eYSIP2016

FreeRTOS on LPC2148

K V S SUMAKAR
KARTIKEYAN V
Rutuja
Deepa

Duration of Internship: 10/06/2016 – 24/07/2016

2016, e-Yantra Publication

Contents

FreeRTOS on LPC2148

Abstract

FreeRTOS provides open source libraries for implementing RTOS on around 32 microcontrollers.The API's are provided in form of C files which can be included into the projects to implement RTOS.

As part of this project we have implemented FreeRTOS on LPC2148. Created modules to illustrate various concepts like MultiTasking Semaphores,Mailbox,Queues and context switching on RTOS.

As part of application of the project we have developed a collision avoidance code both on RTOS and on C.We have also implemented state collector using RTOS and written a python script which would give a csv file of the data.

Completion status

Give details for work/project completed successfully. If work is not complete, mention the details till which task is done.

- The implementation of following concepts have been done
  - Multitasking
  - Binary semaphore
  - Counting Semaphore
  - Mutex
  - Mailbox through task notification.
  - Queues
  - Context switching
- As part of Documentation/tutorial

  - Documentation for Setting up FreeRTOS on keil.
  - Documentation for above mentioned concepts in form of a Book.

- As part of Mini-project
  - Comparision of Collision avoidance Robot using C and RTOS.
  - State collection using RTOS

- Interfacing With smartphone via USB.
  Status : Unsuccessful .

1.1 Hardware parts

- List of hardware
  - FireBird V with LPC2148 adapter board.
  - Xbee modules with Adapter boards.
  - USB A to USB B type cable.
  - DB9 cable.
  - serial to usb converter.
  - Sharp IR Sensors.

1.2 Software used

- FreeRTOS
  version 9.0.0 download link, Sourceforge llink
- Keil uVision
  version 4, download link,
  Installation steps ,Git link
- Flash Magic
  version 5.7, download link,
  Installation steps ,Git link
- Python IDLE
  Version 3.5, download link,
- X-CTU
  Version 6.3.1, download link,

1.3 Code

- **MultiTasking :** Multitasking is the ability to perform various tasks at the same in a single core processors like LPC2148. RTOS will schedule each task according to their priorities and higher priority task will be executed first and the rest will be scheduled later. Following state diagram will explain it better.

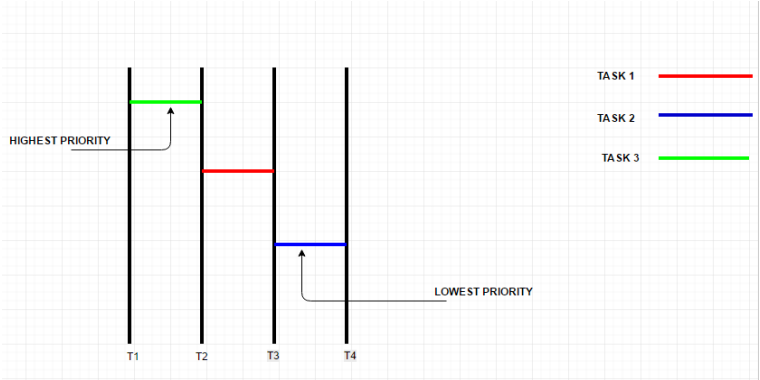

Figure 1.1: Multitasking state diagram

- **Binary Semaphore:**Semaphores are used for task synchronization,the above code illustrates how two tasks are ensured resources(i.e control of motors) with the help of Binary semaphore.

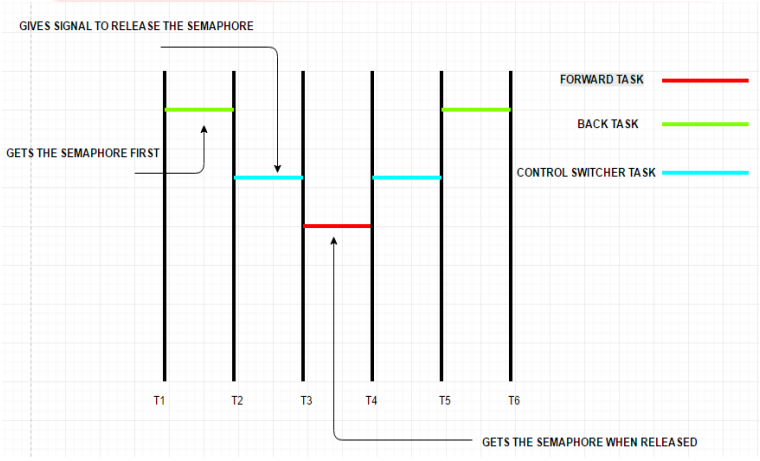

Figure 1.2: Binary Semaphore state diagram

- **Mutex:**Mutex are semaphores which ensure that the resources are only used by only one process. Once the task having the semaphore releases it after its use then only then the other tasks can get the semaphore. This is illustrated with an example which has output similar to that of Binary semaphore but has difference on implementation level.

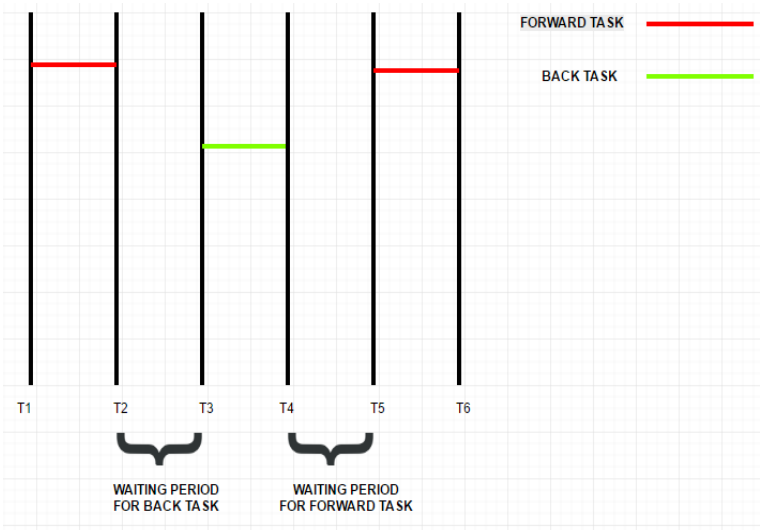

Figure 1.3: Mutex Semaphore state diagram

- **Counting Semaphore:**The concept of counting semaphore has been illustrated by implementing a simulation of the dining philosophers problem.

- **MailBox :**FreeRTOS doesn't directly support Mailbox but it can be implemented using TaskNotification.The message is communicated through predefined functions whose parameters can be varied to perform different operations on message,send different messages and manipulate existing or incoming message.

- **Queues :**Queues can be used to share data between multiple tasks,the data to be shared can be pushed into the queue and the tasks can receive data by receiving from the queue.FreeRTOS provides API's to make static queues,send data to queue and receive data from queue.A timeout period can also be mentioned for receiving/sending data to queue.

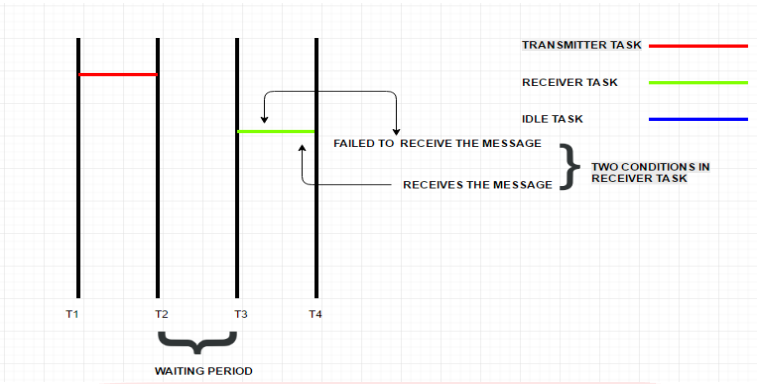

Figure 1.4: Queue state diagram

- **Context Switching:**Context switching is saving the state of the process in the memory when another process is allocated the processor time.This has been illustrated with switching between tasks of different priorities.

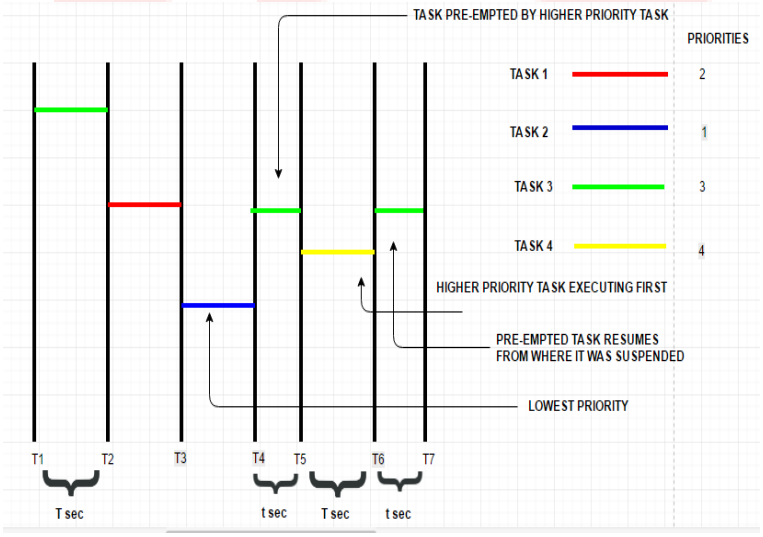

Figure 1.5: Context Switching

- **State collection :**This code consists of a Task which stores data in a variable and another which sends it through UART,using these "tasks" sensor values can be obtained for later use.

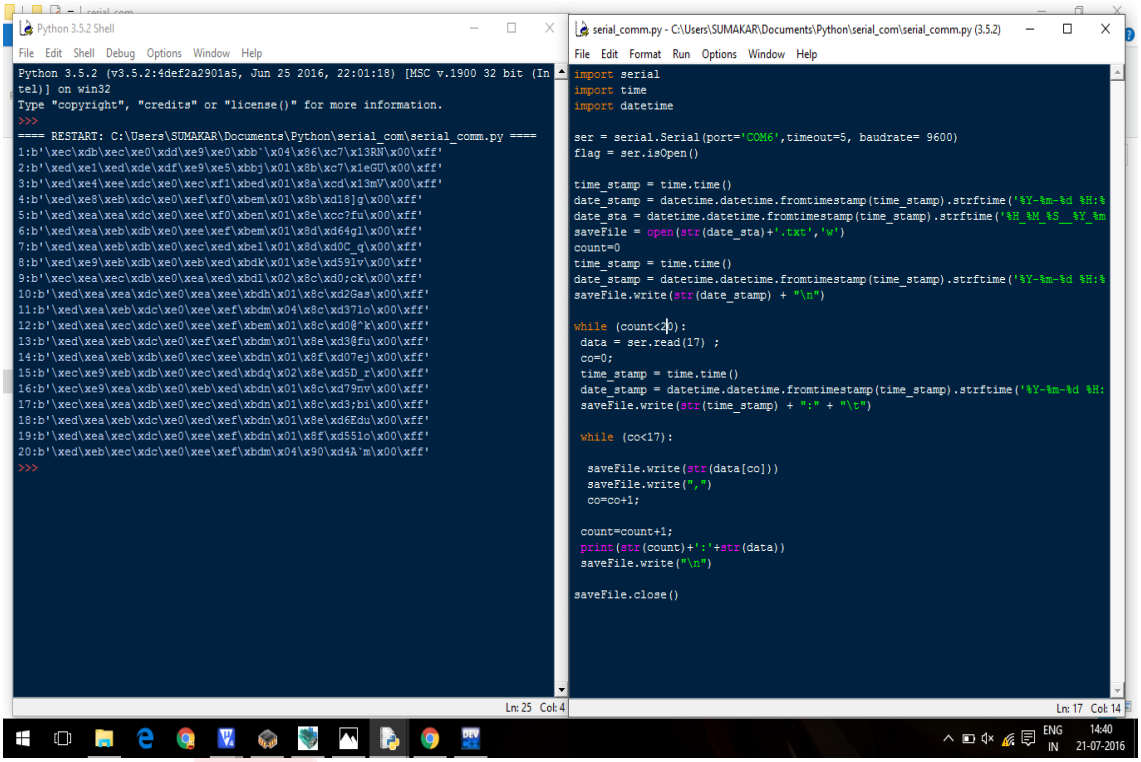

Figure 1.6: State collection

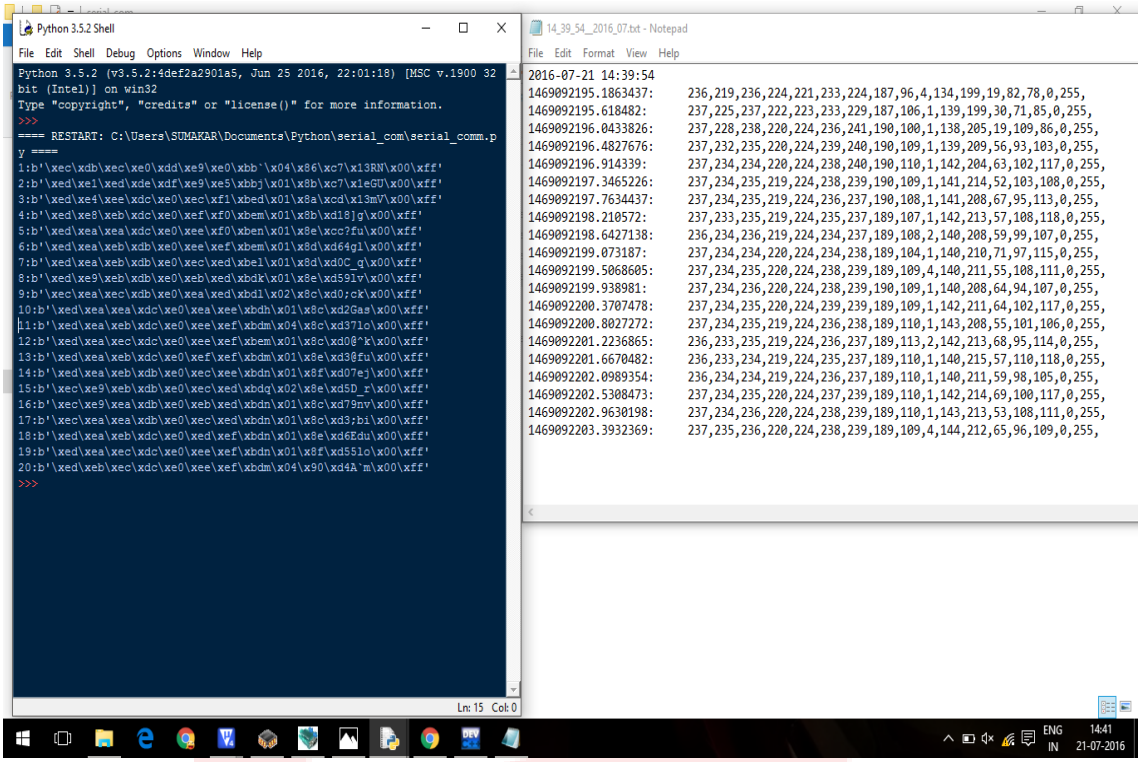

Figure 1.7: State collection

- **Serial to file:**This python script can be used to read the state collection data from a COM port(via serial communication or Xbee) and save the obtained data into a text file(comma seperated values with time stamp).

- **SPI1 :**This code was an improvement of an already existing code.This code prints the values of sensors connected to slave(2 Atmega 8 uC) on an LCD display.

- **Collision Avoidance:**With the help of this code we are trying to illustrate the difference between RTOS and 'normal C'.The difference in the behaviour of the robot can be observed for a similar logic for different implementation.

1.4 Future Work

RTOS is something which is useful only when required therefore we think developing modules for LPC2148 with support to RTOS would be a better direction,here are some of the suggestions/ideas we were trying to work upon.

1. Making FireBird more acessible
    - FireBlocks for LPC2148 and android support for the same.
    - Interfacing FireBird to Smartphone via USB to directly program it by transferring the .bin file.
2. Interfacing Lower resolution camera modules like ov7670 to perform/learn basic level image processing.
3. Improvements in state collection.
    - A SD card module can be added to store state data in the SD card instead of EEPROM or transmitting via Xbee.
    - Interfacing a Wifi module with LPC2148 to log data directly on servers.
    - Encoding the data in the bot itself before transmission.
4. Improvements in Collision avoidance.
    - Using UltraSonic sensors to detect obstacles from far and calculate its course of action/velocity etc.

1.5 Bug report and Challenges

Bug report

- The UART FIFO of LPC2148 is of 16 Bytes ,if more than 16 bytes of data is sent at the same time data loss may occur.

- Data obtained in python script is not validated and is directly stored into the files so loss of data wouldn't be countered.

- In the collision avoidance Experiment if the obstacle is in its blind spot the bot would collide.

- FreeRTOS provides a common header file for all LPC21xx microcontrollers, using it had given us warnings/errors we had replaced the contents of lpc21xx.h file with that of lpc214x.h .

Challenges

- Learning how to port FreeRTOS.
- Getting Sensor data from SPI1.
- Understanding the implementation level difference between Binary semaphore and Mutex.
- IR's connected to Master were giving incorrect values we instead used Sharp sensors.
- Bringing out an example which can illustrate the advantage of using RTOS.

Failures :

- Unable to interface with smart phone.