Lesson 3 –
Physical computing using Python

- S.P. Chong

Objectives

- In this lesson, you will learn to interface RPi (Raspberry Pi) to various **digital** / **analogue** I / O (**Input** / **Output**) devices.

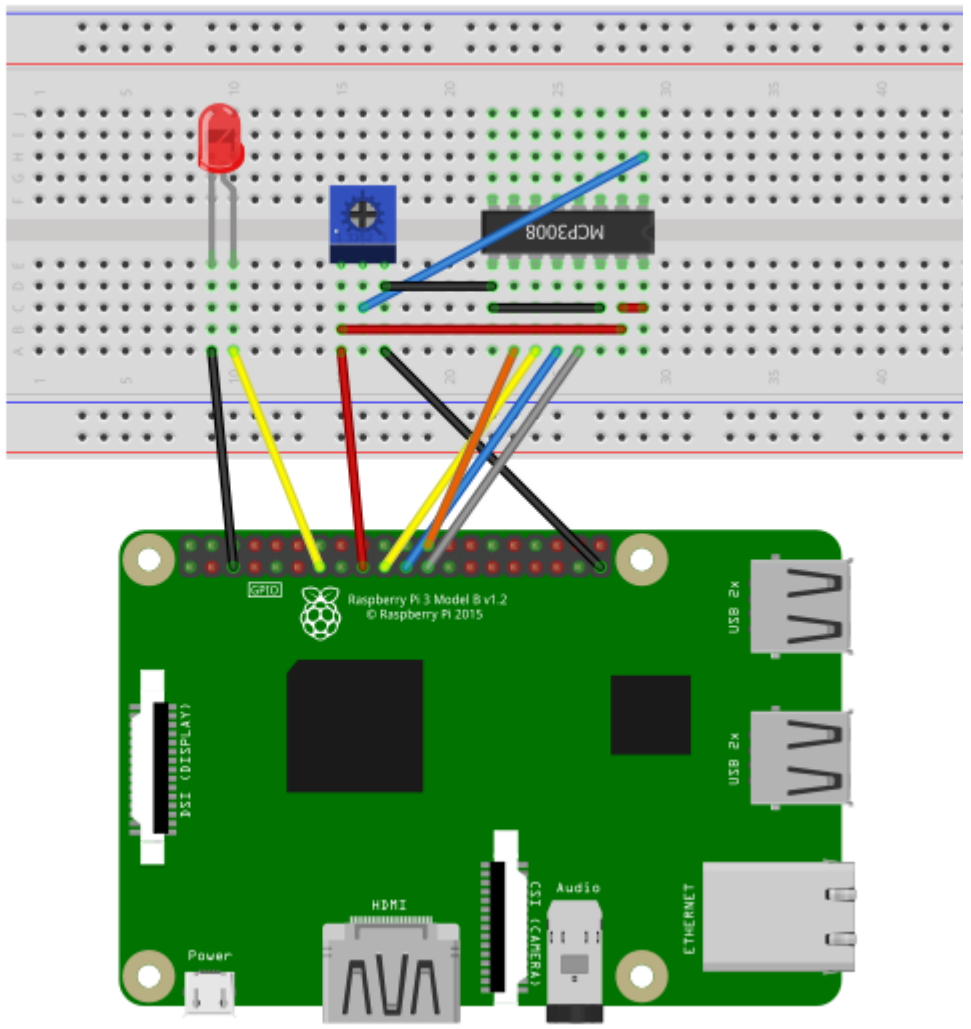
- You will also learn how various **sensors** & **actuators** can be used in many applications.

- We may even talk about some more advanced **serial interfaces** such as **UART** (Universal Asynchronous Transmitter Receiver), **SPI** (Serial Peripheral Interface), **TWI** (Two Wire Interface) for connecting I / O devices.

GPIO pins

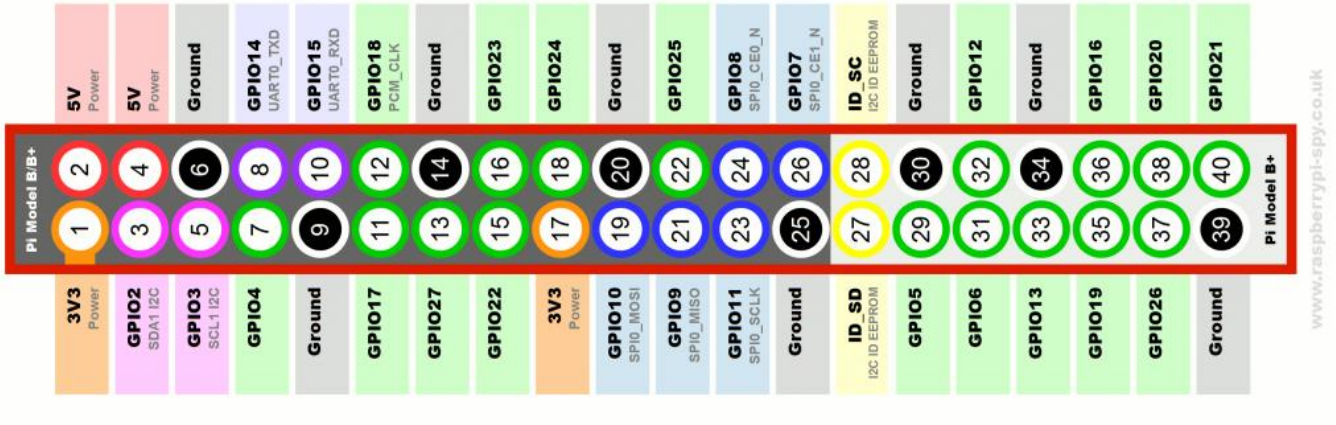
An LED and a potentiometer connected to the RPi's GPIO pins.

- RPi has 2 x 20 **GPIO** (General Purpose Input Output) pins at one edge.
- These pins can be connected to various I/O devices such as LED & push button.
- You can use **breadboard** and **wires** to connect up a circuit.



GPIO pins (cont.)

Power supply, general purpose, T.W.I., U.A.R.T. and S.P.I. pins…

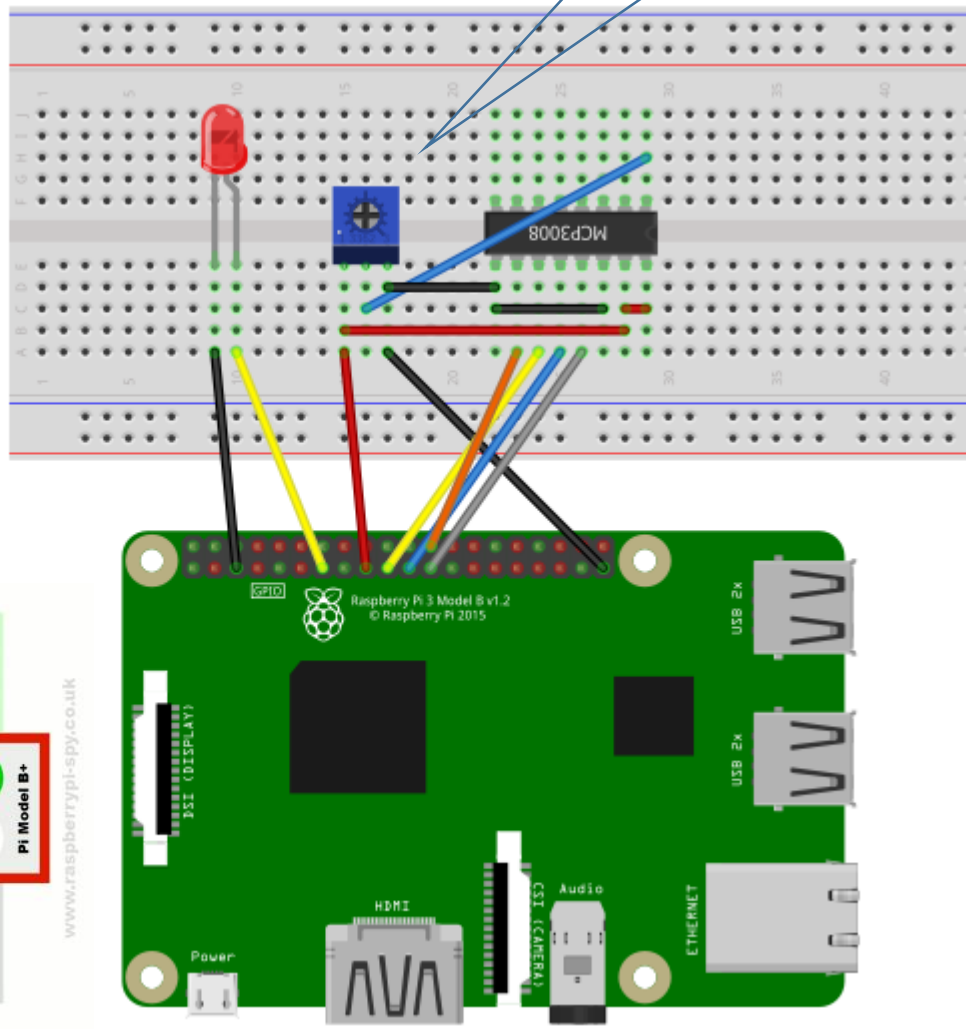


- This diagram shows the 40 GPIO pins.
- **Power supply** pins: Black (ground), Red (5 volt), Orange (3.3 volt).
- **General purpose** pins: Green.
- **T.W.I.** (Two Wire Interface) pins: Pink – more on TWI later.
- **U.A.R.T.** (Universal Asynchronous Receiver Transmitter) pins: Purple.
- **S.P.I.** (Serial Peripheral Interface) pins: Blue – more on SPI later.
- Note that pins are not coloured on the RPi. Identification is by counting and referring to this "**pin diagram**".

GPIO pins (cont.)

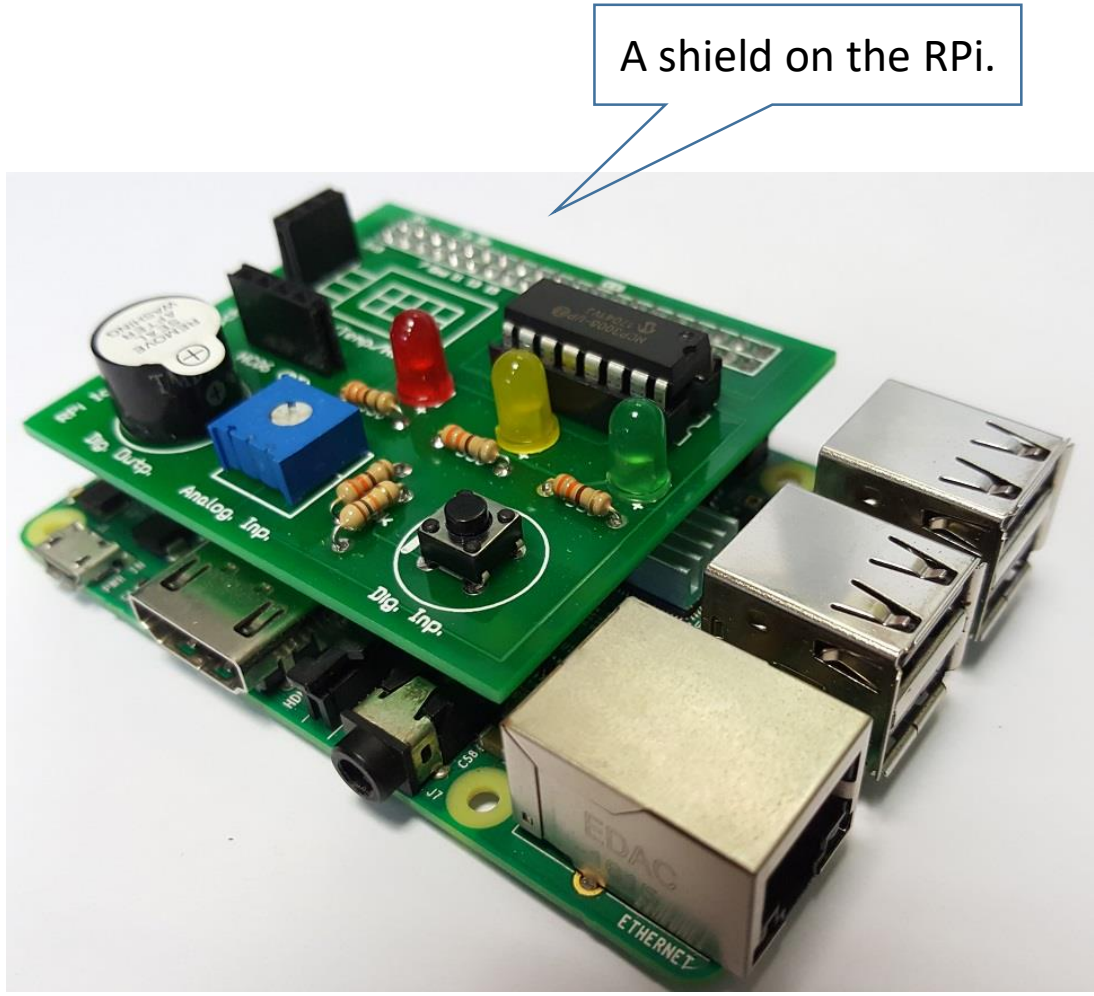
Can you describe how the LED and the IC (Integrated Circuit) chip MCP3008 are connected to the RPi?

How about the potentiometer (trimmer)?



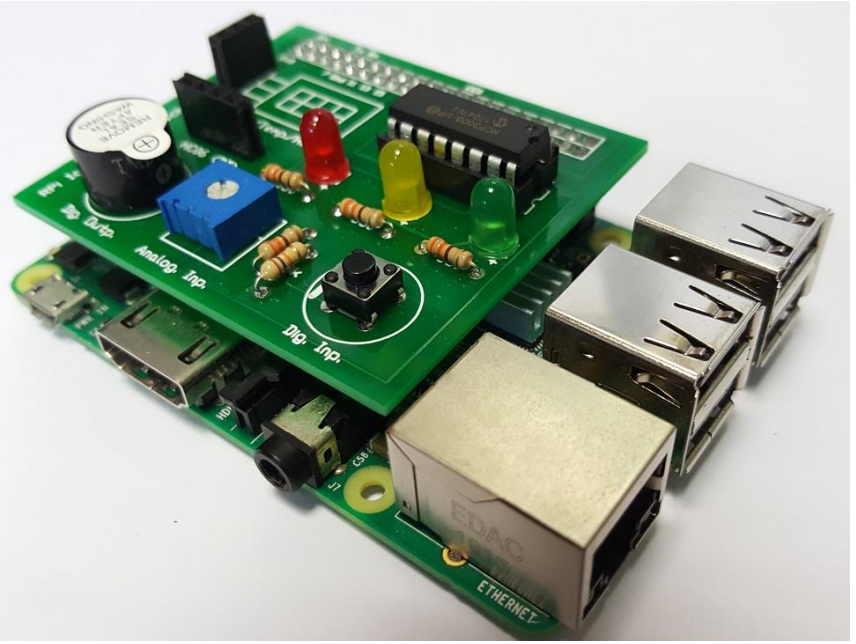
GPIO pins (cont.)

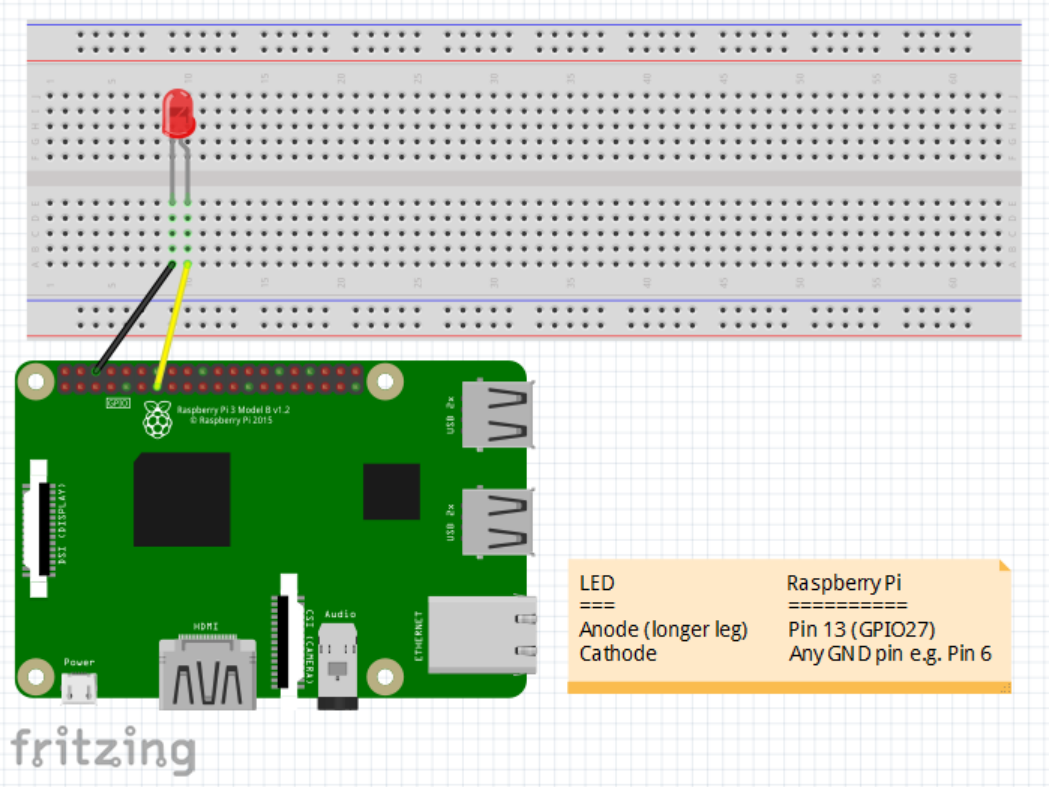
- It is also possible to make a **P.C.B.** (Printed Circuit Board) to connect the RPi's GPIO pins to various I / O devices.
- Such a P.C.B. is often called a "**shield**".
- For implementing the project, you can use:
  - ✓ the provided RPi, I / O devices on the various shields provided, or I / O devices that can be connected to the USB ports.
  - ✓ your own RPi, breadboard, wires & I / O devices.
  - ✓ Emulated I / O devices by "graphical programming" – more on this later.



Blinking an LED

Red LED at pin 27.





- Connect a red LED to the RPi's GPIO **pin 27**, or use the **red LED** on the shield.
- Let's see how a **Python program** can be written to blink this LED on and off.

Blinking an LED (cont.)

BlinkLED.py - /home/pi/BlinkLED.py (3.5.3)

File Edit Format Run Options Window Help

```
import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(27,GPIO.OUT)
while(True):
    GPIO.output(27,1)
    print("LED turns on...")
    sleep(1)
    GPIO.output(27,0)
    print("LED turns off...")
    sleep(1)
```

- Write this Python program, save it as BlinkLED.py and run it.
- What happens?

Press F5 to run it.

Blinking an LED (cont.)

- The program is explained below:

Imports RPi's GPIO module and call that GPIO.

From the time module, import the sleep function.

To get rid of warning, if the GPIO pins you are using have been used before.

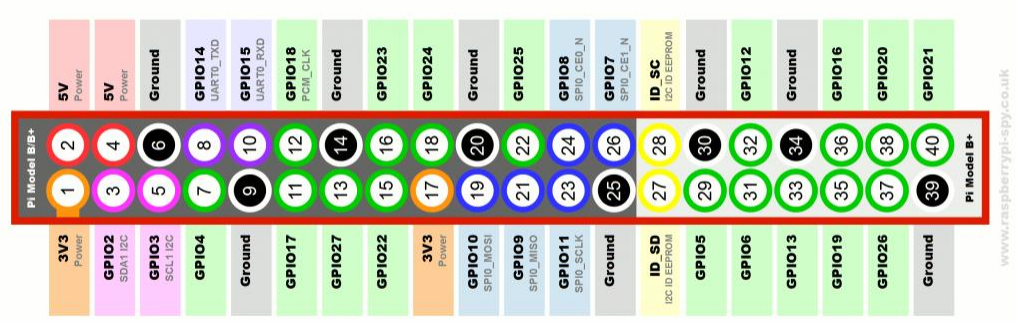
Pin 13 is also GPIO 27.
This line says refer to the pins using their GPIO numbers.

Alternative is:
GPIO.setmode(GPIO.BOARD)

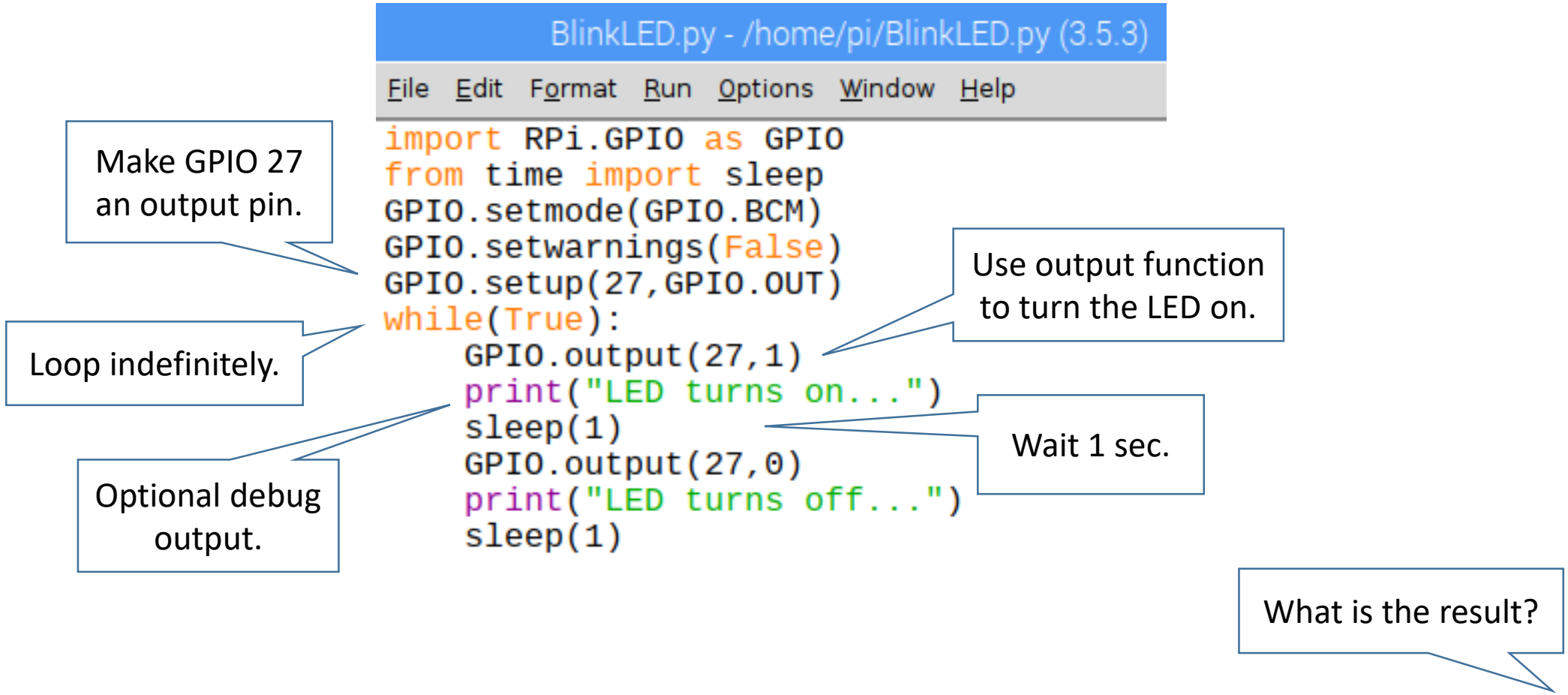
BlinkLED.py - /home/pi/BlinkLED.py (3.5.3)

File Edit Format Run Options Window Help

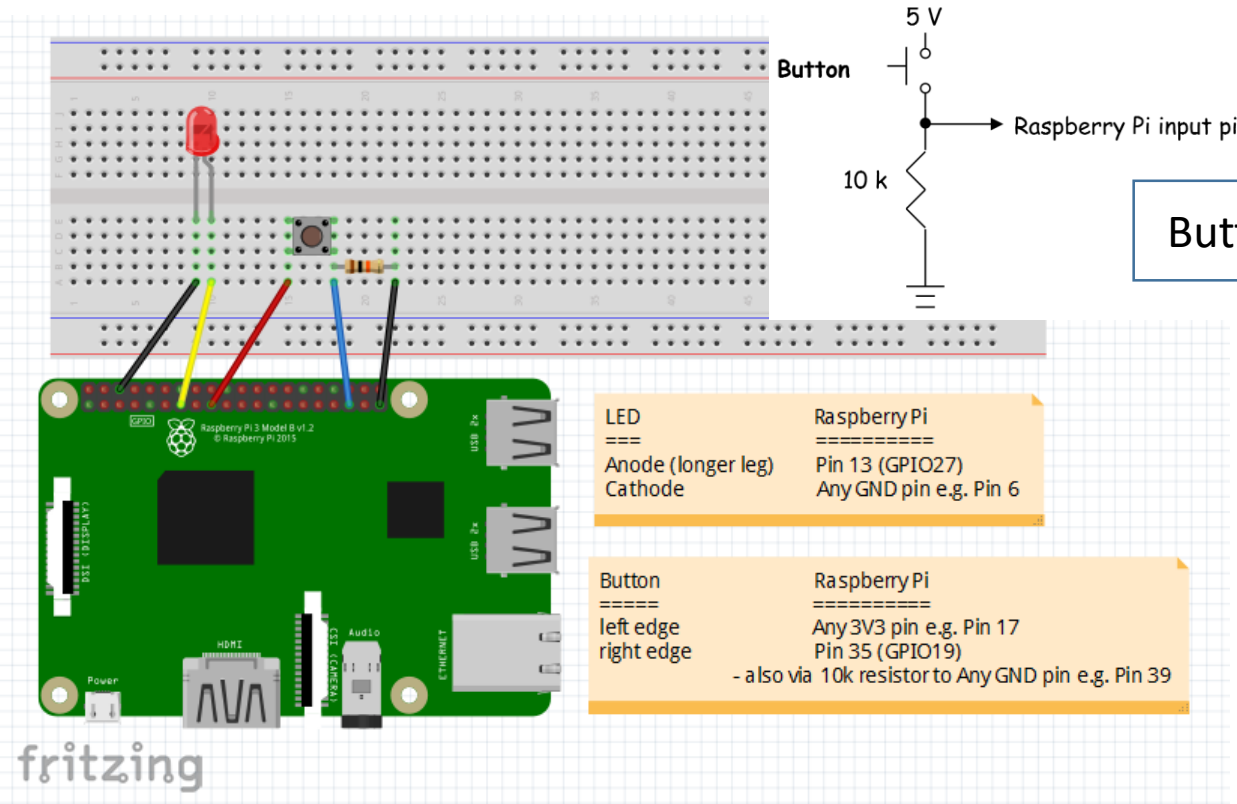
```
import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(27,GPIO.OUT)
while(True):
    GPIO.output(27,1)
    print("LED turns on...")
    sleep(1)
    GPIO.output(27,0)
    print("LED turns off...")
    sleep(1)
```



Blinking an LED (cont.)



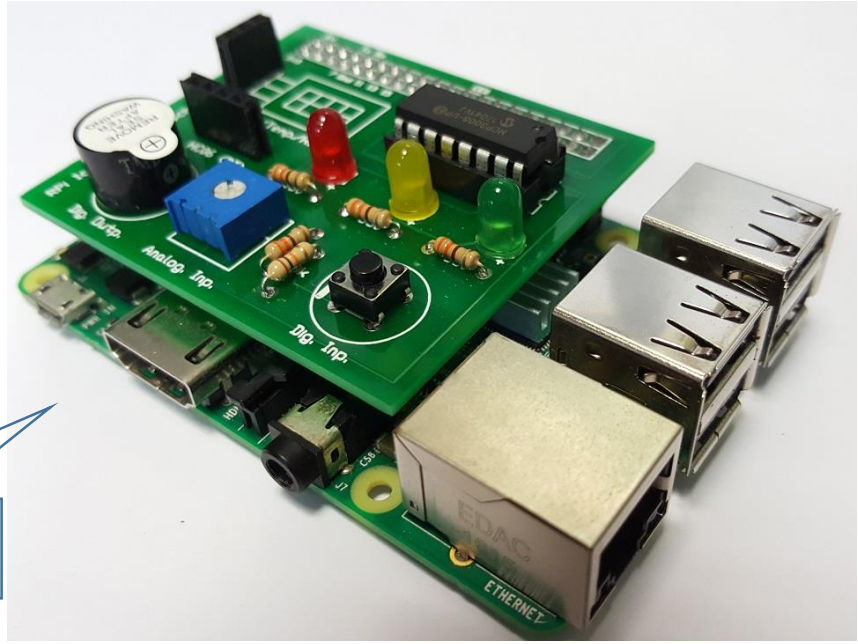
Controlling an LED using a button



LED	Raspberry Pi
Anode (longer leg)	Pin 13 (GPIO27)
Cathode	Any GND pin e.g. Pin 6

Button	Raspberry Pi
left edge	Any 3V3 pin e.g. Pin 17
right edge	Pin 35 (GPIO19)
	- also via 10k resistor to Any GND pin e.g. Pin 39

Button at pin 19.



- Connect a **push button** switch to the RPi's GPIO **pin 19**, or use that on the shield.
- Let's see how a **Python program** can be written to use this button to control the LED on and off.

Controlling an LED using a button (cont.)

Button_LED.py - /home/pi/Button_LED.py (3.5.3)

File Edit Format Run Options Window Help

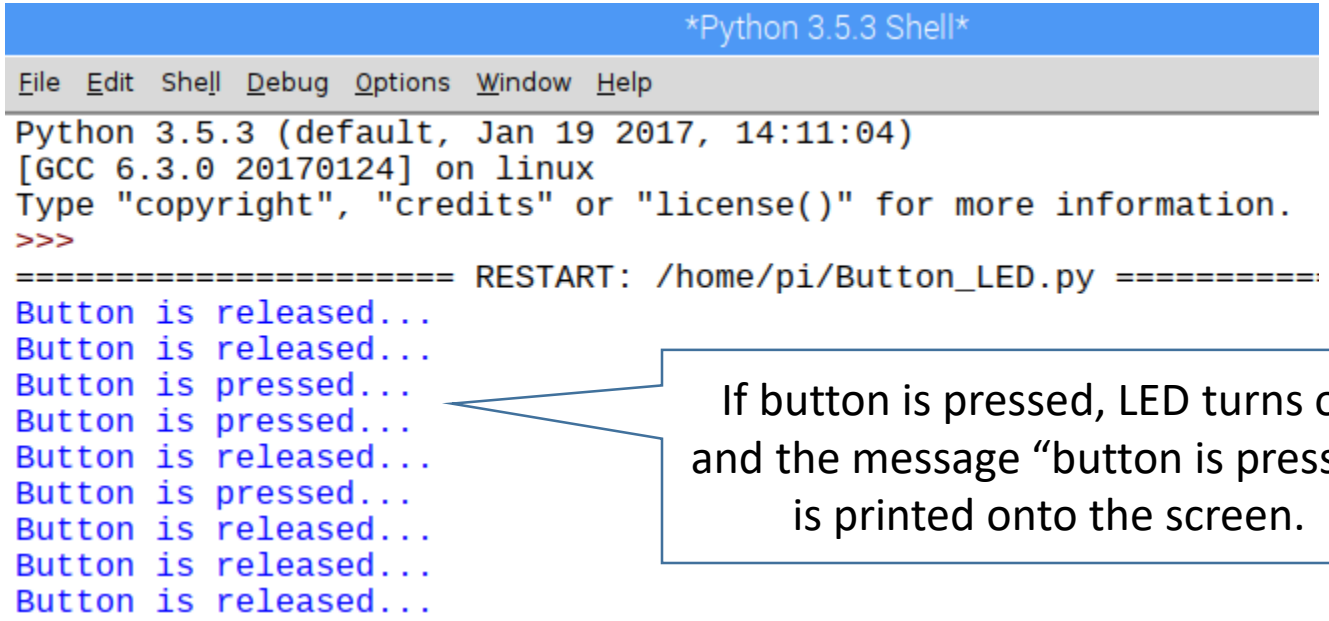
```
import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(27,GPIO.OUT)
GPIO.setup(19,GPIO.IN)
while(True):
    if GPIO.input(19):
        GPIO.output(27,1)
        print("Button is pressed...")
        sleep(1)
    else:
        GPIO.output(27,0)
        print("Button is released...")
        sleep(1)
```

Button pin is input.

If button is pressed, LED turns on. Otherwise, LED turns off.

- Modify the BlinkLED.py program to become this, save it as Button_LED.py and run it.
- What happens?

Controlling an LED using a button (cont.)



If button is pressed, LED turns on, and the message "button is pressed" is printed onto the screen.

Similarly when button is released.

Printing message onto the screen is a useful debugging technique.

Other digital I / O's

- An LED is a **digital output**.
- Other digital outputs are: Buzzer, RGB LED etc.

- A push button switch is a **digital input**.
- Other digital inputs are: slide switch, tilt switch, moisture sensor, PIR motion sensor, IR sensor etc.

Other digital I / O's (cont.)

PIR motion sensor.

Moisture sensor.

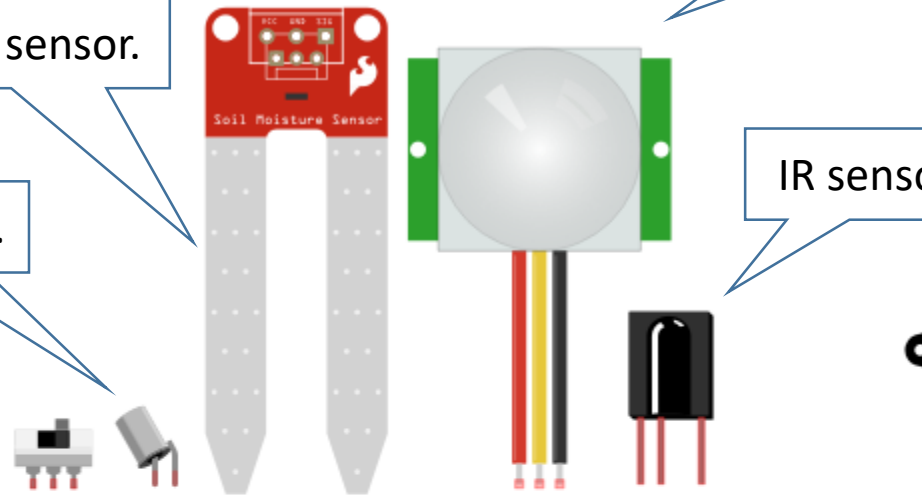
Tilt switch.

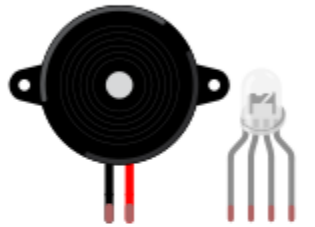
Slide switch.

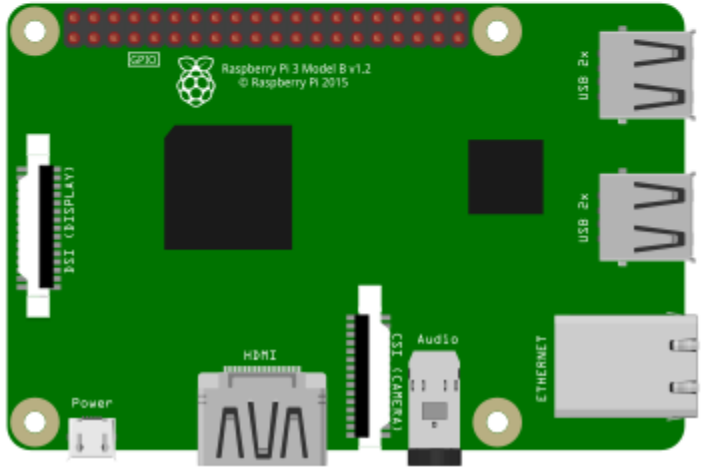
IR sensor.

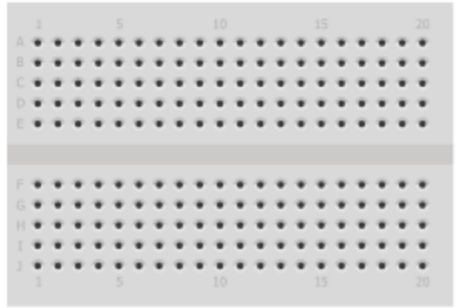
Buzzer.

RGB LED.







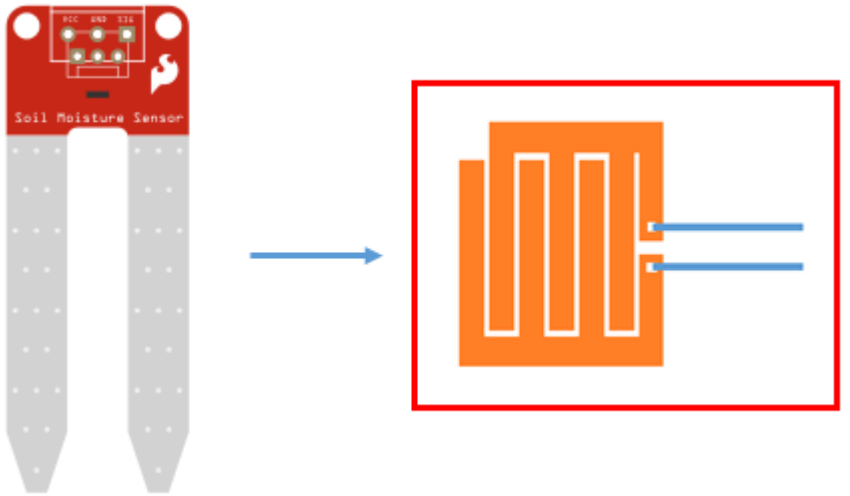


You may want to find out how these work, and what you can use them for.

fritzing

Other digital I / O's (cont.)

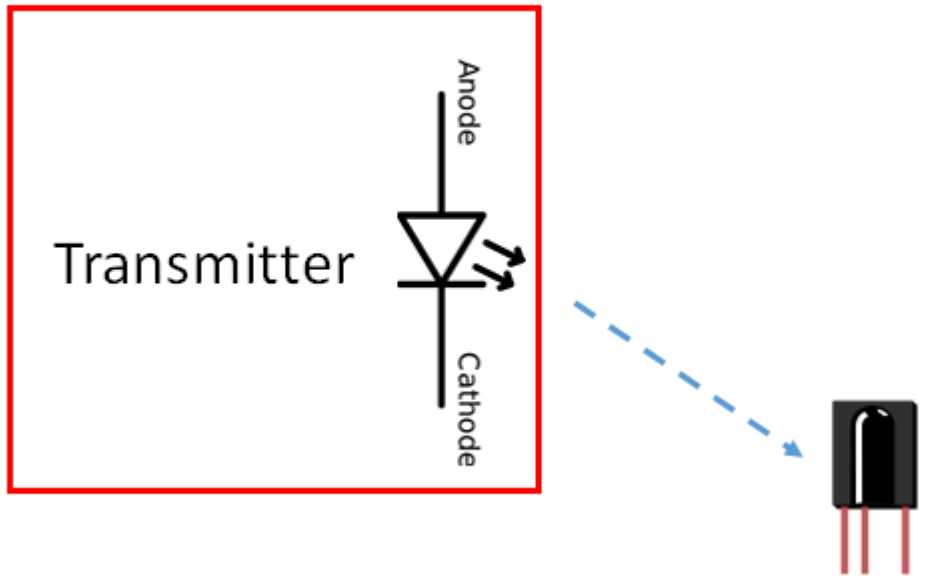
- This is a **moisture sensor**.
- When there is moisture, the 2 copper strips will be shorted.
- So a moisture sensor works as a "**normally open**" switch.



What can you use this for?

Other digital I / O's (cont.)

- This is an **IR sensor**, consisting of a **transmitter** and a **receiver**.
- The receiver is powered up (by 5V and ground connected to 2 of its pins).
- When the IR ray from the transmitter can reach the receiver, the receiver's output pin will give a logic 1 (or HIGH).
- A microcontroller such as RPi can read this output pin.

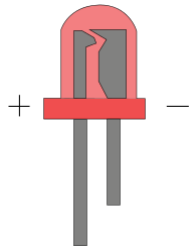


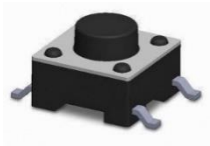
What can you use this for?

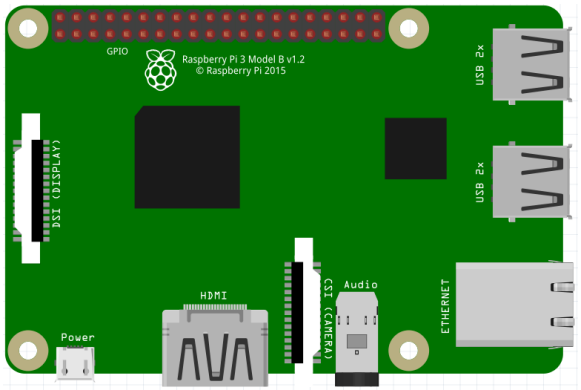
Analogue vs. digital

- What is the difference between a **digital** device and an **analogue** device?

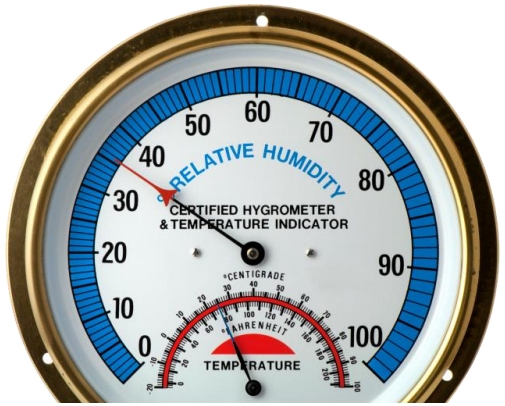
Digital







Analogue



Analogue vs. digital (cont.)

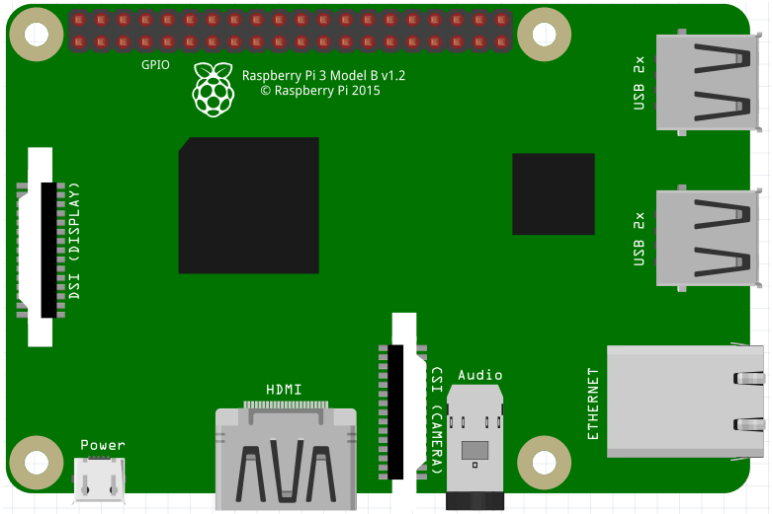
Physical quantity e.g. temperature of 36.9 deg C, is converted into electrical quantity e.g. voltage of 3.69V with the use of a sensor / transducer.

- How can RPi, a digital device, be interfaced to the **physical world**, which is analogue in nature?

Analogue World

Analogue to Digital Converter (ADC)

Digital World

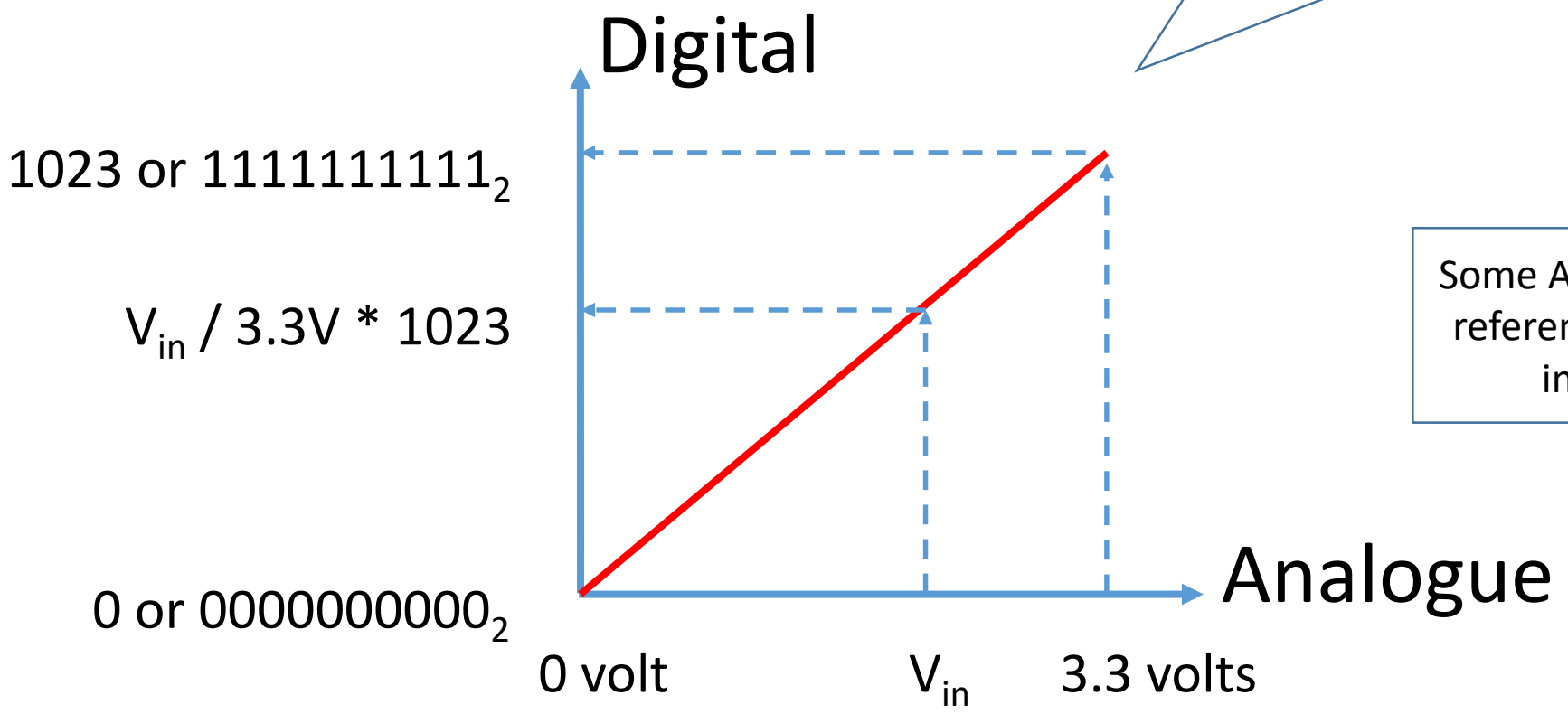


Digital to Analogue Converter (DAC)

Analogue World

Analogue vs. digital (cont.)

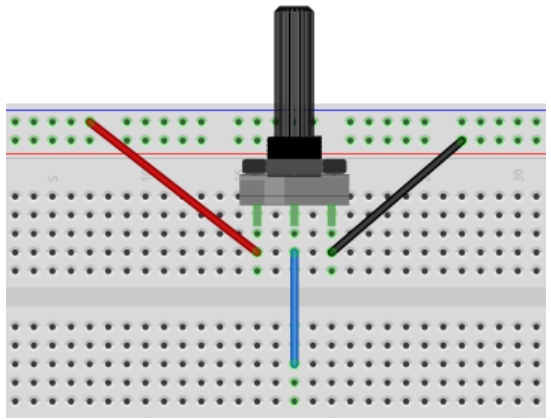
The analogue electrical quantity can then be converted into the corresponding digital representation, a string of 1's and 0's, with the help of an ADC (Analogue to Digital Converter).



Some ADC's use a different reference voltage e.g. 5 V instead of 3.3V.

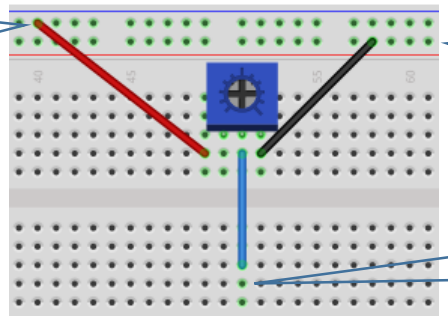
Reading potentiometer value – ADC, SPI

- A **potentiometer** is a **variable resistor**, so connected as to produce a variable voltage, from a range e.g. 0 to 3.3V.



2 common types of variable resistor.

The shield uses this.

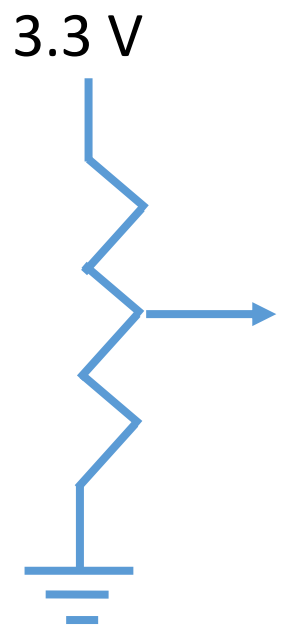


3.3V.

0V.

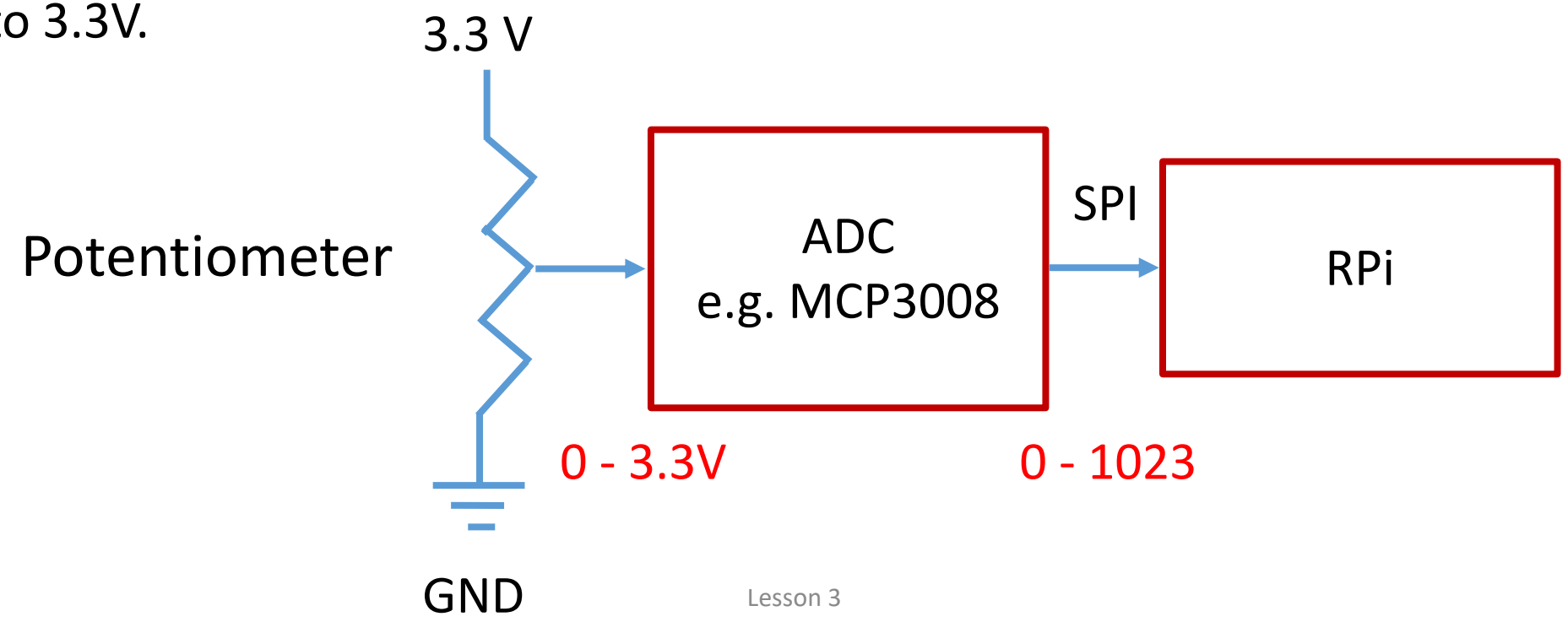
Analogue voltage: 0-3.3V.

Potentiometer



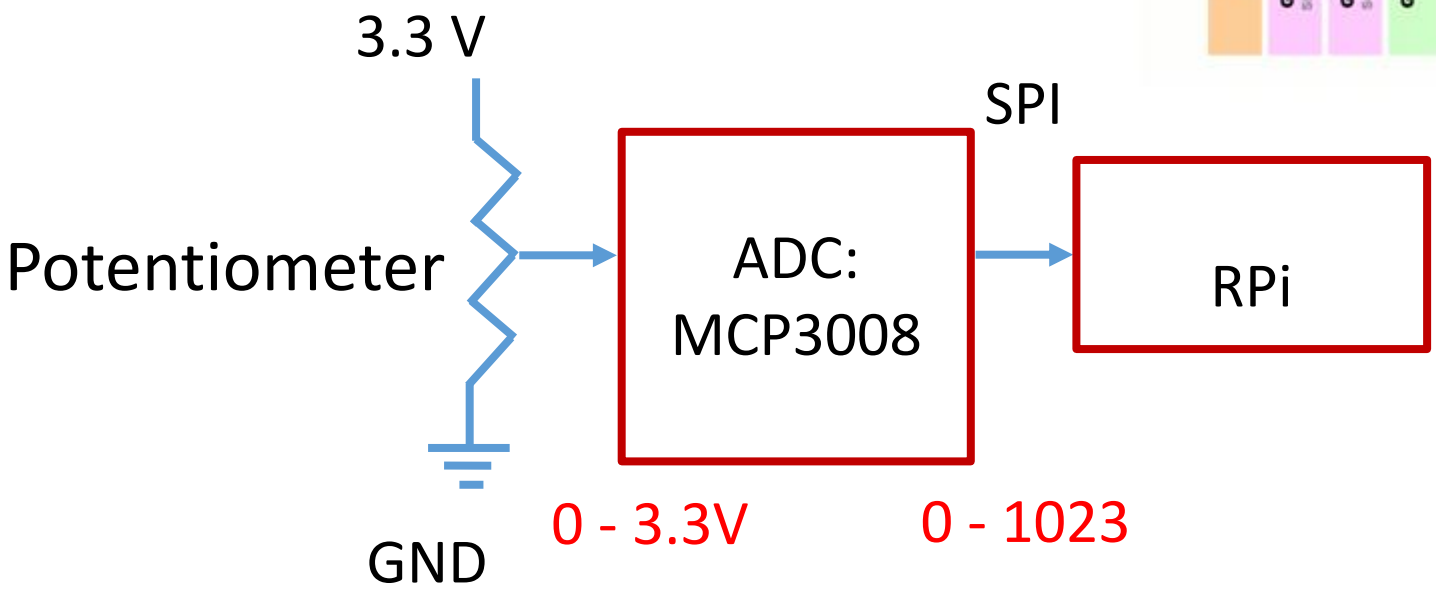
Reading potentiometer value – ADC, SPI (cont.)

- RPi does not have any analogue input pin.
- So to read an analogue voltage input (from the potentiometer), an **ADC** (Analogue to Digital Converter) such as **MCP3008** is required.
- This produces a 10-bit number 0 to 1023, corresponding to the analogue voltage 0 to 3.3V.



Reading potentiometer value – ADC, SPI (cont.)

- The MCP3008 transfers the 10-bit results (of conversion) to the RPi using a serial data transfer method called **SPI** (Serial Peripheral Interface).

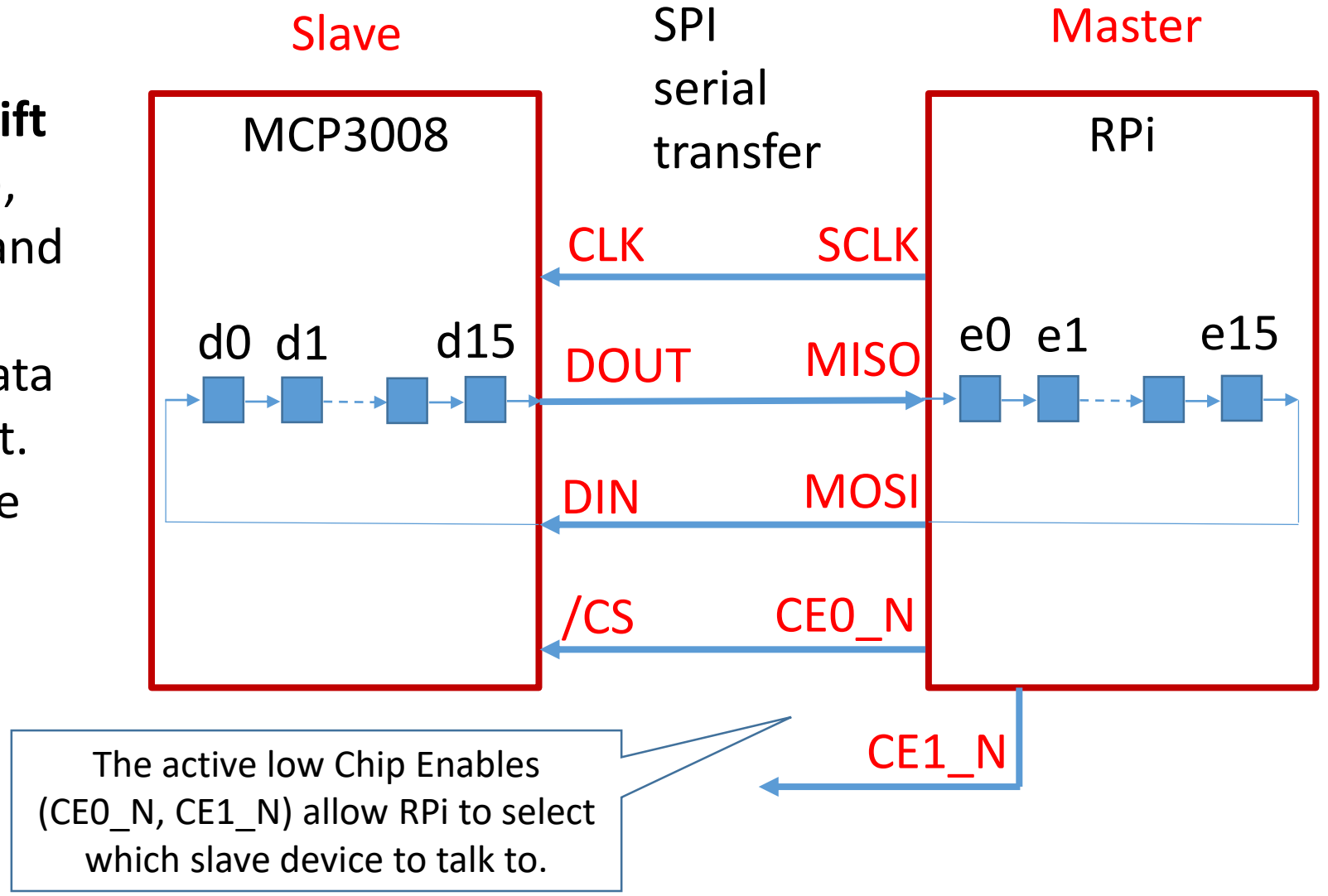


These are the 4 SPI pins:
SCLK (GPIO11), MOSI (GPIO10),
MISO (GPIO9) and CE0_N (GPIO8)

How does SPI work?
See next page…

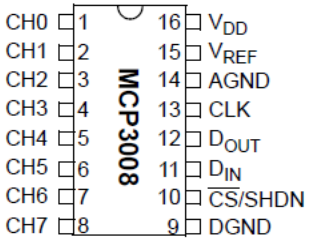
Reading potentiometer value – ADC, SPI (cont.)

- In simple terms, there is a **shift register** (a chain of flip-flops), half in the slave (MCP3008) and half in the master (RPi).
- Each **clock pulse** shifts the data in the shift register by one bit.
- After several clock pulses, the master and slave have **exchanged data**.
- Note that RPi and MCP3008 name the pins differently.

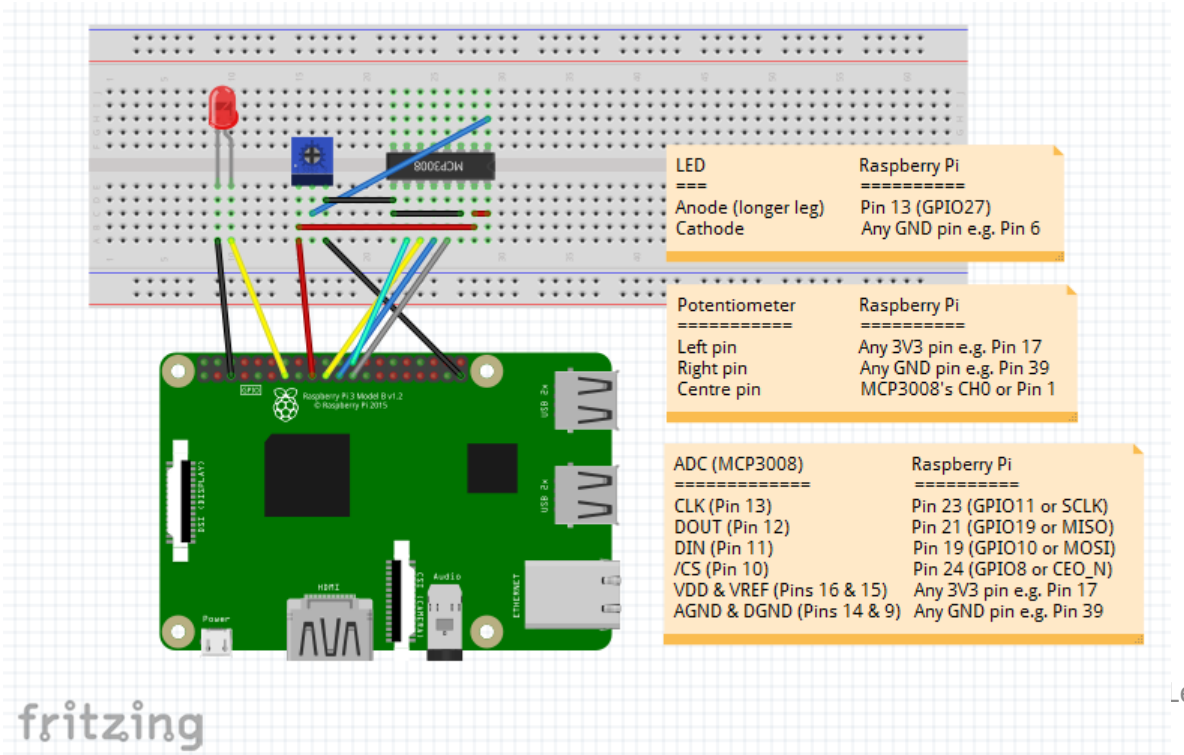


Reading potentiometer value – ADC, SPI (cont.)

- MCP3008 has 8 **channels** CH0 to CH7.
- On the shield, the potentiometer output is connected to CH0.



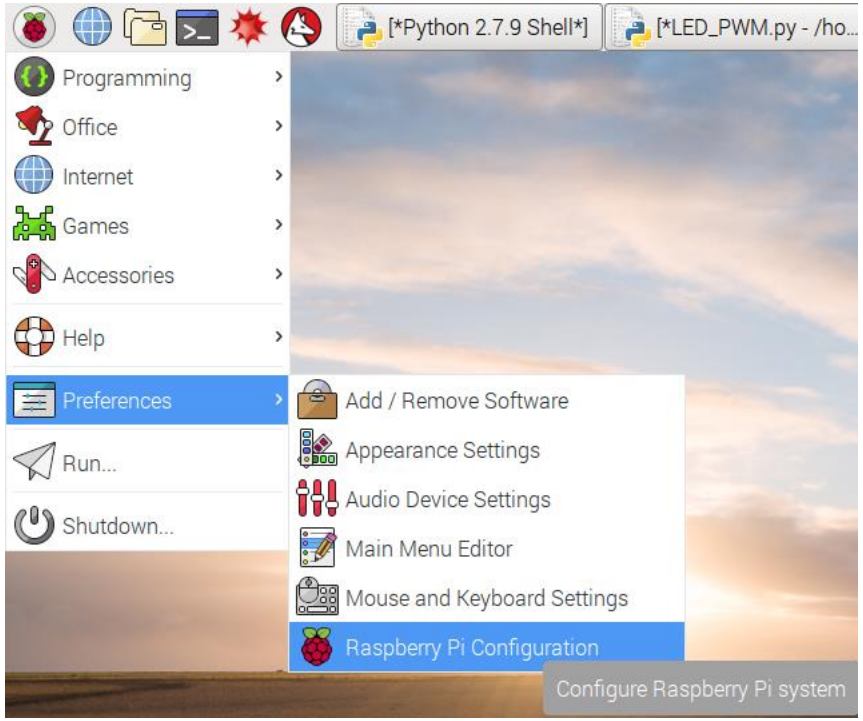
Pin	Symbol	Description
1	CH0	Analog Input
2	CH1	Analog Input
3	CH2	Analog Input
4	CH3	Analog Input
5	CH4	Analog Input
6	CH5	Analog Input
7	CH6	Analog Input
8	CH7	Analog Input
9	DGND	Digital Ground
10	$\overline{CS}$/SHDN	Chip Select/Shutdown Input
11	D_{IN}	Serial Data In
12	D_{OUT}	Serial Data Out
13	CLK	Serial Clock
14	AGND	Analog Ground
15	V_{REF}	Reference Voltage Input
16	V_{DD}	+2.7V to 5.5V Power Supply

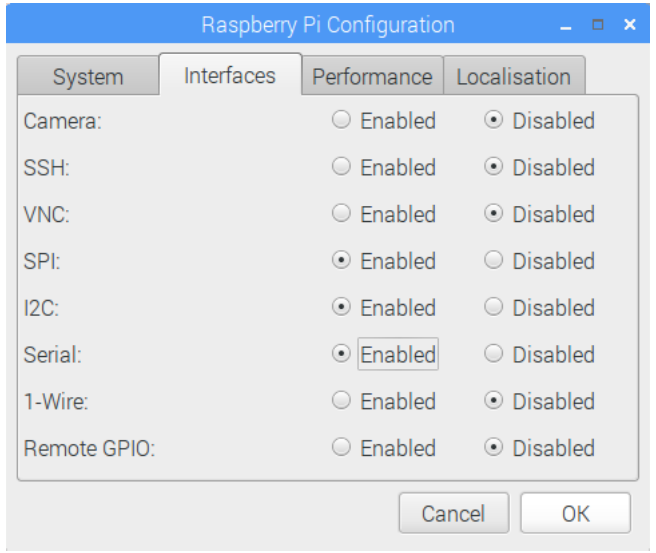


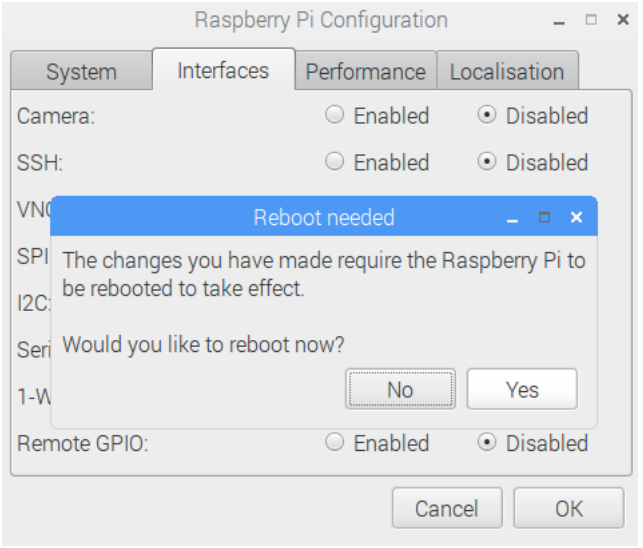
Arrows show the pins connected to the RPi or to the potentiometer.

Reading potentiometer value – ADC, SPI (cont.)

- Let's see how a Python program can be written to read the potentiometer output.
- First, **enable** the SPI interface. You may need to reboot your RPi.

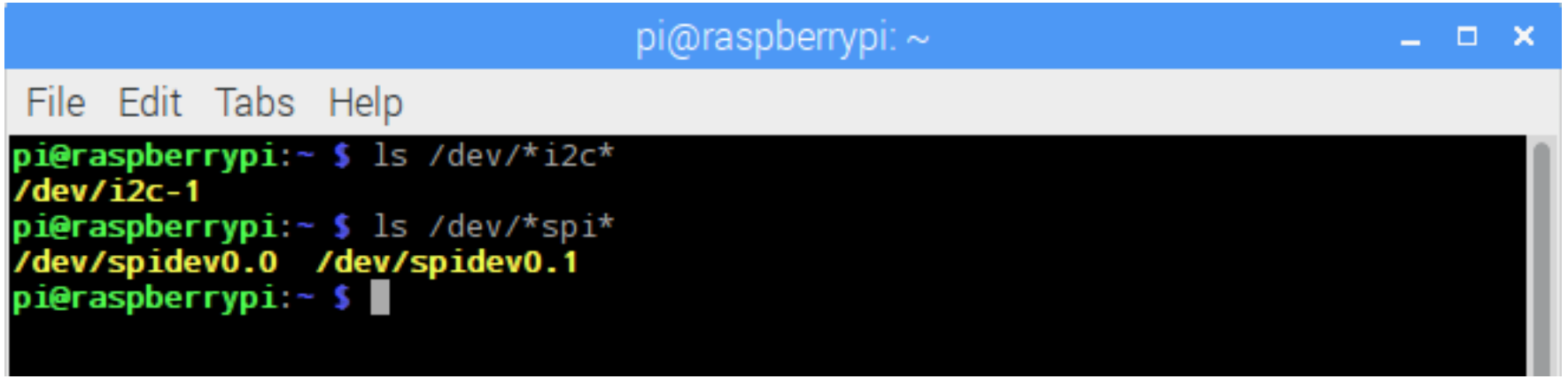






Reading potentiometer value – ADC, SPI (cont.)

- At a terminal, type ls /dev/*spi* to check if SPI has been enabled.



Reading potentiometer value – ADC, SPI (cont.)

The comments help to explain the code.

Pot_ADC_SPI.py - /home/pi/Pot_ADC_SPI.py (3.5.3)

File Edit Format Run Options Window Help

```
import spidev #import SPI library
from time import sleep

my_spi=spidev.SpiDev() #create SPI object
my_spi.open(0,0) #open SPI port 0, device (CS) 0

def read_ADC(ch): #read SPI data from MCP3008 (ADC), 8 channels available
    if ch>7 or ch<0: #invalid channel no
        return -1
    my_spi.max_speed_hz=1350000 #limit SPI speed to 1.35MHz, otherwise ADC cannot cope
    r=my_spi.xfer2([1,8+ch<<4,0]) #construct list of 3 items, and send to ADC
        #1(start),(single-ended+channel no) shifted left 4 bits,0(don't care)
        #see MCP3008 datasheet for details
    data=((r[1]&3)<<8)+r[2] #AND first byte with 3 or 0b00000011 - masking operation
        #shift left 8 bits, then add second byte, to get 10-bit ADC result
    return data

while(True):
    pot_val=read_ADC(0) #read potentiometer value
    print(pot_val)
    sleep(1)
```

- In Python IDLE, enter and save this Python program.

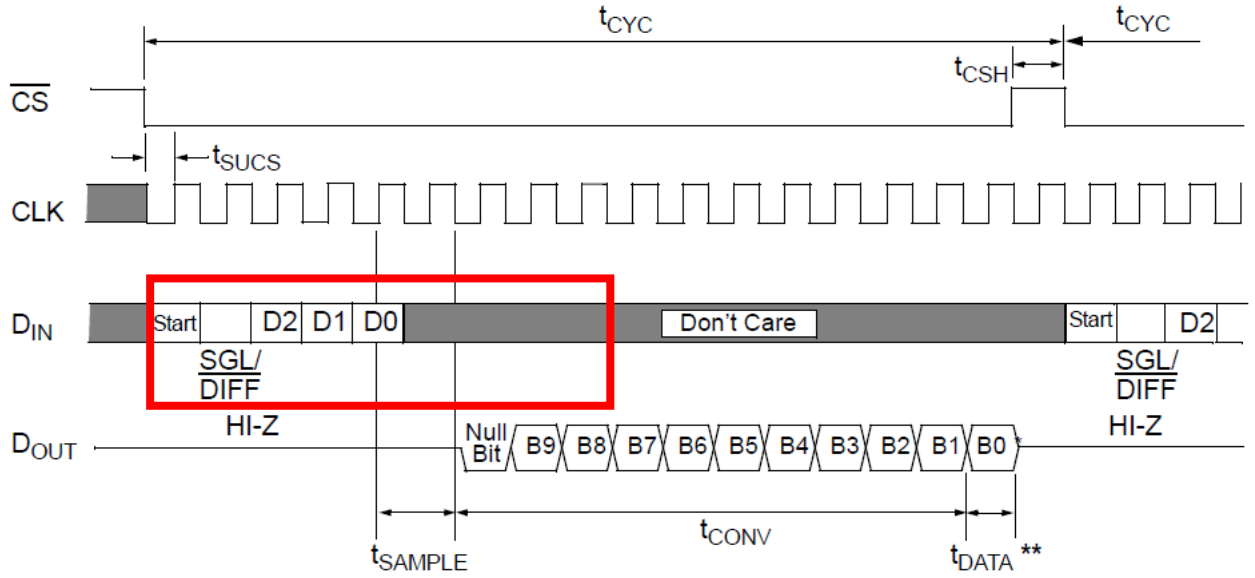
Read up on OOP, to know how a “class” can be used to create an “instance”.

The read_ADC function is defined here.

10-bit result is constructed from 2 bytes of data.

Reading potentiometer value – ADC, SPI (cont.)

Control Bit Selections				Input Configuration	Channel Selection
Single /Diff	D2	D1	D0		
1	0	0	0	single-ended	CH0
1	0	0	1	single-ended	CH1
1	0	1	0	single-ended	CH2
1	0	1	1	single-ended	CH3
1	1	0	0	single-ended	CH4
1	1	0	1	single-ended	CH5
1	1	1	0	single-ended	CH6
1	1	1	1	single-ended	CH7
0	0	0	0	differential	CH0 = IN+ CH1 = IN-
0	0	0	1	differential	CH0 = IN- CH1 = IN+



r = spi.xfer2 ([1, 8 + adcnum << 4, 0])

This line sends:
1, 0b10000000, 0
to MCP3008

0b1000

0b10000000

Start, 0b10000000, don't care

Reading potentiometer value – ADC, SPI (cont.)

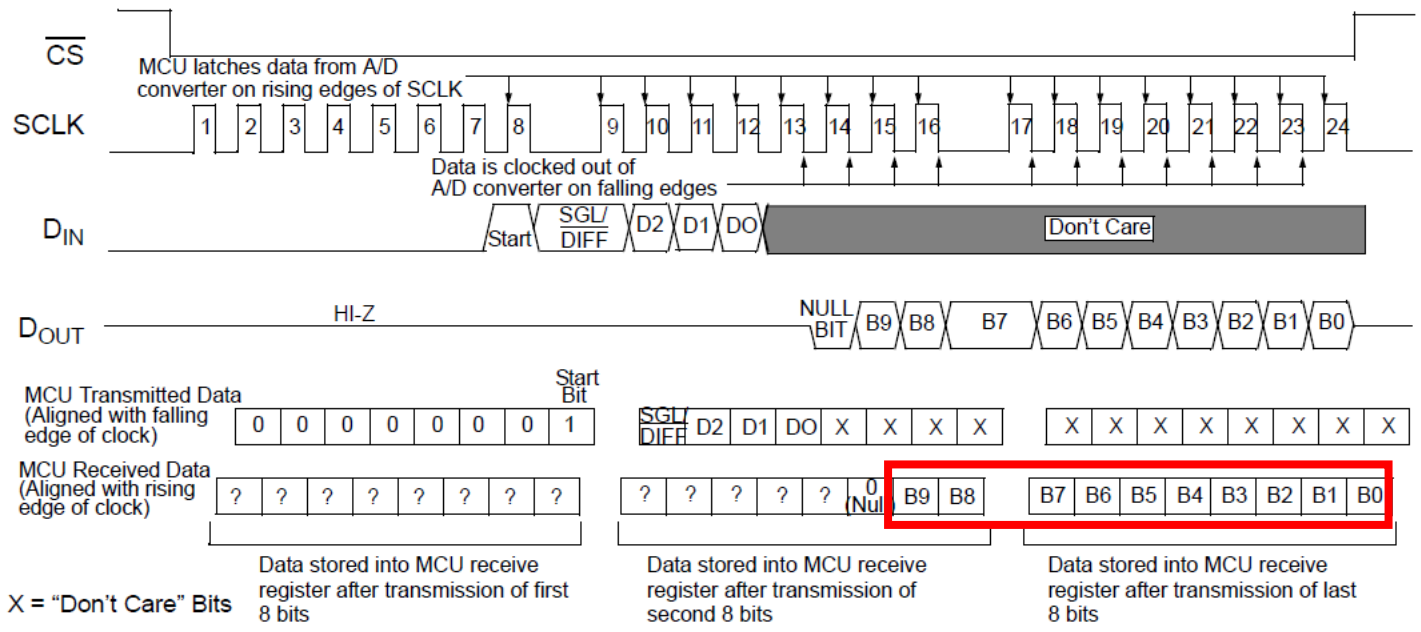
This line constructs a 10-bit ADC result from 2 bytes r[1],r[2].

data = ((r[1] & 3) << 8) + r[2]

AND r[1] with 3 or 0b00000011

Shift result left by 8 bits

OR result with r[2]



Reading potentiometer value – ADC, SPI (cont.)

- When the Python program is run, and the potentiometer turned, you can see 10-bit results (ranging from 0 to 1023) printed on the screen.

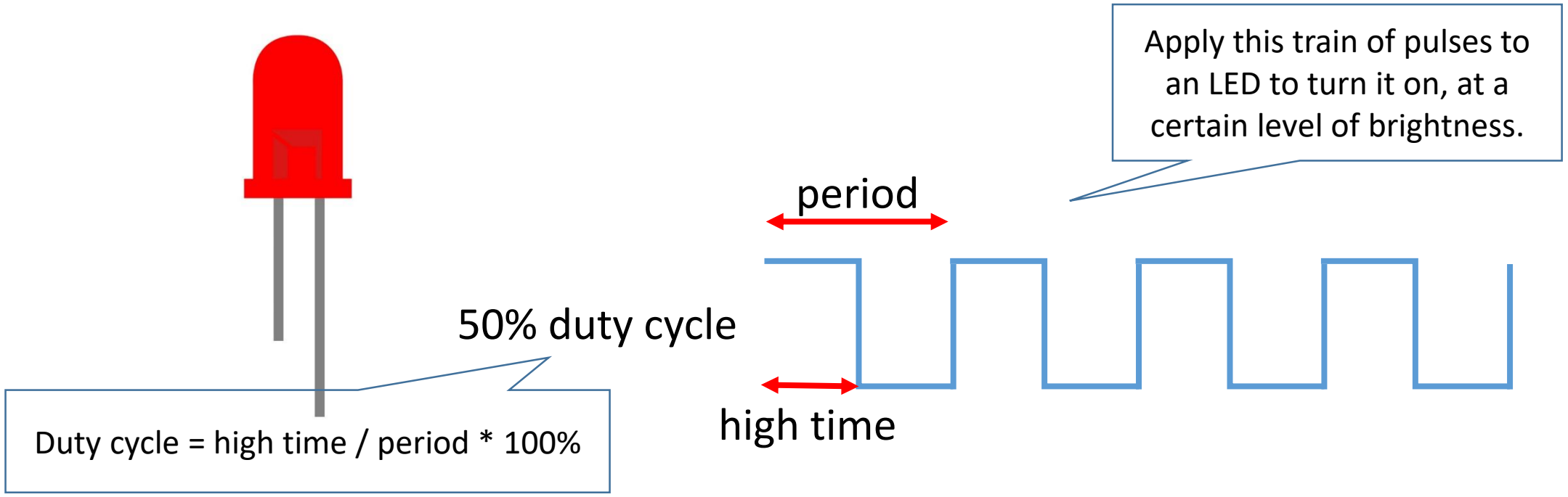
```
*Python 3.5.3 Shell*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: /home/pi/Pot_ADC_SPI.py ==========
284
134
308
443
688
801
```

Challenge: how can the potentiometer and an LED be used to make a dimmer light?

Is LED really a digital output device i.e. “ON” or “OFF” only?

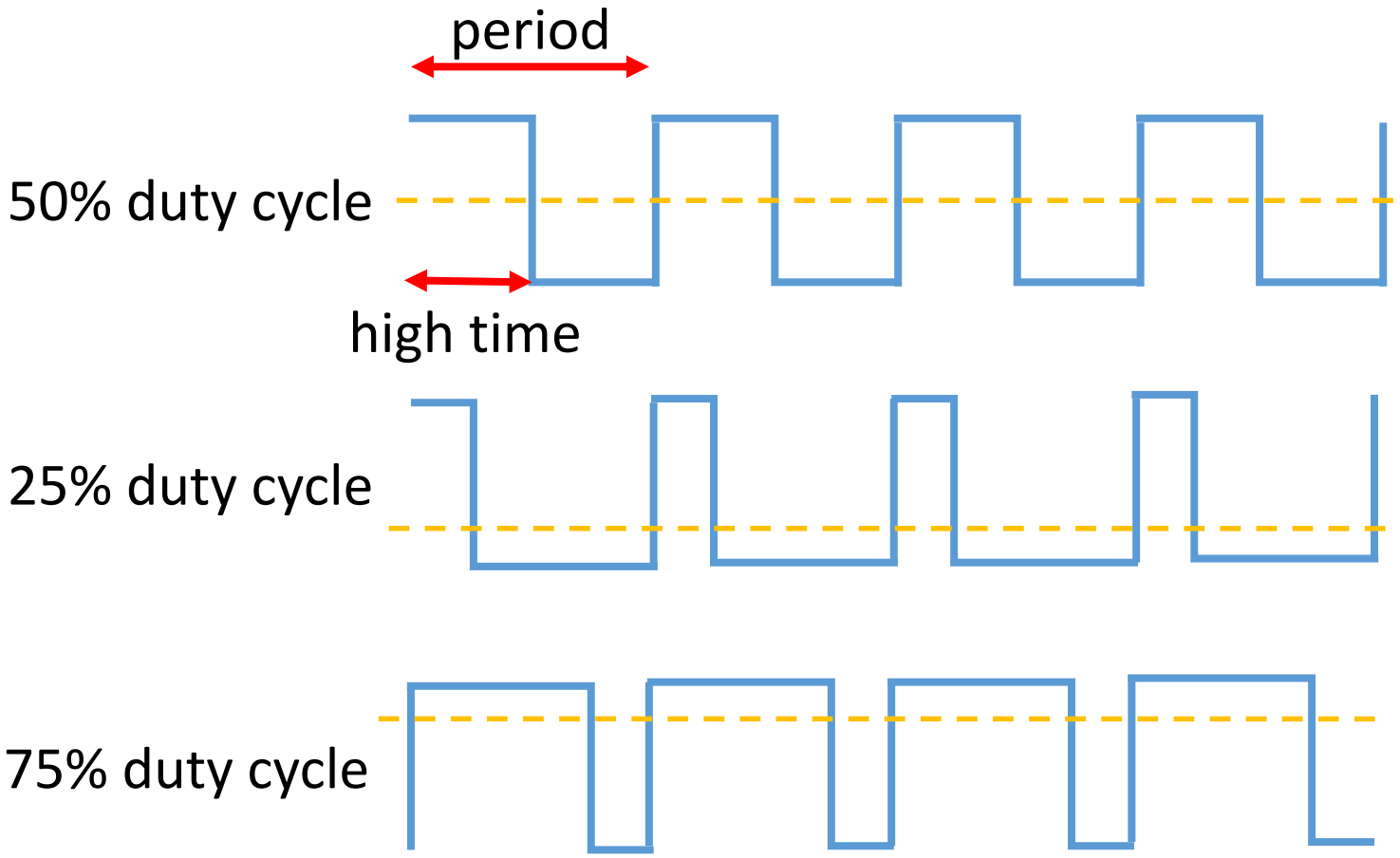
Controlling LED brightness by PWM

- **PWM** (Pulse Width Modulation) is a method to emulate an analogue output, by using a rapidly changing digital signal.
- It can be used to control the brightness of an LED, for instance, or the speed of a DC (Direct Current) motor etc.



Controlling LED brightness by PWM (cont.)

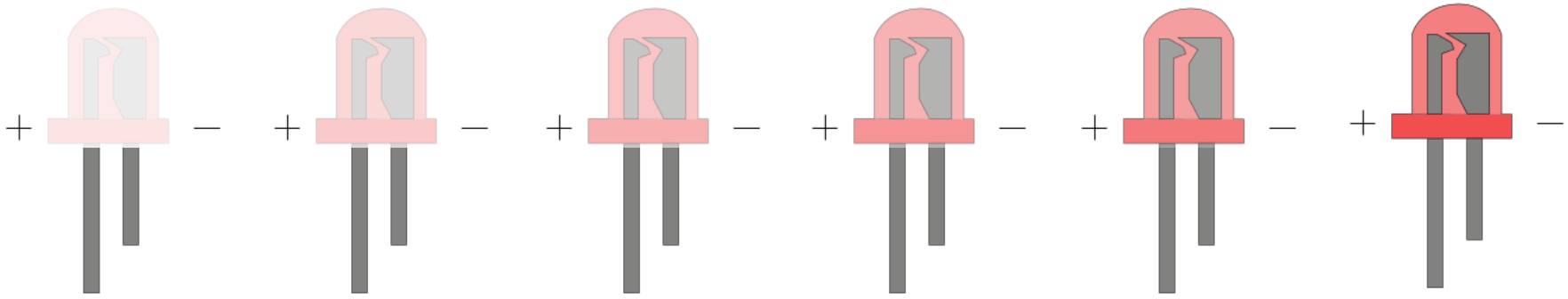
- 25%, 50% & 75% **duty cycle** waveforms produce different **average** output voltages.



Different duty cycles ⇨⇨ different LED brightness
⇨⇨ different DC motor speed
⇨⇨ different servo motor arm position

Controlling LED brightness by PWM (cont.)

- By changing the duty cycle, the LED brightness can be controlled.



Controlling LED brightness by PWM (cont.)

- Let's see how a Python program can be written to vary the LED brightness.

Most lines of code are familiar to you.

LED_PWM.py - /home/pi/LED_PWM.py (3.5.3)

File Edit Format Run Options Window Help

```
import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(27,GPIO.OUT)
my_pwm=GPIO.PWM(27,50) #PWM output at GPIO27, freq=50Hz
while(True):
    for x in range(0,101,20): #duty cycle starts at 0%, end at 100%, in step of 20%
        my_pwm.start(x)
        print('duty cycle:',x)
        sleep(1)
```

The PWM function from the GPIO module is used.

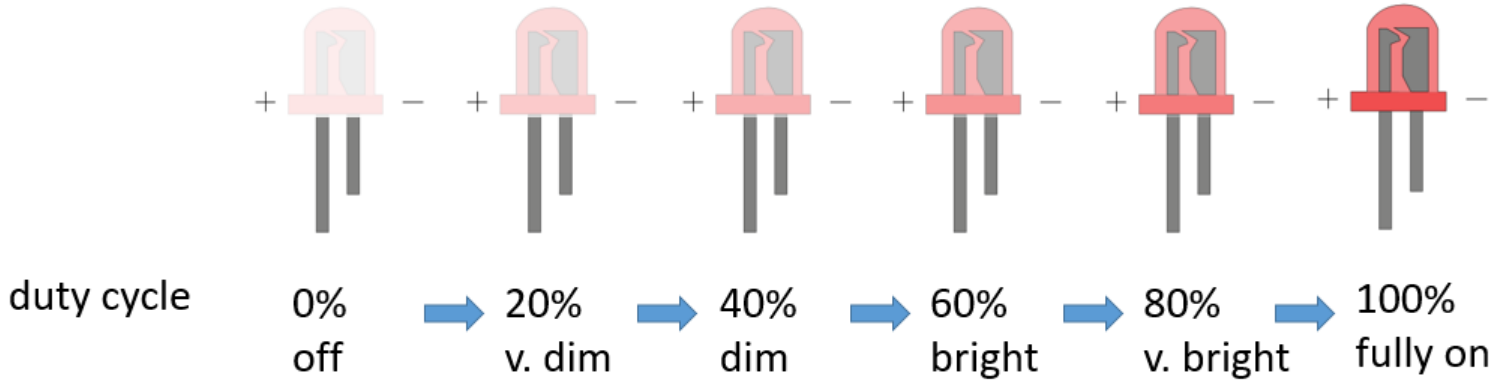
If x = 40, the PWM waveform of 50Hz, 40% (i.e. period=20ms, high time=8ms) will appear at pin 27.

0, 20, 40, 60, 80, 100 i.e. 6 levels of brightness.

Controlling LED brightness by PWM (cont.)

- The debug print shows how the duty cycle changes, as LED goes from off to dim, to bright...

```
*Python 3.5.3 Shell*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: /home/pi/LED_PWM.py ==========
duty cycle: 0
duty cycle: 20
duty cycle: 40
duty cycle: 60
duty cycle: 80
duty cycle: 100
duty cycle: 0
```



Other analogue I / O's

- A potentiometer is an **analogue input**.
- Other analogue inputs are: LDR (Light Dependent Resistor), LM35-based temperature sensor etc.

- An LED with PWM brightness control is (sort of) an **analogue output**.
- Other analogue outputs are: DC (Direct Current) motor (speed), servo motor (arm position), audio etc.

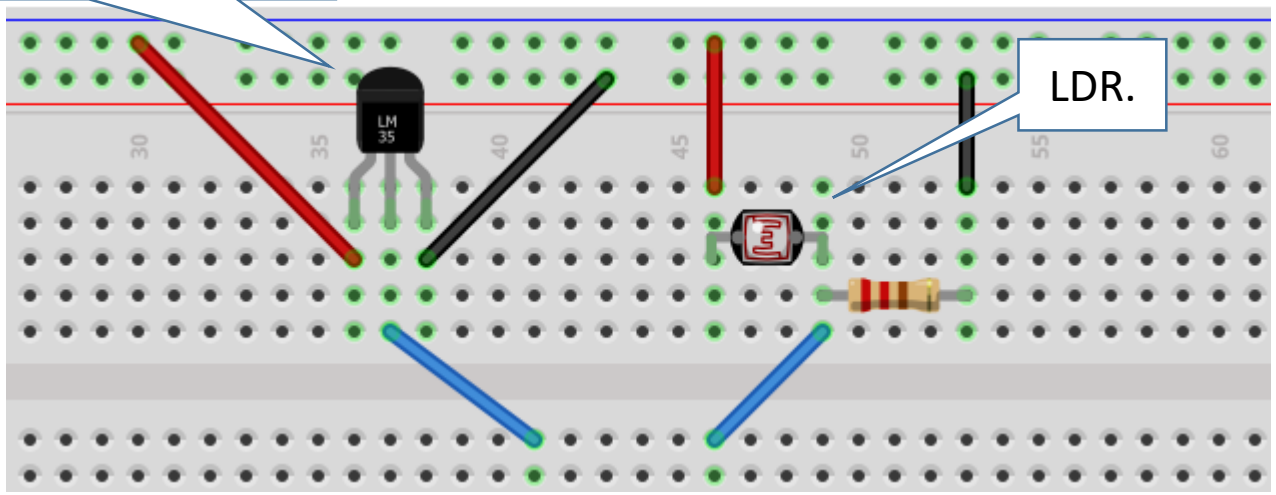
Other analogue I / O's (cont.)

DC motor.

Servo motor.

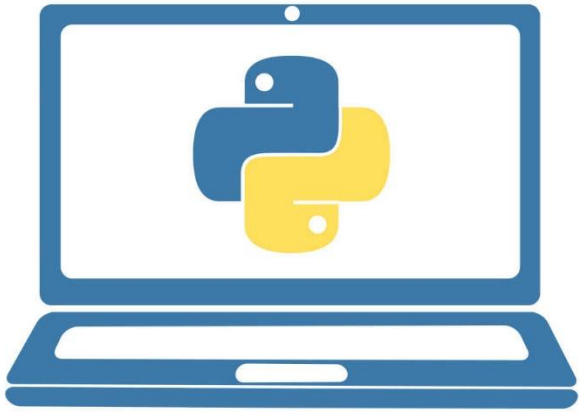
LM35 (temperature sensor).

LDR.



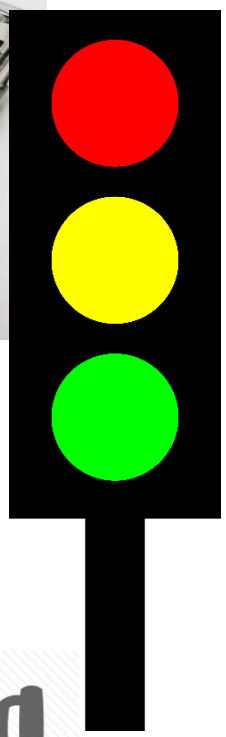
You may want to find out how these work, and what you can use them for.

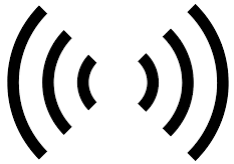
Lab Exercises



- Exercise 3.1 – Traffic light control
- Exercise 3.2 – Dimmer lamp
- Exercise 3.3 – A novel project idea
- Exercise 3.4 – Music, voice, video, camera

Exercise 3.1 – Traffic light control





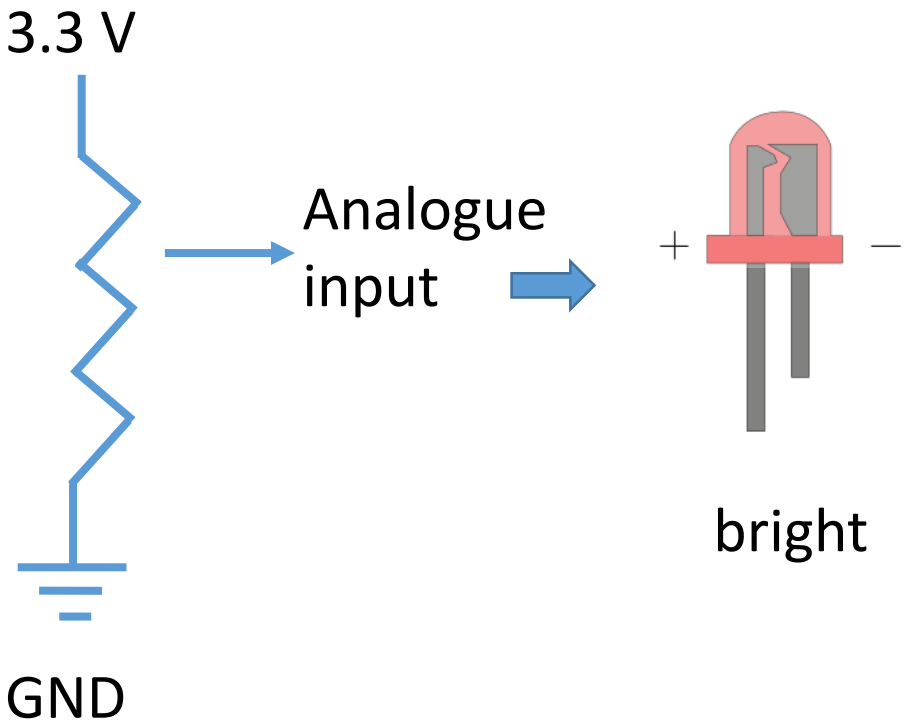
On the shield, the red, yellow, green LEDs are connected to GPIO pins 27, 22, 26 respectively, the push button is connected to GPIO pin 19, while the buzzer is connected to GPIO pin 17.

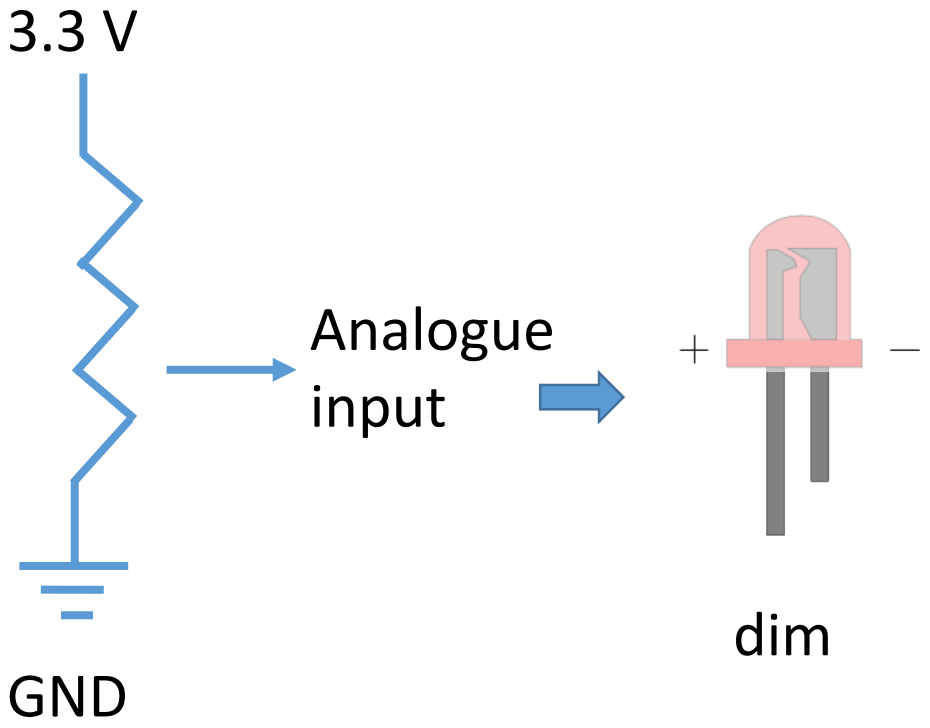
Write a Python program to control the traffic lights.

The green LED should light up for 15 sec, followed by the yellow LED for 2 sec, followed by the red LED for 10 sec. The buzzer should beep (on and off) at 1 sec interval when the red LED is lighting up (- we don't have a "green man" which should light up when the red LED is lighting up). Pressing the button will cause the green LED's time to be reduced to 10 sec.

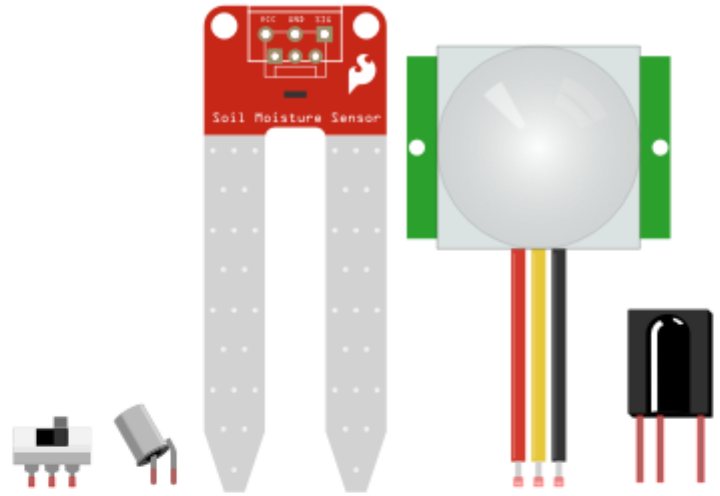
Exercise 3.2 – Dimmer lamp

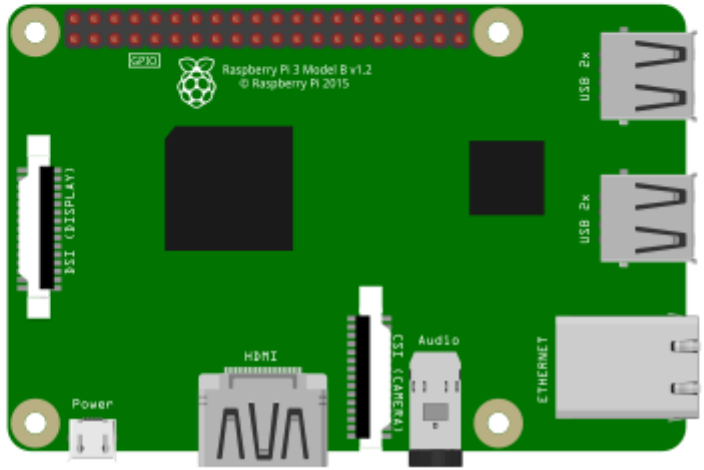
Write a Python program to use the potentiometer to control the red LED's brightness.

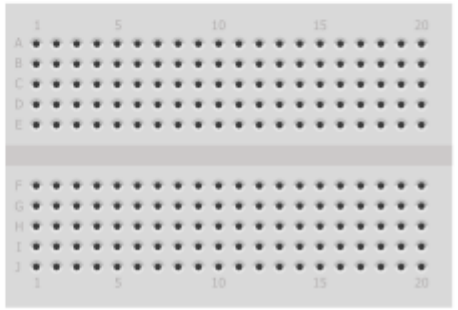




Exercise 3.3 – A novel project idea







Discuss with a classmate to come up with a novel project idea that uses at least 3 of these I / O devices:

- LED
- button
- Buzzer
- RGB LED
- slide switch
- tilt switch
- moisture sensor
- PIR motion sensor
- IR sensor.

Present the idea to the class when you are ready.

fritzing

Exercise 3.4 – Music, voice, video, camera

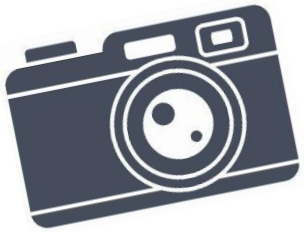
To make your mini-project interesting, you can use music / voice / video playback, or record still pictures (photos) or motion pictures (videos).

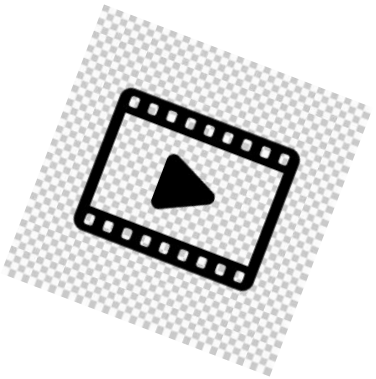
There are tonnes of learning resources out there as RPi & Python are “open sourced”.

Do internet search, and figure out how you can:

1. Convert text to speech.
2. Use Python to play back a music / voice file.
3. Use Python to play back a video file.
4. Use Python to record a still picture.
5. Use Python to record a video clip.

An RPi camera or webcam is needed for these.



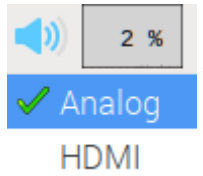


Share what you have learnt with your classmates.

Exercise 3.4 – Music, voice, video, camera (cont.)

Hints:

1. Convert text to speech: http://www.fromtexttospeech.com/
2. Use Python to play back a music / voice file: use the Python code in 3 below, with the correct file name & extension.



You may need to switch to the correct audio output.

3. Use Python to play back a video file:

You can also play back .mp3, .mp4 files.

E1_Ex3_42.py - /home/pi/PythonElectives/E1_Ex3_42.py (3.5.3)

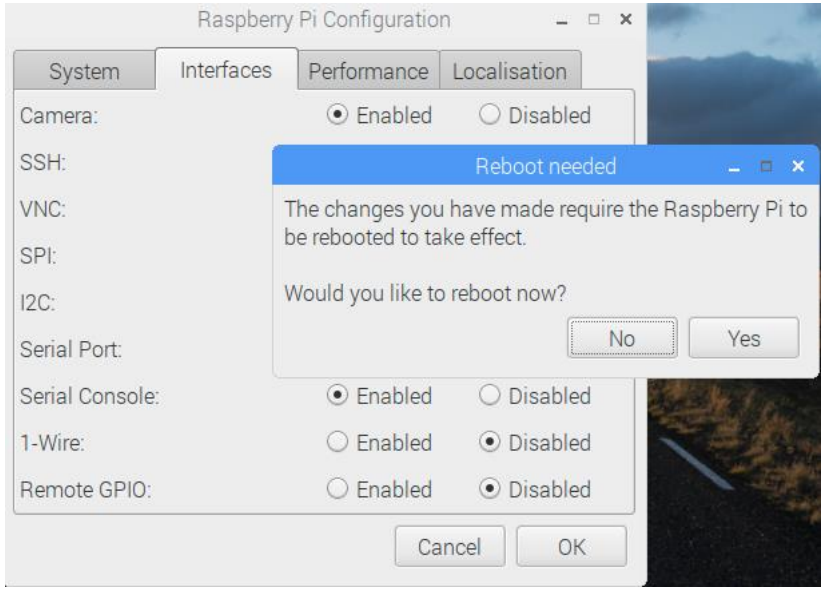
File Edit Format Run Options Window Help

```
import subprocess
my_subprocess=subprocess.Popen(['omxplayer','/home/pi/video.h264'])
```

Exercise 3.4 – Music, voice, video, camera (cont.)

Hints:

To use camera, you need to enable it first:



4. Use Python to record a still picture:

E1_Ex3_44.py - /home/pi/E1_Ex3_44.py (3.5.3)

File Edit Format Run Options Window Help

```
from picamera import PiCamera
from time import sleep

my_camera=PiCamera()
my_camera.resolution=(1920,1080)
my_camera.vflip=True
my_camera.hflip=True

print('taking 5 photos...')

for i in range(5):
    print(i)
    sleep(2)
    my_camera.capture('image{0:03d}.jpg'.format(i))

print('done')
```

This will take 5 photos, named as image000.jpg to image004.jpg.

```
taking 5 photos...
0
1
2
3
4
done
```

Exercise 3.4 – Music, voice, video, camera (cont.)

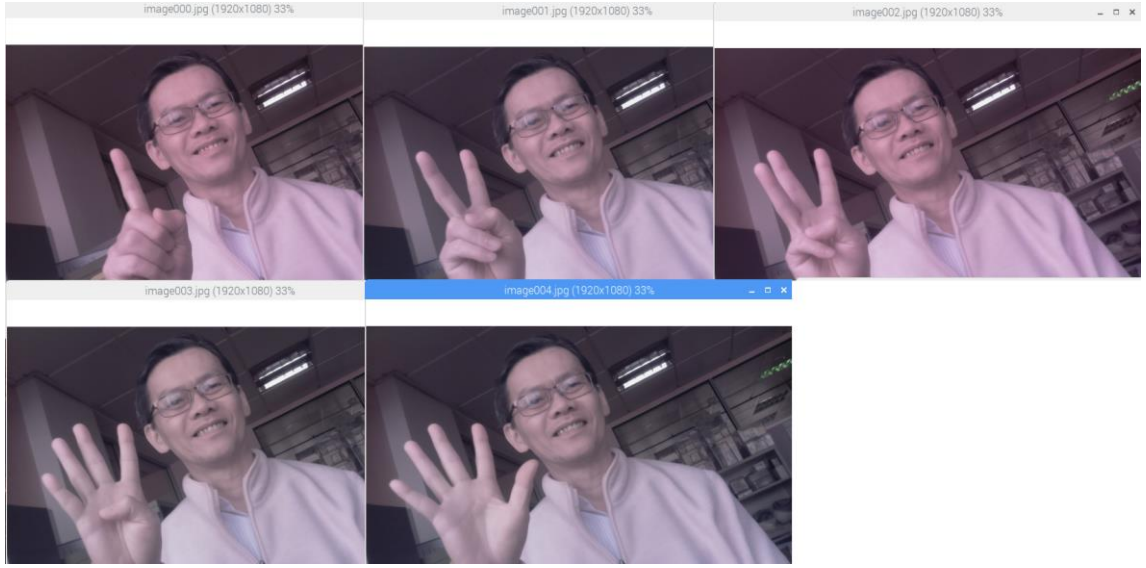


image000.jpg
image001.jpg
image002.jpg
image003.jpg
image004.jpg

Note the “tint” in the photos. This is because an IR (Infra-Red) camera was used. This, together with IR illuminator, allows photos & videos to be taken in darkness.

5. Use Python to record a video clip:

E1_Ex3_45.py - /home/pi/E1_Ex3_45.py (3.5.3)

File Edit Format Run Options Window Help

```
from picamera import PiCamera
from time import sleep

my_camera=PiCamera()
my_camera.resolution=(1920,1080)
my_camera.vflip=True
my_camera.hflip=True

print('taking a 5-sec video clip...')

my_camera.start_preview()
my_camera.start_recording('video.h264')
sleep(5)
my_camera.stop_recording()
my_camera.stop_preview()

print('done')
```

```
taking a 5-sec video clip...
done
```


Thank You!