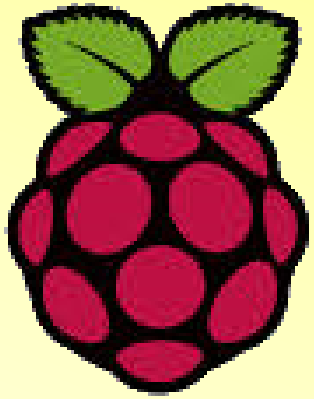
# Raspberry Pi



By Declan Fox

# Raspberry Pi

Hello world

Programming from the command line

In putty type the following

```
pi@raspberrypi ~ $ nano hello.py
```

This creates a file called hello.py and opens it in the Nano text editor

# Raspberry Pi

Type the following

```
  GNU nano 2.2.6                                File: hello.py

print("Hello, World")
```

Press ctrl and x

```
Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
 Y Yes
 N No           ^C Cancel
```

Press y

Press ENTER

# Raspberry Pi

In the command line type

python hello.py

```
pi@raspberrypi ~ $ python hello.py
Hello, World
```

# Raspberry Pi

If we want our script to run like a normal Linux app we need to add the following line.

```
#!/usr/bin/python
```

This line tells Linux to run the script using Python.

# Raspberry Pi

Then we need to make our script executable.

```
pi@raspberrypi ~ $ chmod +x /home/pi/hello1.py
```

Then we can run it using

```
pi@raspberrypi ~ $ /home/pi/hello1.py
```

# Raspberry Pi

Input from keyboard

```
pi@raspberrypi ~ $ nano hello_input.py
```

Then type the following

```
#!/usr/bin/python3
print ("What is your name?")
my_name = input()
print ("Hi good to meet you " + my_name)
```

Then

```
pi@raspberrypi ~ $ chmod +x /home/pi/hello_input.py
```

And finally

```
pi@raspberrypi ~ $ /home/pi/hello_input.py
```

# Raspberry Pi

Electronics

Using a Breadboard

What is a Resistor

Resistance calculator

Spice simulator

Raspberry Pi GPIO pins

# Raspberry Pi

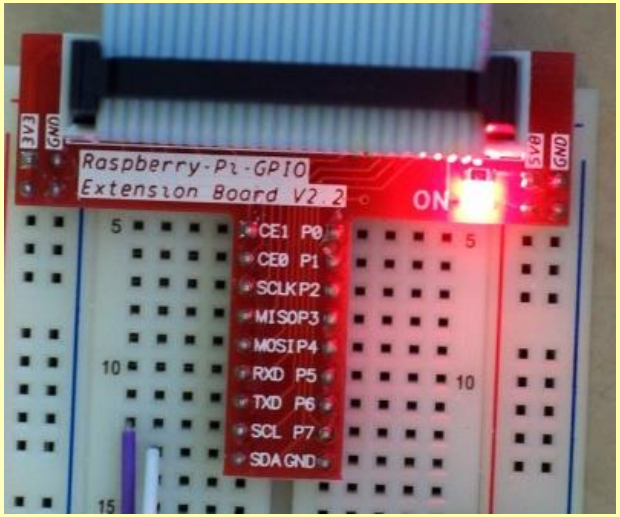
## GPIO pins

The biggest advantage of the Raspberry Pi is the GPIO pins.

GPIO stands for **G**eneral **P**urpose **I**nput **O**utput, all this means is that each pin can be used as an input or an output.

# Raspberry Pi

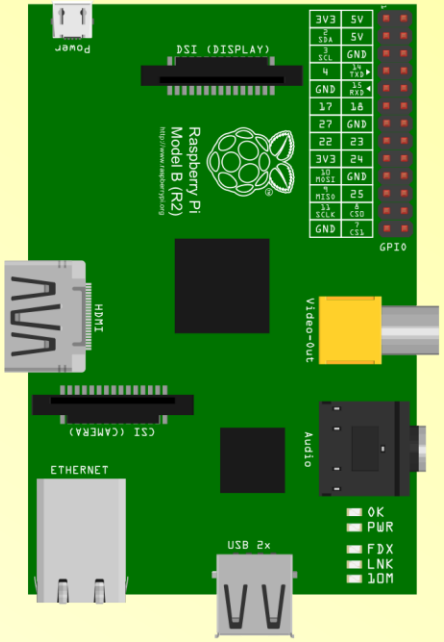
## Extension Board

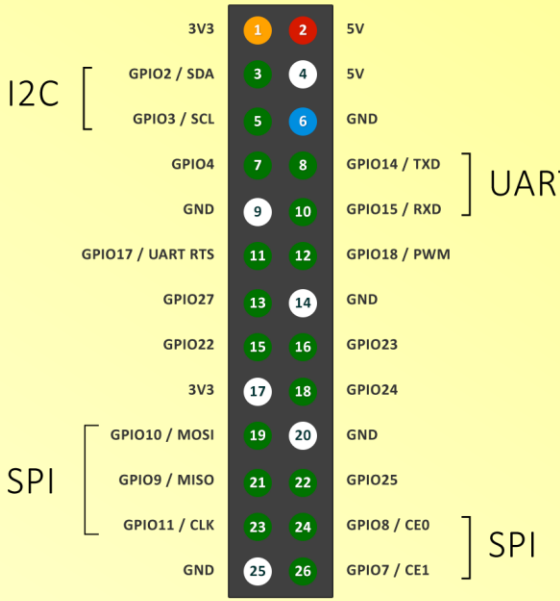


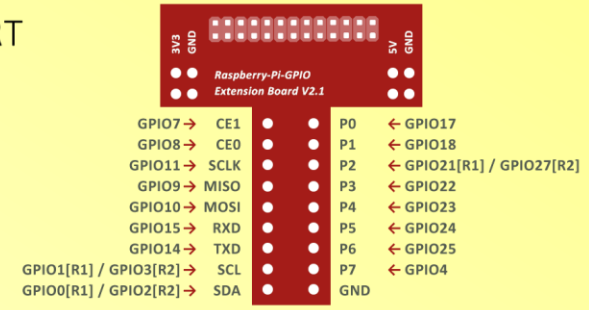

We’re going to use an extension board as this will make it easier to see what’s going on

# Raspberry Pi

## Extension board pins



# Raspberry Pi

First we need to set up the GPIO pins

```
import RPi.GPIO as GPIO
```

This imports the GPIO module

```
GPIO.setmode(GPIO.BCM)
```

This is just a pin numbering scheme

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

This sets up pin 23 as an output

```
GPIO.setup(25,GPIO.IN)
```

This sets up pin 25 as an input

# Raspberry Pi

To set an output high

```
GPIO.output(23,True)
```

This sets the voltage on pin 23 to 3.3v

To set an output low

```
GPIO.output(24,False)
```

This sets the voltage on pin 24 to 0v

# Raspberry Pi

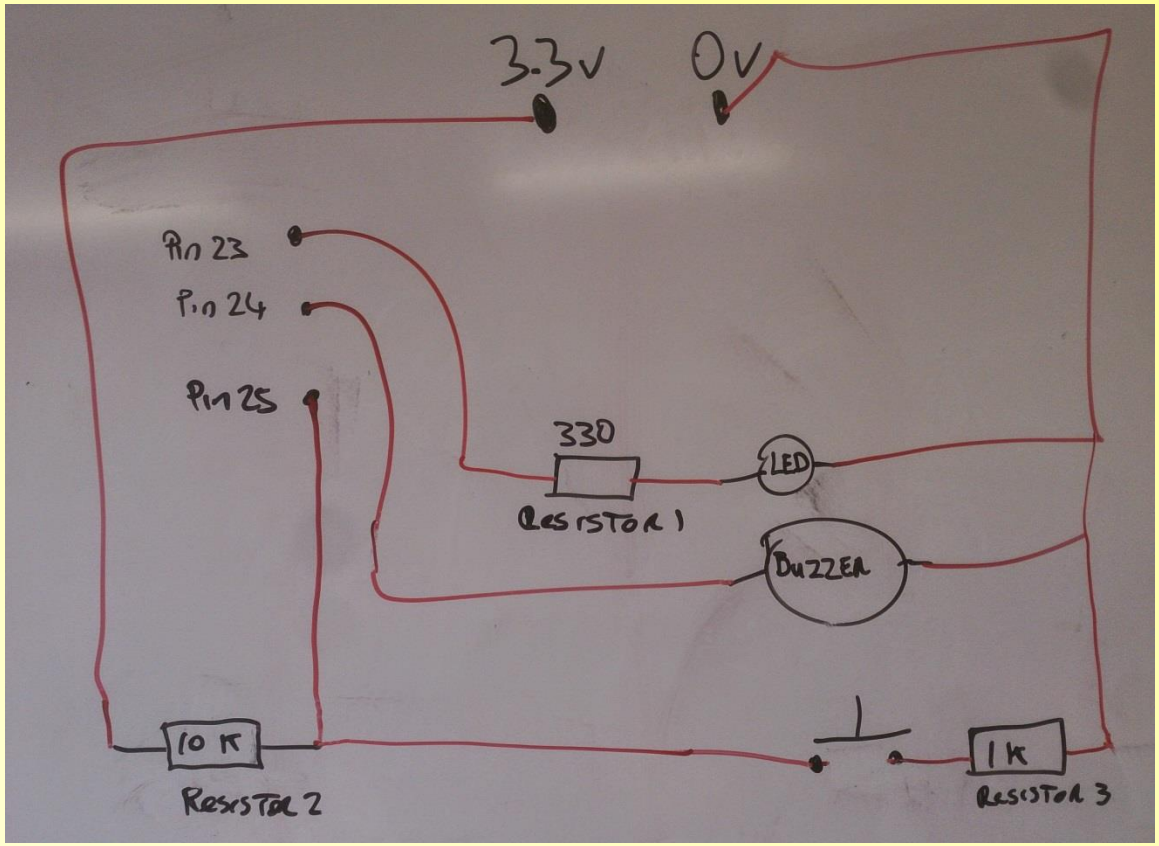
To read an input

```
if (GPIO.input(25) == False):
```

This if statement checks what voltage is on pin 25 and if it is 0v it will execute the block of code following it.

# Raspberry Pi

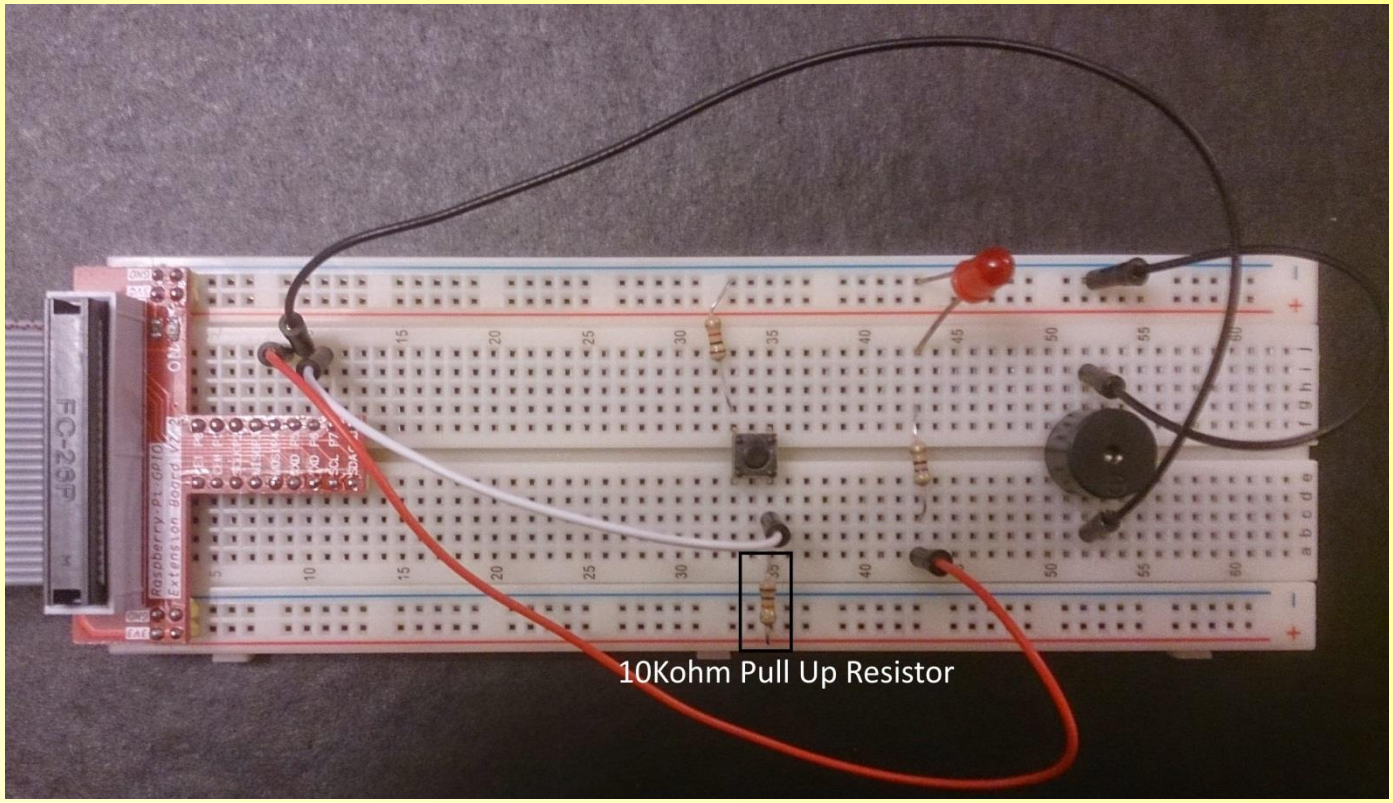
We used the following diagram to wire up our breadboard



# Raspberry Pi

When a GPIO pin isused as an input it is “floating” and has no defined voltage level.
we need to tie to 0v or 3.3v it so that it is always connected and reads high or low.
We use a Pull up resistor to connect the pin to 3.3v, this means that when the switch is open it will read high. When the switch is pressed (with the other side connected to 0V) there is a lower resistance path to 0v and so the pin will read low.
The large (10kΩ) resistor ensures that only a little current is drawn when the switch is pressed.

# Raspberry Pi




Our breadboard ready to run our code

# Raspberry Pi

```
#!/usr/bin/env python

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO.setup(25,GPIO.IN)
GPIO.setup(23,GPIO.OUT)
GPIO.setup(24,GPIO.OUT)

running = 1
while running:

        if (GPIO.input(25) == False):
                GPIO.output(23,True)
                GPIO.output(24,True)
                time.sleep(0.5)
                GPIO.output(24,False)
                time.sleep(0.5)
                GPIO.output(24,True)
                time.sleep(0.5)
                GPIO.output(24,False)
                GPIO.output(23,False)
                running = 0
GPIO.cleanup()
```

Our Finished Code