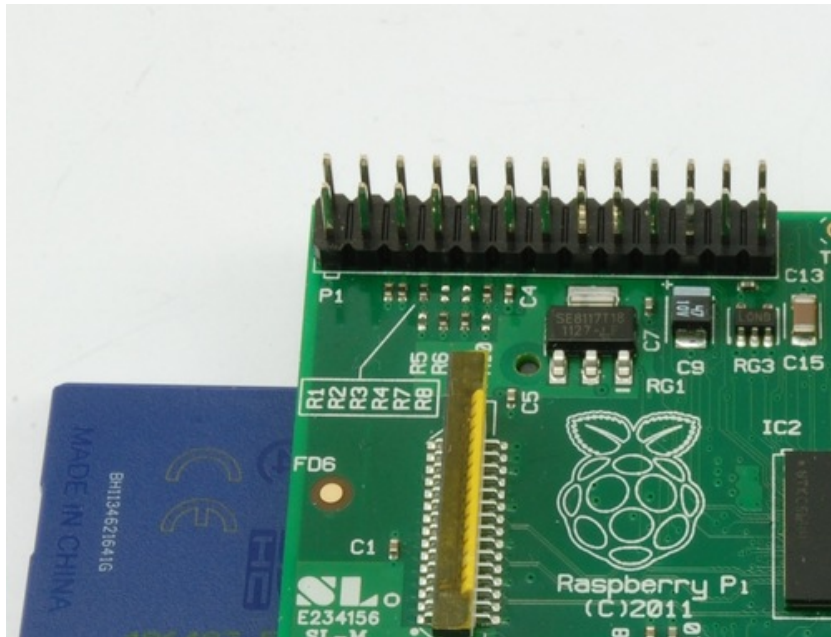


Adafruit's Raspberry Pi Lesson 4. GPIO Setup

Created by Simon Monk



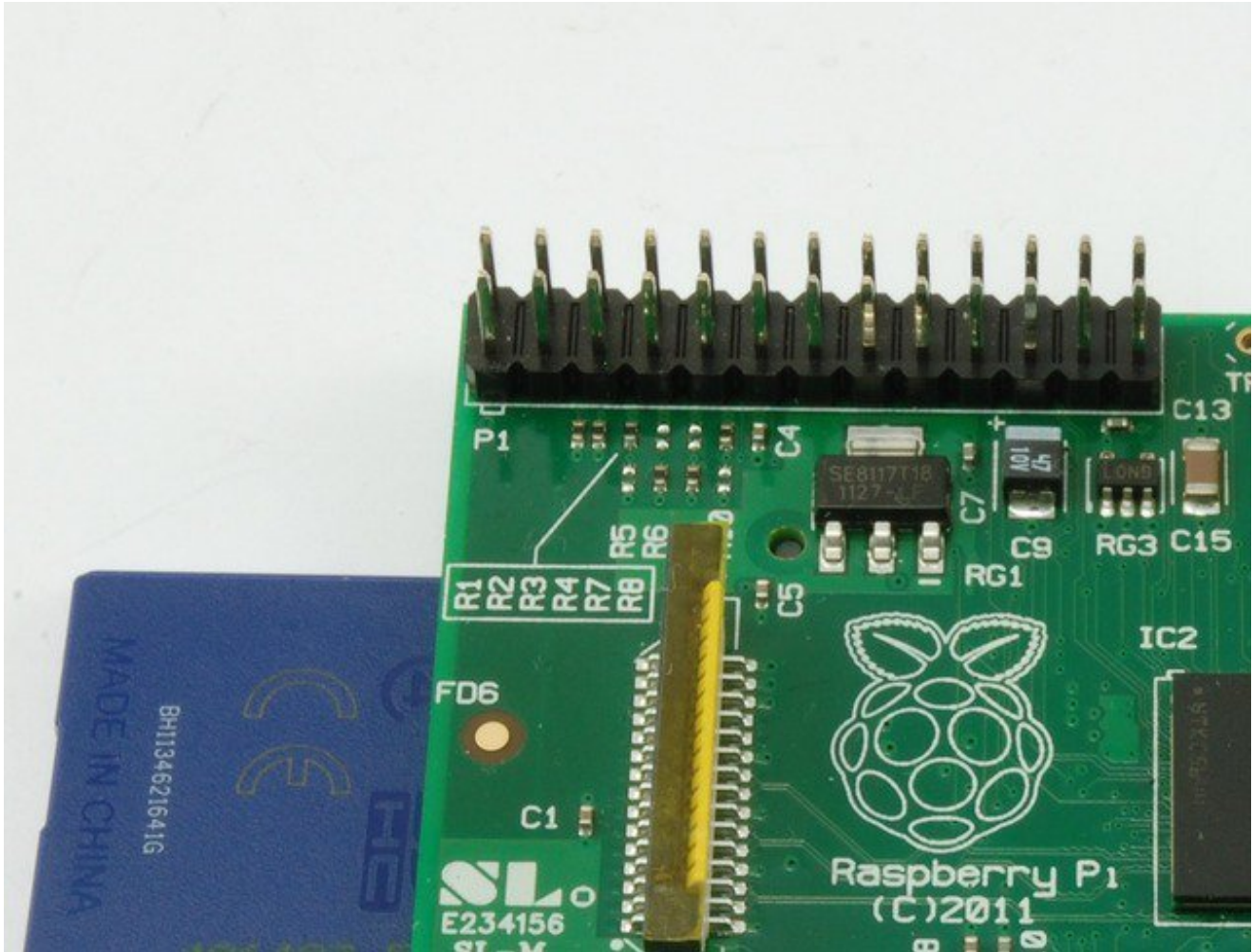
Last updated on 2017-08-15 02:53:17 AM UTC

Guide Contents

Guide Contents	2
Overview	3
The GPIO Connector	5
Adafruit Pi Code	7
Configuring GPIO	11
Configuring I2C	13
Installing Kernel Support (with Raspi-Config)	13
Installing Kernel Support (Manually)	17
Testing I2C	19
Configuring SPI	21
Test and Configure	23

Overview

One of the great things about the Raspberry Pi is that it has a GPIO connector to which you can attach external hardware.



The GPIO connector actually has a number of different types of connection on them. There are:

- True GPIO (General Purpose Input Output) pins that you can use to turn LEDs on and off etc.
- I2C interface pins that allow you to connect hardware modules with just two control pins
- SPI interface with SPI devices, a similar concept to I2C but a different standard

- Serial Rx and Tx pins for communication with serial peripherals

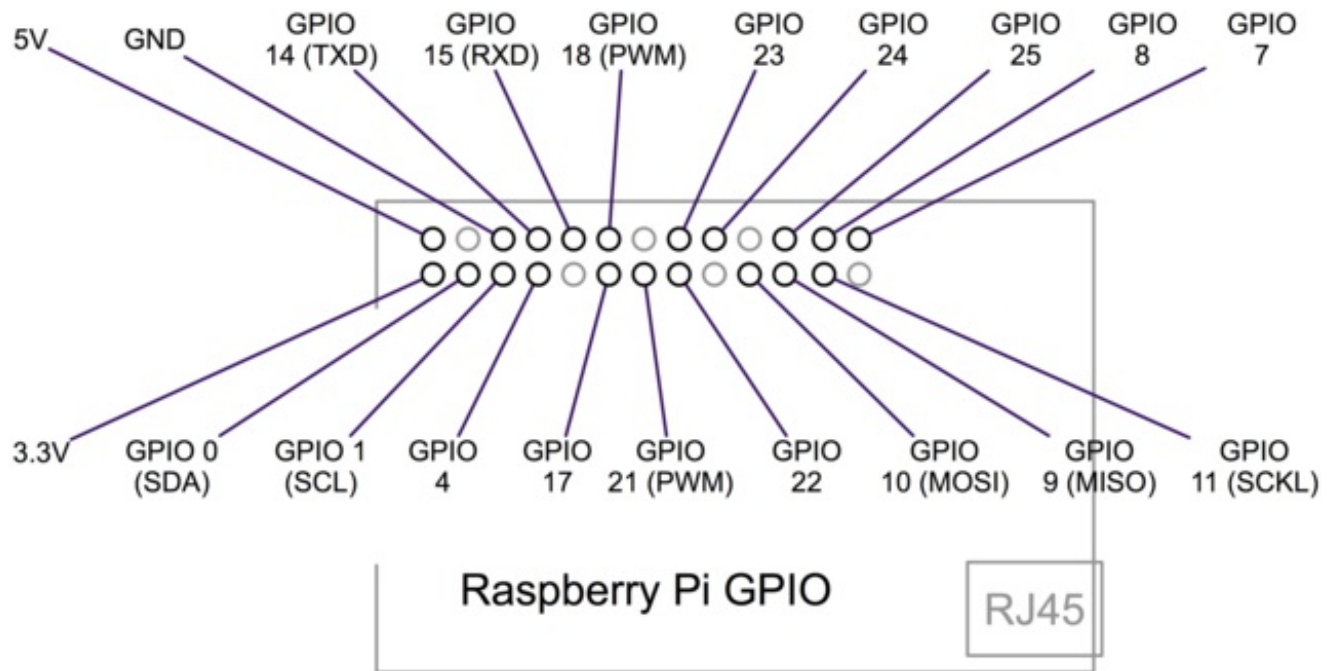
In addition, some of the pins can be used for PWM (pulse Width Modulation) for power control and another type of pulse generation for controlling servo motors called PPM (Pulse Position Modulation).

In this tutorial, you are not actually build anything, but you will learn how to configure your Raspberry Pi and install useful libraries ready to start attaching some external electronics to it.

This tutorial is written for Raspbian & Raspbian-derived installations (like Occidentalis) only

The GPIO Connector

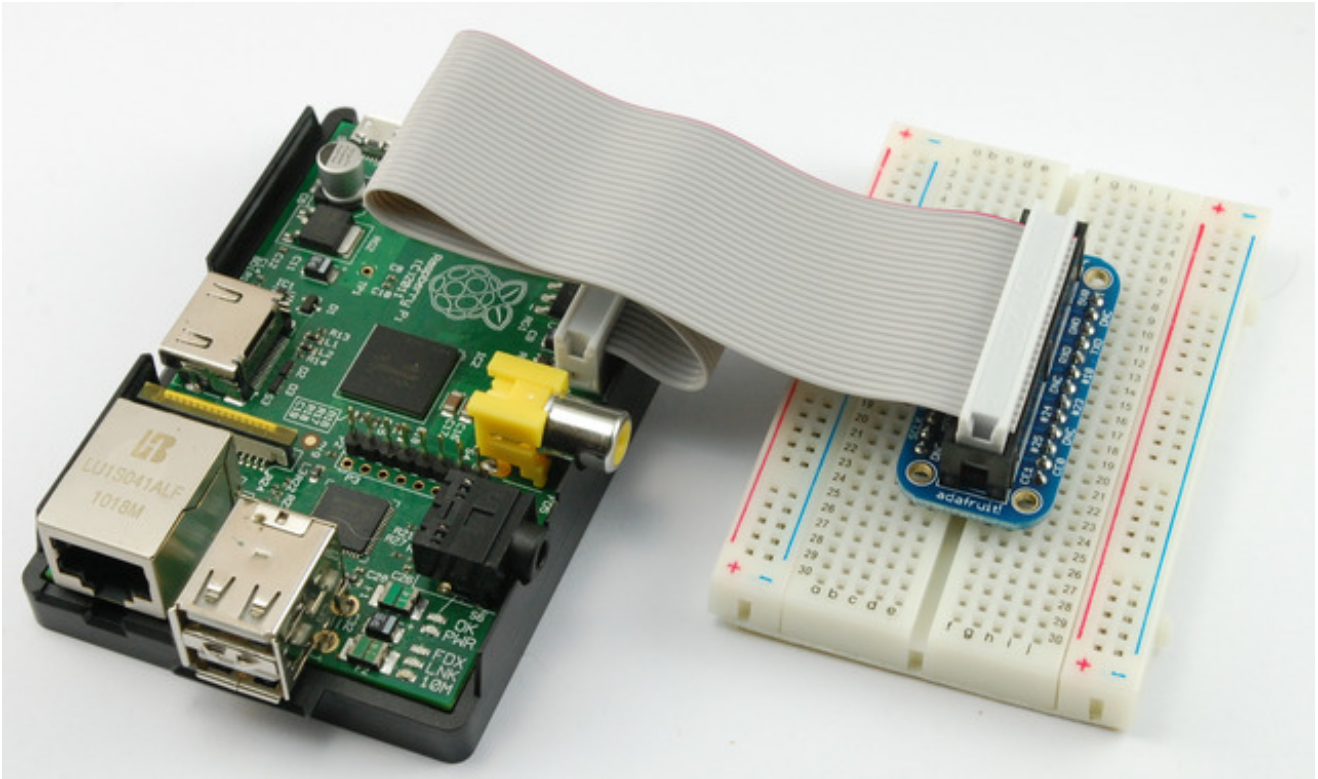
The diagram below show the pins on the GPIO connector for a **Raspberry Pi Version 1** (which is what existed when this tutorial was released) **Version 2** has pin 27 replacing pin 21 but it otherwise the same



As well as supplying power (GND, 3.3V and 5V) all the GPIO pins can be used as either digital inputs or outputs. The pins labelled SCL and SDA can be used for I2C. The pins labelled MOSI, MISO and SCKL can be used to connect to high speed SPI devices.

All the pins have 3.3V logic levels and are not 5V-safe so the output levels are 0-3.3V and the inputs should not be higher than 3.3V. If you want to connect a 5V output to a Pi input, [use a level shifter \(http://adafru.it/aM5\)](http://adafru.it/aM5)

A popular way to actually make the connections to the Raspberry Pi is to use a Pi Cobbler.



Make extra extra double-check sure that the PIN 1 indicator is in the corner of the Pi. If you have a gray cable its probably a red stripe, for black cables, a white stripe. That pin must not be next to the TV connector. Turn around or twist the cable until it is right

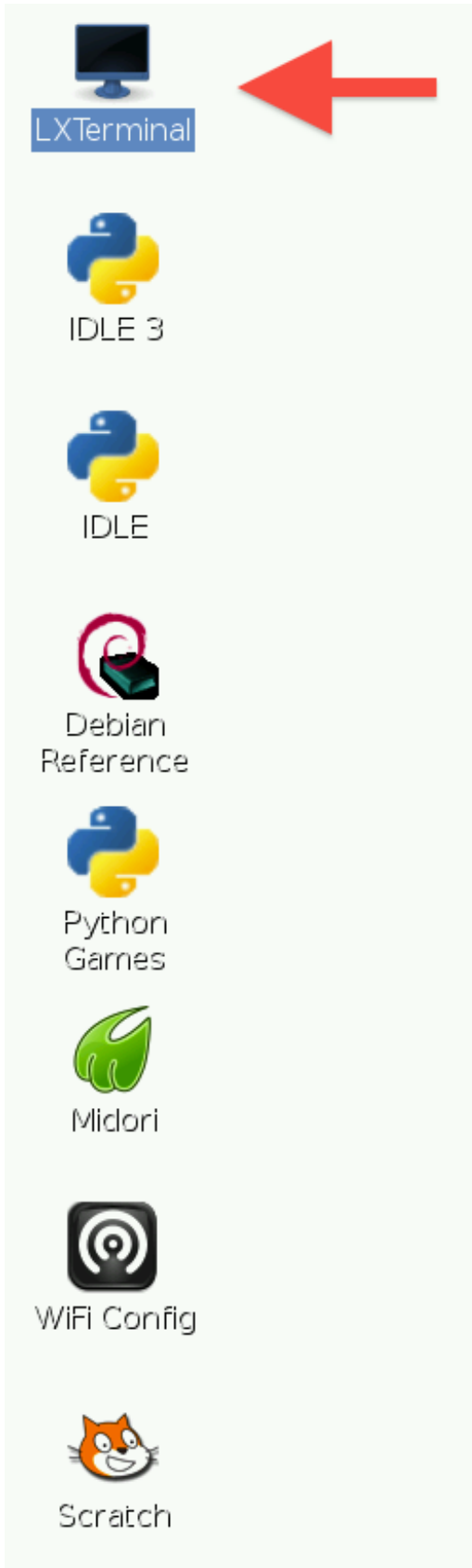
This uses a ribbon cable to connect the GPIO connector to solderless breadboard, where you can add your own components.

Adafruit Pi Code

To make life easy for those wishing to experiment with attaching electronics to their Pi, Adafruit have produced an extensive and extremely useful collection of code. This includes simple Python libraries for a large number of modules, including displays, sensors and PWM controllers etc.

To fetch this code, you need to use some software called 'git'. This comes pre-installed on Occidentalis, but on Raspbian you must install it by entering the following commands into LX Terminal.

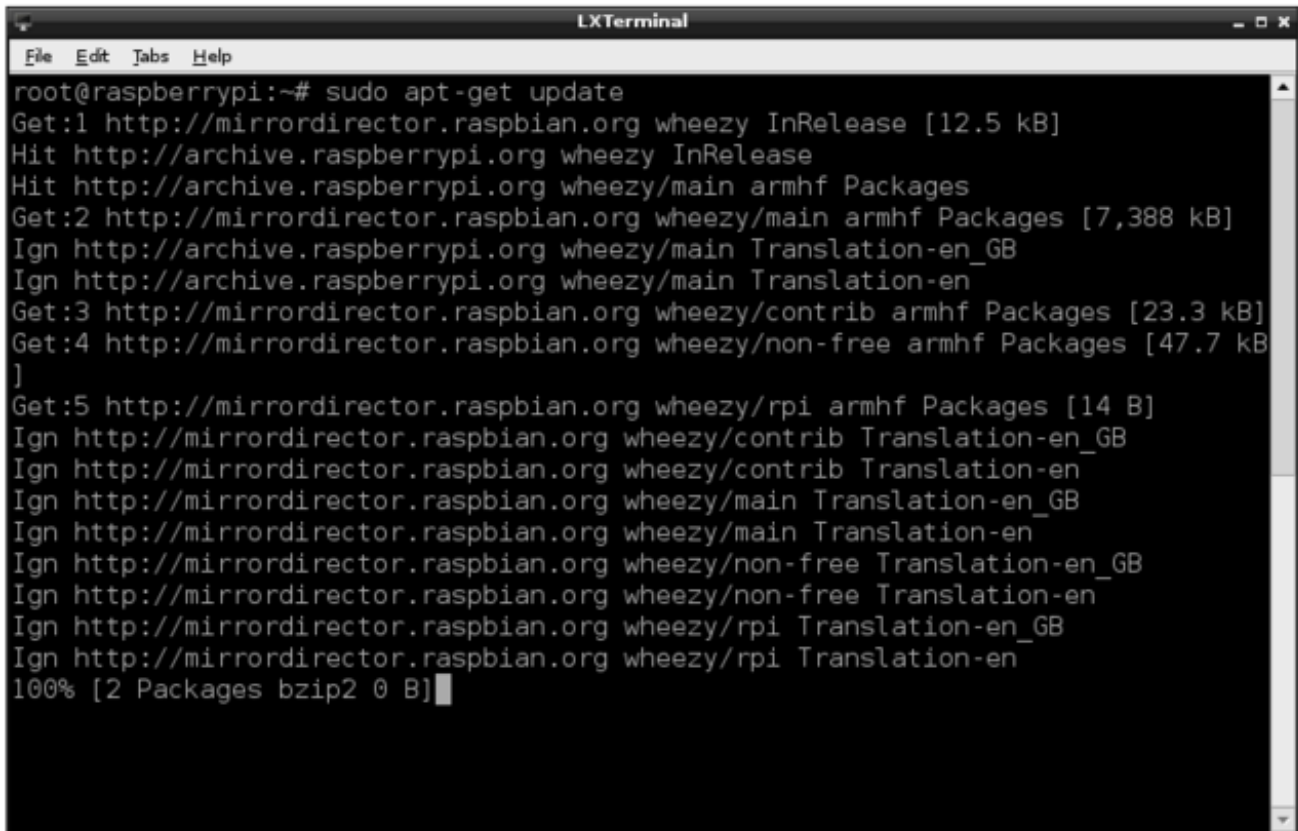
You will find the icon for LX Terminal on your desktop.



Before we go any further, issue the following command in LXTerminal. This will ensure

your package can be found and that you get the latest version. It does not matter which directory you are in.

```
sudo apt-get update
```



```
root@raspberrypi:~# sudo apt-get update
Get:1 http://mirrordirector.raspbian.org wheezy InRelease [12.5 kB]
Hit http://archive.raspberrypi.org wheezy InRelease
Hit http://archive.raspberrypi.org wheezy/main armhf Packages
Get:2 http://mirrordirector.raspbian.org wheezy/main armhf Packages [7,388 kB]
Ign http://archive.raspberrypi.org wheezy/main Translation-en_GB
Ign http://archive.raspberrypi.org wheezy/main Translation-en
Get:3 http://mirrordirector.raspbian.org wheezy/contrib armhf Packages [23.3 kB]
Get:4 http://mirrordirector.raspbian.org wheezy/non-free armhf Packages [47.7 kB]
]
Get:5 http://mirrordirector.raspbian.org wheezy/rpi armhf Packages [14 B]
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-en_GB
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-en
Ign http://mirrordirector.raspbian.org wheezy/main Translation-en_GB
Ign http://mirrordirector.raspbian.org wheezy/main Translation-en
Ign http://mirrordirector.raspbian.org wheezy/non-free Translation-en_GB
Ign http://mirrordirector.raspbian.org wheezy/non-free Translation-en
Ign http://mirrordirector.raspbian.org wheezy/rpi Translation-en_GB
Ign http://mirrordirector.raspbian.org wheezy/rpi Translation-en
100% [2 Packages bzip2 0 B]
```

The update may take a while, especially if this is the first time you have run it on your Pi. Eventually it should give you another command prompt '\$' and it will be ready for you to type the next command which is:

```
sudo apt-get install git
```

Once git is installed (if its not already there) you can "check out" the Adafruit Pi Python repository onto your Pi using the following commands

```
git clone http://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git
cd Adafruit-Raspberry-Pi-Python-Code
ls
```

If there is any problem during any of the steps above, you will see an error message. The most common reasons why something should fail to install are:

- a problem with your Internet connections
- a mis-typed command. Remember everything in Linux is case sensitive. It is best to open this page on your Raspberry Pi so you can just copy and paste the commands.

You will find all sorts of goodies in here, many of which we will use in later tutorials.

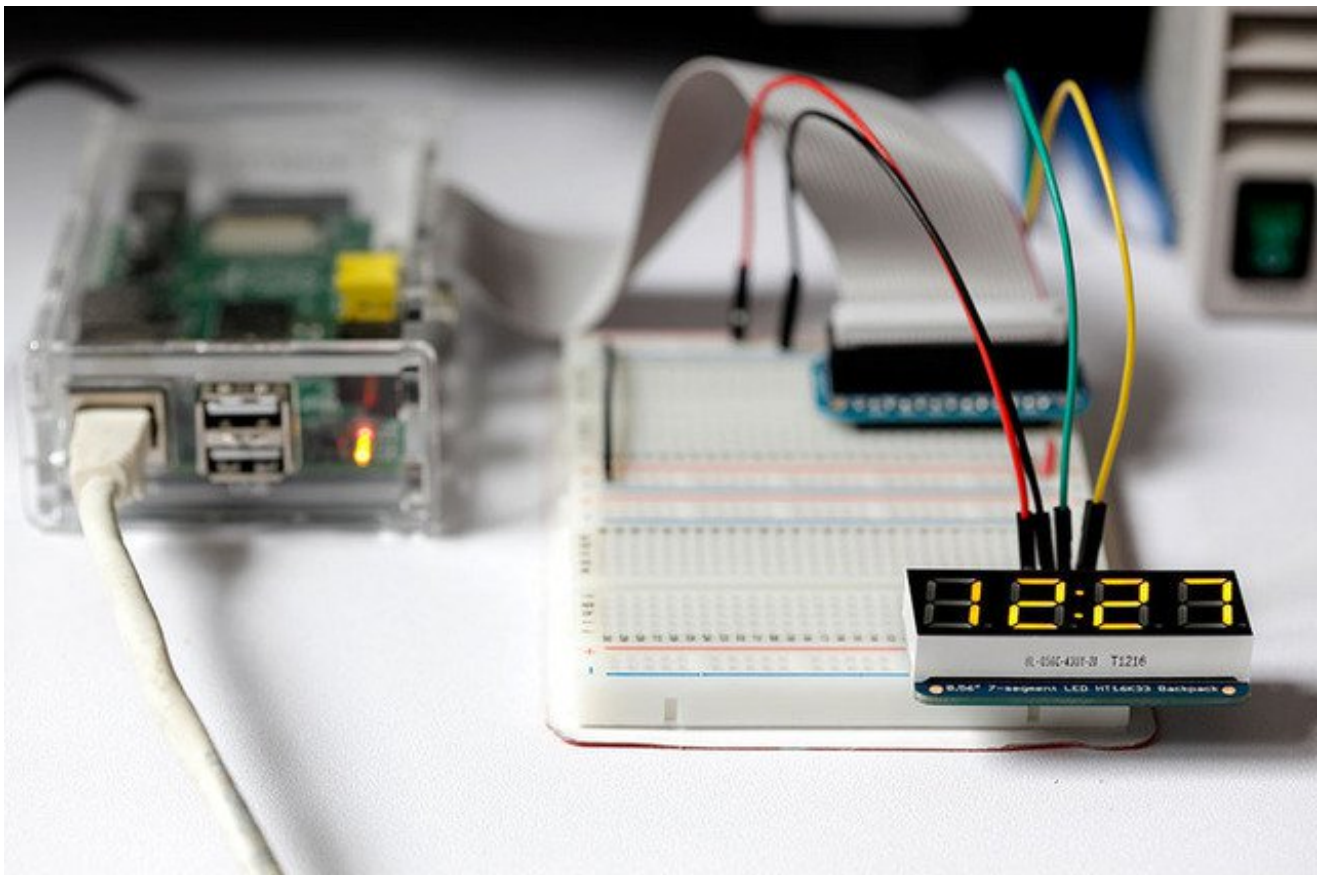
Configuring GPIO

The GPIO pins can be used as both digital outputs and digital inputs. As digital outputs, you can write programs that turn a particular pin HIGH or LOW. Setting it HIGH sets it to 3.3V setting it LOW sets it to 0V. To drive an LED from one of these pins, you need a 1k Ω resistor in series with the LED as the GPIO pins can only manage a small amount of power.

If you use a pin as a digital input, then you can connect switches and simple sensors to a pin and then be able to check whether it is open or closed (that is, activated or not)

Some Adafruit projects that use just GPIO.

- <http://learn.adafruit.com/raspberry-pi-e-mail-notifier-using-leds> (<http://adafru.it/aJ5>)
- <http://learn.adafruit.com/playing-sounds-and-using-buttons-with-raspberry-pi> (<http://adafru.it/aTD>)
- <http://learn.adafruit.com/basic-resistor-sensor-reading-on-raspberry-pi> (<http://adafru.it/aTE>)



To program the GPIO ports in Python, we need to install a very useful Python 2 library called Rpi.GPIO. This module gives us a simple to use Python library that will let us control the GPIO pins.

The installation process for this is the same whether you are using Raspbian or Occidentalis. In actual fact, some versions of Raspbian include this library, but these instructions will also have the effect of updating to the latest version, which is worth doing.

```
sudo apt-get update
```

To install RPi.GPIO, you first need to install the Python Development toolkit that RPi.GPIO requires.

To do this enter the following command into LXTerminal:

```
sudo apt-get install python-dev
```

Then to install Rpi.GPIO itself type:

```
sudo apt-get install python-rpi.gpio
```

You will probably be prompted to confirm by entering 'Y'.

That's all there is to it. You are ready to try some of the projects I mentioned at the top of this section.

Configuring I2C

I2C is a very commonly used standard designed to allow one chip to talk to another. So, since the Raspberry Pi can talk I2C we can connect it to a variety of I2C capable chips and modules.

Here are some of the Adafruit projects that make use of I2C devices and modules:

- <http://learn.adafruit.com/mcp230xx-gpio-expander-on-the-raspberry-pi> (<http://adafru.it/aTF>)
- <http://learn.adafruit.com/adafruit-16x2-character-lcd-plus-keypad-for-raspberry-pi> (<http://adafru.it/aTG>)
- <http://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi> (<http://adafru.it/aPm>)
- <http://learn.adafruit.com/matrix-7-segment-led-backpack-with-the-raspberry-pi> (<http://adafru.it/aPj>)
- <http://learn.adafruit.com/mcp4725-12-bit-dac-with-raspberry-pi> (<http://adafru.it/aPh>)
- <http://learn.adafruit.com/adafruit-16-channel-servo-driver-with-raspberry-pi> (<http://adafru.it/aPi>)
- <http://learn.adafruit.com/using-the-bmp085-with-raspberry-pi> (<http://adafru.it/aPg>)

The I2C bus allows multiple devices to be connected to your Raspberry Pi, each with a unique address, that can often be set by changing jumper settings on the module. It is very useful to be able to see which devices are connected to your Pi as a way of making sure everything is working.

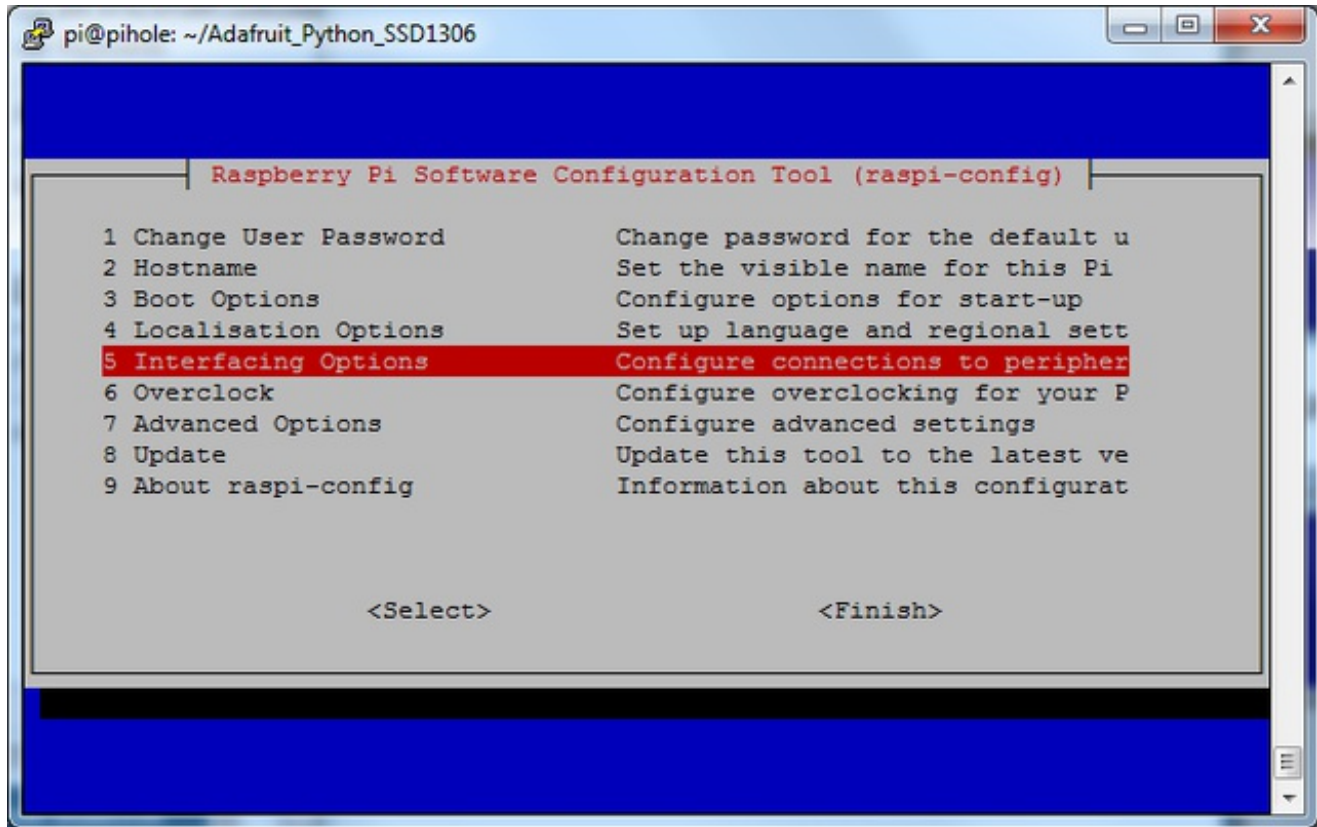
To do this, it is worth running the following commands in the Terminal to install the i2c-tools utility.

```
sudo apt-get install -y python-smbus  
sudo apt-get install -y i2c-tools
```

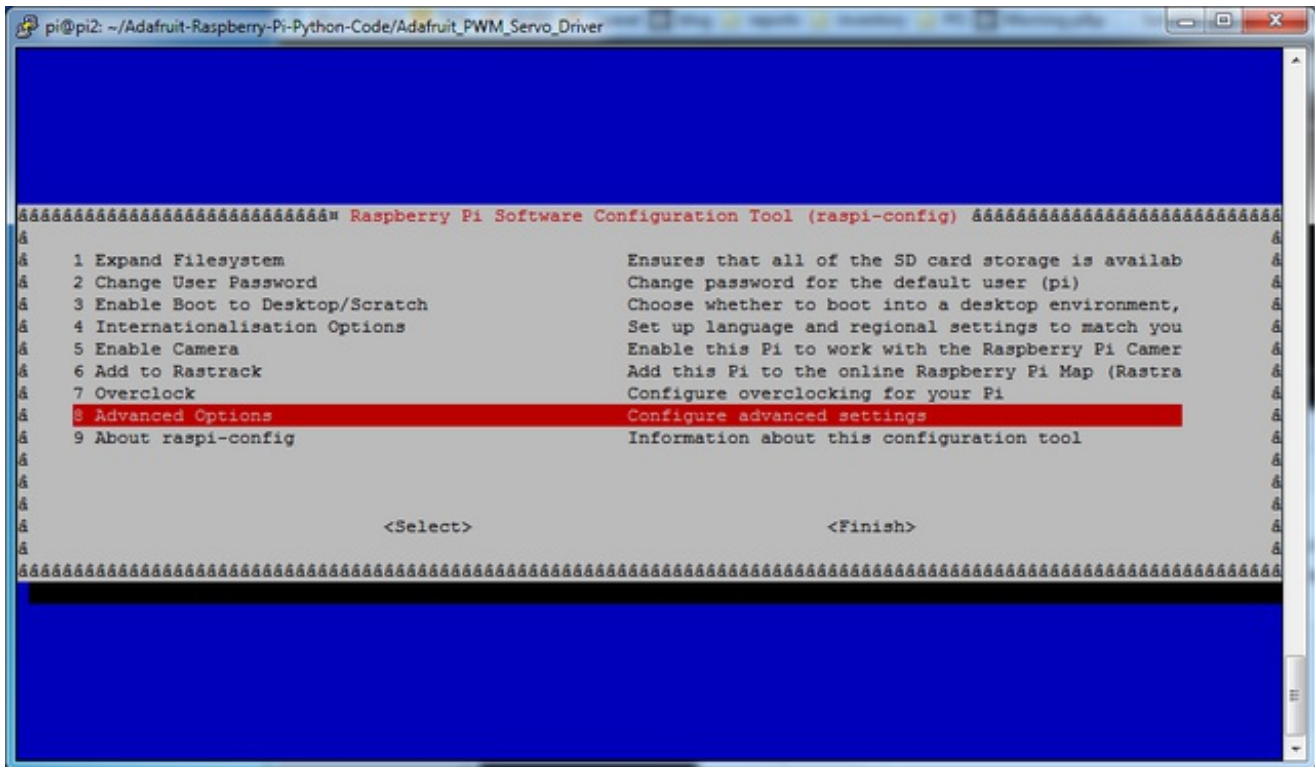
Installing Kernel Support (with Raspi-Config)

Run **sudo raspi-config** and follow the prompts to install i2c support for the ARM core and linux kernel

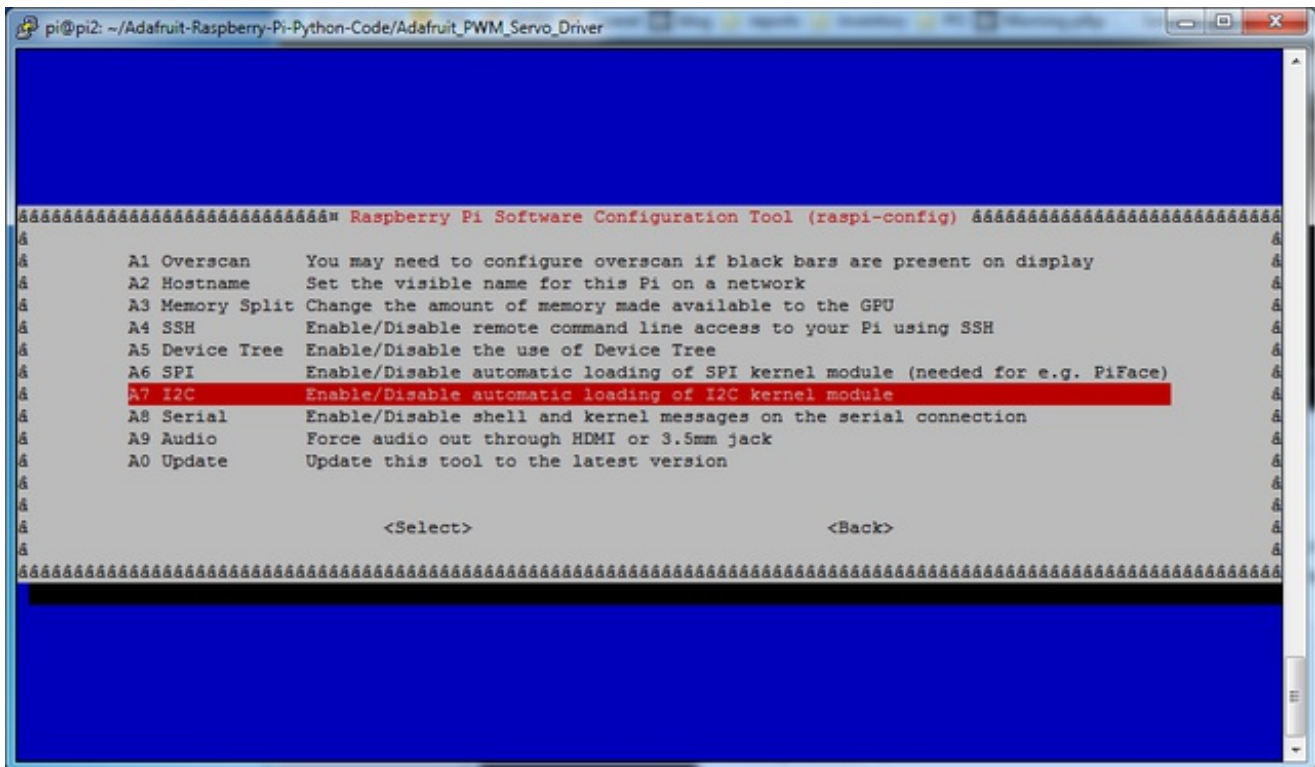
Go to **Interfacing Options**



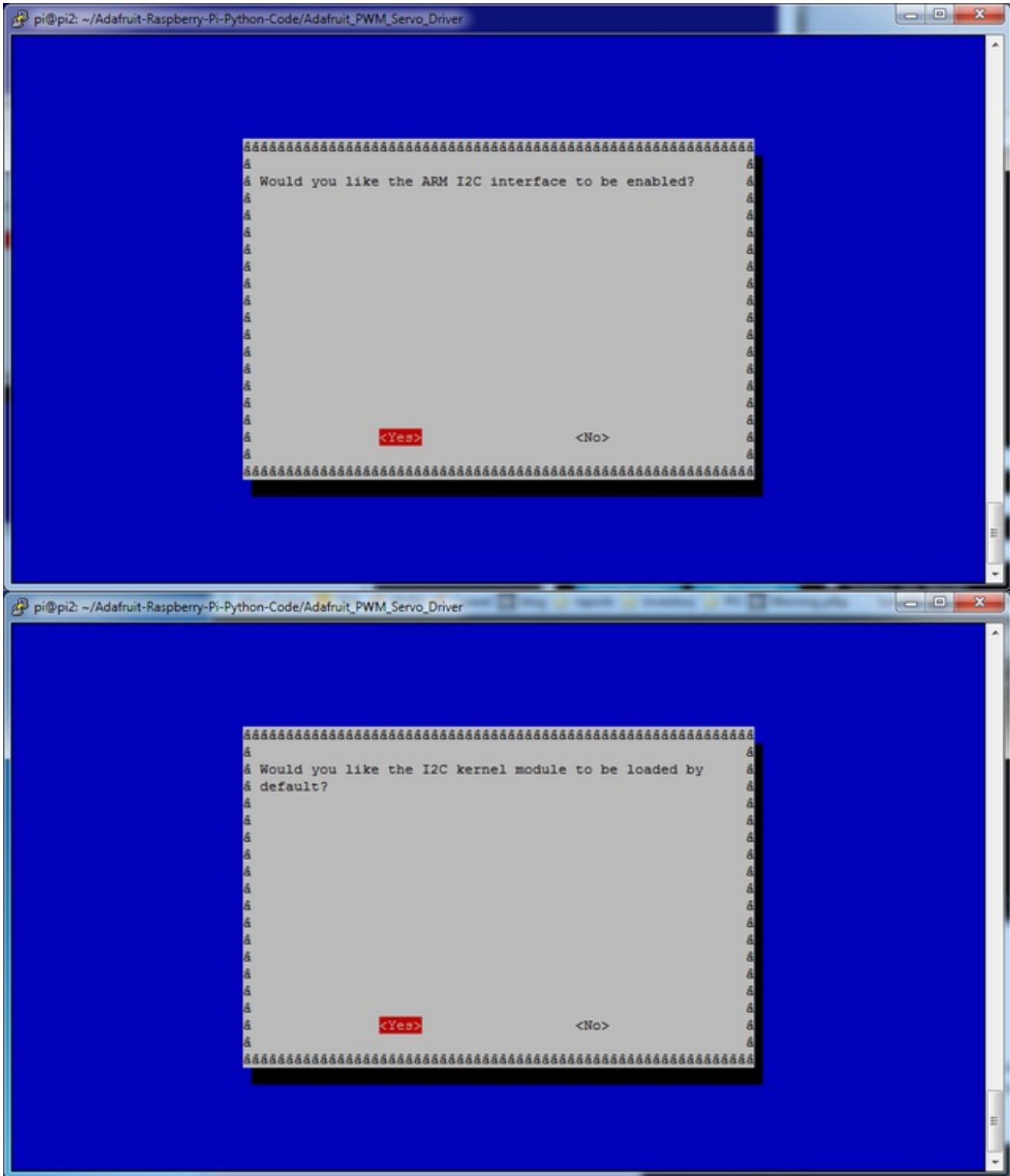
On older versions, look under **Advanced**



then I2C



Enable!



Then reboot!

We also recommend going through the steps below to manually check everything was added by raspi-config!

Installing Kernel Support (Manually)

If you're not using a modern Raspbian or you want to do it by hand, you can! Open LXTerminal or console or ssh and enter the following command:

```
sudo nano /etc/modules
```

and add these two lines to the end of the file:

```
i2c-bcm2708  
i2c-dev
```

like so:



```
LXTerminal  
File Edit Tabs Help  
GNU nano 2.2.6 File: /etc/modules  
# /etc/modules: kernel modules to load at boot time.  
#  
# This file contains the names of kernel modules that should be loaded  
# at boot time, one per line. Lines beginning with "#" are ignored.  
# Parameters can be specified after the module name.  
  
snd-bcm2835  
i2c-bcm2708  
i2c-dev  
  
[ Read 9 lines ]  
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos  
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Then save the file with **Control-X Y** <return>

Depending on your distribution, you may also have a file called **/etc/modprobe.d/raspi-blacklist.conf**

If you do not have this file then there is nothing to do, however, if you do have this file, you need to edit it and comment out the lines below:

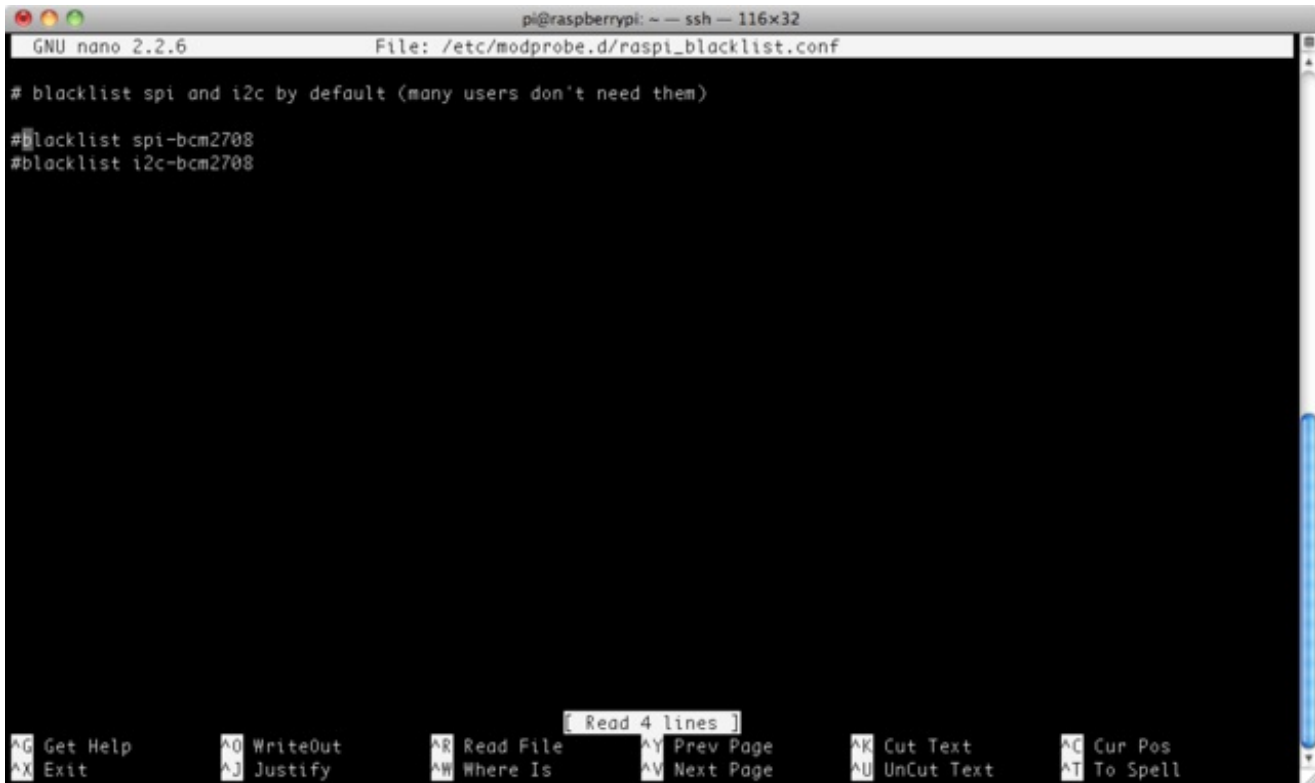
```
blacklist spi-bcm2708
blacklist i2c-bcm2708
```

.. by putting a # in front of them.

Open an editor on the file by typing:

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

.. then edit the file so that it appears as below, and then save and exit the file using CTRL-x and Y.

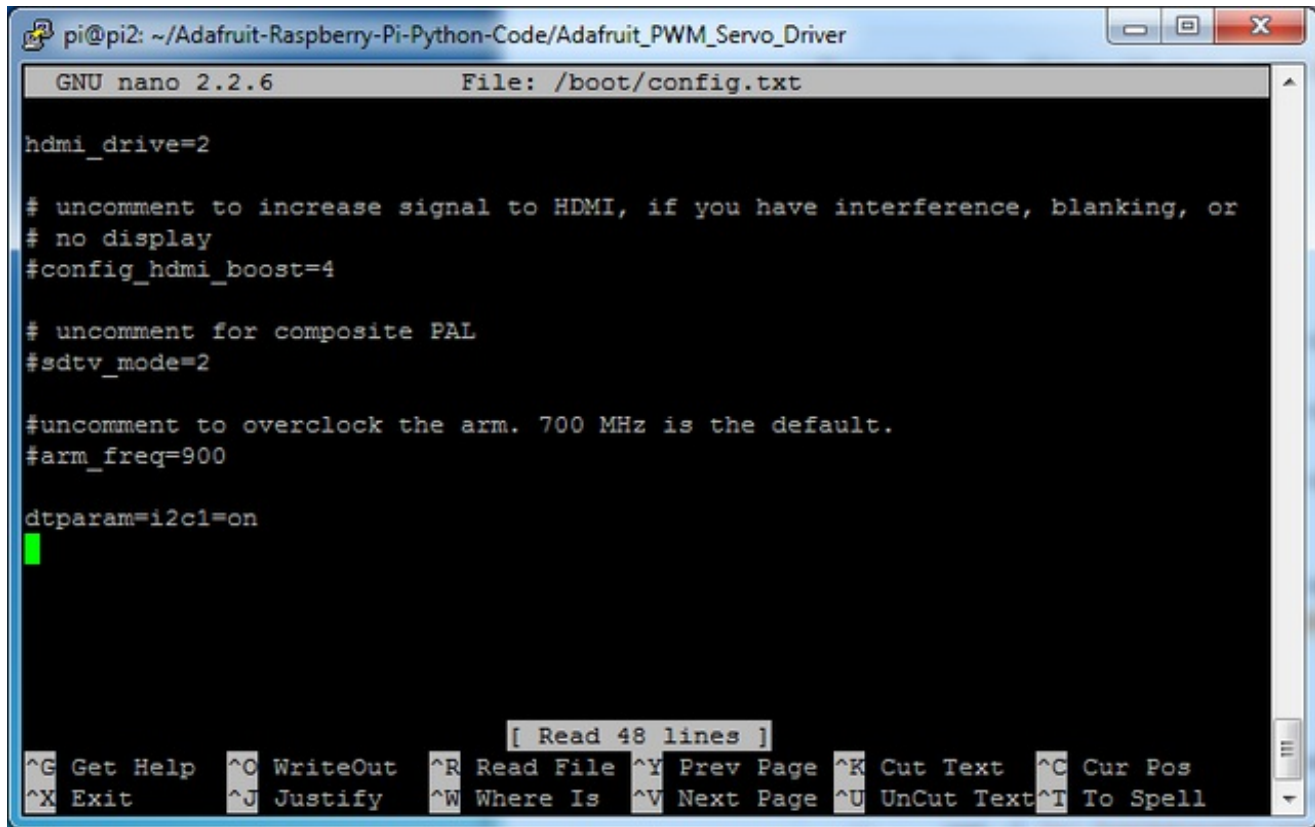


```
pi@raspberrypi: ~ -- ssh -- 116x32
GNU nano 2.2.6 File: /etc/modprobe.d/raspi_blacklist.conf
# blacklist spi and i2c by default (many users don't need them)
#blacklist spi-bcm2708
#blacklist i2c-bcm2708
[ Read 4 lines ]
^G Get Help      ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

If you are running a recent Raspberry Pi (3.18 kernel or higher) you will also need to update the **/boot/config.txt** file. Edit it with **sudo nano /boot/config.txt** and add the text

```
dtoverlay=i2c1=on
dtoverlay=i2c_arm=on
```

at the bottom. note that the "1" in "i2c1" is a one not an L!



```
pi@pi2: ~/Adafruit-Raspberry-Pi-Python-Code/Adafruit_PWM_Servo_Driver
GNU nano 2.2.6 File: /boot/config.txt

hdmi_drive=2

# uncomment to increase signal to HDMI, if you have interference, blanking, or
# no display
#config_hdmi_boost=4

# uncomment for composite PAL
#sdtv_mode=2

#uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=900

dtparam=i2c1=on

[ Read 48 lines ]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Once this is all done, reboot!

sudo reboot

Testing I2C

Now when you log in you can type the following command to see all the connected devices

`sudo i2cdetect -y 1`

```
LXTerminal
File Edit Tabs Help
root@raspberrypi:~# sudo i2cdetect -y 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40: 40  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70: 70  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
root@raspberrypi:~#
```

This shows that two I2C addresses are in use – 0x40 and 0x70.

Note that if you are using one of the very first Raspberry Pis (a 256MB Raspberry Pi Model B) then you will need to change the command to:

```
sudo i2cdetect -y 0
```

The Raspberry Pi designers swapped over I2C ports between board releases. Just remember: 512M Pi's use i2c port 1, 256M ones use i2c port 0!

Configuring SPI

Start by removing any blacklist for the spi module by running

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

And look for the line

blacklist spi-bcm2708

and put a **#** in front like so:

```

pi@pi2: ~/Adafruit-Raspberry-Pi-Python-Code/Adafruit_PWM_Servo_Driver
GNU nano 2.2.6 File: /etc/modprobe.d/raspi-blacklist.conf
# blacklist spi and i2c by default (many users don't need them)
#blacklist spi-bcm2708
#blacklist i2c-bcm2708
blacklist snd-soc-pcm512x
blacklist snd-soc-wm8804
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Tex ^T To Spell

```

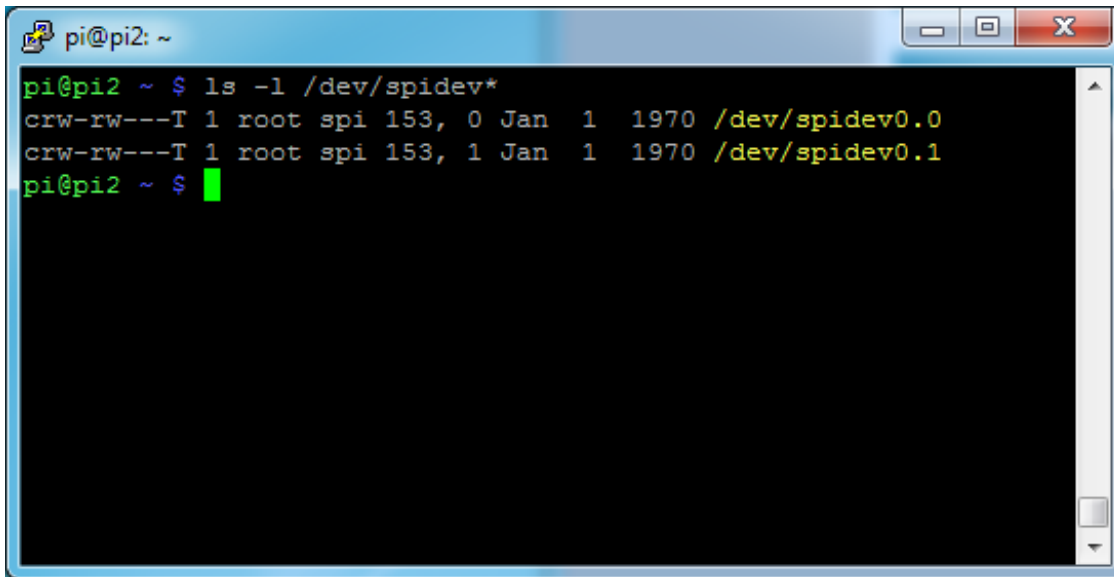
Now type in **control-X Y** and hit return to save the file.

reboot with **sudo reboot**

Next time you log in you can check that you can see the devices with

ls -l /dev/spidev*

you should see two 'devices' one for each SPI bus



```
pi@pi2: ~  
pi@pi2 ~ $ ls -l /dev/spidev*  
crw-rw---T 1 root spi 153, 0 Jan  1  1970 /dev/spidev0.0  
crw-rw---T 1 root spi 153, 1 Jan  1  1970 /dev/spidev0.1  
pi@pi2 ~ $
```

Test and Configure

The best way to test what you have done is to pick one of the projects (perhaps a fairly simple one to start with) and give it a try.

Here are some of the tutorials I would recommend as a first project:

- <http://learn.adafruit.com/raspberry-pi-e-mail-notifier-using-leds> (<http://adafru.it/aJ5>)
- <http://learn.adafruit.com/playing-sounds-and-using-buttons-with-raspberry-pi> (<http://adafru.it/aTD>)
- <http://learn.adafruit.com/matrix-7-segment-led-backpack-with-the-raspberry-pi> (<http://adafru.it/aPj>)

[Click Here for the Next Lesson](#)

<http://adafru.it/aUA>