Lab 04: Using sensors with a Raspberry Pi

One powerful feature of the Raspberry Pi are the GPIO (general purpose input/output) pins along the top edge of the board. These pins are a physical interface between the Pi and the outside world. At the simplest level, you can think of them as switches that you can turn on or off (input) or that the Pi can turn on or off (output).

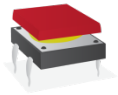

Figure 1 : Push Button

A push button or switch is an input component that you can add to the Raspberry Pi GPIO pins. It will complete a circuit when the button is pressed. What that means is that a current will not flow across the button until it is pressed. When it is released, the circuit will be 'broken'. Sensors are another type of input that work in the same way as a button or switch. Other sensors can be added to the pins, either as individual components connected to GPIO pins, or through an add-on board called a HAT which stands for Hardware Added on Top. Here are a few other popular sensors:

Individual Component Sensors

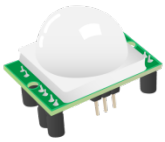

Figure 2 : Passive Infrared Sensor

A Passive Infra-Red sensor or PIR detects movement. You might have seen these before as they are very common. You would most often find them in the corners of rooms for burglar alarm systems. All objects whose temperatures are above absolute zero emit infra-red radiation. Infra-red wavelengths are not visible to the human eye, but they can be detected by the electronics inside one of these modules. The sensor is regarded as passive because it doesn't send out any signal in order to detect movement. It adjusts itself to the infra-red signature of the room it's in and then watches for any changes. Any object moving through the room will disturb the infra-red signature, and will cause a change to be noticed by the PIR module.

Figure 3 : Light Dependent Resistor called LDR

A Light Dependent Resistor or photocell is a component whose resistance will change depending on the intensity of light shining upon it. It can therefore be used to detect changes in light. They are commonly used to turn on the street lighting when it gets dark at night, and turn them off when it gets bright in the morning.

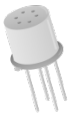

Figure 4 : Air Quality Sensor

An air quality sensor is used to determine air quality by detecting polluting gases. When air enters the sensor, it is energized by a small heater which allows its electrical resistance to be measured. This is done by passing a low level of electricity across a small gap of energized air. The more contaminated the air is, the less resistance it has and the better it will conduct electricity (like a variable resistor). The output of the sensor is therefore an analogue voltage that goes up and down according to how contaminated the air is. The more contaminants, the higher the voltage output.

Sense HAT (Hardware Attached on Top)

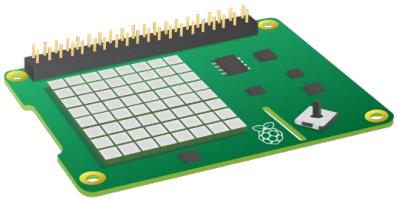

Figure 5 : Sense HAT

The Sense HAT board for the Raspberry Pi has the ability to sense a wide variety of conditions and provide output via the built-in LED matrix. It was designed especially for

the Raspberry Pi as part of the Astro Pi education program, and there are two on board the International Space Station that can be programmed by competition winners.

The Sense HAT has the following sensors:

1. A gyroscope measures the orientation of an object. The gyroscope allows three degrees of movement. These are normally called: Pitch (up and down like a plane taking off and landing), Yaw (left and right like steering a car), and Roll (imagine a corkscrew movement, like barrel rolling a fighter jet).

2. An accelerometer measures an object's increase in speed (acceleration). At rest, it will measure the direction and force of gravity, but in motion it measures the direction and force of the acceleration acting on it – as if you were swinging it around your head on a rope. Because accelerometers can detect the direction of gravity, they are often found in devices that need to know when they are pointing downwards, such as a mobile phone or tablet. When you turn the screen sideways the accelerometer inside detects that the direction of gravity has changed, and therefore changes the orientation of the screen.

3. A magnetometer is used to measure the strength and direction of a magnetic field. Most often they're used to measure the Earth's magnetic field in order to find the direction of North. If your phone or tablet has a compass, it will probably be using a magnetometer to find North. They are also used to detect disturbances in the Earth's magnetic field caused by anything magnetic or metallic; airport scanners use them to detect the metal in concealed weapons, for instance.

4. A temperature sensor is used to measure hot and cold. It is exactly like the thermometer that you would put in your mouth to take your own temperature, except it's an electronic one built into the Sense HAT and reports the temperature as a number in Celsius.

5. A humidity sensor measures the amount of water vapour in the air. There are several ways to measure it, but the most common is relative humidity. One of the main properties of air is that the hotter it is, the more water vapour can be suspended within it. So relative humidity is a ratio, usually a percentage, between the actual amount of suspended water vapour to the maximum amount that could be suspended for the current temperature. If there was 100% relative humidity, it would mean that the air is totally saturated with water vapour and cannot hold any more.

6. A pressure sensor (sometimes called a barometer) measures the force exerted by tiny molecules of the air we breathe. There's a lot of empty space between air molecules and so they can be compressed to fit into a smaller space; this is what happens when you blow up a balloon. The air inside the balloon is slightly compressed and so the air molecules are pushing outwards on the elastic skin; this is why it stays inflated and feels firm when you squeeze it. Likewise, if you suck all the air out of a plastic bottle, you're decreasing the pressure inside it and so the higher pressure on the outside crushes the bottle.

Explorer HAT (Hardware Attached on Top)

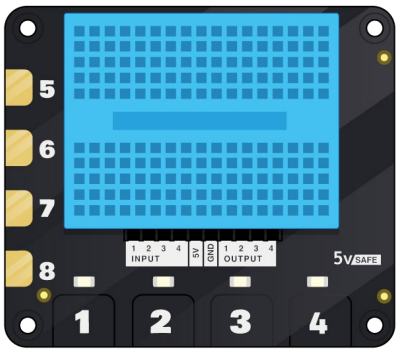

Figure 6 : Explorer HAT

The Explorer HAT add-on board for the Raspberry Pi has useful input and output components built in as well as some useful sensors. In particular, eight capacitive touch pads.

A capacitive touch sensor detects when the metal pads contact human skin, or an object carrying a small electrical charge. Touching one of the eight capacitive touch pads on the Explorer HAT triggers an event, acting as a switch or button. Besides human beings, loTs of objects like plants, fruits and graphite pencils can also be used as contact surfaces.

After reading through the small range of sensors mentioned in this article, what ideas do you have for something you could make with them?

Analogue Vs. Digital

Using the GPIO pins on the Raspberry Pi, it is easy to send a signal to an output component and turn it on or off. You can also detect whether an input component is on or off quite easily. Components that operate in this way are called digital components.

An LED is an example of a digital *output* component. It can either be on or off, and there is no value in-between. We can think of the **on** and **off** states as being either **1** or **0**. You can send a **1** to the LED to illuminate it on and a **0** to the LED to turn it off again.

A button is an example of a digital *input* component. It can either be on or off as well. When the button is pressed, it sends a **1** to the Raspberry Pi GPIO pin it is connected to.

When the button is release, it sends a **0** to the GPIO pin. There is no other value that can be sent, as you can't **half-press** a button.

Look at the graph below. This shows a button being pushed and released over time. When it is pushed it sends a 1 and when it is released it sends a 0.

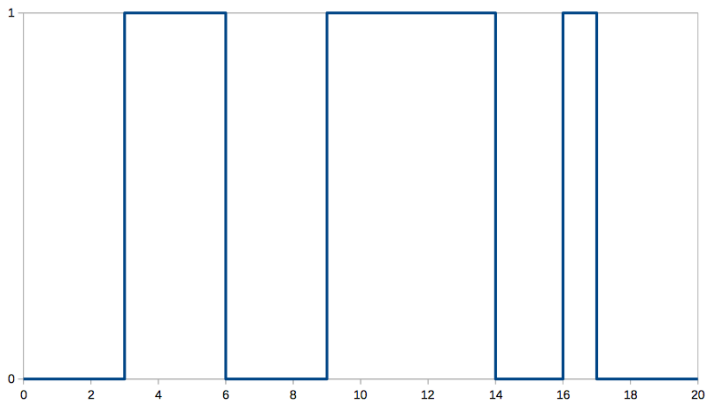

Figure 7 : Digital Input/Output Signal

Digital input and output components are easy to use with the Raspberry Pi, as the GPIO pins are all digital. They can only send or receive 1s and 0s. Not all components are digital however. Some are called analogue components. Analogue components can send and receive values in-between 1 and 0.

A motor is an example of an analogue *output* component. You can send it values between 1 and 0, which will control the speed of the motor. If you send the motor a 1 it will drive at full speed. If you send it 0.5 it will drive at half speed. Sending a 0 will stop the motor.

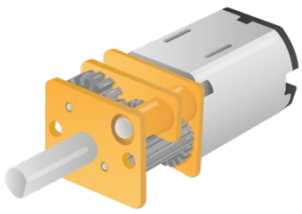

Figure 8 : A Motor

An example of an analogue input component is a Light Dependent Resistor (LDR). When there is no light shining on the component, it has a high resistance. And as light

increases, the resistance of LDR will gradually decrease. This change in resistance can be translated in different voltage drops across LDR, and used as a trigger switch. The graph below shows how the voltage drop across an LDR will impact the voltage on one of it's node over the course of a 24 hour period, as it goes from dark to light and back again.

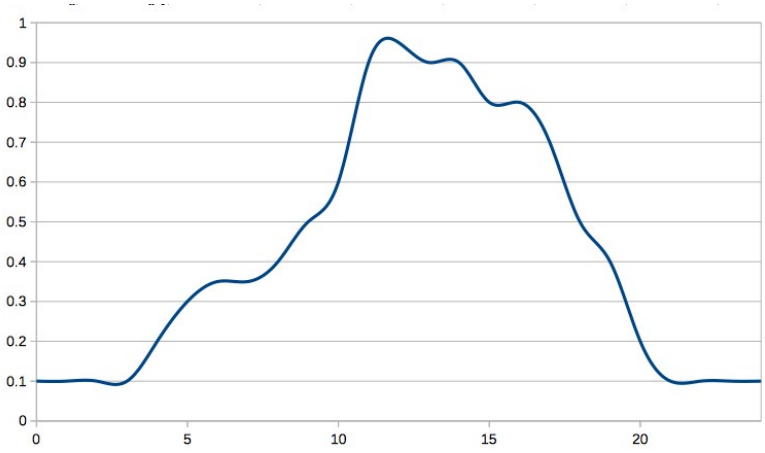

Figure 9 : Light Dependent Resistor (LDR) Input Graph Example

Using analogue components with the Raspberry Pi is a little trickier than using digital components.

To use an analogue output component with the GPIO pins, you need to use a technique called Pulse Width Modulation (PWM). This sends very rapid pulses of 1s and 0s to the component, which when taken as an average can be received as values in-between 1 and 0.

Look at the graph below. The blue line shows the digital signal, over a period of time, moving from 0 to 1 and back again. The signal is 1 for a third of the total time and 0 for the remaining two thirds. This then averages out at around 0.33, which would be the value that is received by the analogue component. You can see this as the red line on the graph.

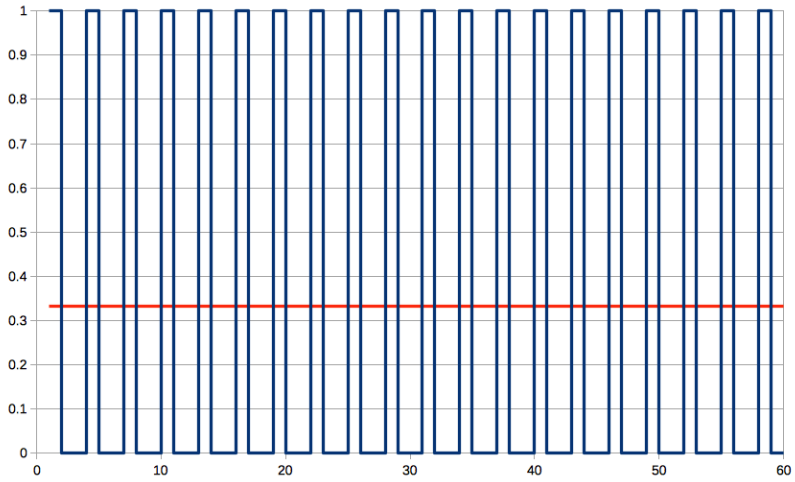

Figure 10 : Pulse Width Modulation (PWM) Signal

To use an analogue input component with the GPIO pins, you need to use an Analogue to Digital Convertor (ADC) that will turn analogue signals into digital signals. Although you can buy small ADCs for use in your circuits, there are several add-on boards you can buy for the Raspberry Pi with ADCs included, such as the Explorer HAT. Another option is to use a capacitor in your circuits along with the analogue component. An example of this can be found on the Laser Tripwire resource on the Raspberry Pi website.

Wiring up your button

Buttons and switches are a class of input component. They allow a user to have some control over a circuit, or to send signals to a computer. The keys on your keyboard are all examples of buttons; when you press a key it sends a signal to the computer that represents the character that has been pressed. You can use physical buttons and switches to send signals to your Raspberry Pi, and trigger all kinds of events. In the diagram below, a button has been wired up to a Raspberry Pi on GP4 and a GND pin. You'll notice that the button has been pushed into the breadboard across the dividing line.

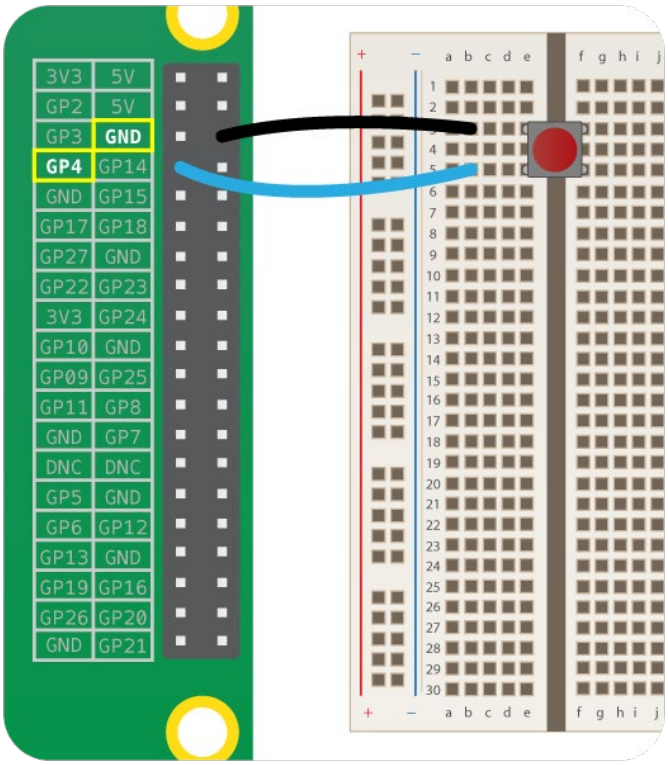

Figure 11 : Wiring Diagram with Button

In this case, GP4 starts off high and when the button is pressed and the circuit closed, the pin is sent low as it is connected to the ground pin. Thus, when the button is pushed, the state of pin 4 is changed.

Test your reflexes

Now that you have gained the skills and knowledge to control LEDs and detect button pushes, it's time to build your first game.

1. Create a new Python file and save it as reaction_game.py.

   The idea of the game is to switch on an LED after a random period of time. The player will have to push a button the moment the LED switches on, and the time between the LED coming on and the button being pressed can be recorded and the reaction time be displayed on the screen.

2. To begin with, you'll need to be able to use a few classes and functions from the modules you've already used. In particular you will need the Button and LED class from gpiozero and the time, sleep and randint functions from the time and random modules.

   At the top of your file, import the classes and function as shown below.

```
from gpiozero import Button, LED
from time import time, sleep
from random import randint
```

3. The next stage will be to create a new LED object on pin 17 and a Button object on pin 4.

```
led = LED(17)
btn = Button(4)
```

4. Now physically connect an LED and a Button to pins 17 and 4 respectively (Don't forget to use a resistor to protect the LED).

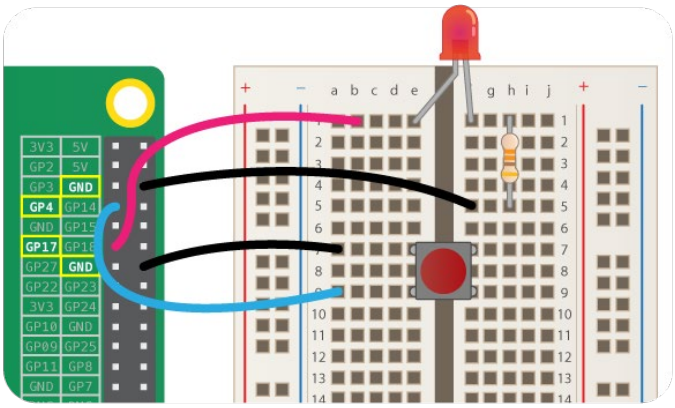

Figure 12 : Wiring Diagram with Button and LED

It's always a good idea to test your wiring, so let's add a little bit of code to make sure it's all working fine.

```
led.on()
btn.wait_for_press()
led.off()
```

Save and run your code. The led should come on, and then turn off again when the button is pressed.

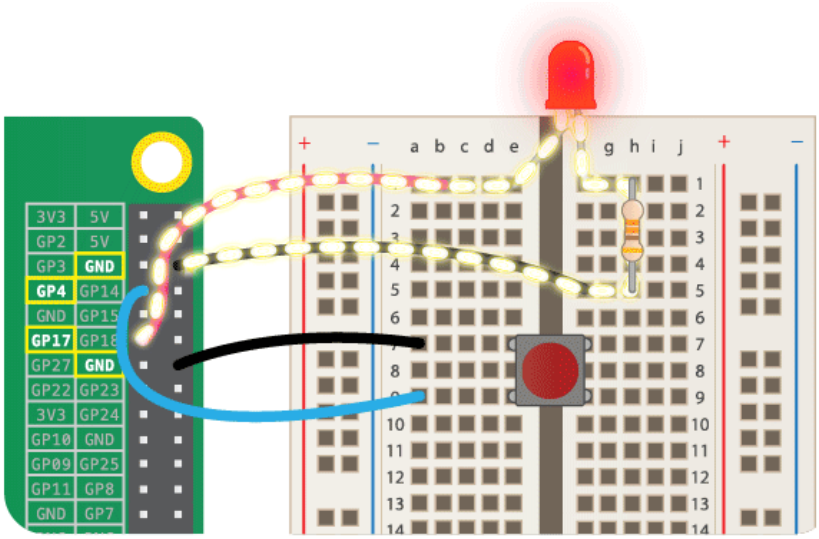

Figure 13 : Wiring Diagram for 'Test Your Reflexes Game'

You already have most of the code to make your game now. Let's first make a little change by placing those last three lines into an infinite loop.

```
while True:
    led.on()
    btn.wait_for_press()
    led.off()
```

If you save and run this, the led should come on, switch off when the button is pressed but then come straight back on again. The loop runs so quickly that you may not even notice the flicker of the LED. You can place a sleep into the loop to stop this behaviour, just as you did in the last lab when you made the LED blink. This time though, instead of making the program sleep for a set amount of time, we can make it sleep for a random number of seconds using the randint function. In the code below the time will be a random integer between 1 and 10 seconds.

```
while True:
    sleep(randint(1,10))
    led.on()
    btn.wait_for_press()
    led.off()
```

To measure the time between the led turning on and the button being pushed, we'll need to use the time function. In the code below the time is saved in two variables - start and end.

```
while True:
    sleep(randint(1,10))
    led.on()
    start = time()
    btn.wait_for_press()
    led.off()
    end = time()
```

To finish off the basic game, you can subtract start from end to calculate the reaction time, and then print it out on the screen.

```
from gpiozero import Button, LED
from time import time, sleep
from random import randint

led = LED(17)
btn = Button(4)

while True:
    sleep(randint(1,10))
    led.on()
    start = time()
    btn.wait_for_press()
    end = time()
    led.off()
    print(end - start)
```

Save and run your game to see what your reaction times are like. You can quit your program using ctrl + c.

Optional Task : Distance Sensor using HC-SR04 Ultrasonic Sensor

Now we will be using the HC-SR04 Ultrasonic Sensor to measure distance with our Raspberry Pi.

Hardware Required:

- HC-SR04 Ultrasonic Sensor
- 3x 1k OHM Resistor

HC-SR04 Ultrasonic Distance Sensor can report the range up to 4 meters away. The sensor consists of two ultrasonic transducers. One transmits a 40kHz ultrasonic sound pulses and the other listen for the transmitted pulses. Once it receives them, it produces an output pulse which we can determine the distance the pulse travelled.

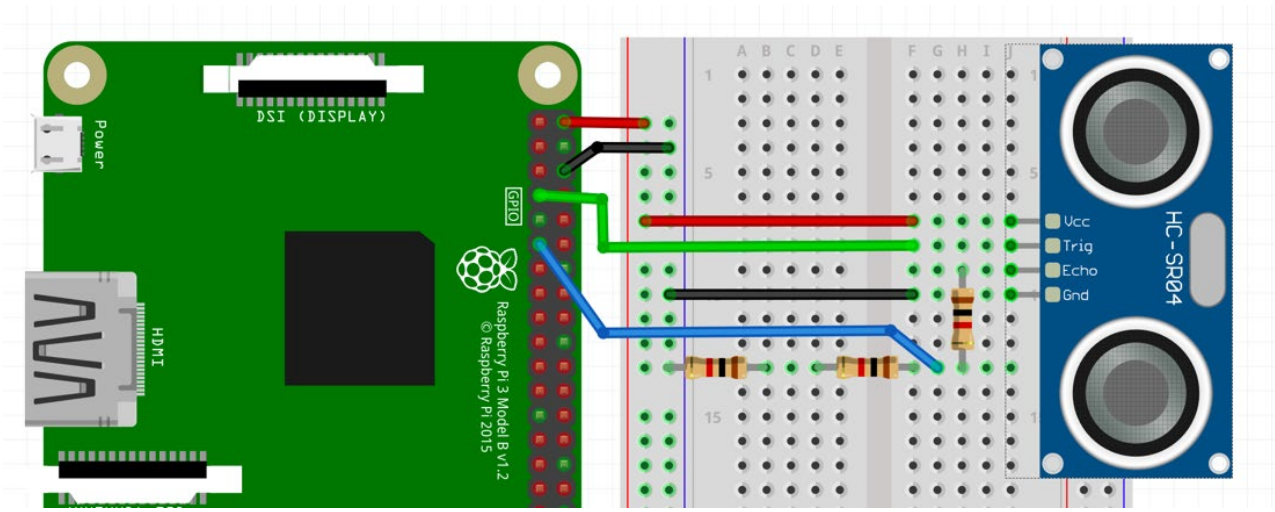

Figure 14 : Wiring Diagram for Distance Sensor using HC-SR04 Ultrasonic Sensor

Follow the guide above to see how to wire your HC-SR04 distance sensor to your Raspberry Pi

- VCC connects to Pin 2 (5V)
- Trig connects to Pin 7 (GPIO 4)
- Echo connects to R1 (1k OHM)
- R2 and R3 (1k OHM + 1 k OHM) connects from R1 to Ground
- Wire from R1 and R2+R3 connects to Pin 11 (GPIO 17)
- GND connects to Pin 6 (Ground)

1. Create a new Python file and save it as distance_sensor.py

2. To begin, you'll need to import **RPi.GPIO** and time. The **RPi.GPIO** allow us to control and interact with the GPIO pins

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
```

3. We then begin our try: statement. This first defines the pin number we are using on the Raspberry Pi for **TRIGGER** and **ECHO**. We also define **PIN_TRIGGER** as an output and **PIN_ECHO** as an input.

```
try:
    GPIO.setmode(GPIO.BOARD)

     PIN_TRIGGER = 7
     PIN_ECHO = 11

     GPIO.setup(PIN_TRIGGER, GPIO.OUT)
     GPIO.setup(PIN_ECHO, GPIO.IN)
```

4. Before we start, we need to initialize the sensor by setting it to "**Low**" so it doesn't send out anything.

```
    GPIO.output(PIN_TRIGGER, GPIO.LOW)

    print("Waiting for sensor to settle")

    time.sleep(2)
```

5. Then the code triggers the sensor by setting it to "**high**" for 1 nanosecond.

```
    print("Calculating distance")

    GPIO.output(PIN_TRIGGER, GPIO.HIGH)

    time.sleep(0.00001)

    GPIO.output(PIN_TRIGGER, GPIO.LOW)
```

6. We then measure the time required for the pulse to travel using a while loop to continually check if **PIN_ECHO** is **low (0).** If it is, we continually set the **pulse_start_time** to the current time until it become **high (1)**. Once **PIN_ECHO** reads high, we set the **pulse_end_time** to the current time.

```
    while GPIO.input(PIN_ECHO)==0:
        pulse_start_time = time.time()
    while GPIO.input(PIN_ECHO)==1:
        pulse_end_time = time.time()
```

7. The two loops allow us to calculate the time it took for the ultrasonic pulse to be sent and received. To calculate the distance, we just multiply the pulse duration with the rough speed of ultrasonic sound which is **34300 cm/s.** Since the pulse duration is the time it took for the ultrasonic sound to hit an object and bounce back, we will use half the speed to calculate the distance.

```
pulse_duration = pulse_end_time – pulse_start_time

distance = round(pulse_duration * 17150, 2)

print("Distance:" + distance + "cm")
```

8. These final two lines of code are crucial as it ensures an end to our try: statement and ensures that we run **GPIO.cleanup()** when the scrip is terminated in anyway. Failure to do so will throw warnings when rerunning the script.

```
finally:
    GPIO.cleanup()
```

9. Save and run your program to measure the distance between the sensor and object.

Acknowledgement: The lab has been developed using the material available on the internet by James Robinson of the Raspberry Pi Foundation and Gus from PiMyLifeUp.

Useful Links

https://shop.pimoroni.com/products/explorer-hat

Sense Hat Emulator

https://trinket.io/sense-hat
https://www.raspberrypi.org/blog/sense-hat-emulator/

Sensing Weather using Raspberry Pi

https://www.raspberrypi.org/learning/sensing-the-weather/

How HC-SR04 Ultrasonic Sensor

https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/