CMPE-240 Laboratory Exercise 03

Timer

By submitting this report, I attest that its contents are wholly my individual writing about this exercise and that they reflect the submitted code. I further acknowledge that permitted collaboration for this exercise consists only of discussions of concepts with course staff and fellow students; however, other than code provided by the instructor for this exercise, all code was developed by me.

Daniel Santoro
Performed: 10/17/16
Submitted : 10/17/16

Lecture Section: 01
Professor: Alessandro Sarra
TA: Kevin Millar, Adrian Cruzat, Humza Syed

Abstract

The goal of the lab is to produce a timed blinking LED to allow for debugging hardware failures. This will be accomplished using the Raspberry Pi's built-in timer to represent error code blink durations.

Design Methodology

The UART connection to the GPIO pins was set up in the given order. (See below figure.)

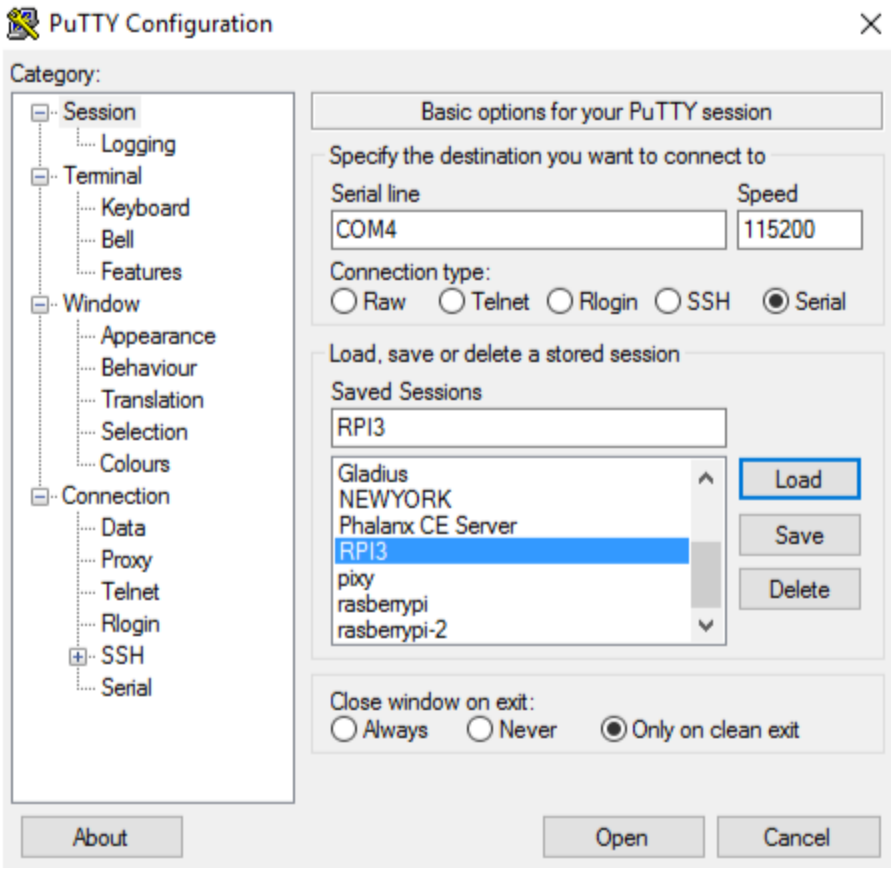
Figure 2.1 – UART Connection to GPIO

In the config.txt file of the Raspberry Pi image, UART needed to be enabled and Bluetooth needed to be disabled. (See below figure).

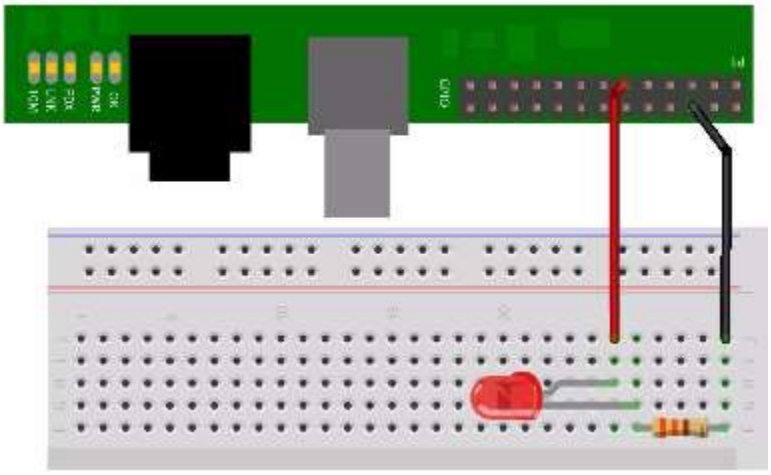
```
# Enable UART
enable_uart=1

# Disable bluetooth (UART and bluetooth share the same resources,
# so both can't be active at the same time)
dtoverlay=pi3-disable-bt
```

Putty was used to connect over serial port via UART to the Pi. (See below figure.) To find the COM number to connect to, device manager was used (on Windows 10) and the COM number was listed under Ports.


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Basic options for your PuTTY session
Specify the destination you want to connect to
Serial line
Speed
COM4
115200
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
Load, save or delete a stored session
Saved Sessions
RPI3
Gladius
NEWYORK
Phalanx CE Server
RPI3
pixy
rasberrypi
rasberrypi-2
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Open
Cancel

A simple circuit for to operate the LED was constructed using the breadboard. The image below shows a working setup. The 16th pin was used as an output controlling pin.



A sleep function was written using the Raspberry Pi's system timer channel; the delay in nano seconds would be inserted into the C0 timer register and the code would spin until the CS register was written to (which signified that the timer expired). Two additional functions which scaled up the timer's function input into milliseconds and seconds respectively were also written.

In the main, a blink_once function was written which would set the registry to turn on a specified port (in this instance pin 16), wait 1 second, and they turn off the port. Another function blink_code would take a number of blinks, and call blink_once function that many times and afterwards wait 5 seconds. The main code emulated various blink codes by running through a while loop which it would look at the current value of a counter, at a count of 10 reset to 0, otherwise send the counter as a parameter for the blink_code function.

Results and Analysis

The program ran and incremented blinking after each 5 second pause. At what would have been 10 blinks, the program restarted at 1 blink.

Conclusions

The lab was successful since the purpose of the lab was completed – using LED blink codes could be visualized (for the potential of hardware debugging).

No problems were encountered when performing the lab.