International Journal of Computer Networks and Communications Security

VOL. 6, NO. 10, OCTOBER 2018, 219–222
Available online at: www.ijcncs.org
E-ISSN 2308-9830 (Online) / ISSN 2410-0595 (Print)

Real Time Implementation of Remote Rover Training System

Anwarul Azim

Assistant Professor, Department of EEE, University of Asia Pacific Bangladesh

azim@uap-bd.edu

ABSTRACT

The main objective of this paper is to highlight the implementation process of a mobile remote rover system with a compact and responsive real-time operating system. The communication process is concentrated on UART, Wi-Fi module & GPS module.

Keywords: *Remote Rover, Real-Time system, GPS Functionality, RTOS, TIVA-C.*

1 INTRODUCTION

This paper follows in the wake of mobile robotic platforms such as Pathfinder and Curiosity. These platforms service sensors, controllers, and interfaces to other systems in a timely manner to ensure proper operation over a long period of time. In order to increase reliability and ensure remote troubleshooting capability, the system must send heartbeat signals to their base station in a timely manner. These requirements impose soft and hard timing constraints on real time robotics systems.
Additionally, it is important that these systems be thoroughly tested to ensure proper operation over extended durations. This was exemplified by the Pathfinder mission, which experienced a priority inversion and required a remote software update. This priority inversion could have caused the major failure of a $150 million dollar mission. Luckily, the developers of the system were able to solve the priority inversion and continue the proper operation of the system.

.

2 SYSTEM DESIGN

The robot consists of four core hardware components, the chassis and electromechanical system, GPS module, Wi-Fi module, and processor board. The robot chassis [2] and motor controllers [3] were purchased through sparkfun electronics. The robot chassis is a simple steel container perforated with mounting holes, four motors and gearboxes. The motor drivers are capable of providing 1.2 amps per motor channel or 3.2 amps continuous current through both channels simultaneously.
GPS functionality is provided by a U-Blox Neo-6M GPS module [5]. This module establishes a GPS lock and transmits GPS location data over UART without additional configuration. This GPS data consists of longitude and latitude, orientation, velocity, and altitude. Figure 1 shows a GPS module used in the system:

Fig. 1. U-Blox Neo-6M GPS module [5]

Wireless functionality is provided by an ESP8266 Wi-Fi module [1]. The ESP8266 is a easily configurable as a client or ad-hoc hotspot and acts as simple Wi-Fi to UART bridge. Figure 2 shows a Wi-Fi module used in the system:

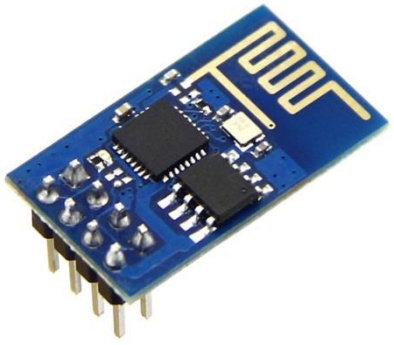

Fig. 2. ESP8266 Wi-Fi module [1]

The main processor is a Tiva-C development module designed by Texas Instruments [4]. This processor utilizes the TM4C123GH6PM microcontroller by TI, which implements the ARM Cortex M4F processor. The module is shown in figure 3:

Fig. 3. Tiva-C series Launchpad by Texas

3 OBJECTIVES

The Pathfinder mission and those that have followed have demonstrated the complexity of remote mobile robotic platforms. It is therefore important for future developers to have a high level of comfort working with these systems and low-cost, low-risk development platforms ensure that architectural or strategic problems can be discovered early in development.

The goal of this paper is to produce such a development platform. This platform will consist of the minimum of components and still be capable of autonomous remote operation to reduce complexity and cost. The primary parts of this system will be the primary processor board, Wi-Fi communication board, the GPS receiver board, and the motor control boards.

4 IMPLEMENTATION

Implementation process of a basic RTOS would depend on successful design of Kernel (Microkernel) that would handle the task management of the real time system. The task management process consists of inter task communication and synchronization, Timers handing, device I/O supervision and dynamic memory allocation. Task is the basic unit of the execution of RTOS, which would go through the scheduler implemented to perform the necessary actions. Scheduling policies can be clock driven or priority driven [6][7].

In addition to Kernel, a Message Passing Inter-process (MPI) communication system has to be designed to interface isolated tasks with each other and with the necessary modules through device drivers and controllers. Device driver is piece of software that enables devices connected to particular processor, via various interfaces. It controls, configures devices connected to the system. The generic task sequence used for this project is shown in figure 4.

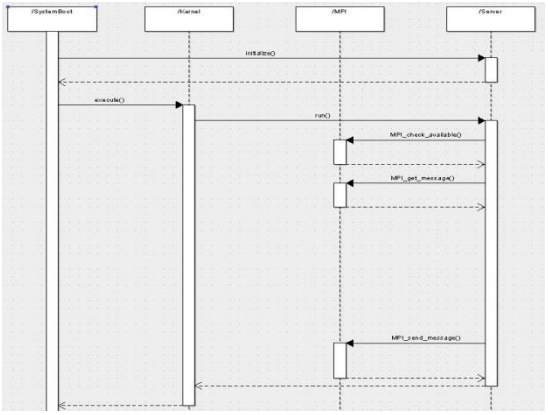

Fig. 4. Generic Task Sequence Diagram

The implemented RTOS in the remote rover system consists of a number of drivers & controllers such as main controller, UART driver, Wi-Fi driver, Wi-Fi controller, Console driver, GPS Driver, PWM Driver, and GPIO Driver as shown in figure 5. Each of the interface elements has the same structure with PID, run, and initialize members.

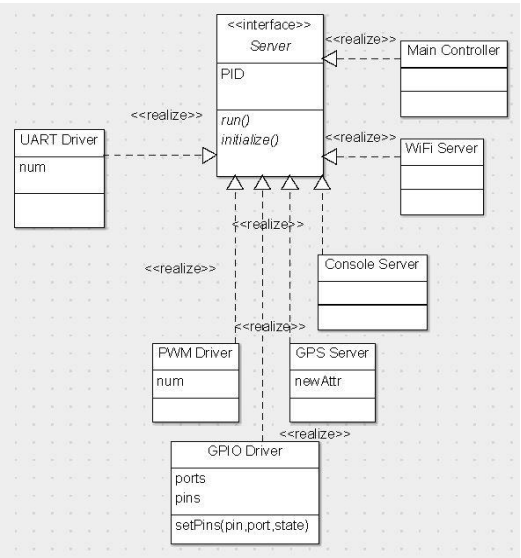

Fig. 5. Task interface diagram

Each of the controller and driver design for the project goes through initialize() and run() functions to perform a correct command sent from an HTML interface through a web server as shown in figure 6.

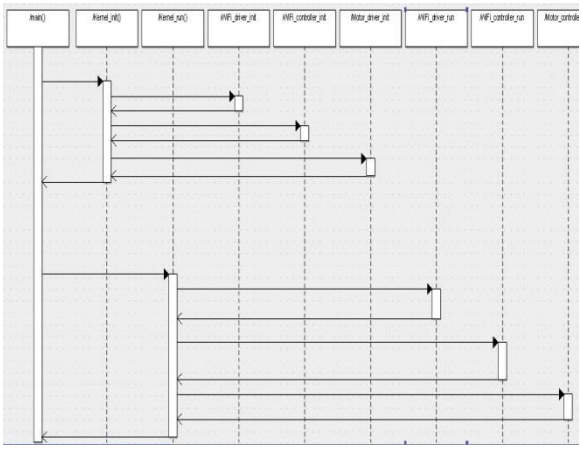

Fig. 6. Kernel Sequence Diagram

The main tasks of remote rover system such as motor controller, Wi-Fi controller, Wi-Fi driver are scheduled by a basic Rate Monotonic (RM) algorithm and their schedulability can checked by calculating worst case execution time (WCET) analysis from Boundt tool and utilization factor from fpscalc tool. As the design of the remote rover system based on an ARM Cortex processor, the machine level codes are not recognized by Boundt tool which was built for ARM7 processor system designs. So the average case execution time analysis through hyperperiods is done the remote rover system to check the schedulability of the tasks as shown below in figure 7.

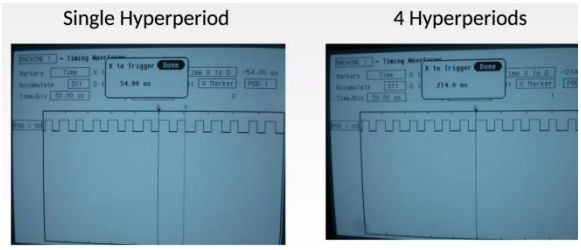

Fig. 7. Average case execution time analysis

The Message Passing Inter-process (MPI) communication in the system goes through the basic operations of availability check, receive and send by executing functions such as MPI_check_available(), MPI_get_message() and MPI_send_message(). Each of these operations would go through error check such as valid memory locations, buffer overflow, command validity, process ID validation, mailbox capacity & time limits etc. All boundary values for error analysis are defined in header files for the corresponding drivers and controllers. All the send and receive operations are synchronized in a way that sender and receiver always suspends activity until proper corresponding validation. The message queuing order is set as first come first serve (FIFO) just to reduce design complexity for now, but it can be updated to a priority-based system. So the process synchronization is done through mutual exclusion depending on whether mailbox is empty or not[8].

The deadlines for the tasks are set by the UART modules, which operate at 9600 baud. At this speed, the system requires that each controller/driver pair which operate a UART module to execute every 0.8 ms. This is by far the most pressing deadline for the current system. Furthermore, if this deadline is not met repeatedly, bytes may accumulate in the UART buffers without being fetched quickly enough, leading to a buffer overflow and loss of data. This would cascade into a failure to interpret a command, and finally a failure to close the TCP connection in a timely manner, causing a network error. This problem can be overcome if another WCET tool is developed or discovered which can analyze modern ARM architectures or languages. Unfortunately, the best that can be done with the current implementation is a hardware reset that detects a failure to meet the deadline, and a subsequent reset. This functionality has been experimented with, but is currently unimplemented. As this project is intended to produce a platform upon which students can run experiments and test various principals of real-time networking and operating systems, it is expected that code development will continue for long after this project has concluded.

In various runs of the implemented RTOS in remote rover system, all the tasks were executed correctly to interface with the commands given through HTML page setup in the web server (bluezone@USU). At first consecutive commands were missing deadlines after a few rounds of correct initial execution due to the limited buffering capacity in Tiva C launchpad. So a delay has been introduced in between consecutive executing tasks so that system can properly recognize the valid commands and execute it within deadlines. It somewhat mitigated the issue for now for a low number of tasks but to be more robust there need be a new approach to schedule the tasks and recognize the validity of commands given.

5 CONTRIBUTION

This project will contribute to the experience and familiarity of developers working with real time systems in mobile applications. Development systems for such applications are often prohibitively complex and expensive. This project will demonstrate that it is possible to create a system, which can be used to emulate a far more complex platform, allowing developers to work in a low-risk environment and gain experience inexpensively.

9 REFERENCES

[1] ESP8266-community. Esp8266 community forum, 2015 [Online]. Available: http://www.esp8266.com/
[2] Sparkfun. Multi-chassis - 4wd kit (basic), 2013 [Online]. Available: https://www.sparkfun.com/products/12089
[3] Sparkfun. Sparkfun motor driver - dual tb6612fng (1a), 2013 [Online]. Available: https://www.sparkfun.com/products/9457
[4] Texas Instruments. Tiva-c series Launchpad evaluation kit, 2015 [Online]. Available: http://www.ti.com/tool/ek-tm4c123gxl
[5] U-Blox. Neo-6, 2011 [Online]. Available: http://www.u-blox.com/en/gps-modules/pvt-modules/previous-generations/neo-6-family.html
[6] J.Corbet, A. Rubini, G.Kroah-Hartman, Linux Device Drivers, 3rd Edition, 2005.
[7] Free RTOS training Materials [Online]. Available : http://www.intelinfo.com/it_training_materials_and_books/free_real_time_operating_systems_training_materials.html
[8] D.E. Comer, Network Systems Design Using Network Processors, 1st Edition, 2009.