Experiment 4

Understanding and connectivity of Raspberry-Pi with Distance measuring using Ultrasonic Sensor.

Write an application to measure the distance of the obstacle using Ultrasonic Sensor.

Ultrasonic Sensor

The HC-SR04 Ultrasonic Sensor (or any Ultrasonic Sensor for that matter), works on the principle that is similar to RADAR and SONOR i.e.

transmits a signal and analyzes the target by capturing the reflected signals.

How HC-SR04 Ultrasonic Sensor Works?

It basically consists of three parts: an ultrasonic transmitter, a control circuit and an ultrasonic receiver.

Coming to the pins of the HC-SR04 Sensor, it has only four pins namely VCC, TRIG (Trigger), ECHO (Echo) and GND.

The basic principle behind the Ultrasonic Sensor is described here.

The Ultrasonic Transmitter in the Sensor generates a 40 KHz Ultrasound.

This signal then propagates through air and if there is any obstacle in its path, the signal hits the object and bounces back.

This bounced signal is then collected by the Ultrasonic Receiver.

Based on the signal's time of travel, you can calculate the distance of the object as you already know the speed of sound.

How to Calculate Distance?

In order to send the 40 KHz Ultrasound, the TRIG Pin of the Ultrasonic Sensor must be held

HIGH for a minimum duration of 10µS.

The Ultrasonic Transmitter, will transmits a burst of 8-pulses of ultrasound at 40 KHz.

Immediately, the control circuit in the sensor will change the state of the ECHO pin to HIGH.

This pins stays HIGH until the ultrasound hits an object and returns to the Ultrasonic Receiver.

Based on the Time for which the Echo Pin stays HIGH, you can calculate the distance between the sensor and the object.

For example, if we calculated the time for which ECHO is HIGH as 588µS, then user can calculate the distance with the help of the speed of sound, which is equal to 340m/s.

Distance = Velocity of Sound / (Time/2) = 340m/s / (588µS /2) = 10cm.

NOTE:

Before making the connections, you have to note a point that Raspberry Pi works at 3.3V Logic while the HC-SR04 Ultrasonic Sensor works at 5V.

The Raspberry Pi needs to read the Echo pin to calculate the time and hence the corresponding GPIO pin on the Raspberry Pi must be configured as Input So, before connecting the Echo Pin to the Raspberry Pi, it must be given to a level converter.

Components Required

- Raspberry Pi 3 Model B
- HC-SR04 Ultrasonic Sensor
- 680 Ω Resistor (1/4 Watt)
- 1.5 KΩ Resistor (1/4 Watt)
- Connecting Wires
- Mini Breadboard
- Power Supply
- Computer

Wiring

There are four pins on the ultrasound module that are connected to the Raspberry:

- VCC to Pin 2 (VCC)

  VCC (voltage at the common collector) meaning that it is the connection for the positive voltage source. Many of these breakout board devices can take a range of voltage unless specified 5v, 3.3v, etc.
- GND to Pin 6 (GND)

- TRIG to Pin 12 (GPIO18) –(10us High pulse given to tell the sensor to sense the distance)
- connect the 330Ω resistor to ECHO.
- On its end you connect it to Pin 18 (GPIO24) and through a 470Ω resistor you connect it also to Pin6 (GND).

User do this because the GPIO pins only tolerate maximal 3.3V.

The connection to GND is to have a obvious signal on GPIO24. If no pulse is sent, the signal is 0 (through the connection with GND), else it is 1.

If there would be no connection to GND, the input would be undefined if no signal is sent (randomly 0 or 1), so ambiguous.

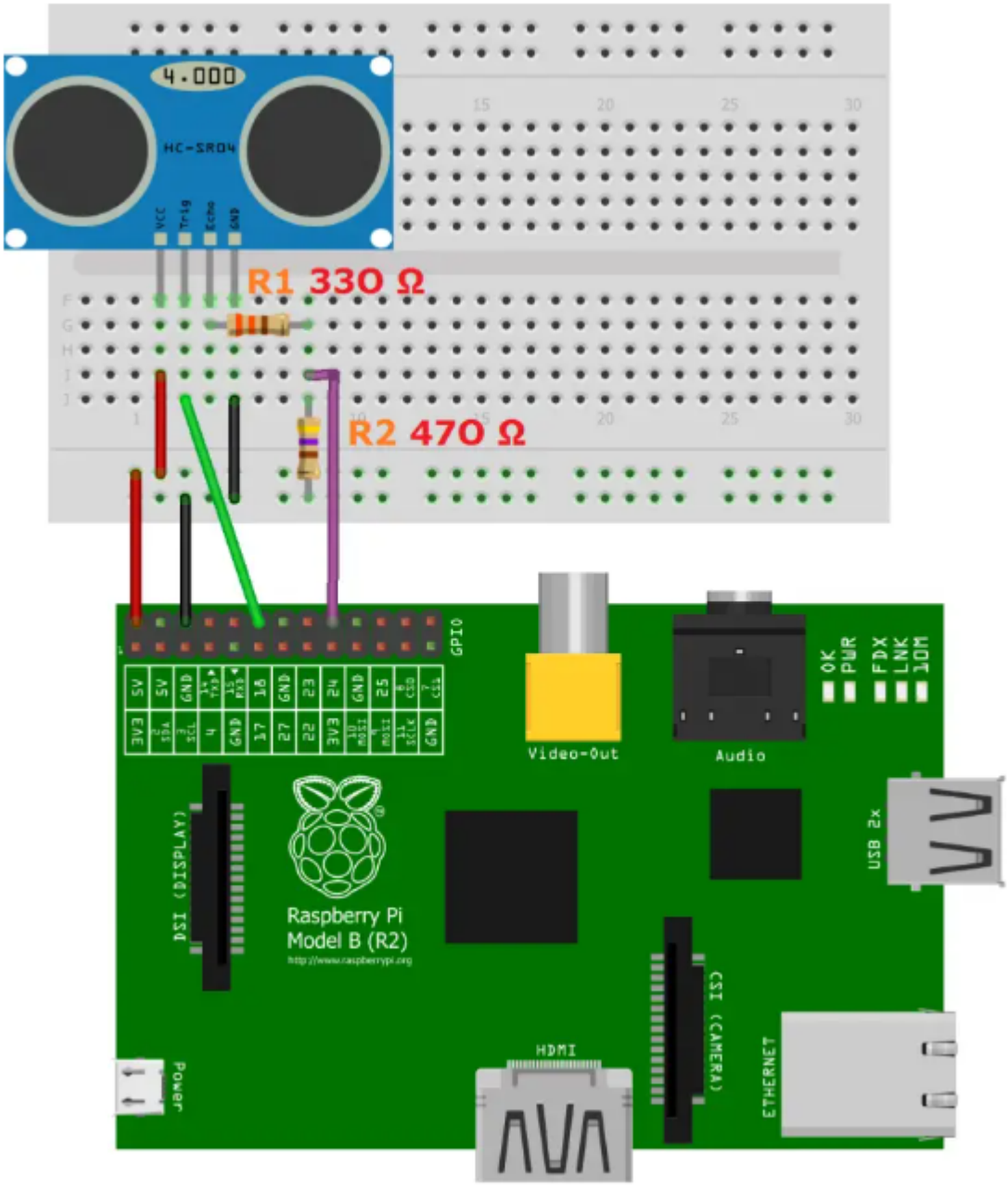
Circuit Design

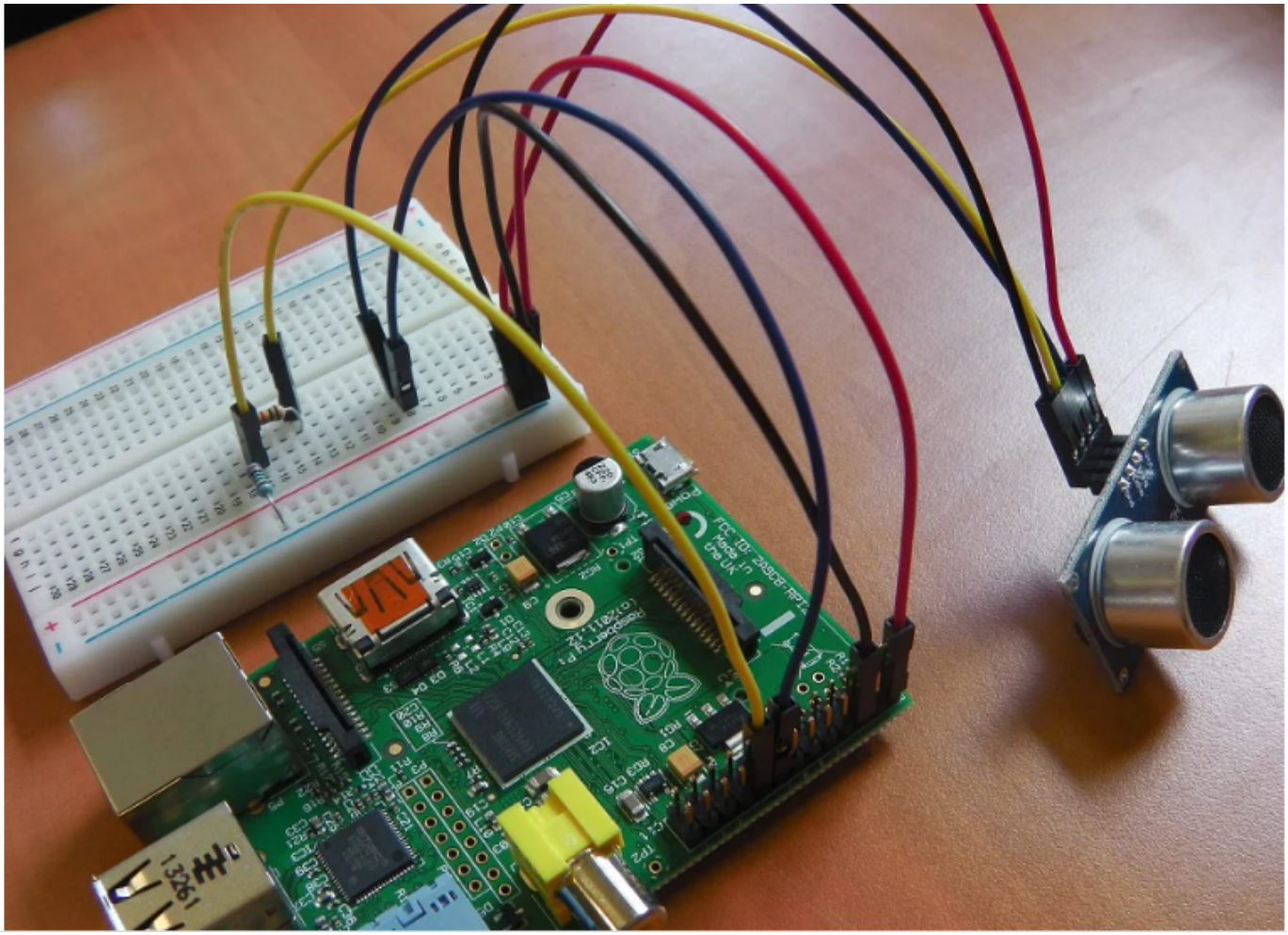
Connect the Trig Pin of the HC-SR04 Ultrasonic Sensor to the Physical Pin 16 i.e. GPIO23 of the Raspberry Pi.

Use a combination of 680Ω and 1.5 KΩ Resistor to convert the Echo pin to 3.3V Logic (approximately)

Connect it to Physical Pin 18 i.e. GPIO24 of the Raspberry Pi.

Finally, provide the +5V and GND connections to the Ultrasonic Sensor from the Raspberry Pi Pins.


4.000
HC-SR04
VCC
Trig
Echo
GND
R1 330 Ω
R2 470 Ω
GPIO
Video-Out
Audio
OK
PWR
FDX
LNK
10M
DSI (DISPLAY)
Raspberry Pi
Model B (R2)
USB 2x
CSI (CAMERA)
HDMI
ETHERNET
POWER



Script for controlling

First of all, the Python GPIO library should be installed

To use the module, user create a new script

sudo nano ultrasonic_distance.py

Python Code

```
#Libraries
import RPi.GPIO as GPIO
import time

#GPIO Mode (BOARD / BCM)
GPIO.setmode(GPIO.BCM)

#set GPIO Pins
```

```
GPIO_TRIGGER = 18
GPIO_ECHO = 24

#set GPIO direction (IN / OUT)
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)

def distance():
    # set Trigger to HIGH
    GPIO.output(GPIO_TRIGGER, True)

    # set Trigger after 0.01ms to LOW
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)

    StartTime = time.time()
    StopTime = time.time()

    # saveStartTime
    whileGPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()

    # save time of arrival
    whileGPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()

    # time difference between start and arrival
    TimeElapsed = StopTime - StartTime
    # multiply with the sonic speed (34300 cm/s)
    # and divide by 2, because there and back
    distance = (TimeElapsed * 34300) / 2

    return distance
```

```
if __name__ == '__main__':
    try:
        while True:
            dist = distance()
            print ("Measured Distance = %.1f cm" % dist)
            time.sleep(1)

        # Reset by pressing CTRL + C
    except keyboardInterrupt:
            print("Measurement stopped by User")
        GPIO.cleanup()
```

After that we run:

```
sudo python ultrasonic_distance.py
```

So every second, the distance will be measured until the script is cancelled by pressing CTRL + C.

Applications

In this project, we have seen how to interface HC-SR04 Ultrasonic Sensor with Raspberry Pi. This setup can be used in a lot applications like:

- Obstacle Avoiding
- Proximity Detection
- Distance Measurement
- Range Meter

Method 2

Assemble the Circuit

User will be using four pins on the Raspberry Pi for this experiment:

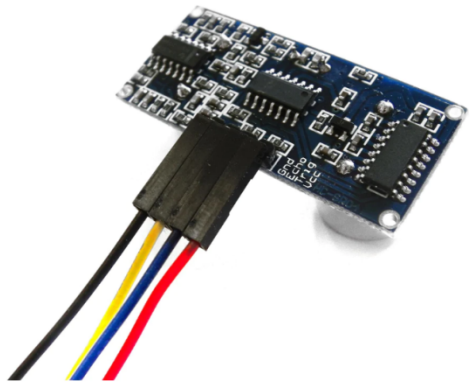
GPIO 5V [Pin 2] - Vcc (5V Power),

GPIO GND [Pin 6] - GND (0V Ground),

GPIO 23 [Pin 16] - TRIG (GPIO Output) and

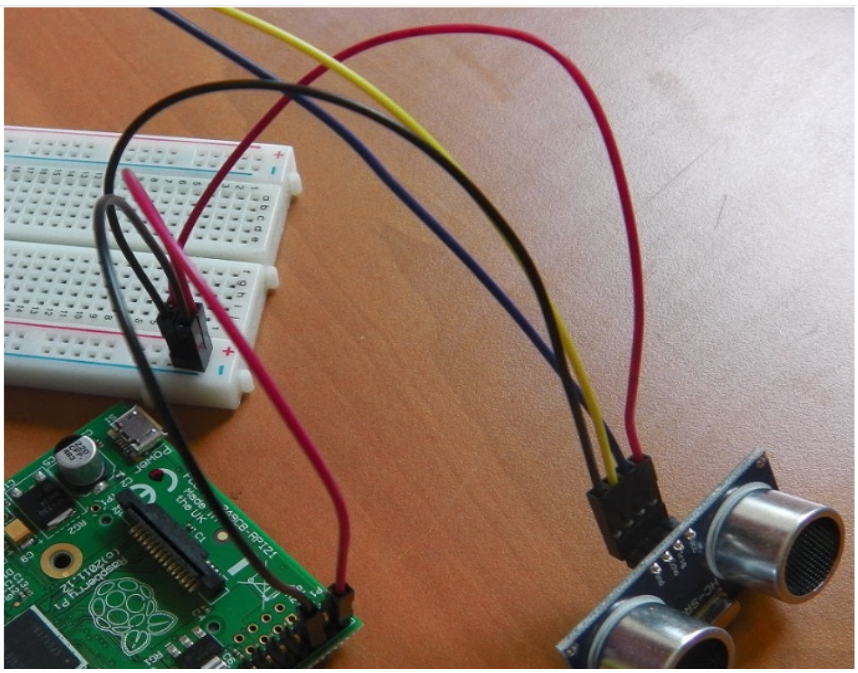
GPIO 24 [Pin 18] - ECHO (GPIO Input)

1. Plug four of your male to female jumper wires into the pins on the HC-SR04 as follows: Red; Vcc, Blue; TRIG, Yellow; ECHO and Black; GND.



2. Plug Vcc into the positive rail of your breadboard, and plug GND into your negative rail.

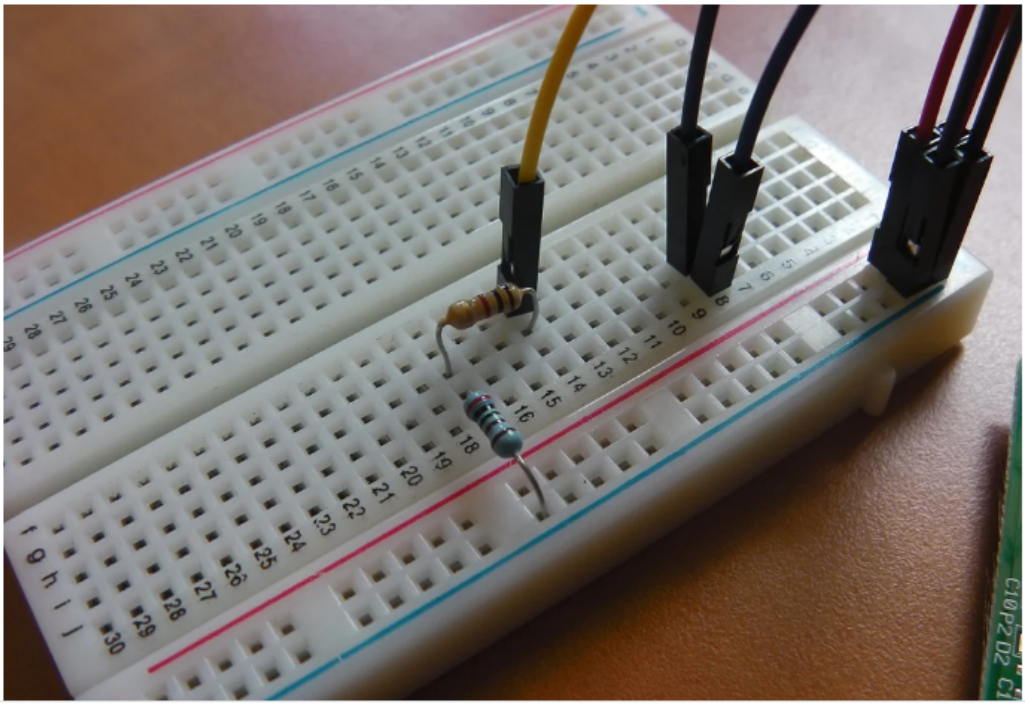
3. Plug GPIO 5V [Pin 2] into the positive rail, and GPIO GND [Pin 6] into the negative rail.



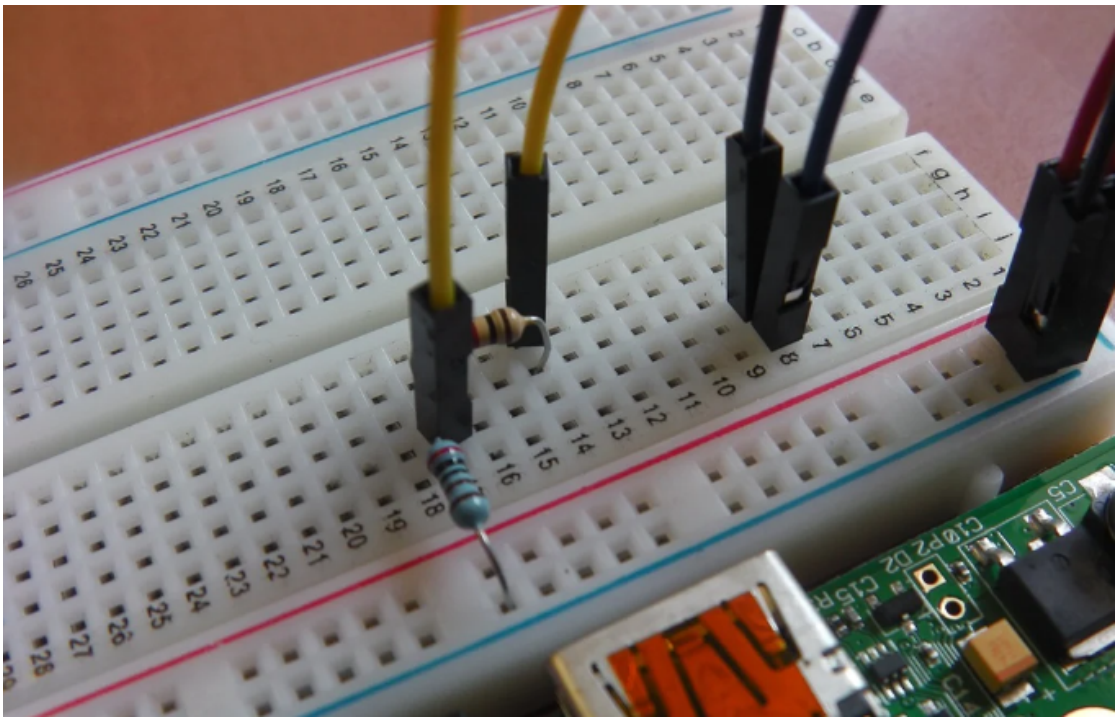
1. Plug TRIG into a blank rail, and plug that rail into GPIO 23 [Pin 16].

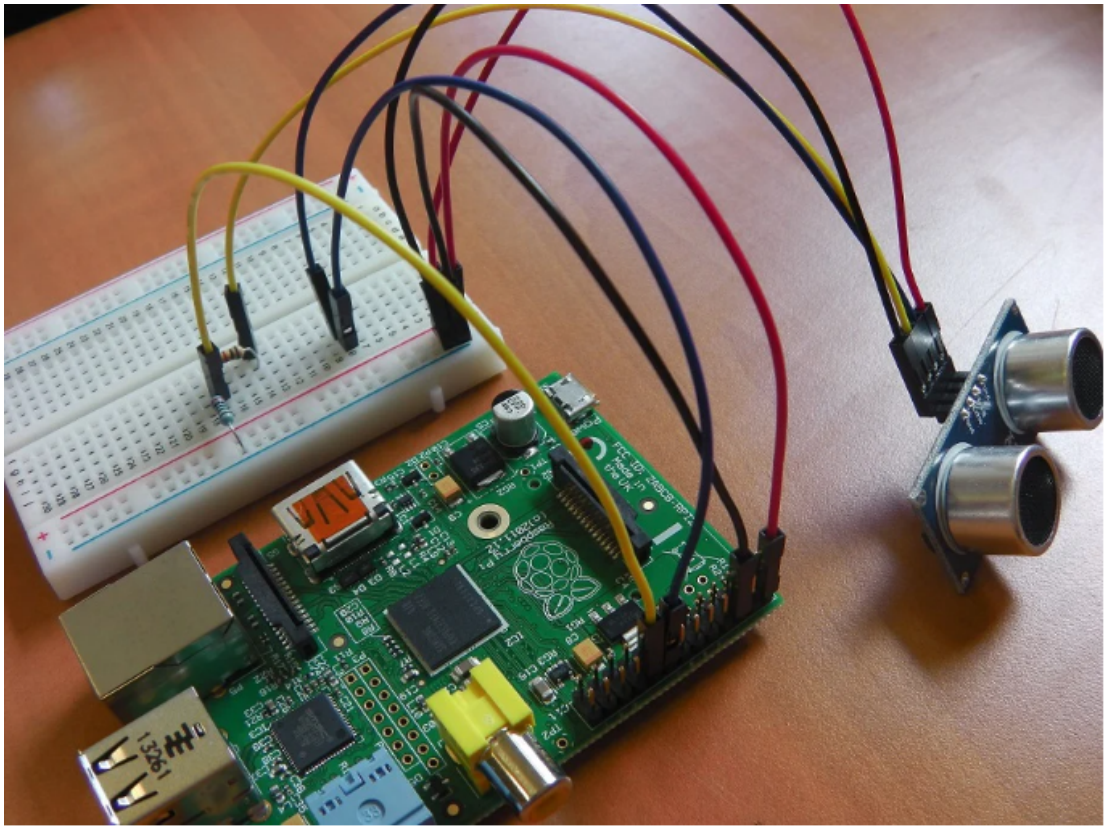
2. Plug ECHO into a blank rail, link another blank rail using R1 (1kΩ resistor)

3. Link your R1 rail with the GND rail using R2 (2kΩ resistor). Leave a space between the two resistors.



4. Add GPIO 24 [Pin 18] to the rail with your R1 (1kΩ resistor). This GPIO pin needs to sit between R1 and R2





Python Code

```
# import the Python GPIO library, import our time library
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
# Next, we need to name our input and output pins, We'll name our output pin (which triggers the sensor) GPIO 23 [Pin 16] as TRIG, and our input pin (which reads the return signal from the sensor) GPIO 24 [Pin 18] as ECHO.
TRIG = 23
ECHO = 24
# print a message to let the user know that distance measurement is in progress
print "Distance Measurement In Progress"
# set your two GPIO ports as either inputs or outputs as defined previously.
GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)

# Then, ensure that the Trigger pin is set low, and give the sensor a second to settle.
GPIO.output(TRIG, False)
```

```
print "Waiting For Sensor To Settle"
time.sleep(2)

# The HC-SR04 sensor requires a short 10uS pulse to trigger the module, which will cause the sensor to start the ranging program (8 ultrasound bursts at 40 kHz) in order to obtain an echo response. So, to create our trigger pulse, we set out trigger pin high for 10uS then set it low again.
GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)

#The time.time() function will record the latest timestamp for a given condition. For example, if a pin goes from low to high,
while GPIO.input(ECHO)==0:
  pulse_start = time.time()
# Once a signal is received, the value changes from low (0) to high (1), and the signal will remain high for the duration of the echo pulse. We therefore also need the last high timestamp for ECHO (pulse_end).
while GPIO.input(ECHO)==1:
  pulse_end = time.time()
# We can now calculate the difference between the two recorded timestamps, and hence the duration of pulse (pulse_duration).
pulse_duration = pulse_end - pulse_start
distance = pulse_duration x 17150

# Now we need to round our distance to 2 decimal places (for neatness!)
distance = round(distance, 2)

#Then, we print the distance. The below command will print the word “Distance:” followed by the distance variable, followed by the unit “cm”
print "Distance:",distance,"cm"

# Finally, we clean our GPIO pins to ensure that all inputs/outputs are reset
GPIO.cleanup()
```

```
pi@raspberrypi ~ $ sudo python range_sensor.py
Distance Measurement In Progress
Waiting For Sensor To Settle
Distance: 12.52 cm
pi@raspberrypi ~ $
```

Save your python script, I called ours "range_sensor.py", and run it using the following command. Running a root (sudo), is important with this script:

With the time it takes for the signal to travel to an object and back again, we can calculate the distance using the following formula.

$$Speed = \frac{Distance}{Time}$$

some physicists have calculated the speed of sound at sea level so we'll take our baseline as the 343m/s.