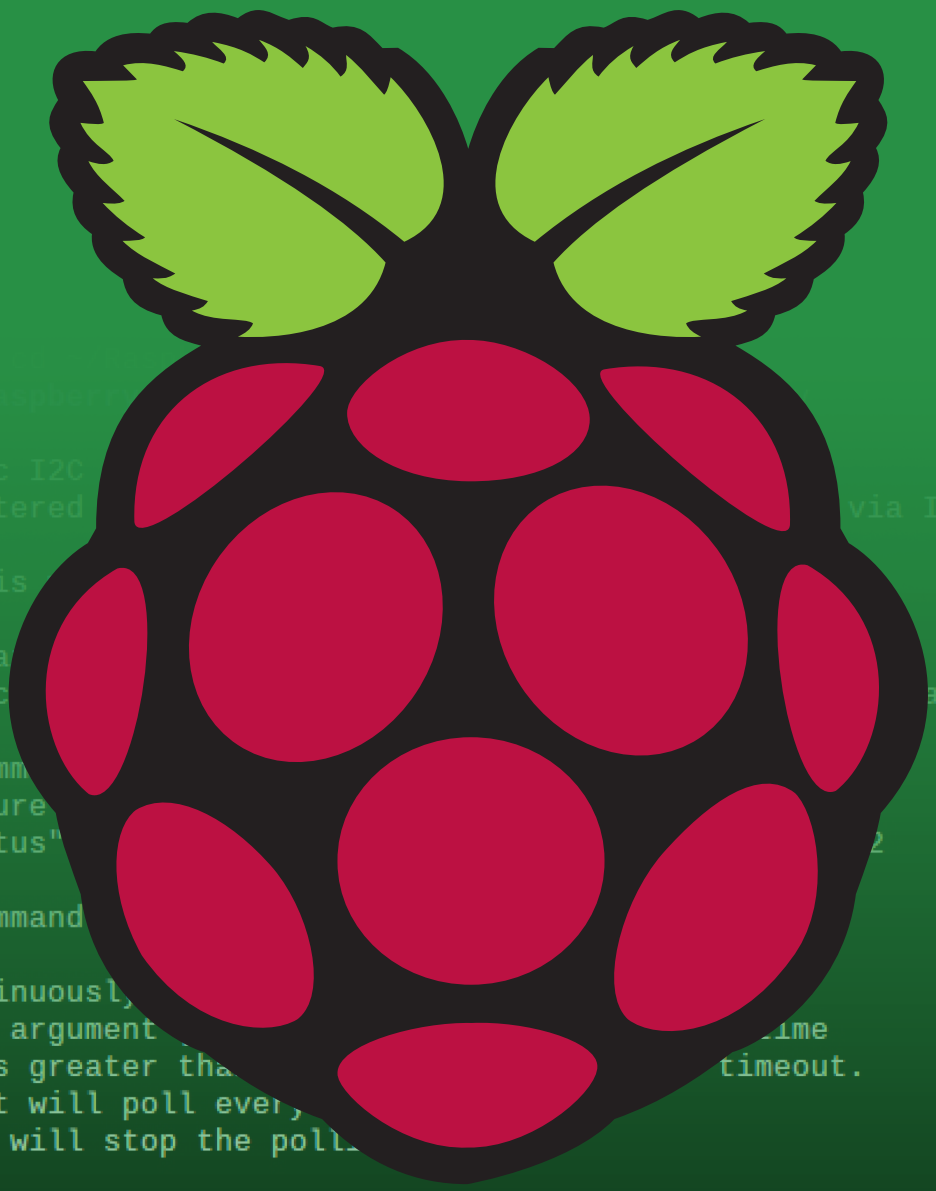
Atlas Scientific™
Environmental Robotics

Raspberry Pi

Sample Code & Wiring Diagrams



```
>> Atlas Scientific I2C
>> Any commands entered                                via I2C except:
   - Help
      brings up this
   - List
      lists the ava
      the --> indic                                    al commands
   - xxx:[command]
      sends the comm
      and sets future
      Ex: "102:status"                                 2
   - all:[command]
      sends the command
   - Poll[,x.xx]
      command continuousl
      the optional argument                            ime
      where x.xx is greater tha                        timeout.
      by default it will poll ever
>> Pressing ctrl-c will stop the poll

--> EC 78
 - DO 97
 - pH 99
>> Enter command: poll,3
-------press ctrl-c to stop the polling
Success EC 78: 22.47
Success DO 97: 9.09
Success pH 99: 3.076
-------press ctrl-c to stop the polling
Success EC 78: 22.47
Success DO 97: 9.09
Success pH 99: 3.053
-------press ctrl-c to stop the polling
```

Preparing Raspberry Pi

Install the latest Raspberry Pi OS

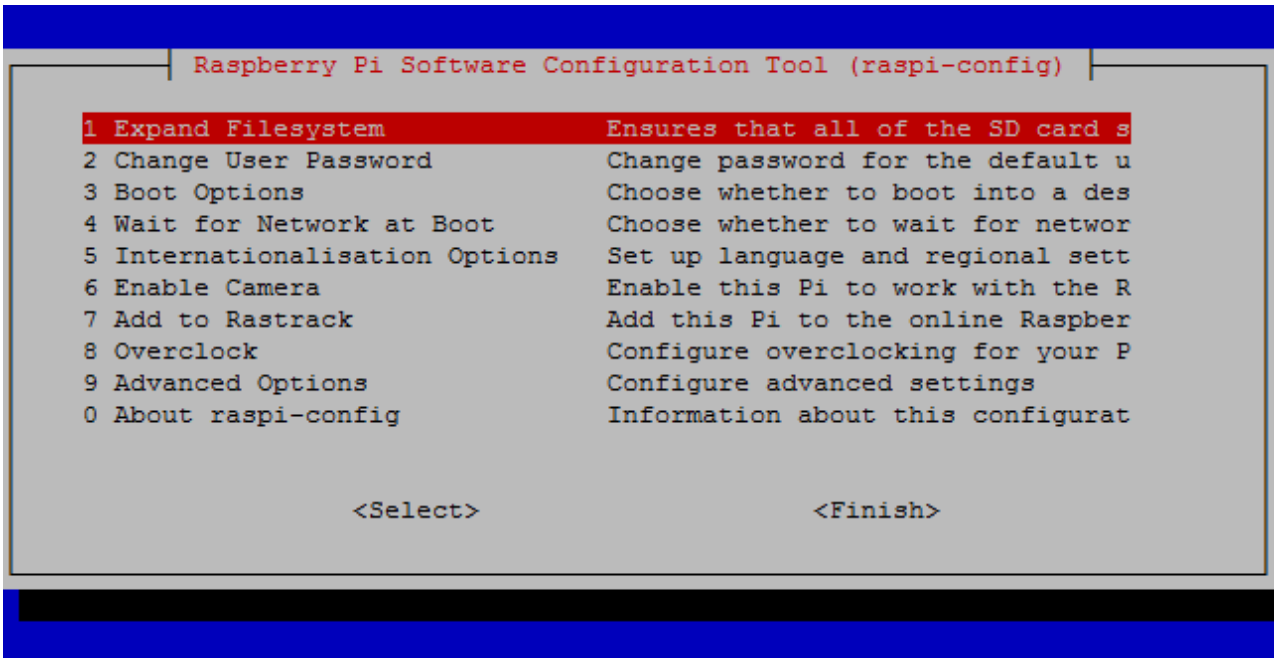
Click **HERE** and follow the instructions to get the Raspberry Pi OS running.

Expand file system

Run the following command line within the Raspberry Pi's terminal.

```
sudo raspi-config
```

You should see a blue screen with options in a gray box in the center, like so



Choose "Expand Filesystem"

Choosing this option will expand your installation to fill the rest of the SD card, giving you more space to use for files. **You will need to reboot the Raspberry Pi to make this available.**

Update and Upgrade Packages

First, you will need to update your system's package list by entering the following command in terminal.

```
sudo apt-get update
```

Next, upgrade your installed packages to their latest versions with the command.

```
sudo apt-get upgrade
```

Running the upgrade may take up to 30 minutes depending on which version of the Raspberry Pi you have.

Download the sample code

To download the Atlas Scientific™ sample code,run the following commands within the Raspberry Pi's terminal.

```
cd ~
```

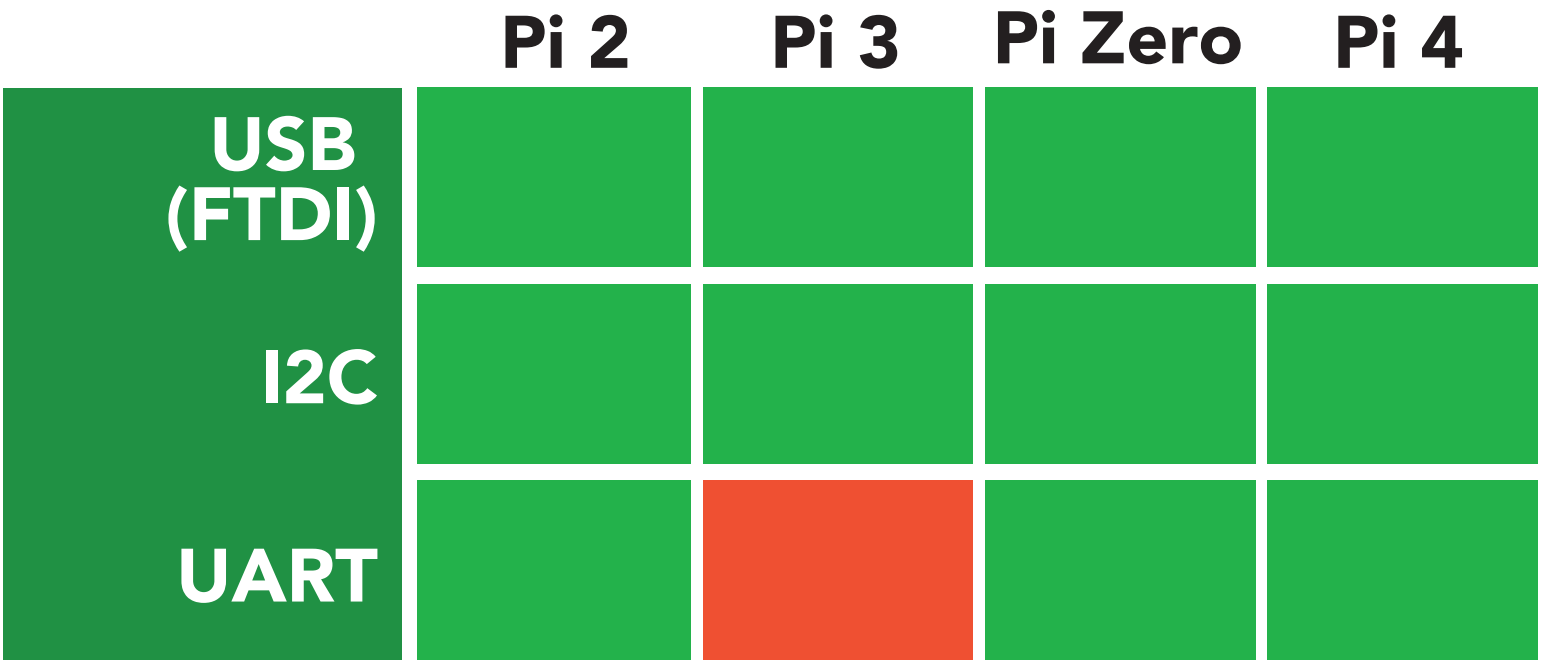
```
git clone https://github.com/AtlasScientific/Raspberry-Pi-sample-code.git
```

Once the sample code has finished downloading, you will be almost ready to begin using the Atlas Scientific™ EZO™ class circuits with your updated Raspberry Pi.

There are three different ways to interact with the Atlas Scientific™ EZO™ class circuits with your Raspberry Pi.

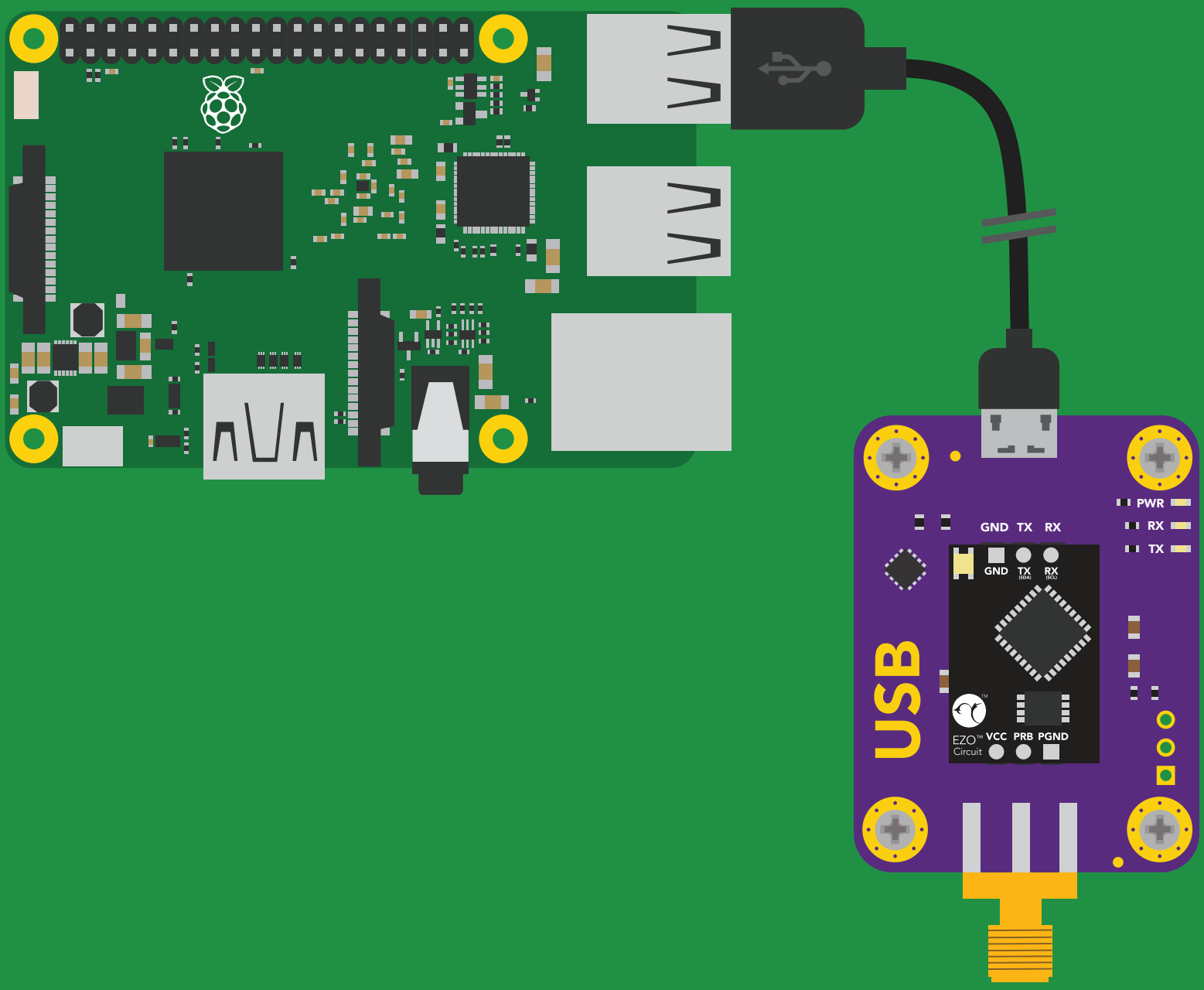
- **USB Mode**
- **I²C Mode**
- **UART Mode**

Sample code compatibility chart



The Raspberry Pi Foundation has failed to make a working UART on the Pi 3. Because of this no UART connected devices can run on a Raspberry Pi 3.

USB Mode


GND TX RX
PWR
RX
TX
GND TX (SDA) RX (SCL)
USB
EZO™ Circuit
VCC PRB PGND

USB Mode

USB mode will let you communicate through the Raspberry Pi's USB port to any FTDI based USB device. This includes all USB based Atlas Scientific™ devices.

First, we need to install the libftdi package.

```
sudo apt-get install libftdi-dev
```

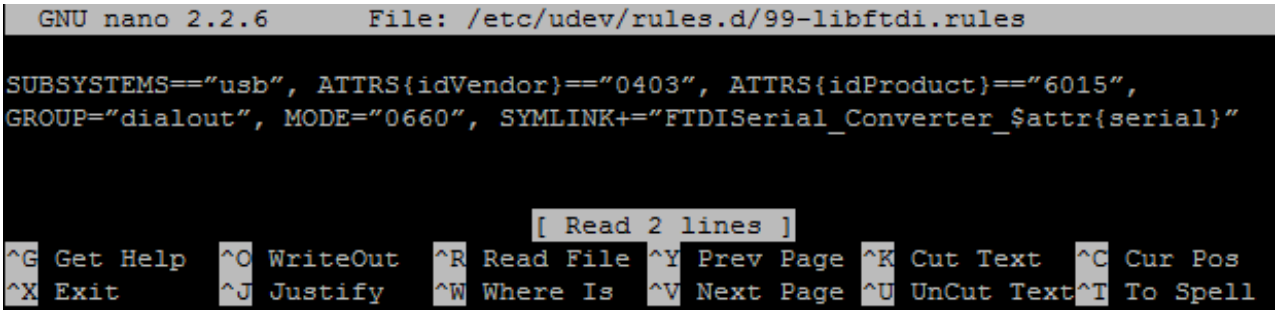
Next, we need to install the pylibftdi python package.

```
sudo pip install pylibftdi
```
← Python 2

```
sudo pip3 install pylibftdi
```
← Python 3

We need to create a udev rule file by entering the following command in terminal.

```
sudo nano /etc/udev/rules.d/99-libftdi.rules
```



Replace the current rule with following revised rule below.

```
SUBSYSTEMS=="usb", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6015",
GROUP="dialout", MODE="0660", SYMLINK+="FTDISerial_Converter_$attr{serial}"
```

Press "CTRL+X", then "Y" and hit Enter to save & exit.

Once the updated udev rule has been saved, a restart is required in order to apply changes to the rule.

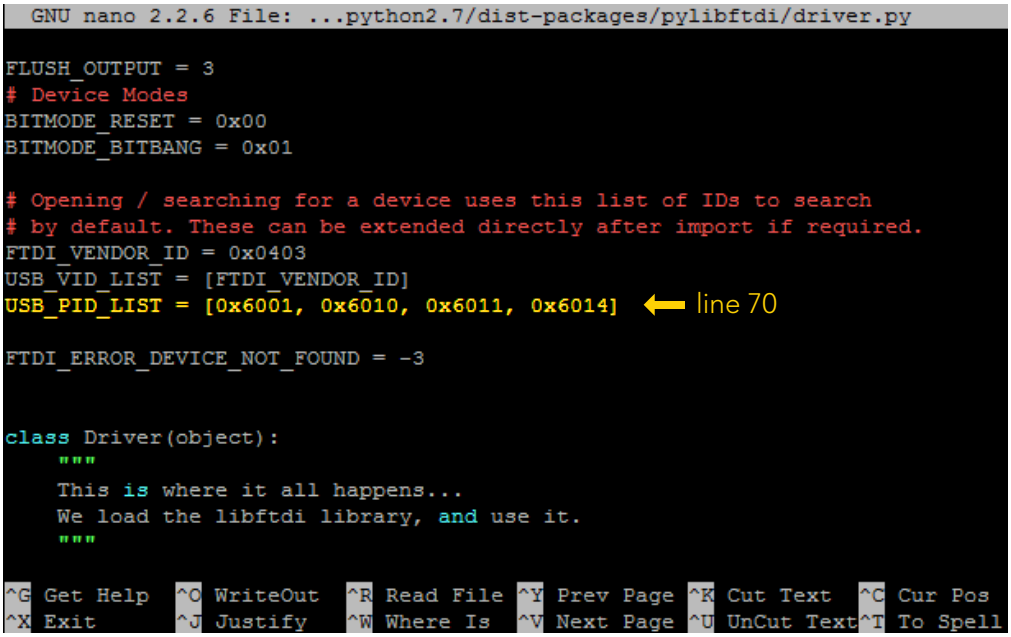
```
sudo service udev restart
```

Lastly, we need to modify the FTDI python driver.

Since Atlas Scientific™ FTDI devices use *USB PID (0x6015)*, we need to tweak the original FTDI driver, by entering the following command in terminal.

```
sudo nano /usr/local/lib/python2.7/dist-packages/pylibftdi/driver.py
```

Move down to the line 70 and add **0x6015** at the end of line.



Original line

USB_PID_LIST = [0x6001, 0x6010, 0x6011, 0x6014]

Modified line

```
USB_PID_LIST = [0x6001, 0x6010, 0x6011, 0x6014, 0x6015]
```

Press “CTRL+X”, then “Y” and hit Enter to save & exit.

Raspberry Pi sample code

Revised 06/22

Your Atlas Scientific™ EZO™ class circuits are almost ready to work with your Raspberry Pi, we just have to run a simple test first.

Connect your FTDI based USB device and run the following command in the terminal.

```
sudo python -m pylibftdi.examples.list_devices
```

The program will report information about each connected device.
You will get result like this:

```
FTDI:FT230X Basic UART:DA00TN6Q
```

Each FTDI adaptor has its own unique serial number.

In the result above, serial number is `DA00TN6Q`

Using pylibftdi module for Atlas Scientific™ EZO™ class circuits

Run the following commands in terminal.

```
cd ~/Raspberry-Pi-sample-code
```

```
sudo python ftdi.py
```

The program will present a list of available FTDI devies. Enter the index of the device you wish to use, and you will now be able to control an Atlas Scientific™ EZO™ class circuit via the USB port.

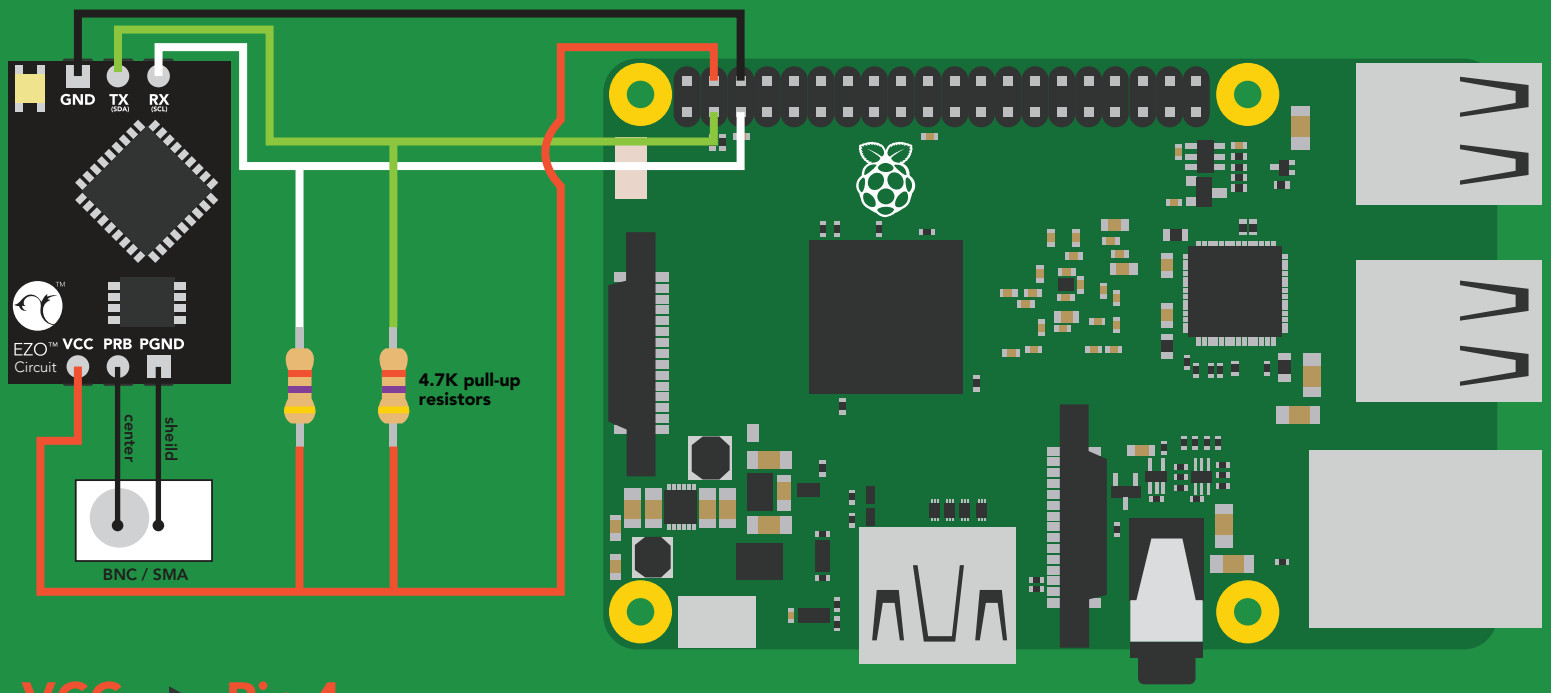
```
Discovered FTDI serial numbers:

Index:  0  Serial:  DA00OIQH

Index:  1  Serial:  DA00OJSH
==================================
Please select a device index:
```

For more details on the commands and responses, please refer to the datasheets of each Atlas Scientific™ EZO™ class circuit in use.

I²C Mode



VCC → Pin 4
GND → Pin 6
SDA → Pin 3
SCL → Pin 5

I²C Mode

Make sure the Atlas Scientific™ EZO™ class circuits are in *I²C mode* before moving further with the following instructions.

Before we can start using the EZO™ class circuits with your Raspberry Pi, we have to install and enable I²C bus on the Raspberry Pi.

Run the following commands in terminal.

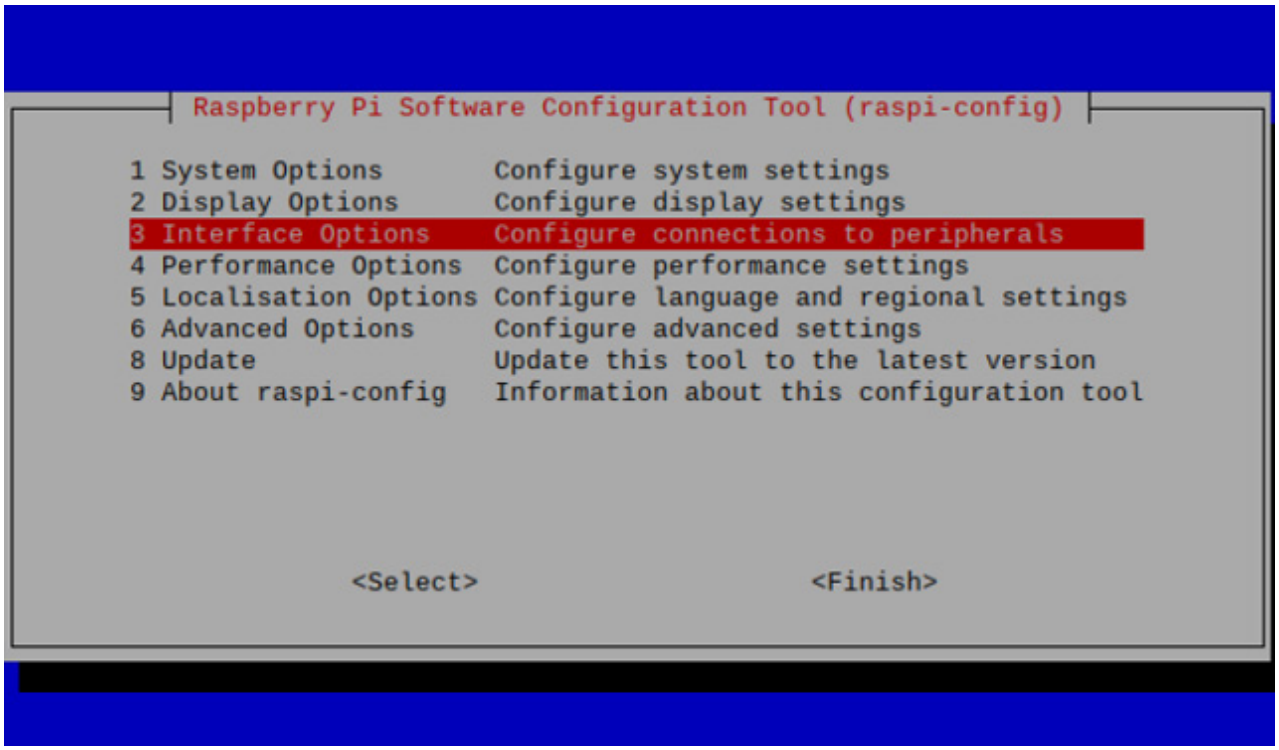
```
sudo apt-get install python-smbus
```

```
sudo apt-get install i2c-tools
```

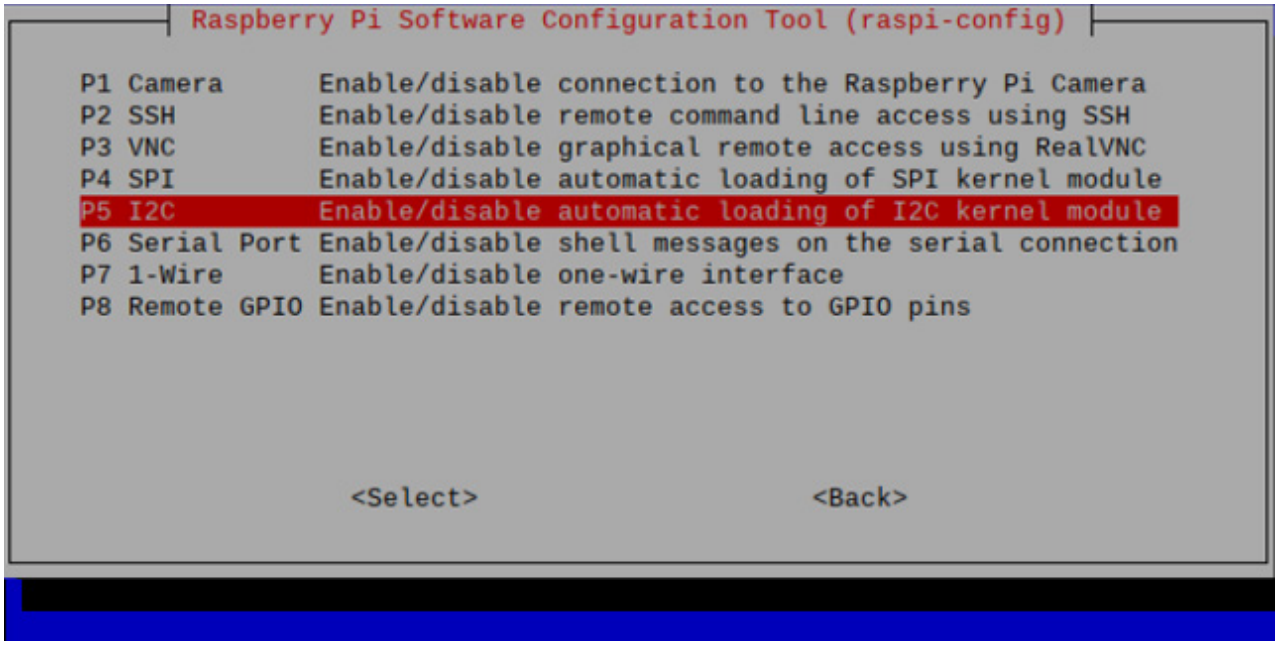
Once those have finished installing, we need to head back to the Raspberry Pi config.

```
sudo raspi-config
```

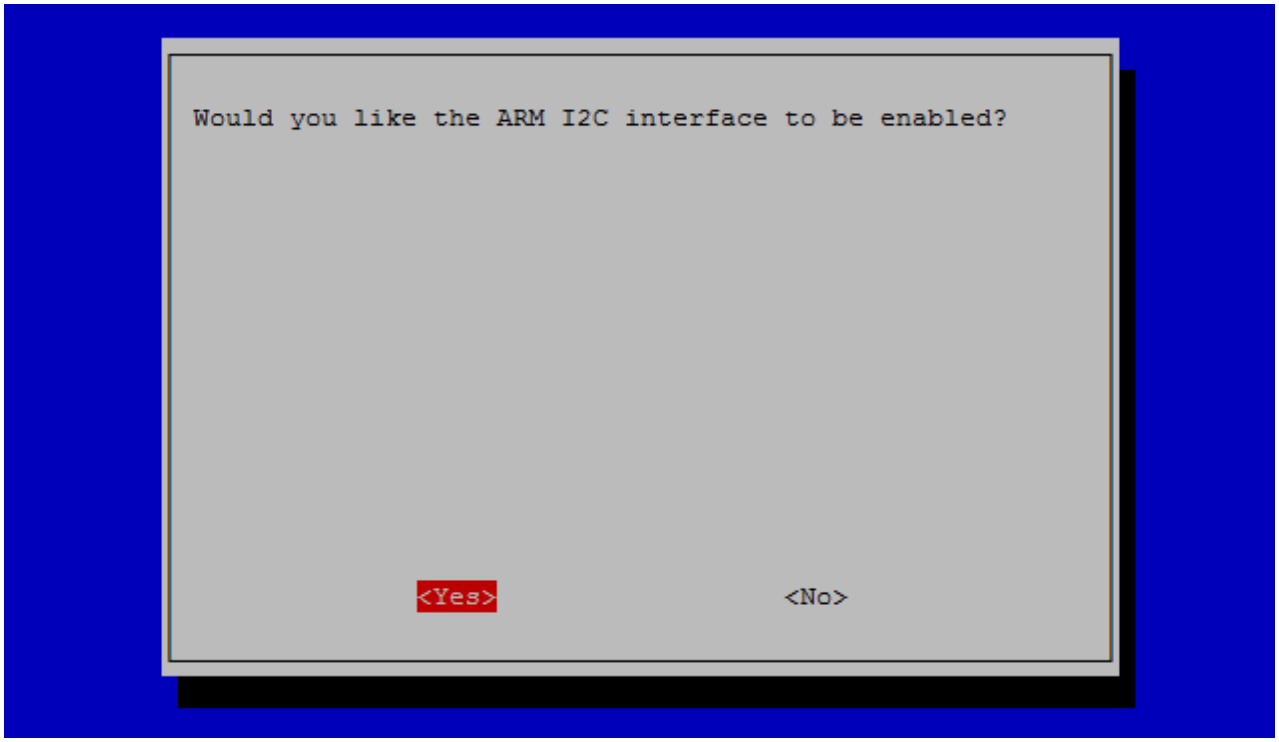
You should see a blue screen with options in a grey box in the center, like so



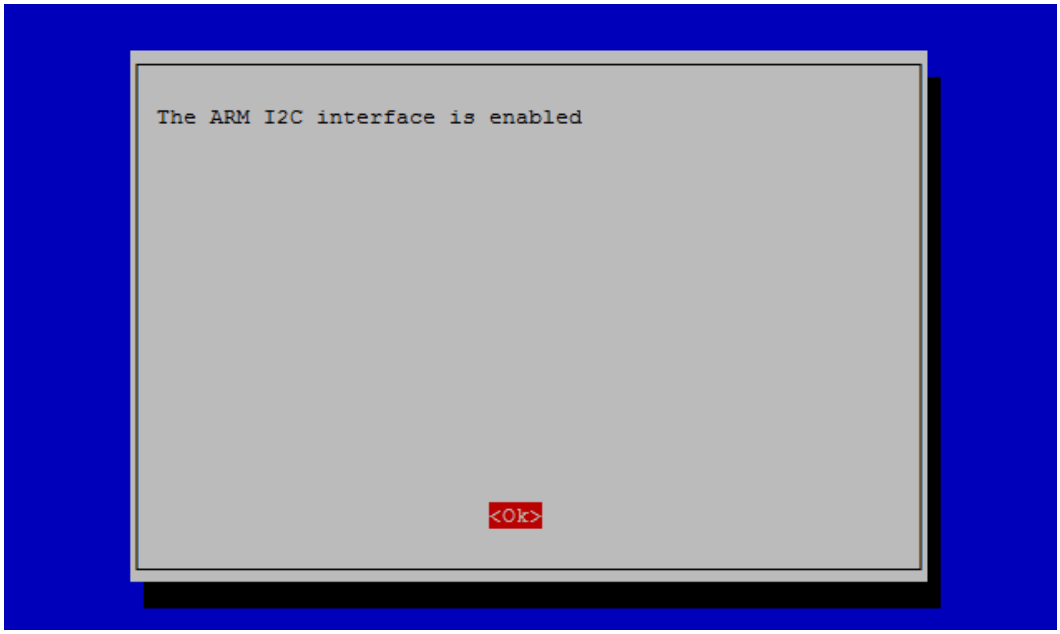
Choose “Interface Options”

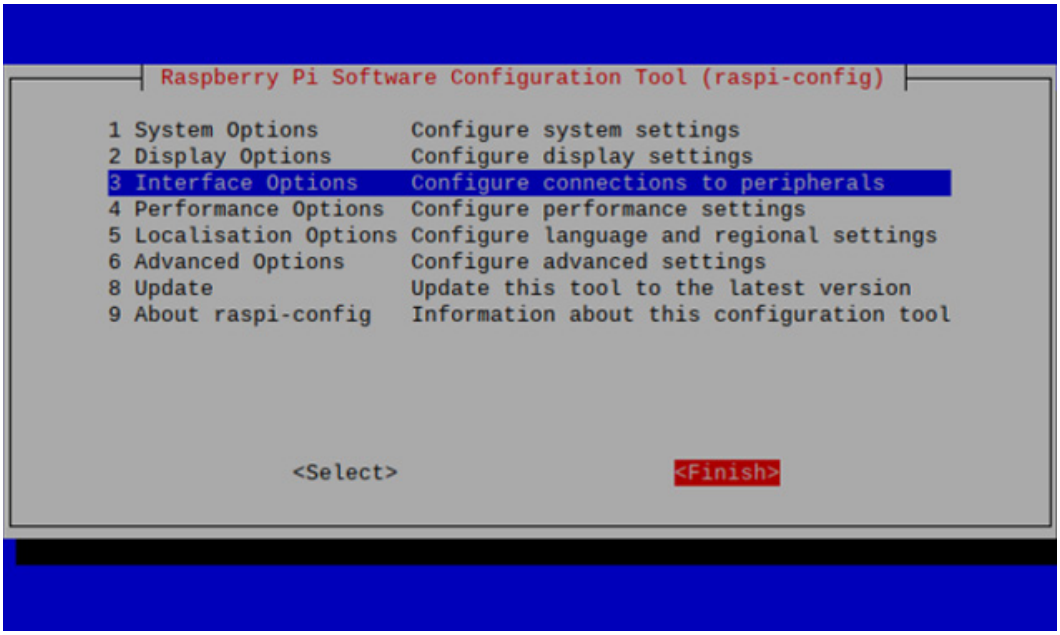


Choose "I2C"



Choose "YES"





Hit "OK" followed by "Finish" and reboot the Raspberry Pi.

```
sudo reboot
```

Raspberry Pi sample code

Revised 06/22

Your Atlas Scientific™ EZO™ class circuits are almost ready to work with your Raspberry Pi, we just have to run a simple test first.

Connect your EZO™ class circuit, and run the following command in terminal.

```
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: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- 63 -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

The program will report information about each connected I²C device. This shows that an I²C address (0x63) is in use.

I2C address list

Device	Decimal	Hex
pH	99	0x63
ORP	98	0x62
DO	97	0x61
EC	100	0x64
RTD	102	0x66
PMP	103	0x67
FLOW	104	0x68
CO2	105	0x69
PRS	106	0x6A
O2	108	0x6c
PMP-L	109	0x6D
HUM	111	0x6F
RGB	112	0x70

Run the following commands in terminal.

```
cd ~/Raspberry-Pi-sample-code
```

```
sudo python i2c.py
```

Each Atlas Scientific™ device has a different default I²C address.
To see a list of connected I²C devices from the program, use the command.

```
List
```

The last step is to tell the Raspberry Pi which circuit you want to talk to. By default the program will talk to the circuit pointed to by the arrow in the list shown by the list command. To talk to other circuits type their I2C address followed by a colon. For example to talk to the pH circuit, type 99: you can also add commands like “I”, “status” or “R”.

```
99:
```

This will now tell the Raspberry Pi to communicate with the EZO™ pH circuit 99 (0x63)

For more details on the commands, responses and I²C addresses, please refer to the datasheets of each Atlas Scientific™ EZO™ class circuit in use.

Raspberry Pi sample code

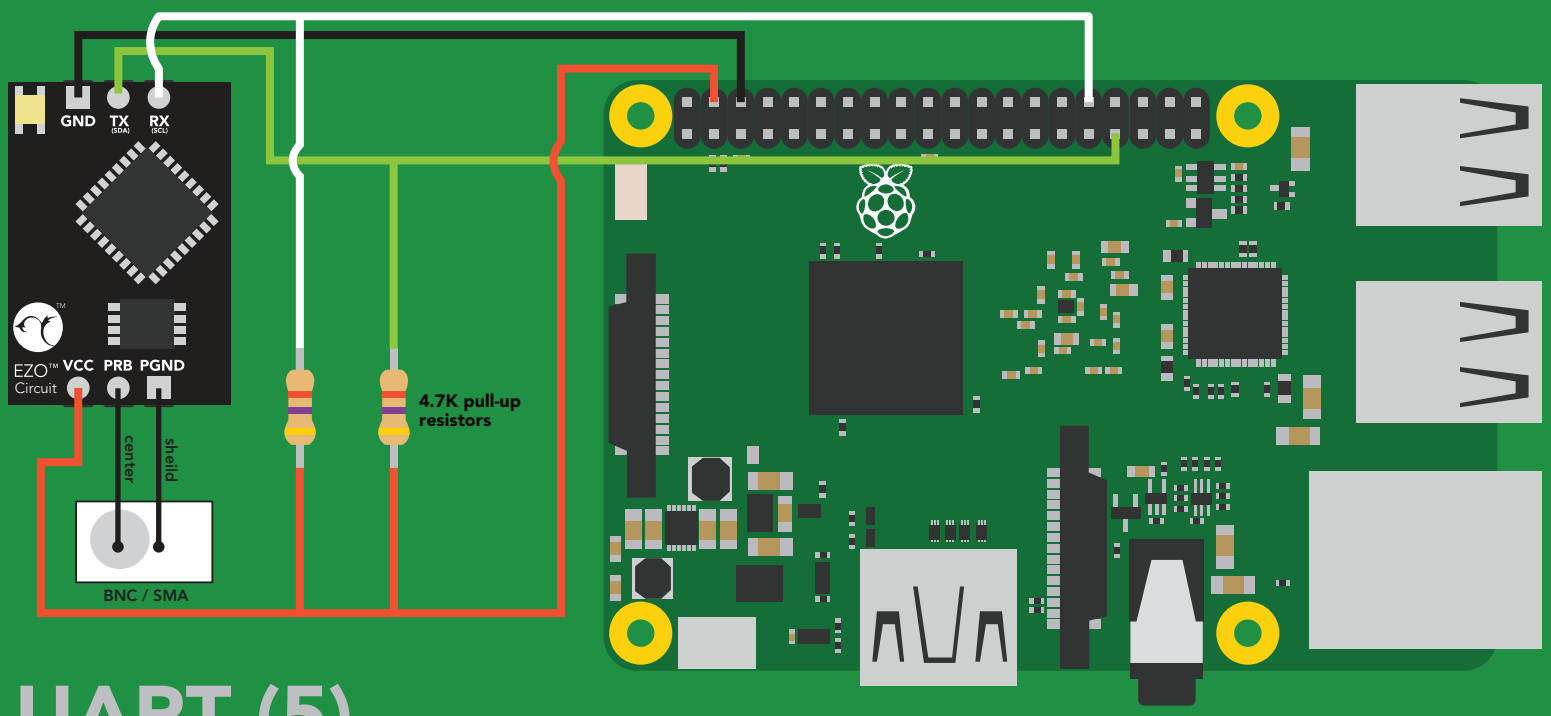
Revised 06/22

```
pi@raspberrypi:~ $ cd ~/Raspberry-Pi-sample-code
pi@raspberrypi:~/Raspberry-Pi-sample-code $ sudo python i2c.py

>> Atlas Scientific I2C sample code
>> Any commands entered are passed to the default target device via I2C except:
  - Help
      brings up this menu
  - List
      lists the available I2C circuits.
      the --> indicates the target device that will receive individual commands
  - xxx:[command]
      sends the command to the device at I2C address xxx
      and sets future communications to that address
      Ex: "102:status" will send the command status to address 102
  - all:[command]
      sends the command to all devices
  - Poll[,x.xx]
      command continuously polls all devices
      the optional argument [,x.xx] lets you set a polling time
      where x.xx is greater than the minimum 1.50 second timeout.
      by default it will poll every 1.50 seconds
>> Pressing ctrl-c will stop the polling

--> EC 78
 - DO 97
 - pH 99
>> Enter command: poll,3
-------press ctrl-c to stop the polling
Success EC 78: 22.47
Success DO 97: 9.09
Success pH 99: 3.076
-------press ctrl-c to stop the polling
Success EC 78: 22.47
Success DO 97: 9.09
Success pH 99: 3.053
-------press ctrl-c to stop the polling
Success EC 78: 22.47
Success DO 97: 9.09
Success pH 99: 3.040
```

UART Mode



UART (5)

VCC → Pin 4
GND → Pin 6
RX → Pin 32
TX → Pin 33

UART Mode

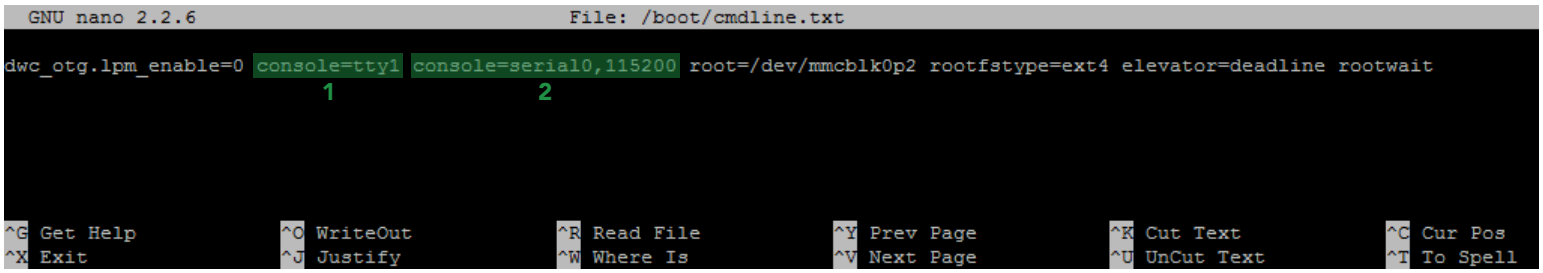
The Raspberry Pi Foundation has failed to make a working UART on the Pi 3. Because of this no UART connected devices can run on a Raspberry Pi 3 GPIO pins.

Before we can start using the Atlas Scientific™ EZO™ class circuits with your Raspberry Pi, we have to make a small tweak to the boot command line.

Run the following command line.

```
sudo nano /boot/cmdline.txt
```

You should see something that looks a lot like this:



You might see two separatecommands listed for the "console".

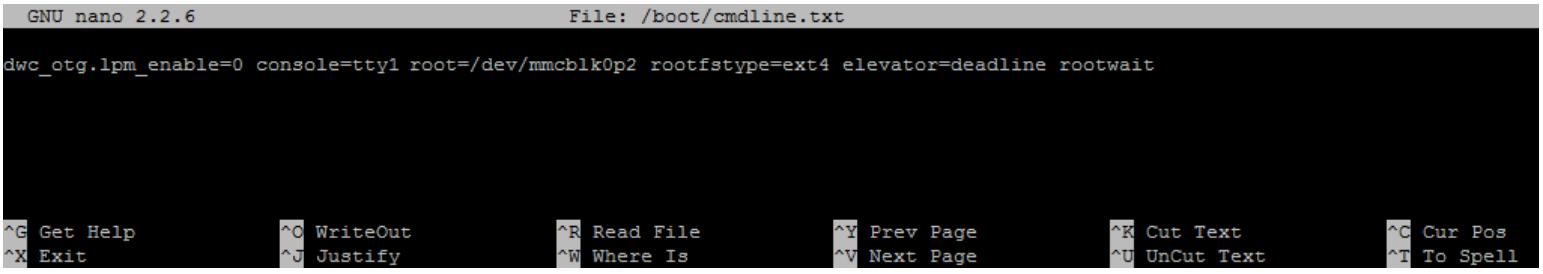
console=tty1 **console=serial0,115200**

1 2

This can cause a conflict in the serial port.

To correct this issue, ***delete*** the command:
console=serial0,115200

The command line should now look like this:



Press "CTRL+X", then "Y" and hit Enter to save & exit.

We need to ensure PySerial is installed for Python

```
sudo pip install pyserial
```

Run the following commands in terminal.

```
cd ~/Raspberry-Pi-sample-code
```

```
sudo python uart.py
```

UARTs on Pi 4

The raspberry pi 4 has 6 UARTs.

First we're going to enable UART 5.

Note that other UARTs share their pins with other peripherals, so those peripherals may have to be disabled to use them.

UART 5 uses pins 32 (TX) and 33 (RX) on the raspberry pi 40 pin header.

Go into the boot configuration

```
sudo nano /boot/config.txt
```

and add the lines

```
enable_uart=1
dtoverlay=uart5
```

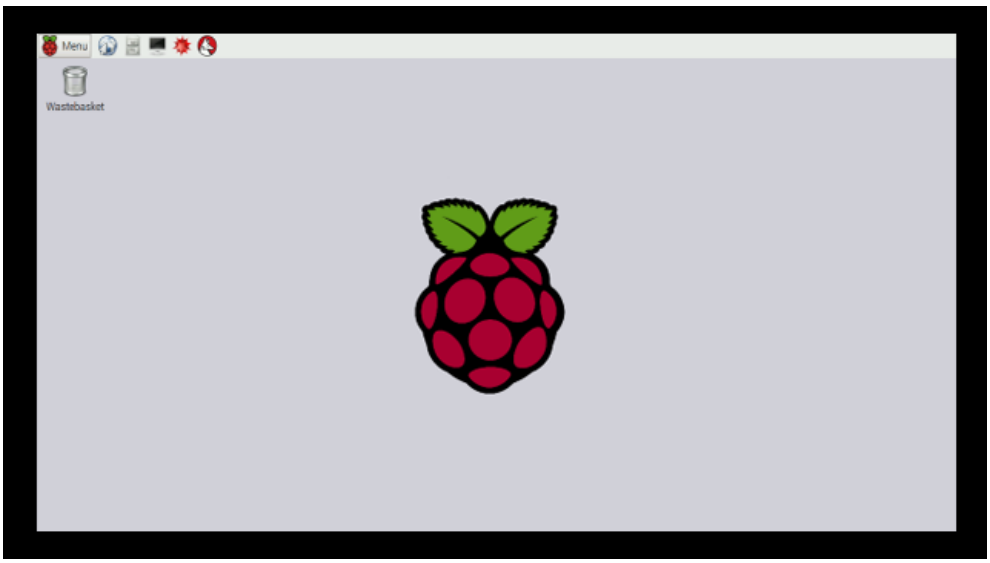
then restart the raspberry pi.
To use this port in the uart sample code **uart.py** change line 70 to:

```
usbport = '/dev/ttyAMA1'
```

Note that it may be a different ttyAMA depending on your setup.

Side note

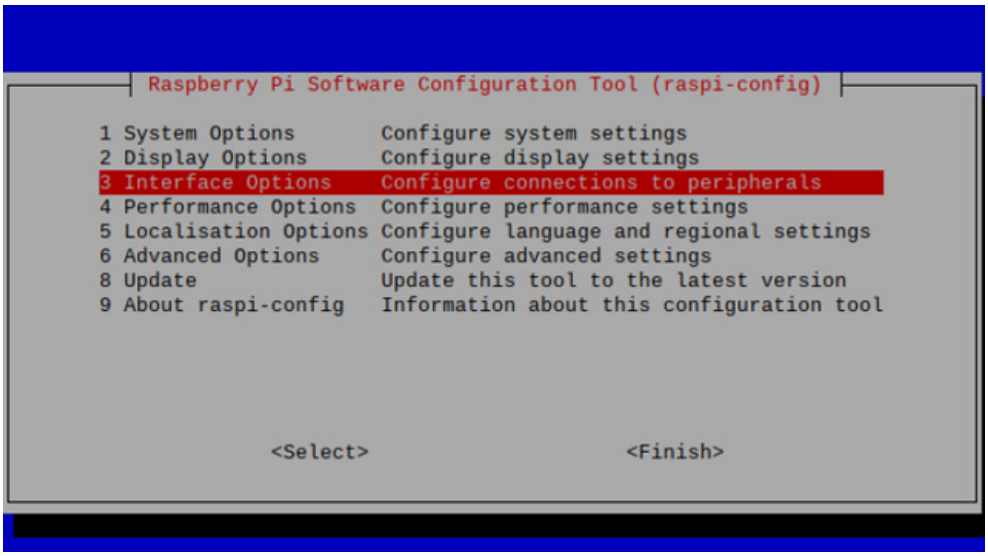
Does your Raspberry Pi have an annoying black border around the OS?



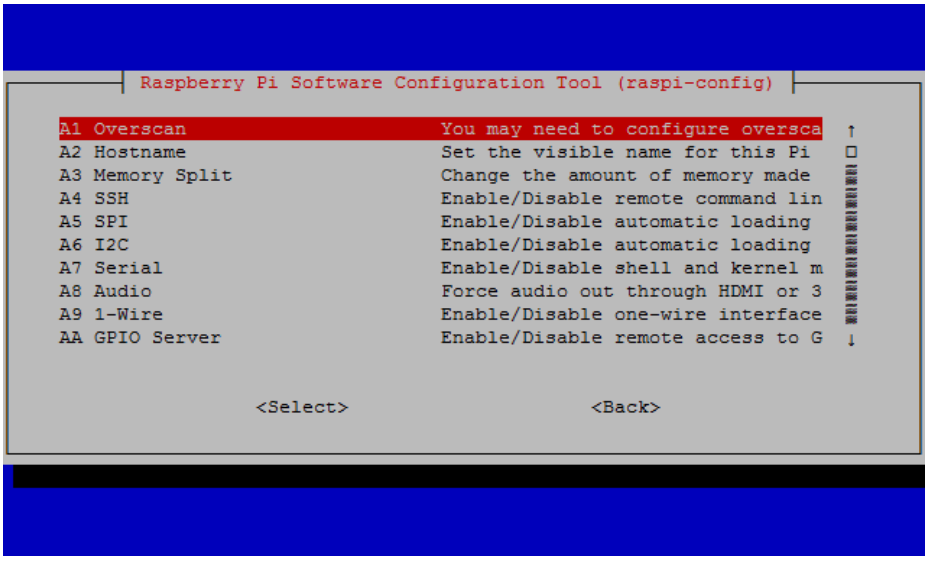
If so, here is how to remove it.

Run the following command line within the Raspberry Pi's terminal.

```
sudo raspi-config
```



Choose the option "Advanced Options"



Then, choose the option "A1 Overscan"

It will ask if you would like to enable compensation for displays with overscan?
say "**NO**"

The black border will know be gone. Enjoy!