of Scientific Engineering and Technology Research Volume.04, IssueNo.09, April-2015, Pages: 1607-1610

[6] 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.

[7] 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.