Using MATLAB and Simulink to Aid in Debugging Raspberry Pi GPIO Programs

Dominic Ottaviano

ECE 3005, Professional and Technical Communication
Section B

Georgia Institute of Technology
School of Electrical and Computer Engineering

Submitted
November 21, 2014

Summary of Document

The purpose of this document is to describe the methods and benefits of using MATLAB and Simulink to analyze the physical inputs and outputs of the Raspberry Pi computer. This document is written at the level of a user who is familiar with basic electrical concepts but no exhaustive circuit's knowledge is required. The reader should also be familiar with Linux and the Raspberry Pi.

In this guide python will be used as the scripting language since it is most commonly used for Raspberry Pi programming. Advanced knowledge of python is not required to understand this guide. A Raspbian Linux image provided by MATLAB for the Raspberry Pi will also be used. Installation and configuration is demoed under MATLAB R2013a but is similar for most versions and will be covered first.

Installation

This section will outline the installation of the support module for MATLAB and the installation of the image for the Raspberry Pi. The support package installation is a straightforward setup and will be outlined in detail below. During the support package installation the image for the Raspberry Pi will be downloaded and copied to an SD card.

To get started the Simulink support package for the Raspberry Pi needs to be installed. It is accessible through the Add-Ons button in the MATLAB toolbar (See Figure 1).

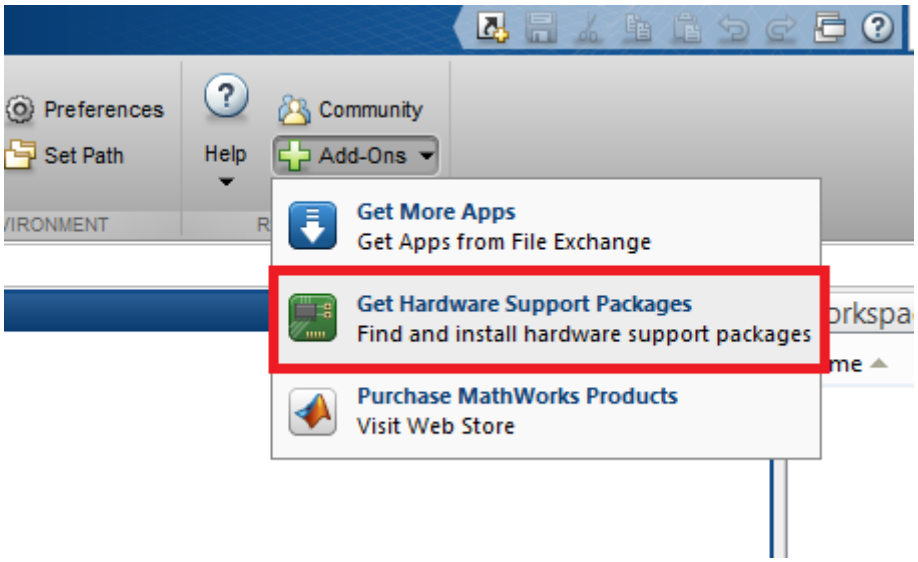

Figure 1: *Location of hardware support package installer.*

After that a window will open asking where the installation package should come from. Select 'Internet'. The next window will show a list of available hardware support packages. Select the one for the Raspberry Pi.

A note for some users: Depending on your configuration and version of Windows, the default installation folder may cause problems with permissions when the Raspberry Pi installation

image is extracted. It is recommended that you change the Installation folder to somewhere inside your user directory (i.e. C:\Users\UserName\MATLAB). After that click ‘next’ to read and accept the MathWorks license agreement and click ‘Continue’.

The next screen will show a list of required software to be installed in addition to the support package. None of this software is optional, so click ‘Next’ then ‘Install’. After some time the window will update to reflect that the installation was successful, and the option to continue to update the firmware will be available.

This next step will require an SD card of at least 4 GB in size. **All of the existing data, if any, on the SD card will be erased!** Choose ‘continue’ and the next window will ask you to choose a hardware to start the firmware update process. Choose ‘Raspberry Pi’ and click next. The next window will inform you that a custom Raspbian Linux image will be downloaded. Click ‘Download’ to continue. The download will take some time depending on the speed of your internet connection.

After the download and decompression completes successfully, the network configuration window will appear. There are two methods for connecting MATLAB to the Raspberry Pi. The first is by connecting via a direct connection to the host computer. This is the method the guide will use since it is the simplest, fastest, and most portable. A connection over a LAN is also possible, and after configuring the network this guide will work for both. Choose your connection type and at this point insert the SD card into the SD card slot if it is not already and click ‘Next’.

The next screen tells MATLAB where to write the image to. Be very careful to select the drive letter of your SD card and not another drive since, again, all of the information will be erased. Once you’re ready click write to write the Raspbian image to the SD card. After the write completes a screen with instructions on what to do next will appear. For the sake of completeness they will be replicated here.

1) Remove the SD card from the computer and insert it into the Raspberry Pi.
2) Connect an Ethernet cable to the Raspberry Pi and to the host computer, or to the local network if you chose that network configuration.
3) Connect a 5 V micro USB power supply to the Raspberry Pi. A minimum of 700 mA is required, but a supply of at least 1 A is recommended to ensure stability for all configurations.
4) One power is connected the Raspberry Pi will begin booting automatically.
5) Connect all needed peripherals to the Raspberry Pi (Keyboard, Mouse, monitor, ect.).
6) Ensure that the Raspberry Pi has booted when the link LED begins flashing.
7) Click ‘next’.

MATLAB will then attempt to connect to the Raspberry Pi. If the connection is unsuccessful, ensure there are no static network settings set on your Ethernet network interface card (NIC). This can prevent the Raspberry Pi from being placed on the correct subnet and communication will fail. You can check the network settings in the Windows Network and Sharing Center. Once it connects, a window will appear displaying the IP address and default username and password of the Raspberry Pi. Keep this information, test the connection if desired, and click

'Next'. A useful piece of information to take away from this screen is that the Raspberry Pi will "speak" its IP address upon boot.

The completion screen will now display. Click 'Finish'. This completes the installation portion of this guide.

Configuration

This section will overview the setup of the Raspberry Pi. The Linux environment will be prepared, then the network will be configured to allow internet access to the Raspberry Pi for updates and optional software packages to be installed.

Configuration of the Raspberry Pi is possible using two different methods. The first is to use a USB keyboard, mouse, and display to the Raspberry Pi. The second is to connect to it over SSH. The default MATLAB Raspbian image has an SSH server running by default, so connection is straightforward. This guide will do all interactions with the Raspberry Pi over SSH. Both methods are equally powerful however using SSH does not require a separate keyboard, mouse, and monitor to interact with the Raspberry Pi. Using either method execute the command 'sudo raspi-config' to launch the Raspberry Pi Software Configuration Tool (See Figure 2).

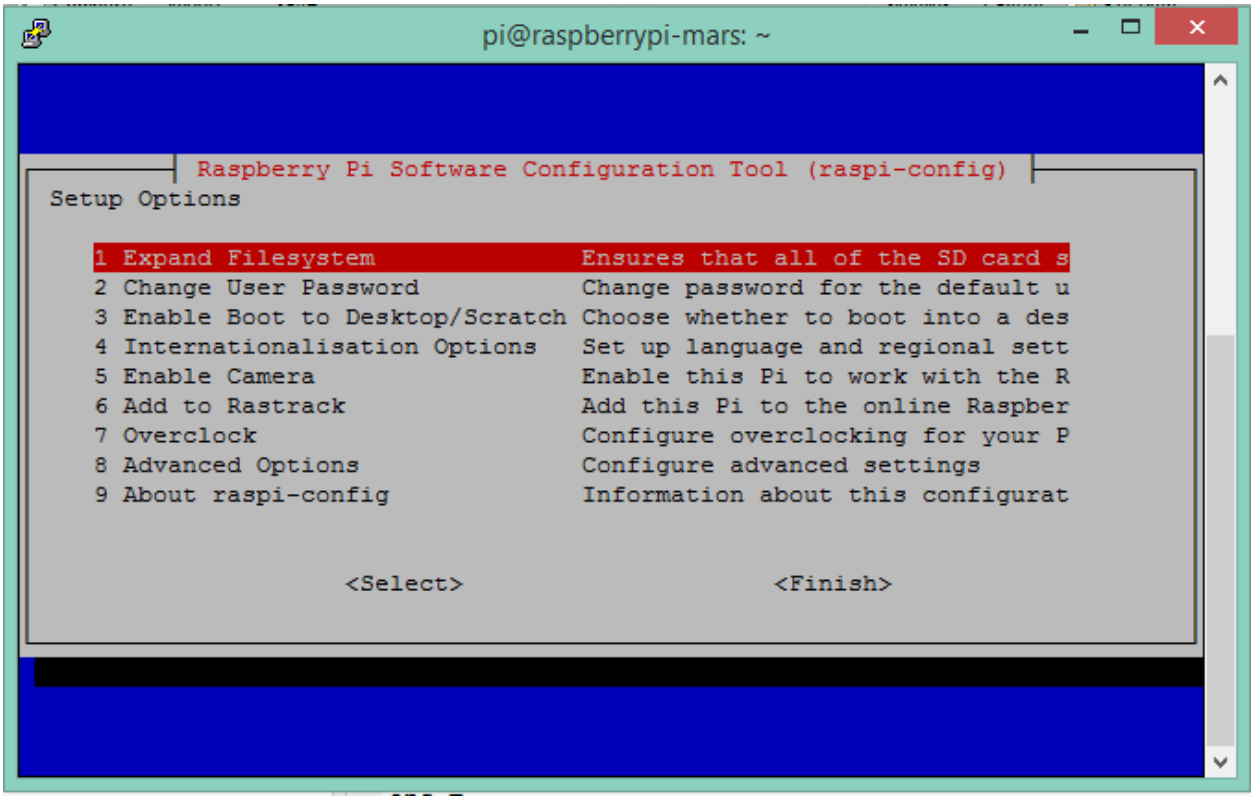

Figure 2: *raspi-config utility.*

The most important option to execute here is the "Expand Filesystem" option. This makes space for updates and additional desired software to be installed. Execute this option and exit the tool. Reboot when prompted.

If the direct connection option was chosen on the network configuration page and you want to provide an internet connection to the raspberry pi, then internet connection sharing will need to be setup on the host computer. The host computer must also have two NICs. One is for the internet connection and one is for the connection to the Raspberry Pi. On Windows operating systems this can sometimes cause problems connecting to the Raspberry Pi after enabling internet connection sharing. To remedy this the network settings on the Raspberry Pi need to be changed to DHCP instead of a static IP address. The configuration file for this is located at /etc/network/interfaces (See figure 3). This is a detail of the Windows ICS service. Then the new dynamic IP address needs to be determined which can be done by plugging speakers into the Raspberry Pi and rebooting it. It will then read off its new IP address during boot up. It can also be done by attaching a monitor and checking manually.

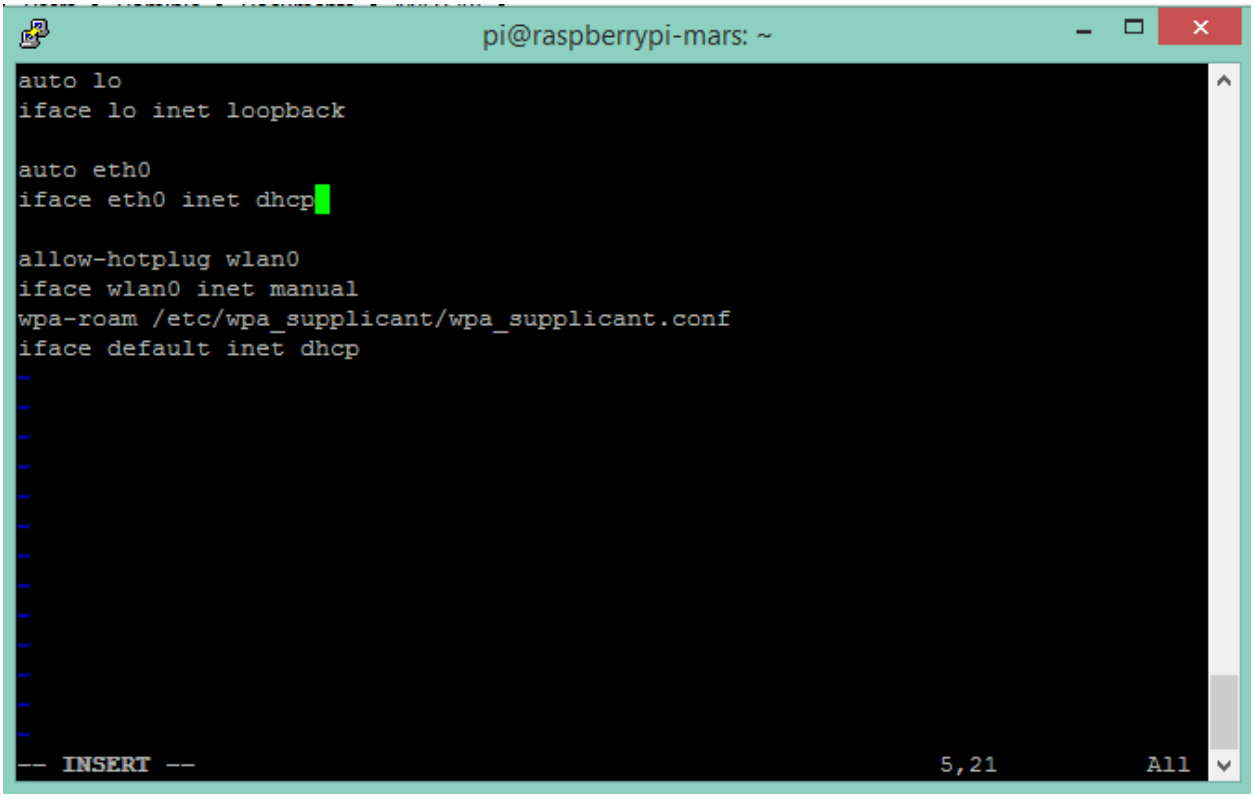

Figure 3: *Editing the /etc/network/interfaces file enable DHCP.*

After an internet connection has been established, run the update command “sudo apt-get update” followed by “sudo apt-get dist-upgrade” to get all the new packages and ensure the Raspberry Pi is up to date. Other desired packages may be installed at this time such as the text editor of your choice. This concludes the configuration portion of this guide.

Basic Python and the GPIO Pins

This section will cover general purpose input/output (GPIO) pin assignments and just enough python to explain the examples.

GPIO Pinout

The Raspberry Pi Model B has 26 pins. 16 of which are GPIO pins. See Figures 4 and 5 for pin out numbering and details.

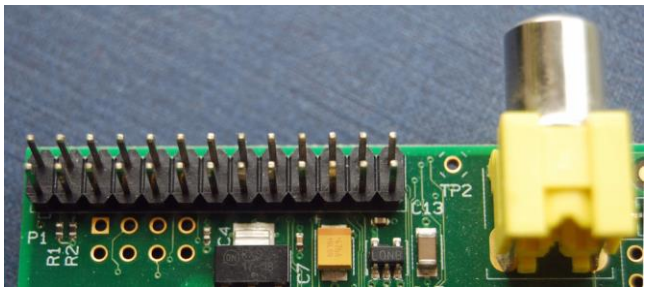

Figure 4: *Raspberry Pi Pins.(Image from [1])*

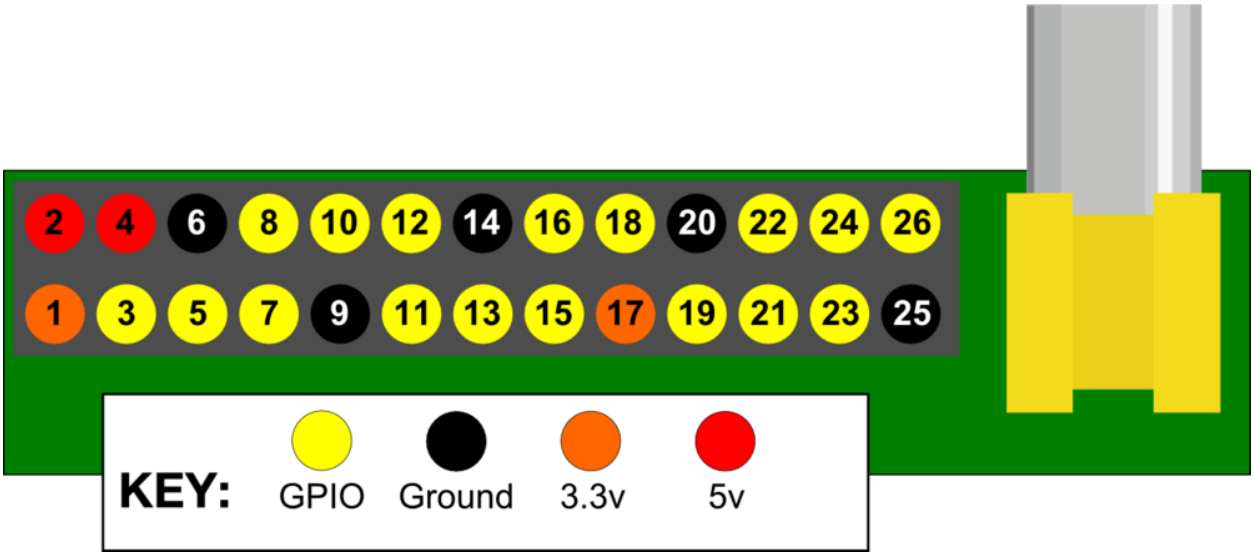

Figure 5: *Raspberry Pi GPIO pin assignments. (Image from [1])*

Building the Demo Circuit

If you don't have a circuit of your own to analyze, this is the circuit used in the examples. To build the circuit used in these examples, you will need:

1) An LED
2) A resister of about 220 Ω
3) A pushbutton
4) Appropriate jumper wires

Connect positive terminal of the LED to pin 23 and the negative terminal to the resistor then to the ground pin 25 on the Raspberry Pi. Connect the button's terminals to ground and pin 21 directly (See Figure 6). Note that you can manually pull up or down the buttons with an appropriate resistor, however the Raspberry Pi has internal resistors so the push buttons can be pulled up or down in software.

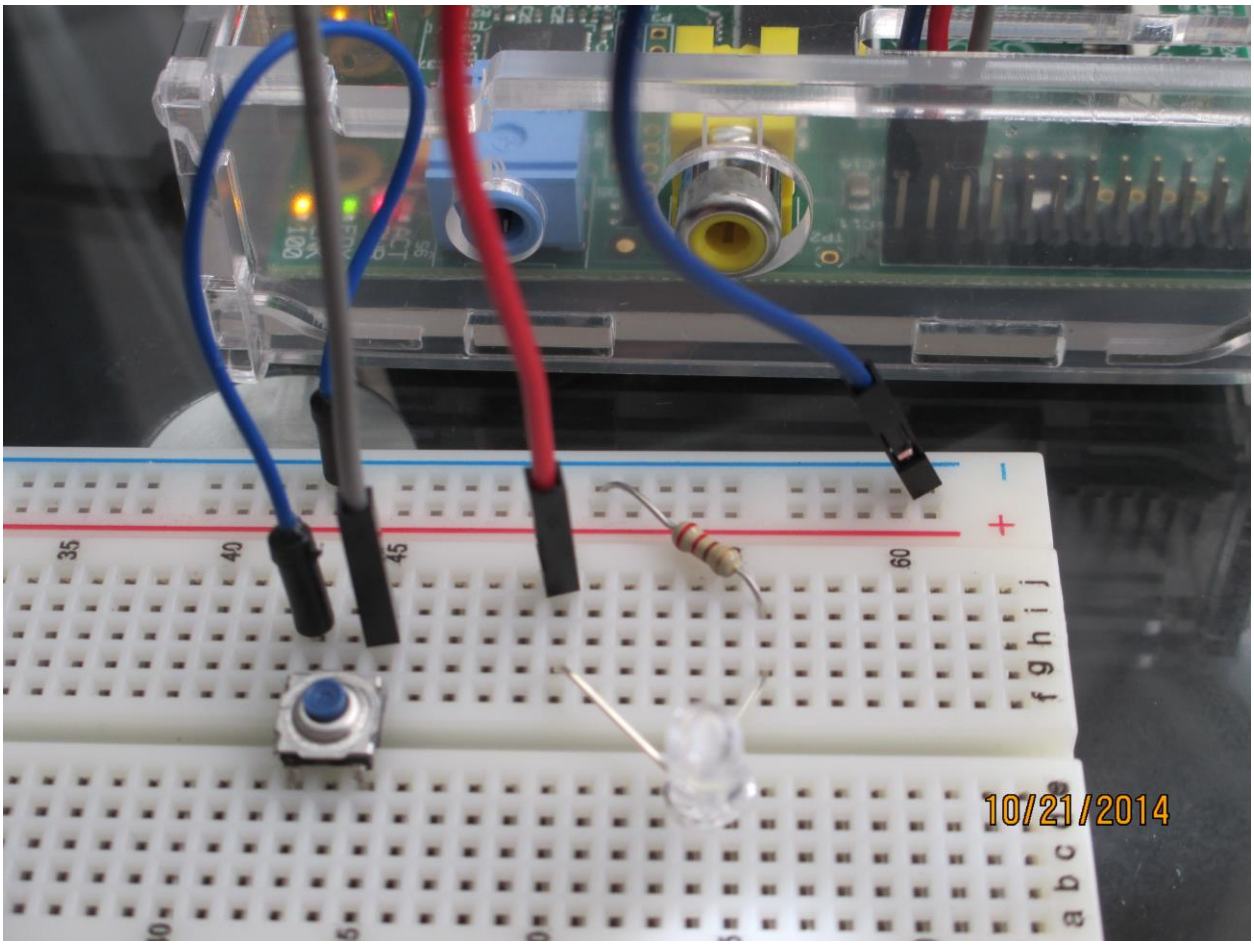

Figure 6: *Image of the complete demo circuit.*

Running Python Programs to Control GPIO

This is a very quick introduction to running Python programs that control GPIO pins on the Raspberry Pi. Python syntax and other details are outside the scope of this document. However a list of quality guides is provided by the python.org wiki page. (https://wiki.python.org/moin/BeginnersGuide)

Create a document in the main user directory called ‘gpio.py’. In that file a few commands must exist for everything to work properly. At the top of the file insert:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
```

These python commands get everything ready for GPIO based programming on the Raspberry Pi. The GPIO library is included by default in the MATLAB image for the Raspberry Pi so no previous configuration is required. Next setup the pin assignments. If the demo circuit is being used, the code below will work.

```
led = 23
GPIO.setup(led, GPIO.OUT)

button = 21
GPIO.setup(button, GPIO.IN, pull_up_down=PUD_UP)
```

These python commands determine whether the pin is considered an input or an output. The pull_up_down=PUD_UP part sets our button to be pull up to a 'high' logic level until pressed. The next section of code is just a simple loop that blinks the LED until the button is pressed.

```
While GPIO.input(button):
     GPIO.output(led,1)
     time.sleep(0.1)
     GPIO.output(led,0)
     time.sleep(0.1)
```

The next and final line of code in the example simply removes all pin assignments set in the program.

```
GPIO.cleanup()
```

For more information on the RPi.GPIO library refer to the library website (https://pypi.python.org/pypi/RPi.GPIO). To compile and run the python script 'gpio.py', run the following command from the terminal:

sudo python3 gpio.py

The LED should blink until the button is pressed. See Figure 7 for the complete code example if it is not working correctly.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)

led = 23
GPIO.setup(led, GPIO.OUT)

button = 21
GPIO.setup(button, GPIO.IN, pull_up_down=GPIO.PUD_UP)

while GPIO.input(button):
        GPIO.output(led,1)
        time.sleep(0.1)
        GPIO.output(led,0)
        time.sleep(0.1)

GPIO.cleanup()
```

Figure 7: *Complete code example in the vim editor.*

Using MATLAB and Simulink

This section is about the techniques of using MATLAB and Simulink to analyze the GPIO pins on the Raspberry Pi. Once everything above is working or you have your own design to analyze start up MATLAB and create a new Simulink model by clicking ‘New’ then ‘Simulink Model’ on the dropdown menu (See Figure 8).

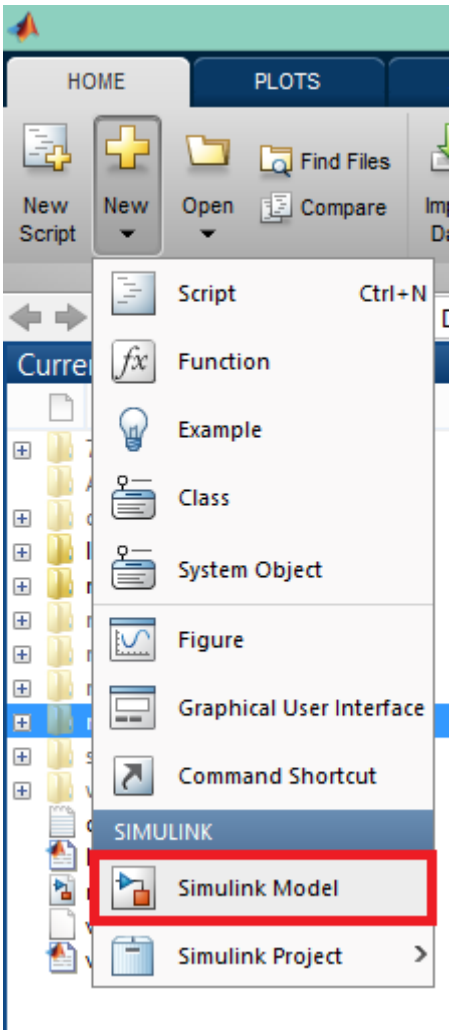

Figure 8: *Location of Simulink Model Button*

Then in the resulting window’s toolbar, click ‘Tools’, then ‘Run on Target Hardware’, then ‘Prepare to Run…’. In the next window ensure the ‘Run on Target Hardware’ tab is selected and set ‘Target Hardware:’ to Raspberry Pi. Check the checkbox for ‘Enable External Mode’ and leave the default TCP/IP port. Set the host name to the IP address of the Raspberry Pi, and leave everything else default unless the default user has been changed (See Figure 9). Click ‘Okay’ to close the window and save the changes.

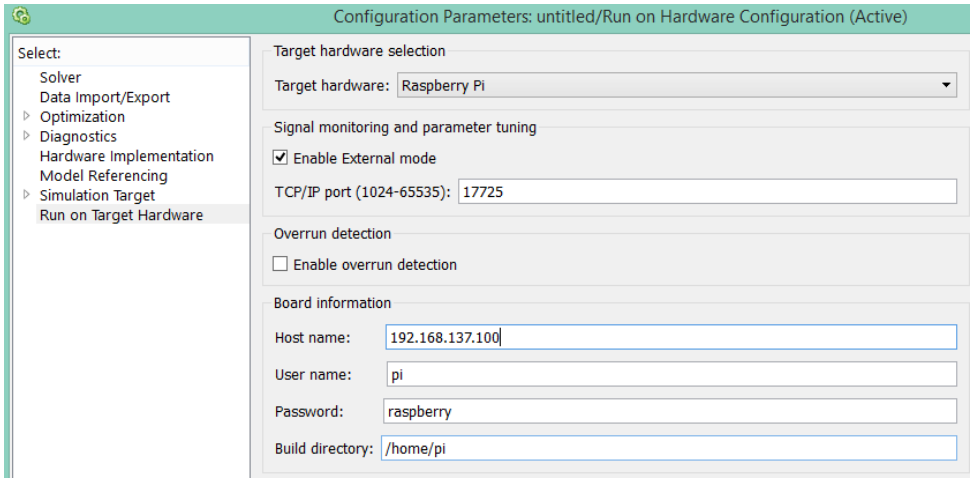

Figure 9: *Configuration page.*

Analyzing GPIO Inputs and Outputs

Save the model. Now click the tools option in the menu and select 'Library Browser'. In the left pane select the 'Simulink Support Package for Raspberry Pi Hardware' tab. Right click on the GPIO Read block and click 'Add to modelname' where modelname is the name you saved the model as. Do this twice to add two of these blocks to the model. This block allows the current state of an individual GPIO pin to be read into MATLAB. If you are analyzing a custom design, add as many of these blocks as pins you want to analyze.

Now in the left pane select and expand the Simulink tab and select 'Commonly Used Blocks' and add two of the 'Scope' blocks. The Scope block serves as a basic time plotted visualization of an input signal. Close the Library Browser and return to the model window. The blocks will be stacked on top of each other, so separate them out as in Figure 10.

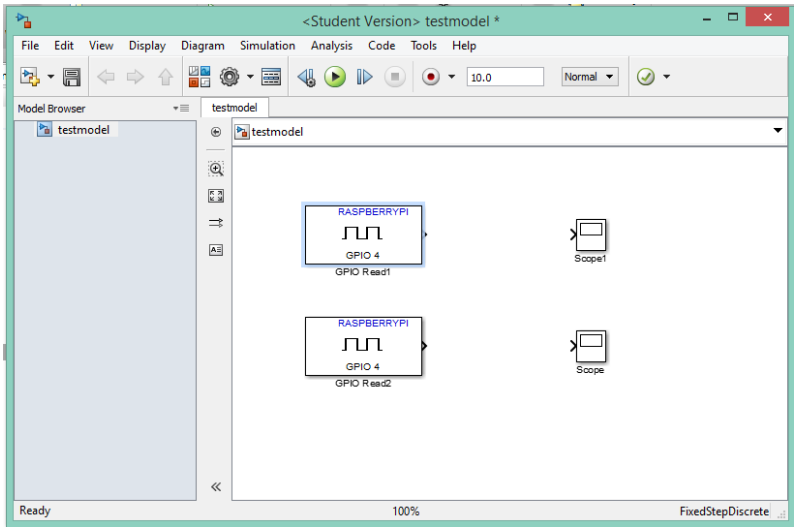

Figure 10: *Model editor window.*

Double click on the first GPIO Read block. Note that the pin numbering scheme MATLAB uses is not the same as the pin order on the Raspberry Pi. MATLAB uses pin numbers as assigned by the Broadcom CPU itself. See Figure 10 for Broadcom pin assignments. For this example Raspberry Pi pins 24 and 23 are going to be referenced in MATLAB so the GPIO Read blocks should be set to pins 11 and 9. Set the sample time to 0.01.

Figure 10: *Broadcom GPIO pin assignments. (Image from [2])*

Now connect each GPIO Read block to a scope by clicking on the small triangle on the side of the GPIO block and dragging to the small port on the side of the Scope block. If you have many of these for a custom design, then renaming the scope blocks to represent which pin they are showing will be helpful. You can rename a block by clicking on the small name below it.

The last step is to change the duration of the simulation from 10 seconds to ‘inf’. This will allow it to continuously run. Enter ‘inf’ in the text box currently containing ‘10’ below the menu bar. See Figure 11 for a complete test model.

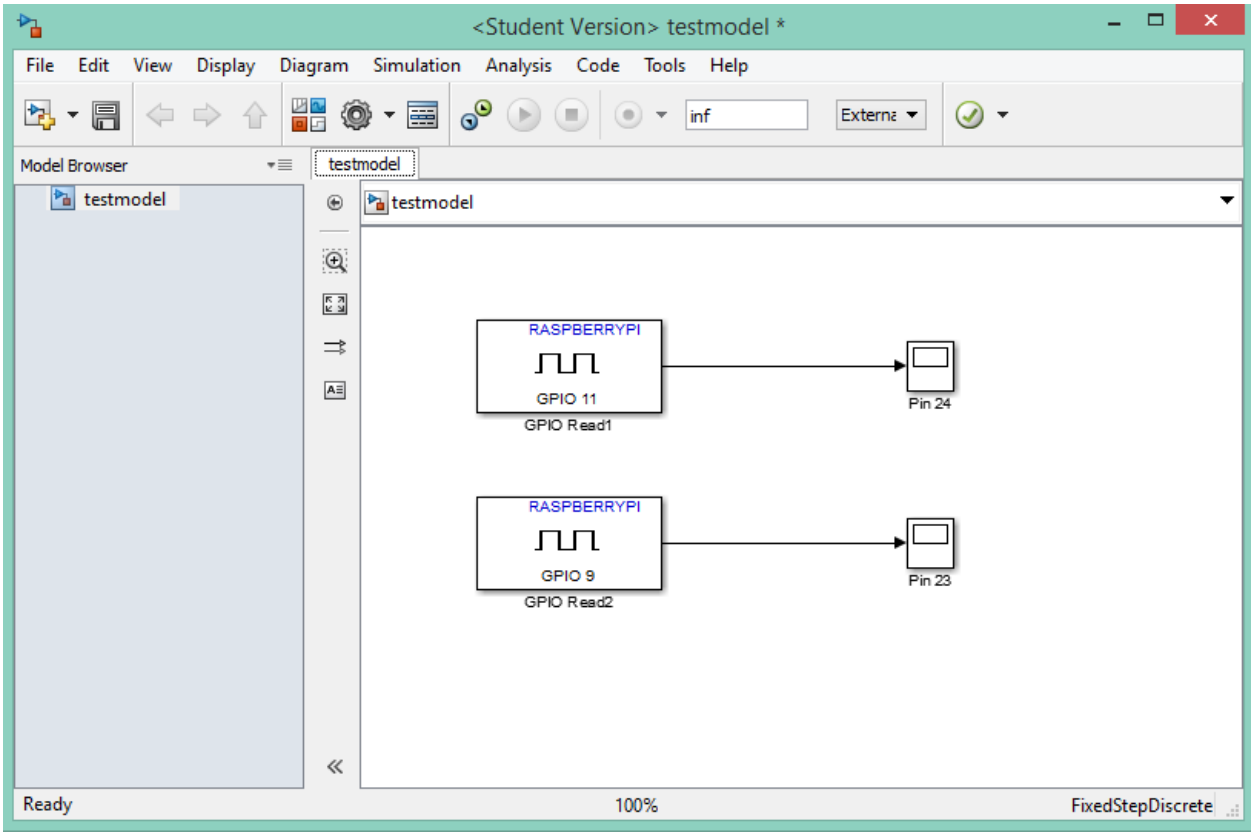

Figure 11: *Completed Simulink model with connected blocks.*

Save the model. Now click ‘Tools’ on the menu bar and select ‘Run on Target Hardware’ and then click ‘Run’. The model will now build and load itself onto the Raspberry Pi. When the model is running a command prompt window will appear, you may minimize this, and in the bottom of the model window a timer will be counting the current run time.

Now double click on each of the Scope blocks to open the scope viewer. The current data populating the scopes will likely be showing the pins in an indeterminate state (See Figure 12). This can be caused by the pins not being properly initialized or a button that has not been pulled down or up correctly.

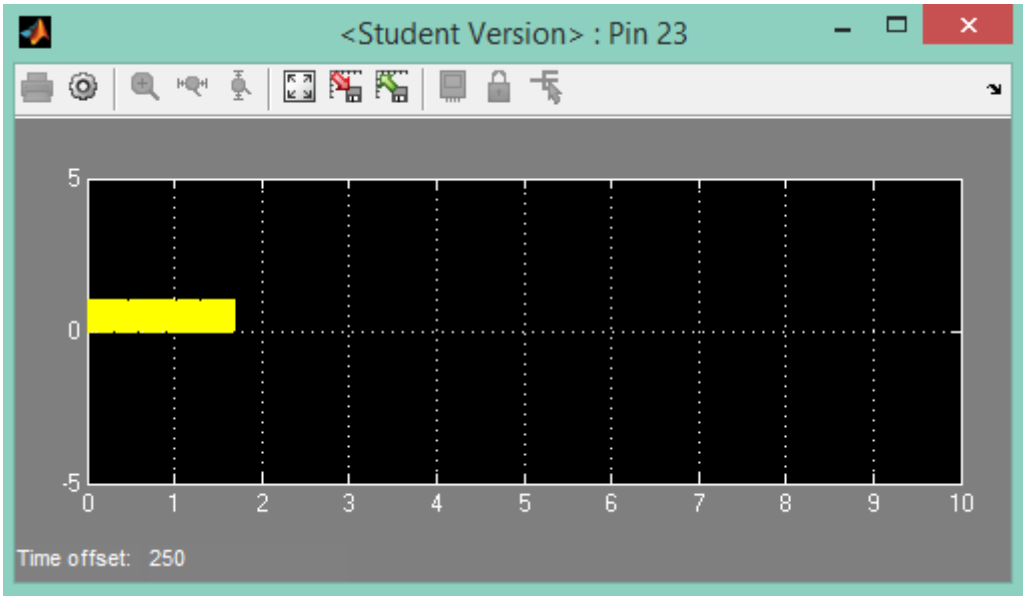

Figure 12: *Indeterminate scope results.*

Run the python script to begin getting meaningful data to populate the scopes. The scopes should now update to begin showing GPIO pin's states in real time as the program executes. A sample of the demo program running is shown in Figure 13.

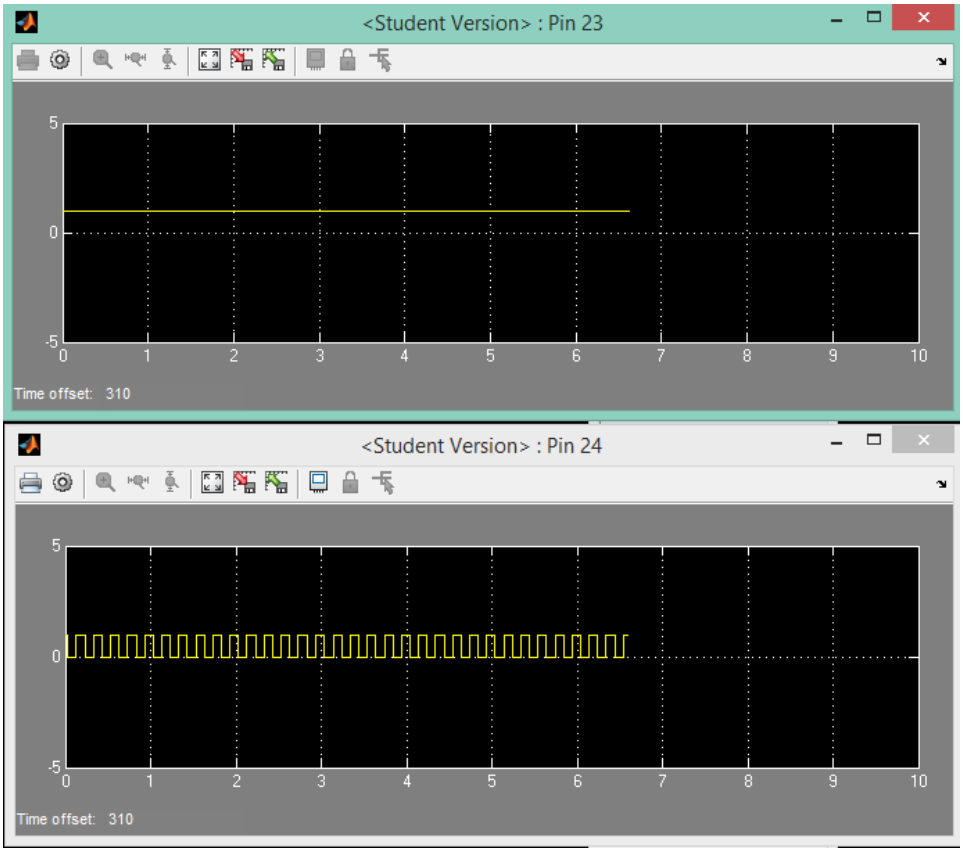

Figure 13: *Sample of demo circuit and program executing.*

In the second scope in Figure 13, the state of the output pin connected to the LED is observed oscillating. This corresponds to the LED flashing in circuit. The first scope shows the state of the button. It is constantly high. This is expected since the button is pulled high. If the button is pressed, the signal in the first scope should go low and in the demo the program will end. The pins should then return to an indeterminate state as the pin assignments are flushed with the GPIO.cleanup() command at the end of the python file. As is seen in Figure 14, this is exactly what happens.

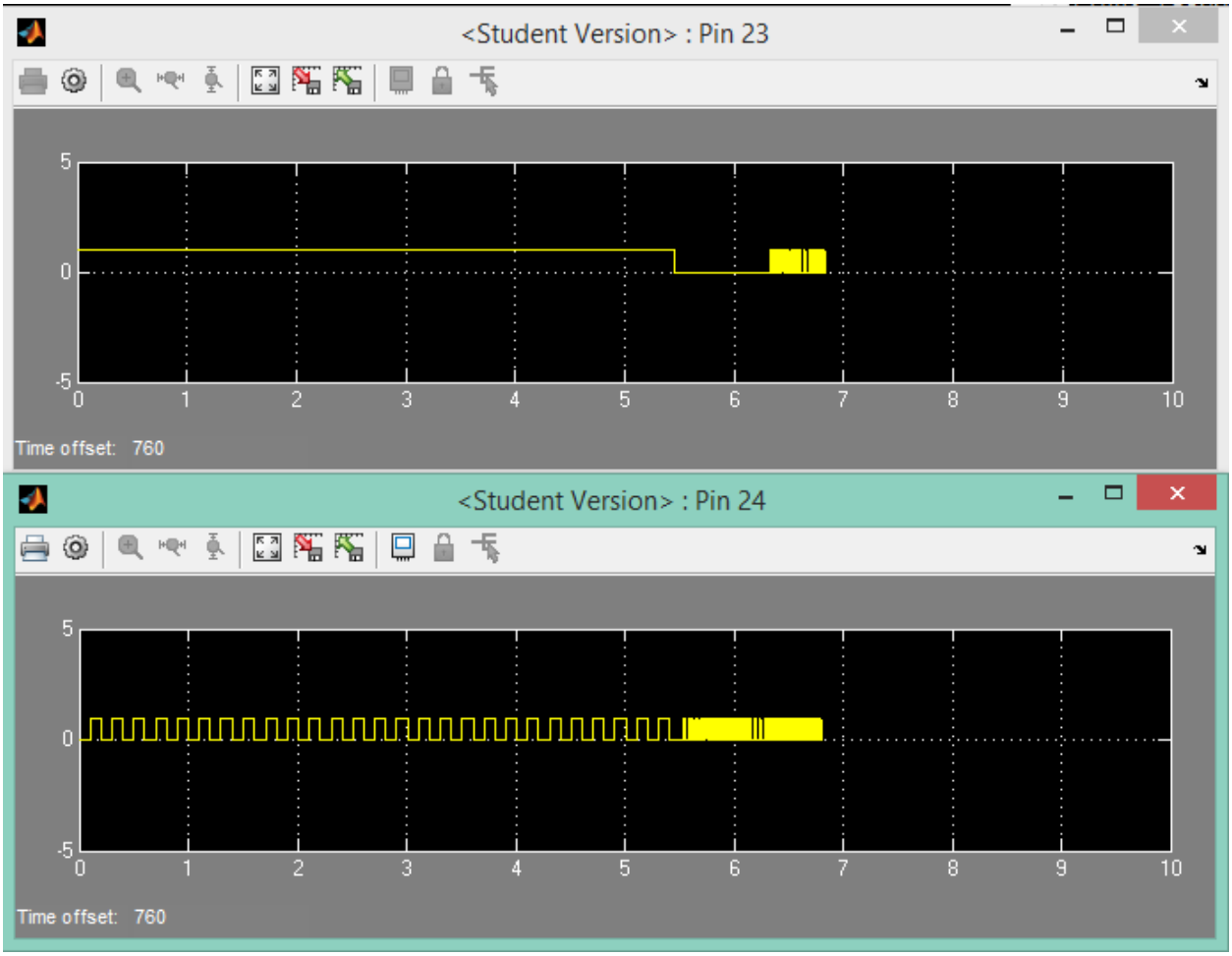

Figure 14: *The demo program running then ending when the pushbutton is pressed.*

This simplistic example illustrates a basic principle of using MATLAB and Simulink to debug Raspberry Pi GPIO programs.

Conclusion

This is by far not the only use of MATLAB to aid in Raspberry Pi programming, however the more advanced topics are outside of the scope of a guide written for programmers beginning with MATLAB and Simulink.

Citations

[1] http://www.raspberrypi.org/documentation/usage/gpio/

[2] http://elinux.org/RPi_Low-level_peripherals