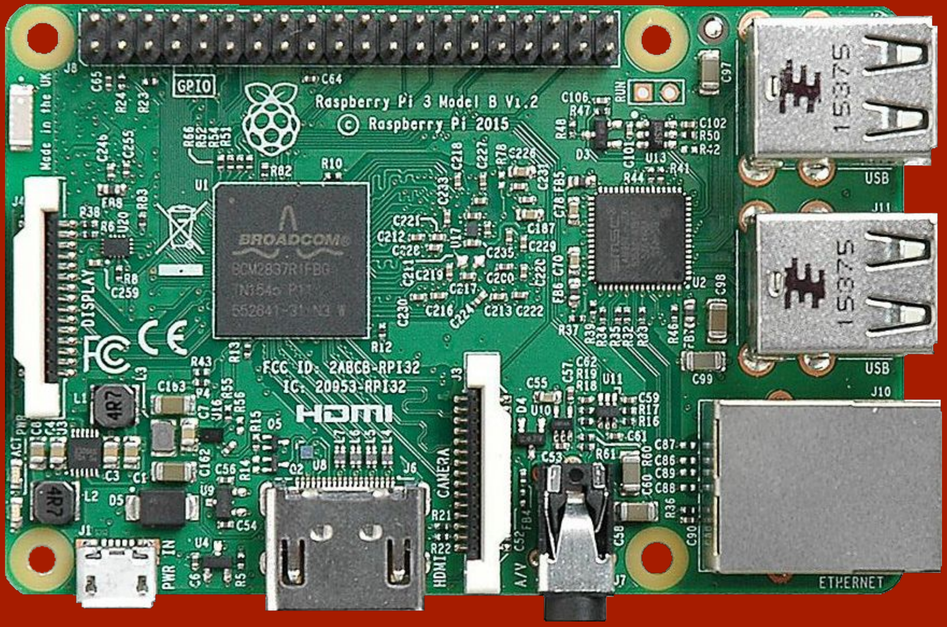
IoT and Practical Applications in Data Science

Raspberry Pi Hardware



TINKERCADEMY

An Infocomm Club Appointed Vendor

Admin

Next class (21 Sept):

You’ll be working on a fun Raspberry Pi project to practice interfacing with all the RPi components.

The following class (28 Sept):

You’ll have time to work on your final projects. In class, each group will be required to meet with me for a short 10-15min consultation regarding your plans for the final project. This will be part of your final probject grade! Come prepared with a plan!

Final Project Details

So far throughout the class, we've covered:

- an introduction to IoT
- how to upload data to a server
- how to handle data in python
- how to create visualisations from that data
- how to connect components to the Raspberry Pi

The final project will have a strong focus on hardware, but will combine all of this knowledge.

Final Project: Weather Station

Groups of 2-3, let me know via Slack

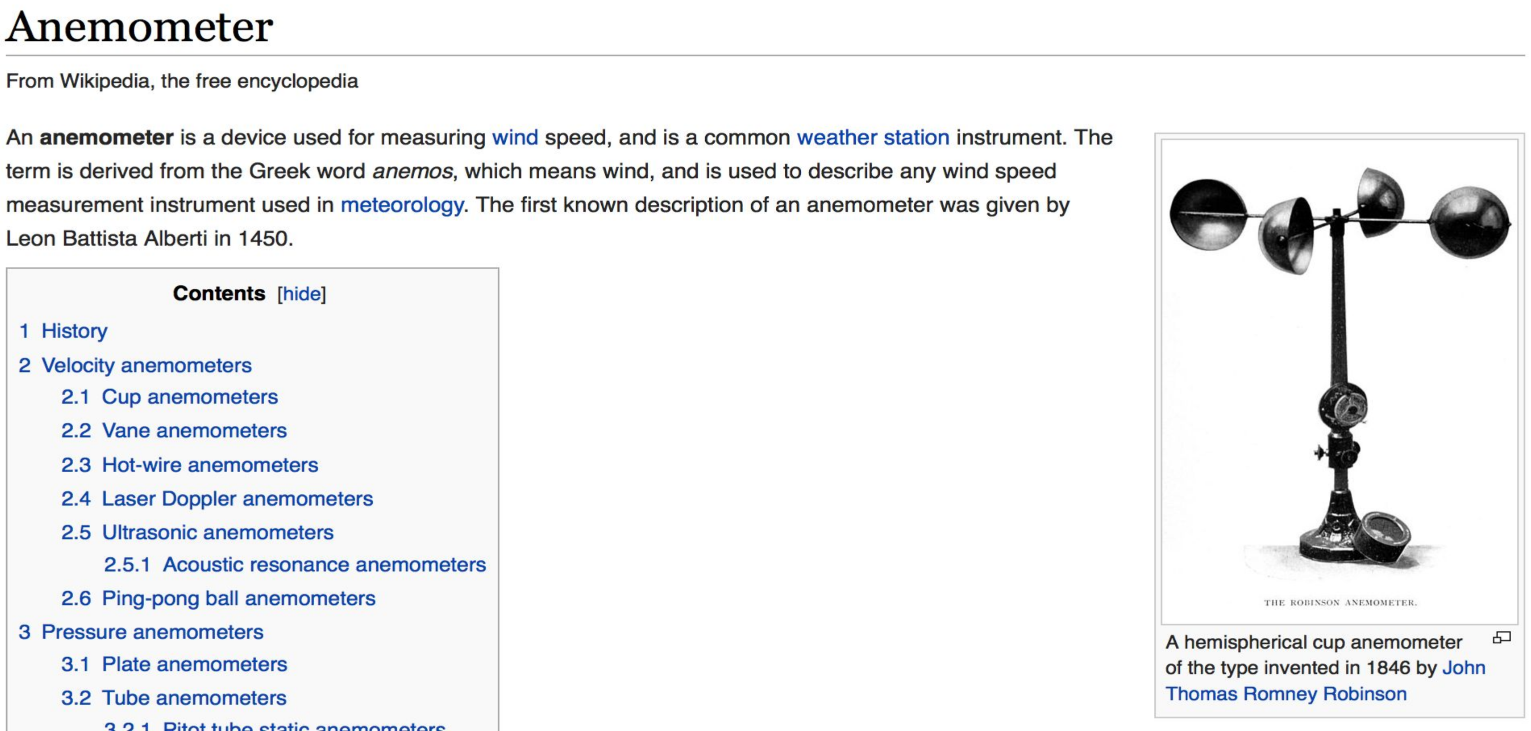
(otherwise I'll randomly assign)

You will be building your own weather station using the Raspberry Pi and collecting weather data.

Final Project Details

Creating a Weather Station

We'll learn today how to use the DHT sensor to capture temperature and humidity, but what about wind speed? You'll need to build your own anemometer!



Anemometer

From Wikipedia, the free encyclopedia

An **anemometer** is a device used for measuring wind speed, and is a common weather station instrument. The term is derived from the Greek word *anemos*, which means wind, and is used to describe any wind speed measurement instrument used in meteorology. The first known description of an anemometer was given by Leon Battista Alberti in 1450.

Contents [hide]

THE ROBINSON ANEMOMETER.

A hemispherical cup anemometer of the type invented in 1846 by John Thomas Romney Robinson

Final Project Details

Creating an anemometer:

This is where your creativity comes in. Think about the components that we've used on the RPi and what you can do with them. It will need to connect to the Raspberry Pi and be able to capture wind speed data (somewhat) accurately.

Note that even once you build an anemometer, you'll have no easy way to convert your data into a common measurement (i.e. km/h). This is okay, make up your own units for this!

Final Project Details

After your hardware is built, you'll need to:

- **Collect weather data**
- **Store it on a server**
- **Download it in CSV format and import into Python**
- **Create data visualisations**
- **Present your project, explain your data, your reasoning for your anemometer build, etc.**

Final Project Details

Additionally:

Final Project Consultations:

During class on 27 Sept, all groups will be working on their final projects while I meet with each group for 10-15 minutes to discuss the plans, problems, and progress of their final project. Please come prepared with any questions you might have, as well as your anemometer plans and ideas for your weather station enhancements.

Final Project Details

Finally:

Peer Ratings:

During final project presentations, you will be rating your classmates on the quality of their:

- Hardware
- Visualisations
- Creativity
- Presentation

Raspberry Pi as an IoT Device?

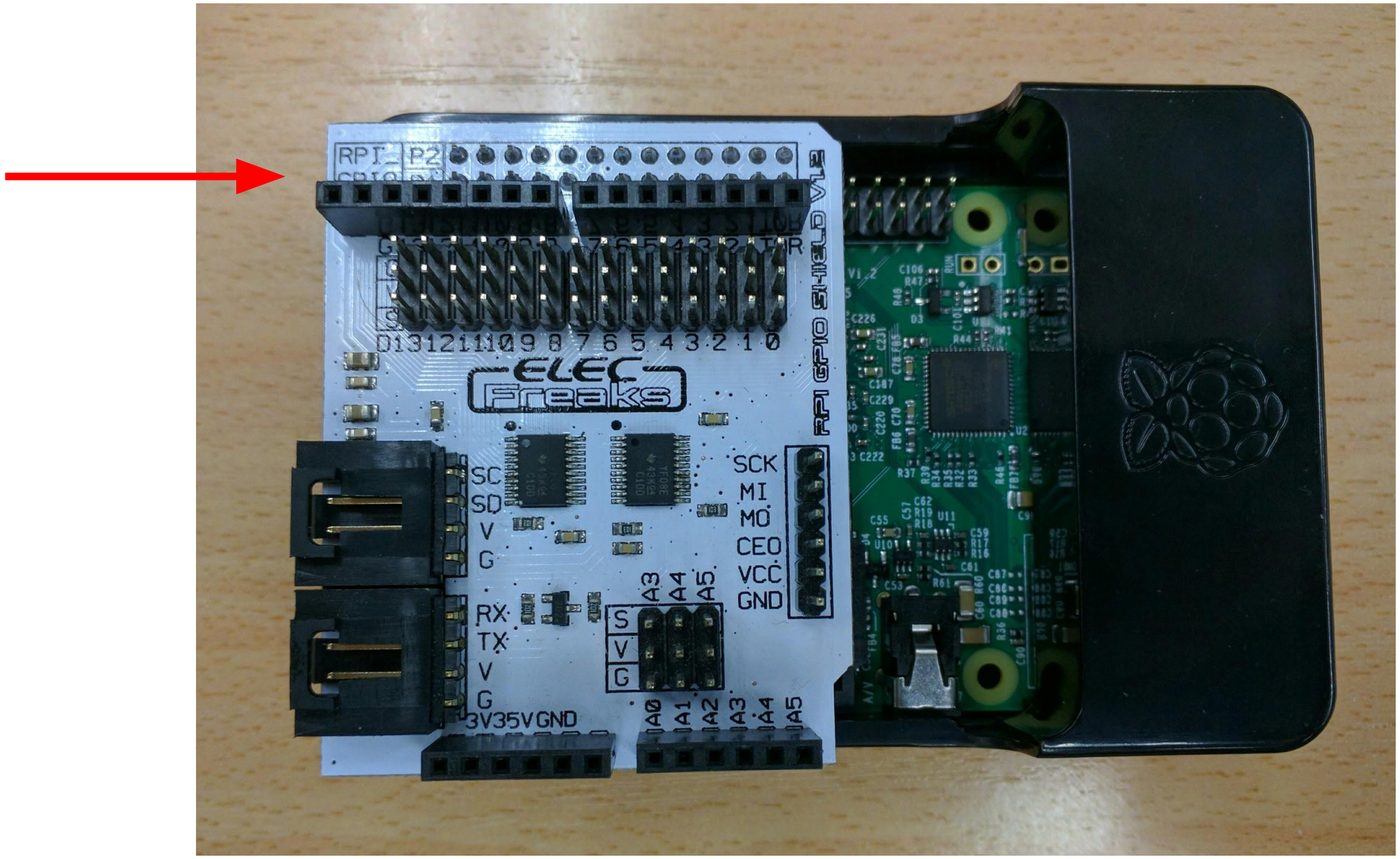
It can be!

If we connect sensors to it, it can function as an IoT device that collects data.

We also have the option of using it as a computer, but connecting a monitor and peripherals (mouse, keyboard, etc.)

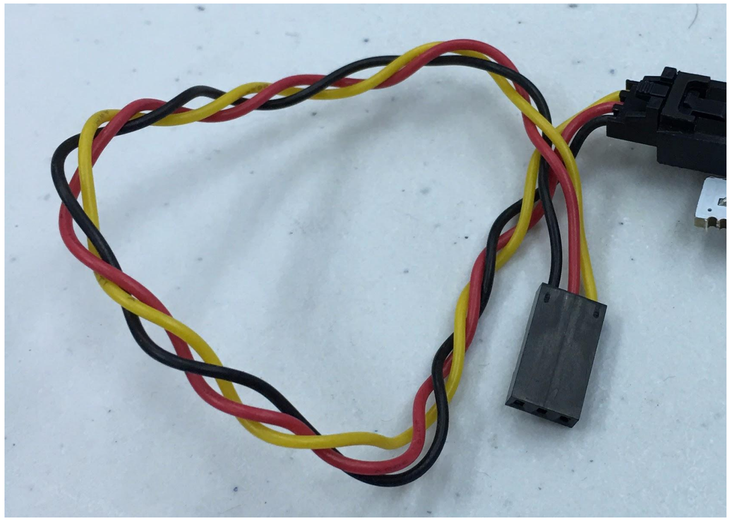
How to Connect the GPIO Shield

The GPIO Shield connects to the leftmost of the 40 pins:



Connecting Sensors

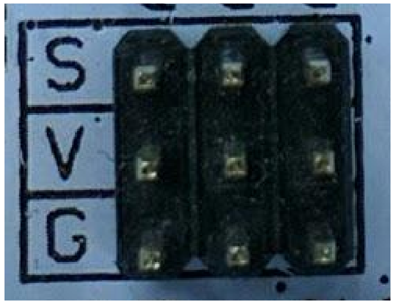
First, we need to connect sensors properly. Notice how each module has three different coloured wires? We need to connect these to the proper pins:



Yellow > S (Signal)

Red > V (Voltage)

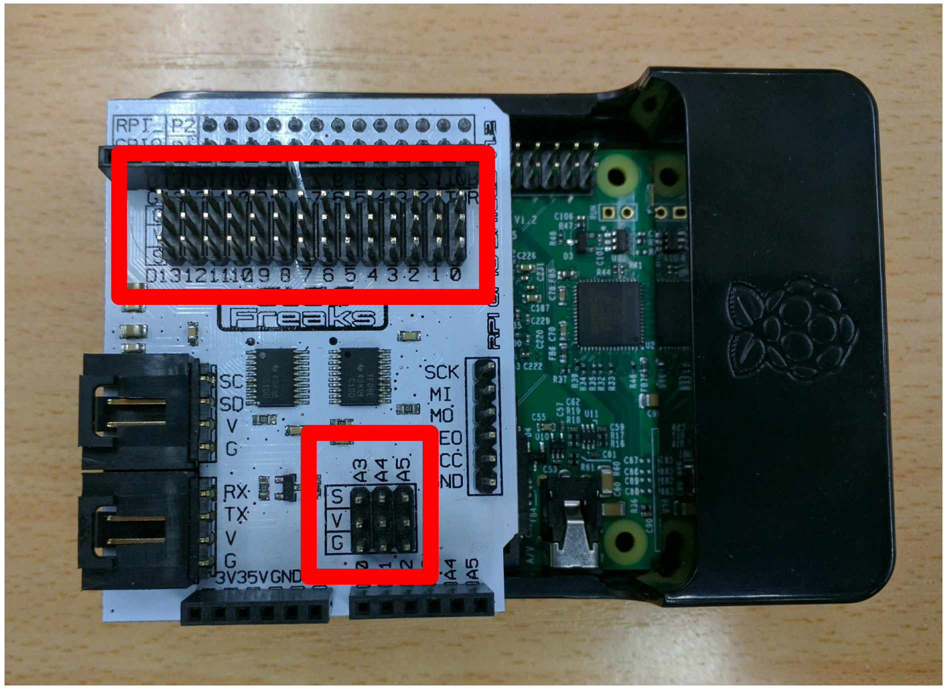
Black > G (Ground)



Analog vs Digital

Notice that we have analog and digital pins.

- Digital pins are labelled D0 to D13.
- Analog pins are labeled A3, A4, and A5.

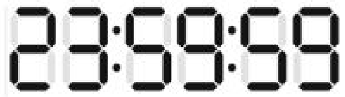


Analog vs Digital

An analog signal has an infinite amount of possibilities.

Analog →

Digital →



While a digital signal has a limited number of possibilities.

Picture from https://learn.sparkfun.com/tutorials/analog-vs-digital

Pin Map

D0 = RX	D9 = GPIO 4
D1 = TX	D10 = GPIO 8
D2 = GPIO 17	D11 = MOSI
D3 = GPIO 18	D12 = MISO
D4 = GPIO 27	D13 = SCLK
D5 = GPIO 22	
D6 = GPIO 23	A3 = GPIO 7
D7 = GPIO 24	A4 = SDA
D8 = GPIO 25	A5 = SCL

How to Interact with the GPIO Shield

Your SD card comes with GPIO libraries pre-installed!

We do need to import them though:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(pin number, GPIO.OUT)
GPIO.output(pin number, True/False)
```

We can run our python programs using IDLE.

Try to make an LED blink

Hint: use ‘import time’ and ‘time.sleep()’

Making an LED Blink

Example code to light up one LED (plug LED into D2):

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)
while True:
   GPIO.output(17, True)
   time.sleep(1)
   GPIO.output(17, False)
   time.sleep(1)
```

Making an LED Blink

How to stop your program from running:

Ctrl + C - sends a signal to kill the program but gives the program a chance to clean itself up first before closing

Ctrl + Z - sends a signal to kill the program, without giving it the chance to cleanup first

Allowing Cleanup

```
def loop():
    ...

if __name__ == '__main__':
    try:
        print('Press Ctrl-C to quit.')
        while True:
            loop()
    except KeyboardInterrupt:
        GPIO.cleanup()
```

Try to make a traffic light

- Your traffic light should:
  - Turn the first LED on for 3 seconds (green light)
  - Turn the first LED off and the second LED on for 1 seconds (yellow light)
  - Turn the second LED off and the third LED on for 3 seconds (red light)
  - And then repeat
- Feel free to customise your traffic light patterns after!

Active vs Passive Buzzer

Passive Buzzer

Requires an AC signal to make a sound; changing the input signal produces different tones.

Active Buzzer

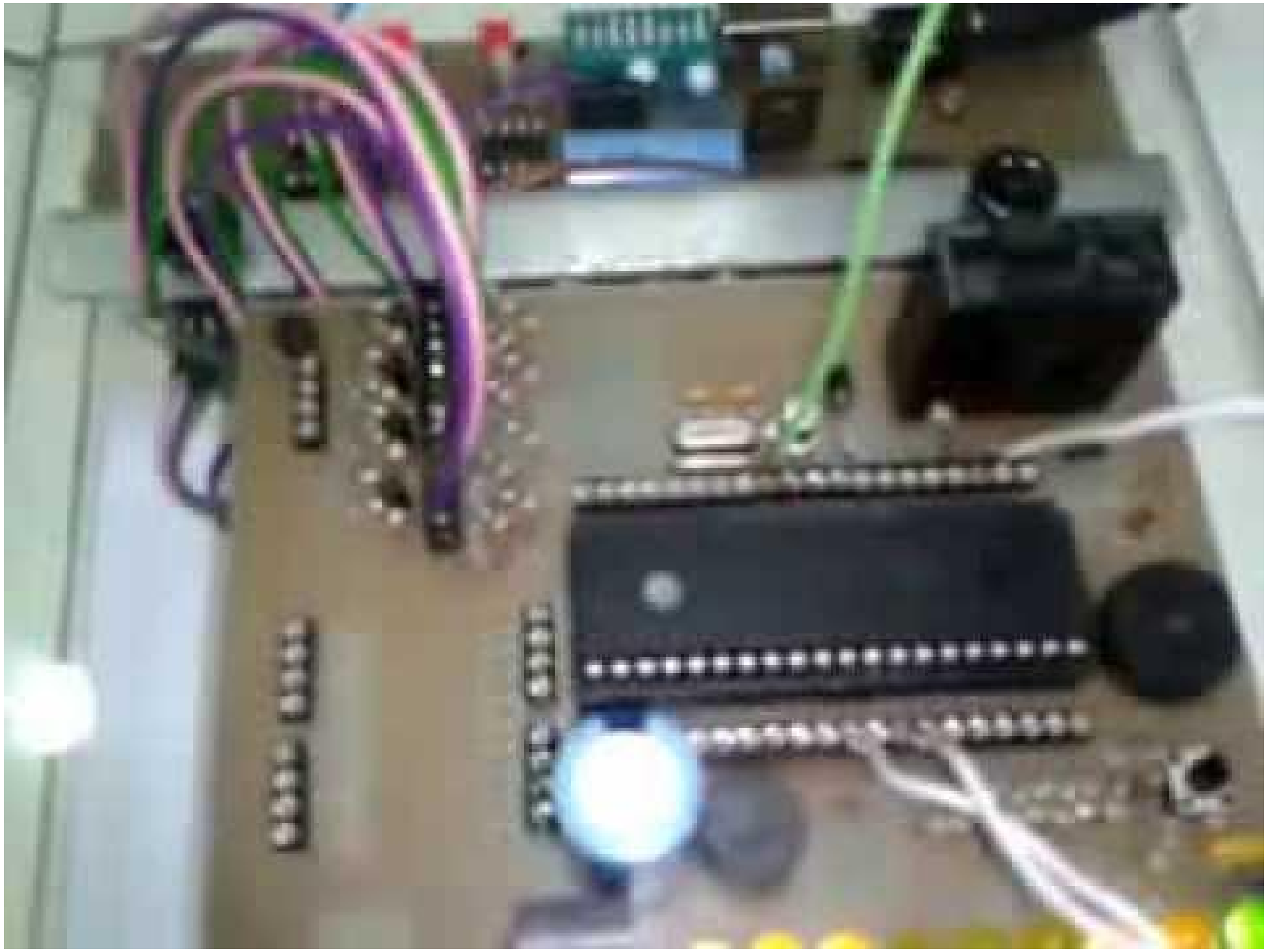
Generates one tone only, can be toggled on/off

Active vs Passive Buzzer

Active Buzzer - generates a tone using an internal oscillator, all you have to do is turn it on/off

Passive Buzzer - requires an AC signal to make a sound; changing the input signal produces the sound, rather than producing a tone automatically. In practice, we need to use some physics knowledge to produce sounds with this.

What Does the Passive Buzzer Sound Like?



Even Floppy Drives Make Better Music...

Better in Surround Sound?

Left as an exercise to the reader...

Pitch and Frequency

The pitch of a sound is determined by the frequency of vibrations of the particles of the medium that the sound wave is passing through.

Frequency measures the number of complete back and forth vibrations over a given time period (i.e. 1 vibration/sec)

We often use Hertz (Hz) to measure frequency.

1 vibration/sec = 1 Hz

If we change the frequency, we can change the pitch.

Passive Buzzer Example Code

Example code:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)
pitch = 523  #C note
duration = 1
#(cont. on next slide)
```

Passive Buzzer Example Code

Example code (cont.):

```
period = 1.0/pitch
delay = period / 2
cycles = int(duration * pitch)

for i in range(cycles):
    GPIO.output(17,True)
    time.sleep(delay)
    GPIO.output(17, False)
    time.sleep(delay)
```

Passive Buzzer Exercise

Try it out:

Program the passive buzzer to play ‘Mary Had a Little Lamb’ (or your favorite song, just make sure it’s not too complicated!)

Precise musical note frequencies can be found here: http://www.phy.mtu.edu/~suits/notefreqs.html

Hint: one option is to use lists to store multiple pitches and durations

Passive Buzzer Answer

Example code to play multiple short songs. View the code at the Gist linked below:

tk.sg/passivebuzzer

Add a buzzer to your traffic light

- Make the buzzer sound when the traffic light is red

Connecting the Button

The e-lock push button connects to a Digital port just like our LEDs, but we need to activate the pin to be an input rather than an output:

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

The pin will either have a value of True or False depending on if the button is pressed.

Hint: Use conditional statements to test if True or False!

Add a button to activate your traffic light

- Your traffic light should only loop through when the button is pressed

Traffic Light Answer

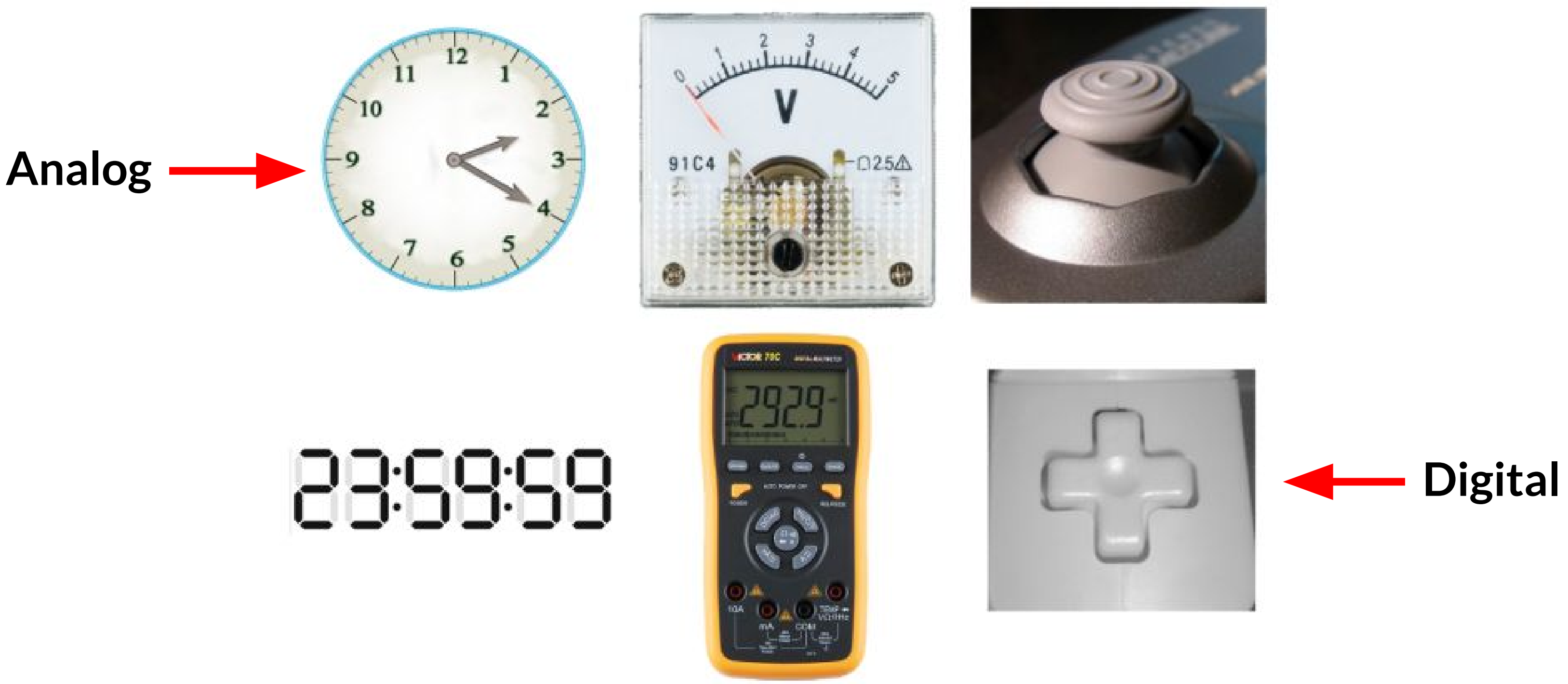
```
...

while True:
    if GPIO.input(23):
        GPIO.output(17, True)
        time.sleep(1)
        GPIO.output(17, False)
        GPIO.output(18, True)
        time.sleep(1)
        GPIO.output(18, False)
        GPIO.output(27, True)
        GPIO.output(22, True)
        time.sleep(1)
    else:
        GPIO.output(17, False)
        GPIO.output(18, False)
        GPIO.output(27, False)
        GPIO.output(22, False)
```

Pulsing Lights

When the Macbook added the “breathing” pulse light, it was a big feature! How do we recreate this on the Pi?

What is Analog?

An analog signal has an infinite amount of possibilities.



Picture from https://learn.sparkfun.com/tutorials/analog-vs-digital

While a digital signal has a limited number of possibilities.

Analog Output

We’ve seen that the Raspberry Pi can write HIGH (3.3V) and LOW (0V) digital voltage levels to light an LED. What if we wanted a brightness somewhere in between?

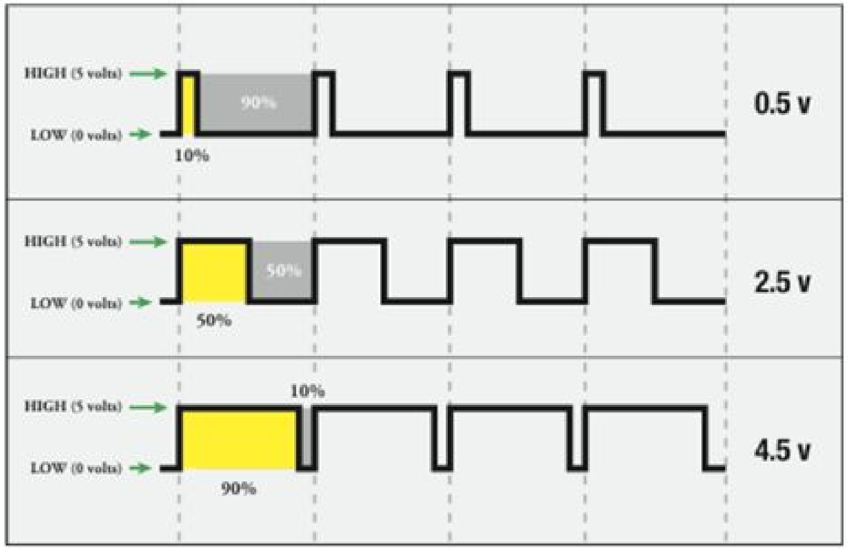
We can use **analog output** to set an arbitrary voltage between LOW (0V) and HIGH (3.3V), for instance 2V.

However, the Raspberry Pi does not have any analog outputs so it cannot truly output an arbitrary analog voltage, but it can *fake* an analog voltage.

Pulse Width Modulation (PWM)

PWM varies the amount of time that the blinking pin outputs HIGH vs. the time it outputs LOW.

Because the pin is blinking much faster than your eye can detect, the RPi creates the illusion of a “true” analog output.



However, do take note that only GPIO pin 18 (D3) is PWM-capable.

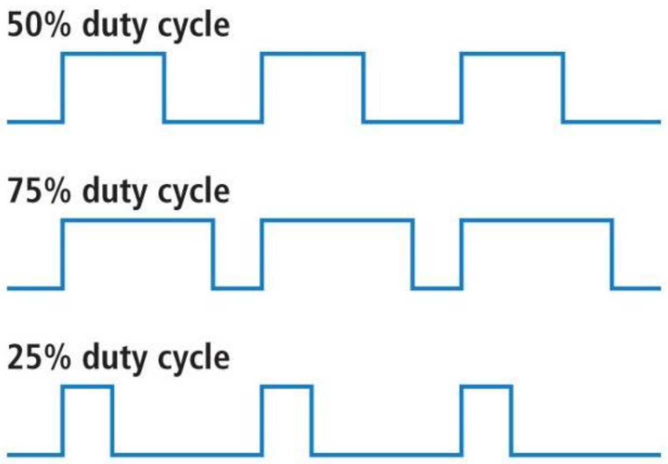
PWM commands

- *To create a PWM instance:*

```
p = GPIO.PWM(channel, frequency)
```

- *To start PWM:*

```
p.start(dc) #dc=duty cycle, 0.0<=dc<=100.0
```



Picture from: https://learn.sparkfun.com/tutorials/pulse-width-modulation

PWM commands

- *To change the frequency:*

  ```
  p.ChangeFrequency(new_freq)
  ```

- *To change the duty cycle:*

  ```
  p.ChangeDutyCycle(new_dc)
  ```

- *To stop PWM:*

  ```
  p.stop()
  ```

PWM Example

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

p = GPIO.PWM(18, 50) #frequency=50Hz
p.start(50) #duty-cycle = 50
```

(We'll keep the frequency constant at 50 throughout)

PWM Exercise

Try it out:

Program your LED to pulse slowly on and off.

PWM Answer (part 1)

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
p = GPIO.PWM(18, 50)
# pin=18 frequency=50Hz
p.start(0)
while True:
    p.ChangeDutyCycle(20)
    time.sleep(.5)
    p.ChangeDutyCycle(40)
    time.sleep(.5)
    p.ChangeDutyCycle(60)
    time.sleep(.5)

(cont. in next column)
```

```
    p.ChangeDutyCycle(80)
    time.sleep(.5)
    p.ChangeDutyCycle(100)
    time.sleep(.5)
    p.ChangeDutyCycle(80)
    time.sleep(.5)
    p.ChangeDutyCycle(60)
    time.sleep(.5)
    p.ChangeDutyCycle(40)
    time.sleep(.5)
    p.ChangeDutyCycle(20)
    time.sleep(.5)
    p.ChangeDutyCycle(0)
    time.sleep(.5)
```

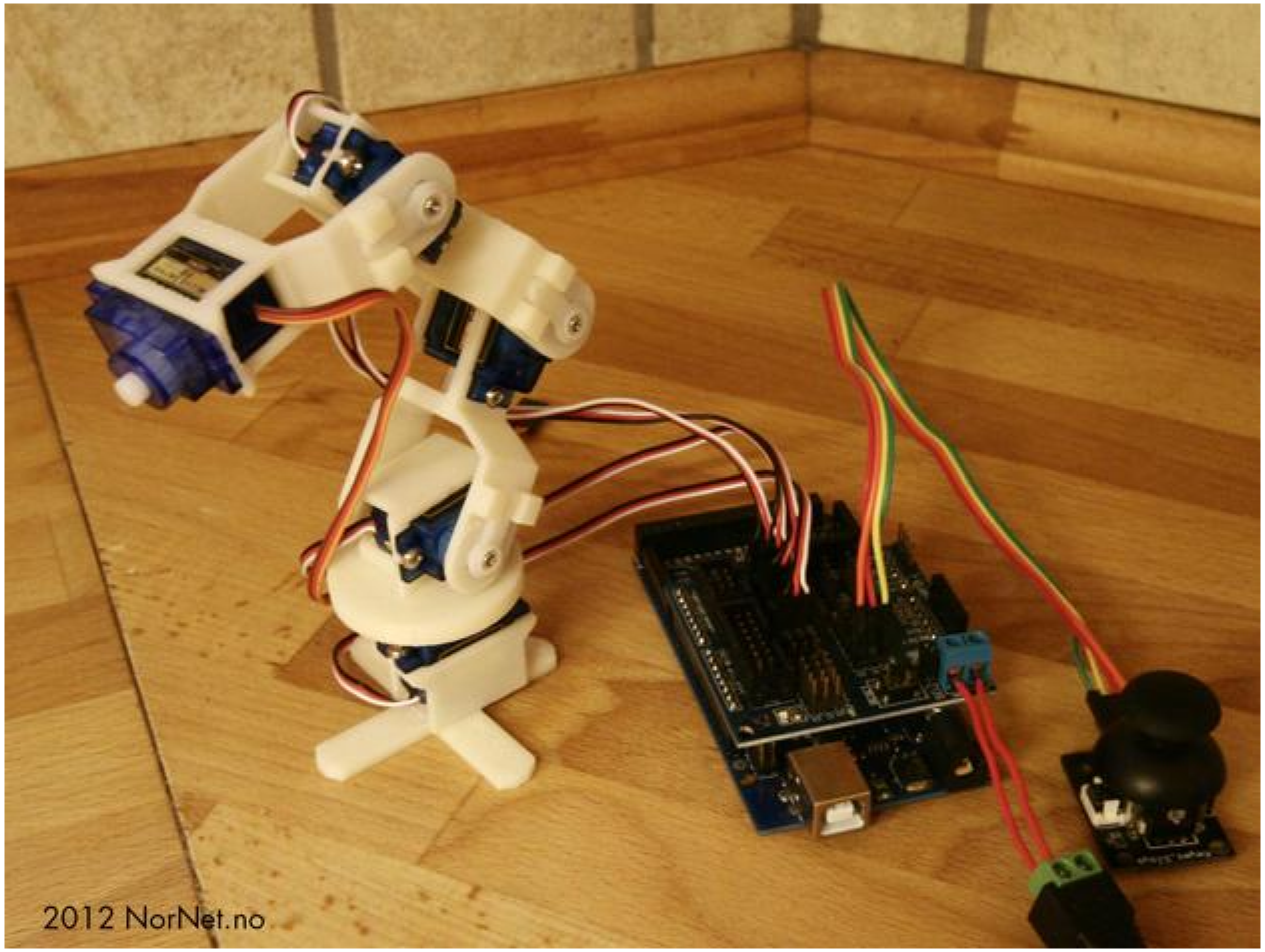
PWM Alternative Answer

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
p = GPIO.PWM(18, 50) #channel=18 frequency=50Hz
p.start(0)
try:
    while True:
        for dc in range(0, 101, 5):
            p.ChangeDutyCycle(dc)
            time.sleep(0.1)
        for dc in range(100, -1, -5):
            p.ChangeDutyCycle(dc)
            time.sleep(0.1)
except KeyboardInterrupt:
    pass
p.stop()
GPIO.cleanup()
```

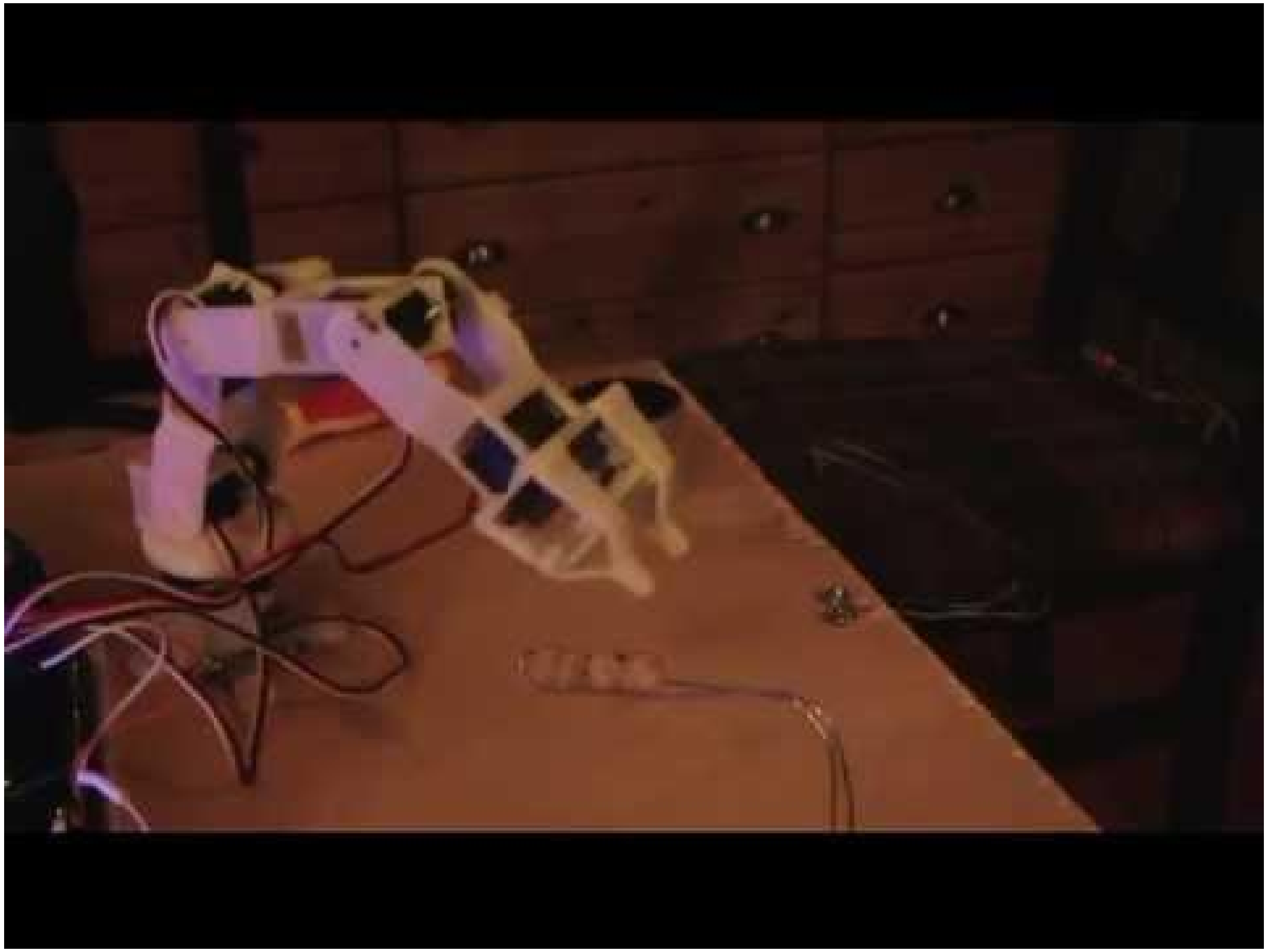
Servo

A servo is similar to a motor, except they typically do not rotate all the way around. Instead, they can be given commands to move precisely to a given angle.

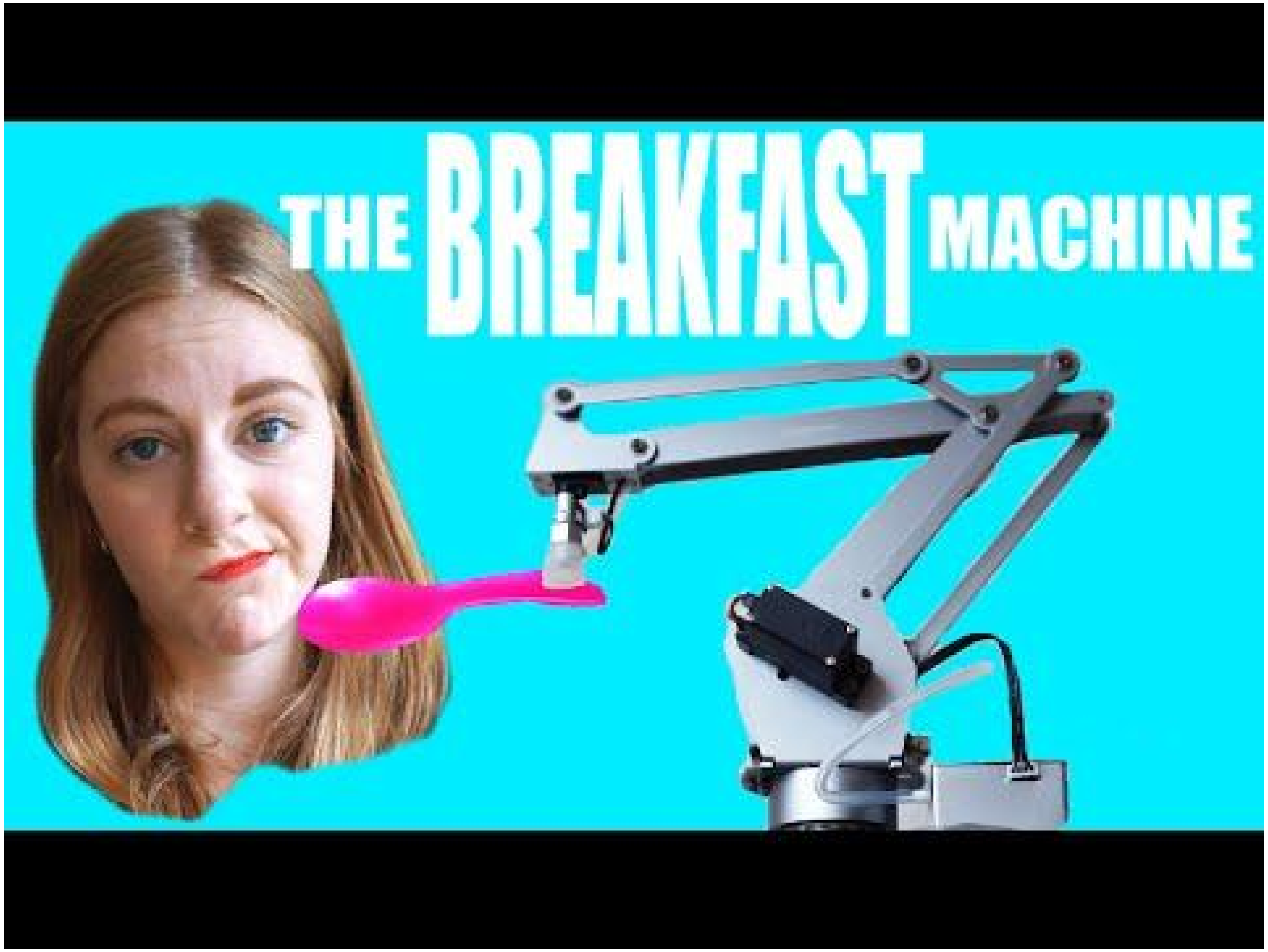
Servo



Servo



Servo



Servo

Servo

Servo

- Our servo isn't as powerful as those used in commercial applications.
- As a result, **BE CAREFUL** turning your servo to angles close to 0° and 180° (or numbers outside these boundaries).
- If the motor strains too much, it can burn out.
- Always include a pause to allow the servo time to move

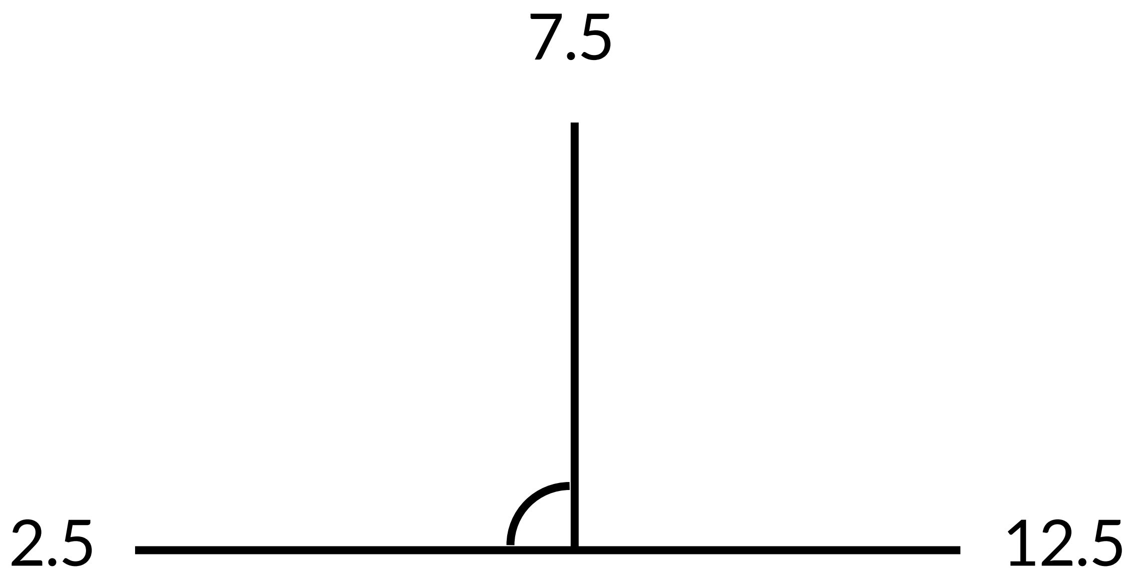
Servo Example

Servo Example

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
p = GPIO.PWM(18, 50)
p.start(7.5)
while True:
    p.ChangeDutyCycle(7.5) #90 degrees
    time.sleep(1)
    p.ChangeDutyCycle(11.5) #just under 180 degrees
    time.sleep(1)
    p.ChangeDutyCycle(3.5) #just over 0 degrees
    time.sleep(1)
```

*Be careful about turning your servo to exactly 0° or 180°!

Servo Example



2.5 corresponds to 0 degrees

12.5 corresponds to 180 degrees

Servo Exercise

Try it out:

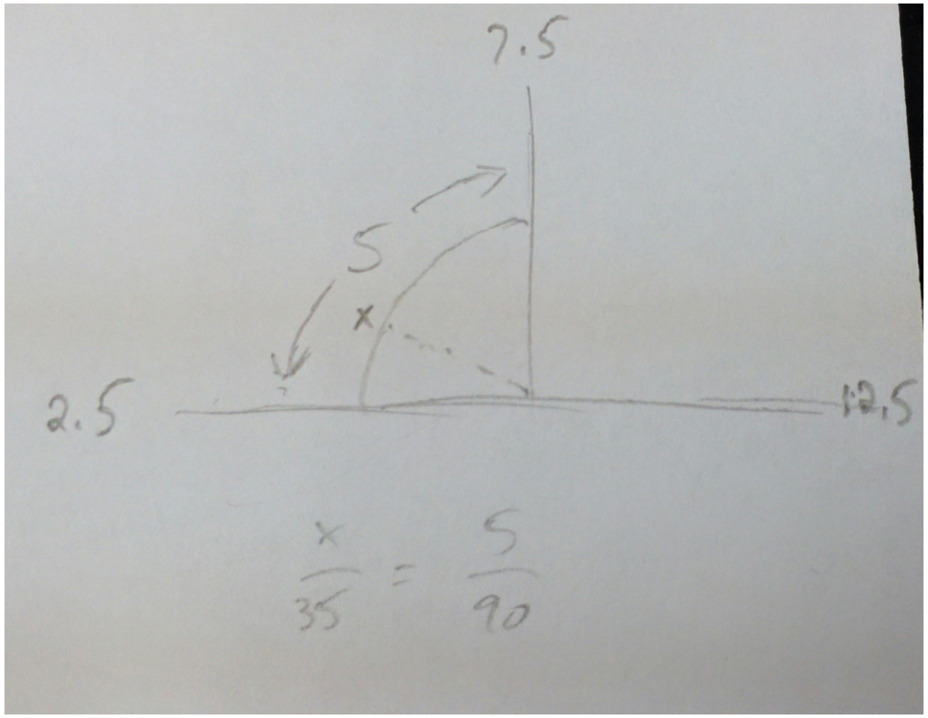
Program your servo to:

- Move to 90° at the start of the program
- Wait until a button is pressed, and then rotate back and forth between 35° and 145° **(be sure to add a short pause in between giving the serve a command to move)**

Servo Exercise – Hint

For precise angles, just use ratios.

(Remember that 2.5 corresponds to 0°, and 12.5 corresponds to 180°)



x=1.94, so the duty cycle for 35° = 1.94 + 2.5 = 3.44

Servo Exercise – Hint

Common angles:

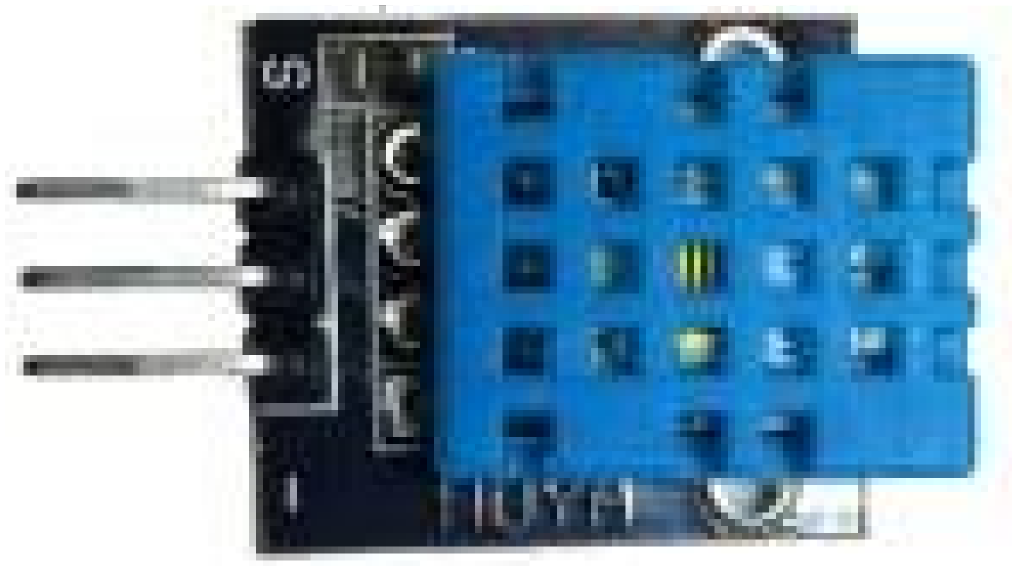
Angle	#
0°	2.50
15°	3.33
30°	4.17
45°	5.00
60°	5.83
75°	6.67
90°	7.50
105°	8.33
120°	9.17
135°	10.00
150°	10.83
165°	11.67
180°	12.50

Servo (Incomplete) Answer

```
p = GPIO.PWM(18, 50)
p.start(7.5)
while True:
   if GPIO.input(27):
      p.ChangeDutyCycle(11.56)
      time.sleep(1)
      p.ChangeDutyCycle(3.44)
      time.sleep(1)
   else:
      p.ChangeDutyCycle(7.5)
      time.sleep(1)
```

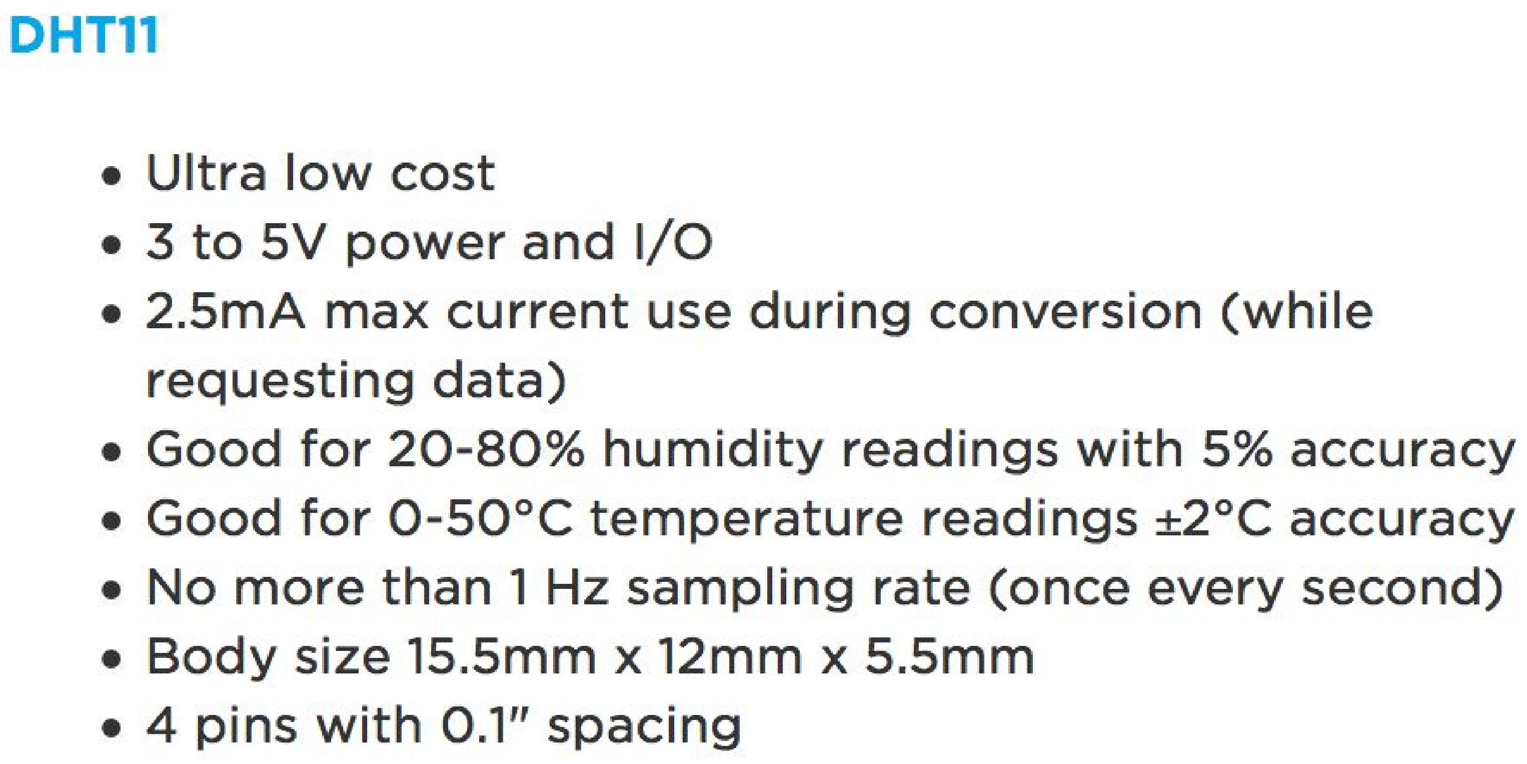
Humidity/Temperature Sensor

Let’s first try to connect the digital humidity/temperature sensor (DHT for short). This is a DHT11 sensor:



Humidity/Temperature Sensor

Note that these sensors are DHT11 as opposed to DHT22. If you're curious the difference:

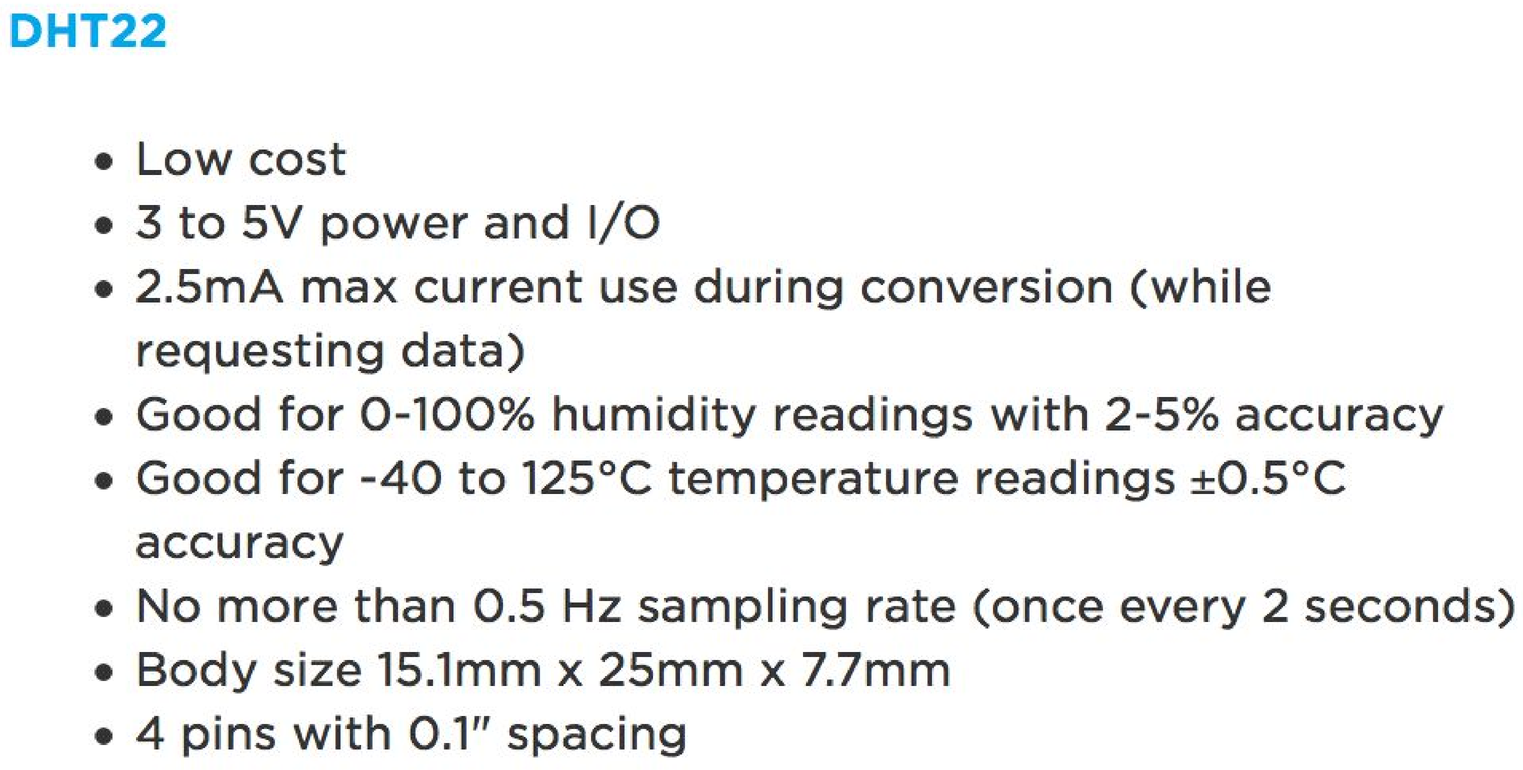


DHT11

- Ultra low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 20-80% humidity readings with 5% accuracy
- Good for 0-50°C temperature readings ±2°C accuracy
- No more than 1 Hz sampling rate (once every second)
- Body size 15.5mm x 12mm x 5.5mm
- 4 pins with 0.1" spacing

Humidity/Temperature Sensor

TL;DR - DHT22 is slightly better (in accuracy and range) than the DHT11. Keep these specs in mind!



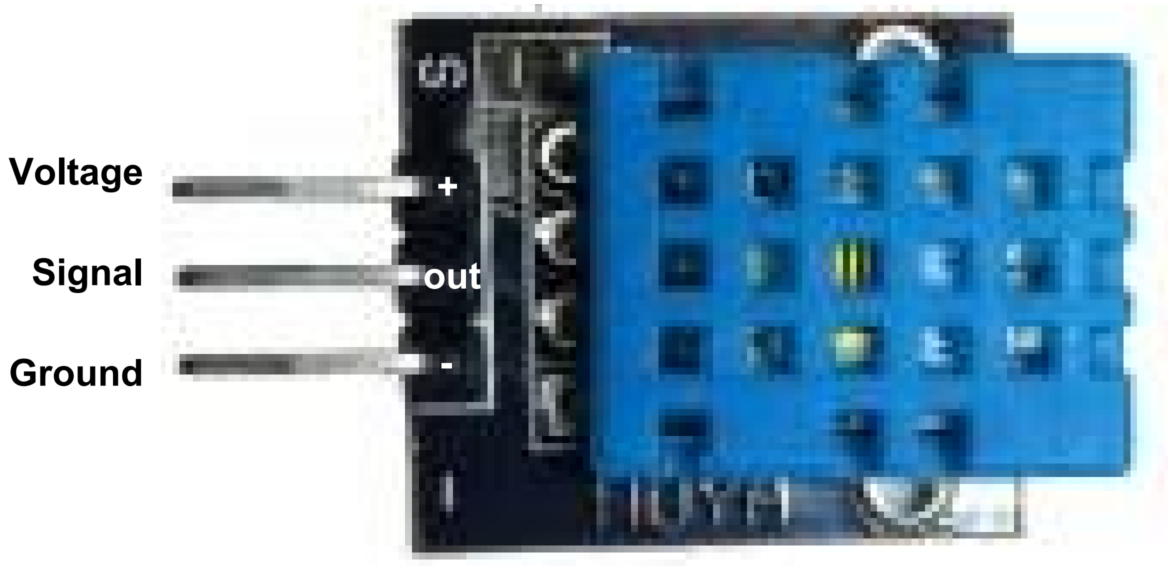
DHT22

- Low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 0-100% humidity readings with 2-5% accuracy
- Good for -40 to 125°C temperature readings ±0.5°C accuracy
- No more than 0.5 Hz sampling rate (once every 2 seconds)
- Body size 15.1mm x 25mm x 7.7mm
- 4 pins with 0.1" spacing

Humidity/Temperature Sensor

Let's connect it to the RPi now. Connect the three pins to the appropriate pins on the RPi. You can connect them to any viable digital pin on the Pin Map, just make sure you match:

Voltage > V, Signal > S, Ground > G



Install Adafruit Library

To install the Adafruit DHT11 library:

1. Enter this at the command prompt to download the library:

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
```

2. Change directories with:

```
cd Adafruit_Python_DHT
```

3. Enter this:

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

4. Install the library with:

```
sudo python3 setup.py install
```

Test your code

```
import Adafruit_DHT

humidity, temperature = Adafruit_DHT.read_retry(
   Adafruit_DHT.DHT11, <pin#>)
```

where <pin#> = the GPIO pin you connected your DHT component to

```
(i.e. humidity, temperature = Adafruit_DHT.read_retry(Adafruit_DHT.DHT11, 17)
```

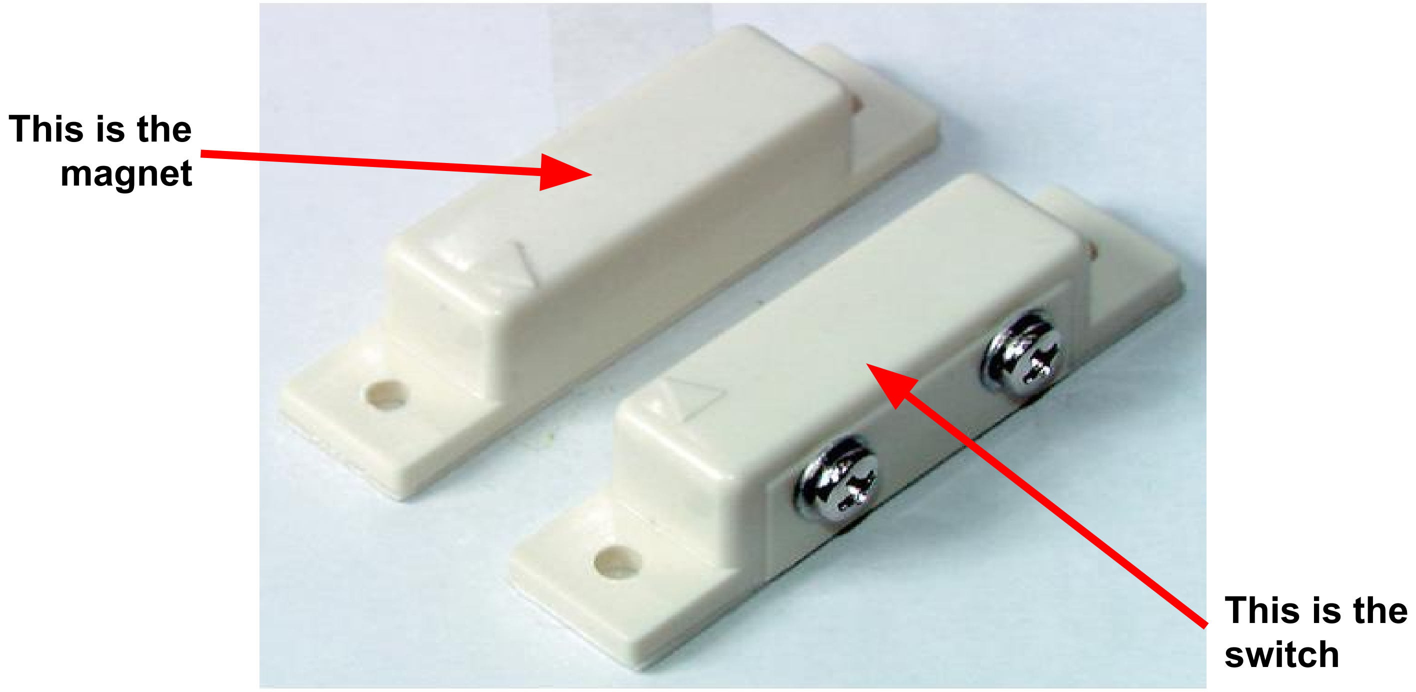
Test your code

Finally, print the sensor values:

```
if humidity is not None and temperature is not None:
   print('temp =', temperature)
   print('humidity =', humidity)
else:
   print('Failed to get reading')
```

Reed Switch

A reed switch uses a magnet to trigger a switch circuit

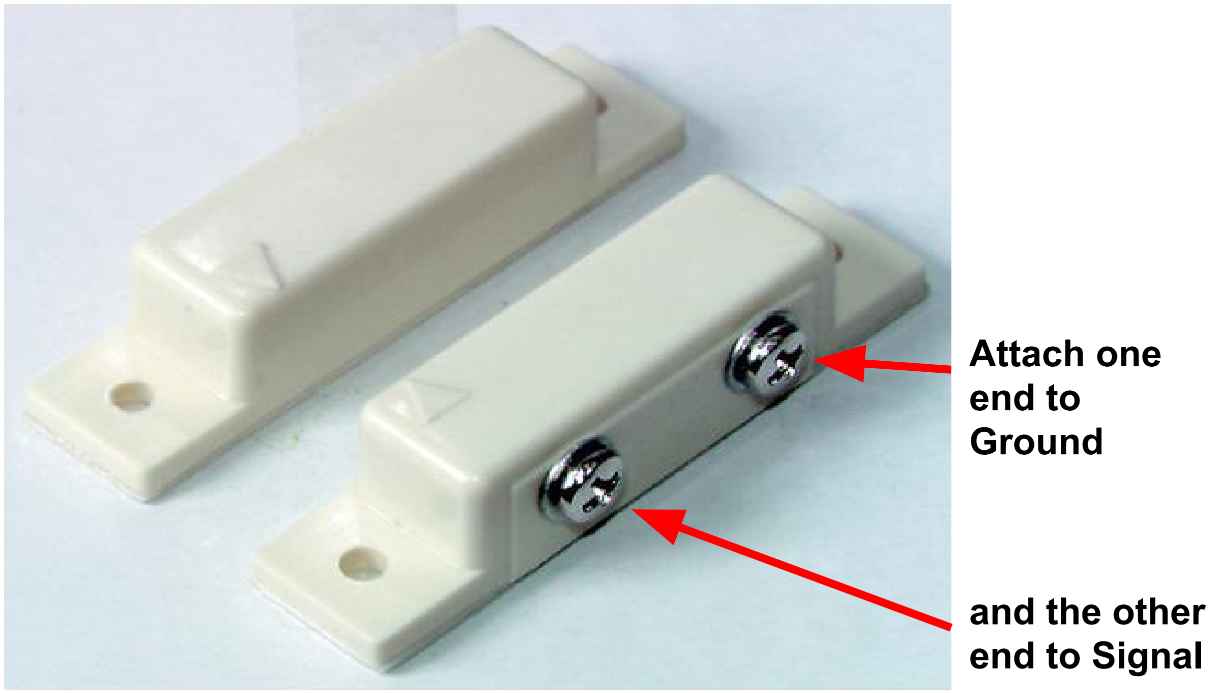


Reed Switch Uses

Reed Switch Uses

Reed Switch

1. Unscrew the screws slightly
2. Obtain/create a male-to-female wire
3. Wind the male wire under the screw head and tighten



Reed Switch code

Once you have your reed switch connected, it’s time to test it out. Remember how we had to initialise GPIO pins before? We need to do the same here and set our GPIO pin to be an input, except we’re going to pass one more parameter as well:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

Reed Switch code

Try it out:

The pin value will switch between True and False depending on the switch. Now, test if the reed switch is working by adding to your code.

Reed Switch code

Answer:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True:
    if GPIO.input(18):
        print("not activated")
    else:
        print("reed switch activated")

GPIO.cleanup()
```