

INTERNET OF THINGS

ASSIGNMENT - 3

PREPARED BY

Garv Baheti

(2020BTechCSE031)

SUBMITTED TO

Mr. Divanshu Jain



JKLU

NAAC 'A' Grade Accredited

**Department of Computer Science Engineering
Institute of Engineering and Technology
JK Lakshmipat University Jaipur**

October 03, 2022

OBJECTIVE

1. Interfacing Ultra Sonic Sensor with the help of RaspberryPi, displaying output on and sending reading to ThingSpeak Server.
2. Interface LM35 sensor with Raspberry Pi using ADC and show the value on serial monitor.

INTERFACING ULTRA SONIC SENSOR WITH THE HELP OF RASPBERRYPI, DISPLAYING OUTPUT ON AND SENDING READING TO THINGSPEAK SERVER

Hardware

- HC-SR04 Module
- Resistors
- Jumper wire
- RaspberryPi

Wiring

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

- VCC to Pin 2 (VCC)
- GND to Pin 6 (GND)
- TRIG to Pin 12 (GPIO18)
- 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).

We 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.

SETTING UP THINGSPEAK ACCOUNT FOR LM35 SENSOR:

Before we proceed towards construction details we need to setup our thingspeak account correctly to receive the sensor data, this procedure need to be done for all three methods mentioned here.

You can sign up for [thingspeak account here](#), if you haven't signed up yet.

- **You need to create a new channel** by clicking “New Channel” button and in the channel.

Ultra-Sonic Sensor Data to Thingspeak

- 1) Go to Channel settings tab and edit the name.
- 2) Enable Field 1 by checking the box and write the label as “Distance”.
- 3) Scroll down and click save.

- Now click on API keys tab:

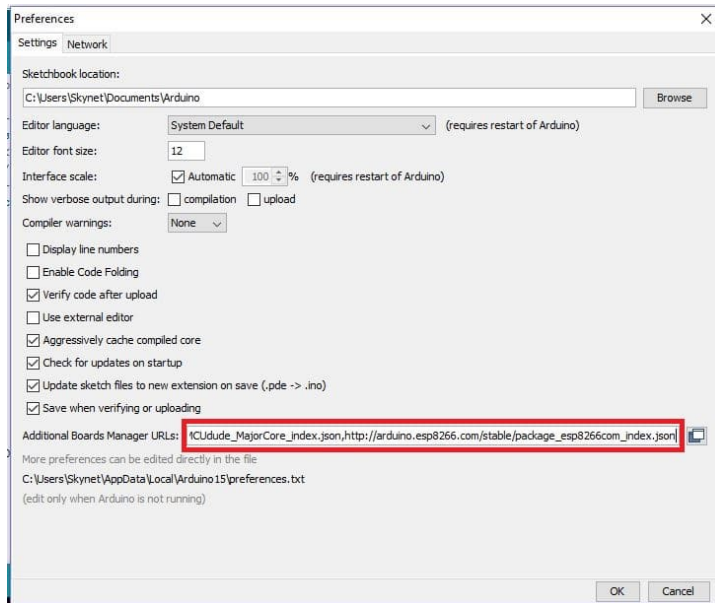
API keys are the access keys to your channel using which you can write and read values. In this project we are using only write API key which is already generated for your channel. You need to take note of it and this write API key need to be inserted to the given program codes.

- Now by clicking “Private View” tab you will see a blank channel. You will see some data once we send to it.

Installing ESP8266 package to Arduino IDE:

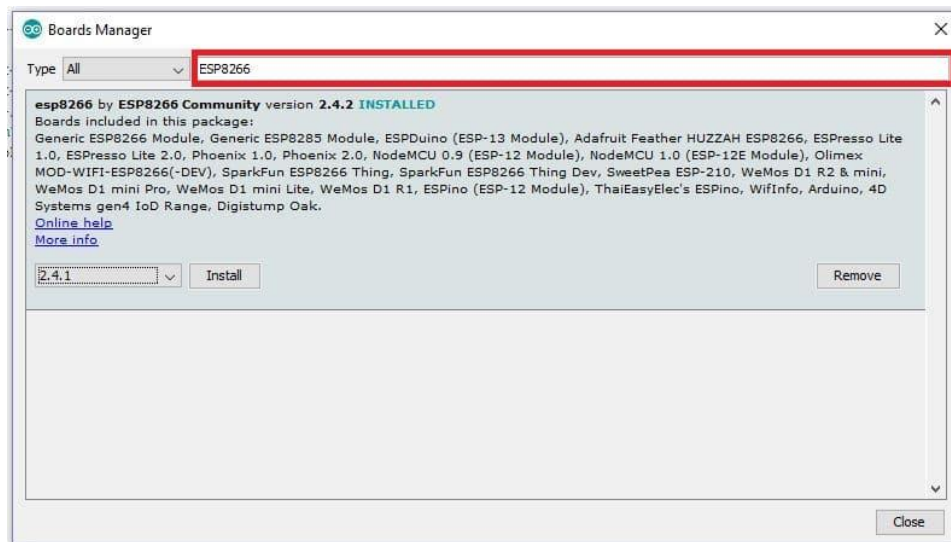
This process needs to be done if you are using NodeMCU or generic ESP8266 to connect to thingspeak. If you are going to use GSM modem to send LM35 data, this step is irrelevant for you, but **for NodeMCU and ESP8266 boards this step is mandatory.**

- In this step you need internet; we are going to install core files to Arduino IDE for IoT based boards.
- Copy this
link: http://arduino.esp8266.com/stable/package_esp8266com_index.json
- Now open Arduino IDE and click on “**File**” >> “**Preferences**”.
- A window will open like this:



Preferences

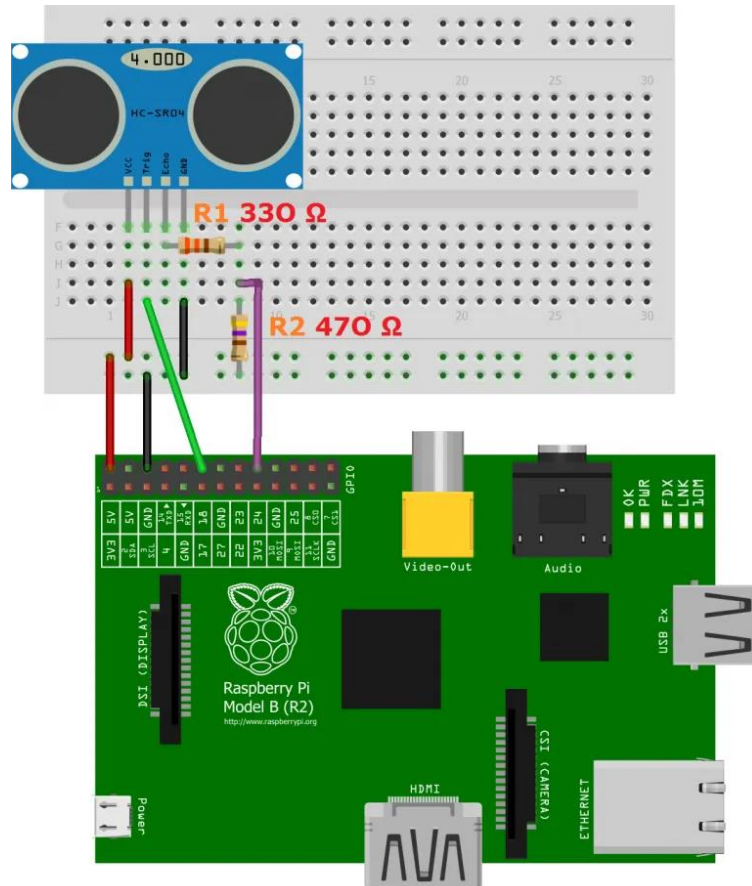
- Paste the URL on the box and click “OK”.
- Now go to **tools > Board > Boards Manager**.
- Now a window will popup:



boards manager

- Type ESP8266 on the box as shown and you will get an installation option, select the latest version and click install.

CIRCUIT

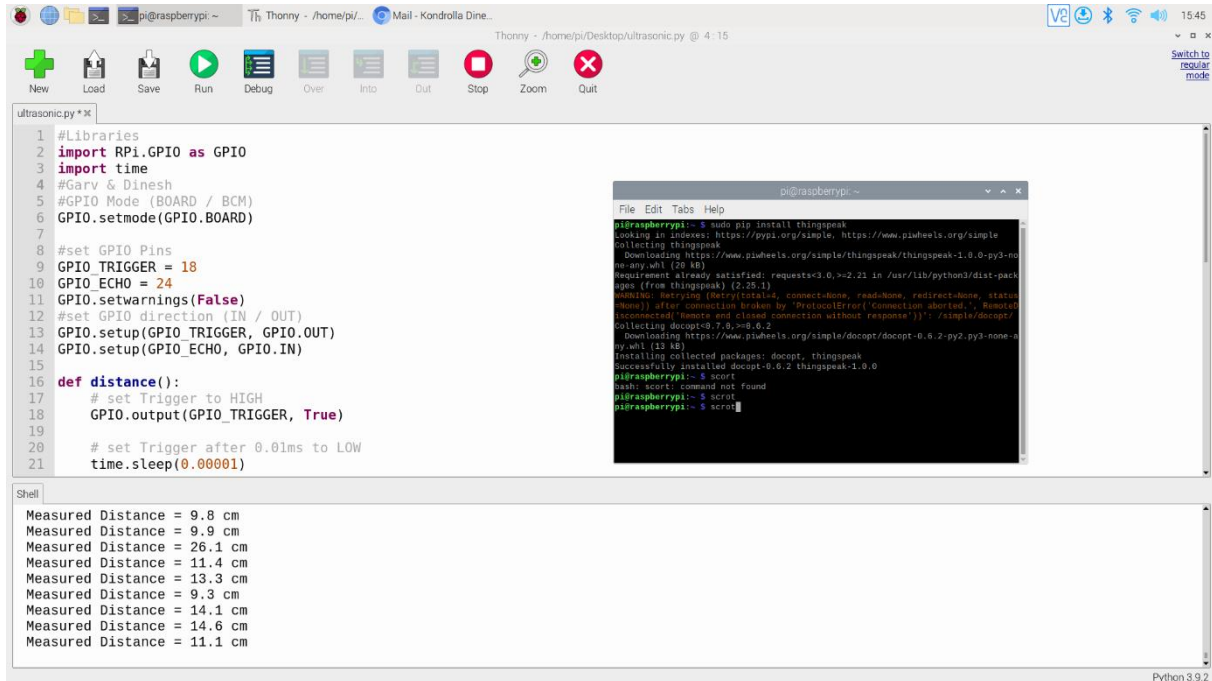


CODE

```
1 #Libraries
2 import RPi.GPIO as GPIO
3 import time
4 import thingspeak
5 channel_id = 1881317
6 write_key = 'C1NR4JF75NEV23XY'
7 read_key = 'QYD3TR3LDFTLRJ5D'
8
9 #GPIO Mode (BOARD / BCM)
10 GPIO.setmode(GPIO.BOARD)
11
12 #set GPIO Pins
13 GPIO_TRIGGER = 18
14 GPIO_ECHO = 24
15 GPIO.setsarnings(False)
16
17 #set GPIO direction (IN / OUT)
18 GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
19 GPIO.setup(GPIO_ECHO, GPIO.IN)
20
21 def distance():
22     # set Trigger to HIGH
23     GPIO.output(GPIO_TRIGGER, True)
24
25     # set Trigger after 0.01ms to LOW
26     time.sleep(0.00001)
27     GPIO.output(GPIO_TRIGGER, False)
28
29     StartTime = time.time()
30     StopTime = time.time()
31
32     # save StartTime
33     while GPIO.input(GPIO_ECHO) == 0:
34         StartTime = time.time()
35
36     # save time of arrival
37     while GPIO.input(GPIO_ECHO) == 1:
38         StopTime = time.time()
39
40     # time difference between start and arrival
41     TimeElapsed = StopTime - StartTime
42     # multiply with the sonic speed (34300 cm/s)
43     # and divide by 2, because there and back
44     distance = (TimeElapsed * 34300) / 2
45
46     return distance
47
48 if __name__ == '__main__':
49     try:
50         while True:
51             dist = distance()
52             channel = thingspeak.Channel(channel_id, write_key, read_key)
53             response = channel.update({'field1': dist})
54             print ("Measured Distance = %.1f cm" % dist)
55             time.sleep(1)
56
57         # Reset by pressing CTRL + C
58     except KeyboardInterrupt:
59         print("Measurement stopped by User")
60         GPIO.cleanup()
```

OUTPUT

Output on RaspberryPi

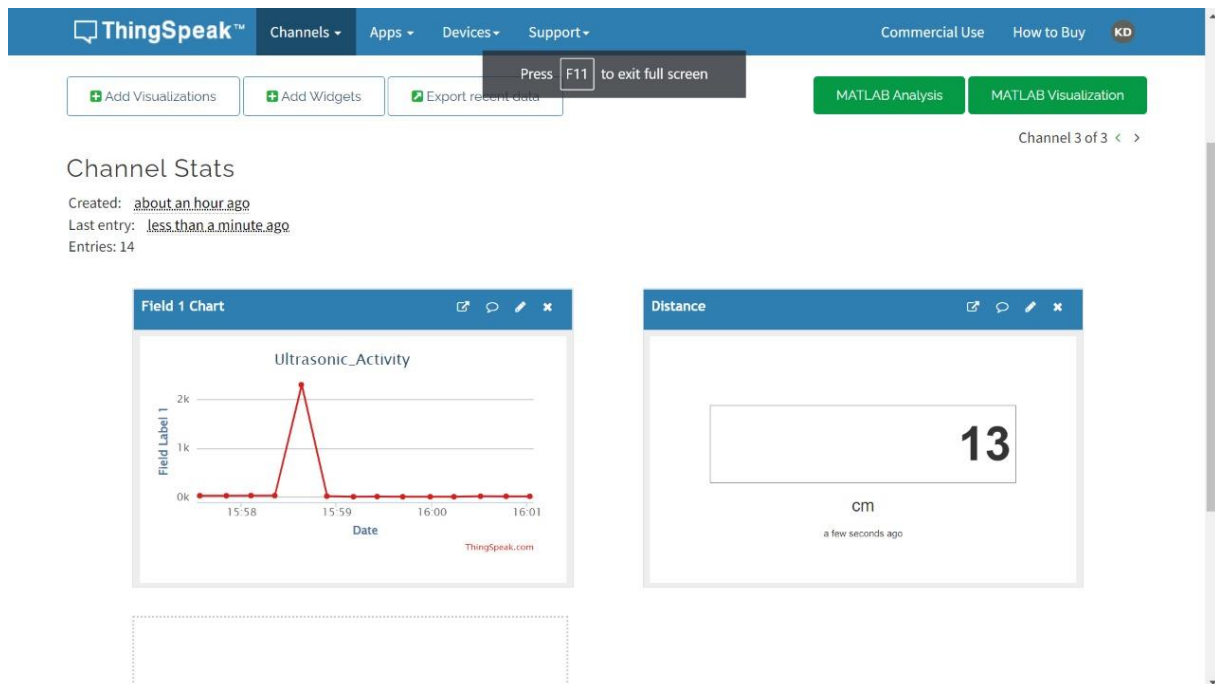


```
1 #Libraries
2 import RPi.GPIO as GPIO
3 import time
4 #Garv & Dinesh
5 #GPIO Mode (BOARD / BCM)
6 GPIO.setmode(GPIO.BOARD)
7
8 #set GPIO Pins
9 GPIO_TRIGGER = 18
10 GPIO_ECHO = 24
11 GPIO.setwarnings(False)
12 #set GPIO direction (IN / OUT)
13 GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
14 GPIO.setup(GPIO_ECHO, GPIO.IN)
15
16 def distance():
17     # set Trigger to HIGH
18     GPIO.output(GPIO_TRIGGER, True)
19
20     # set Trigger after 0.01ms to LOW
21     time.sleep(0.00001)
```

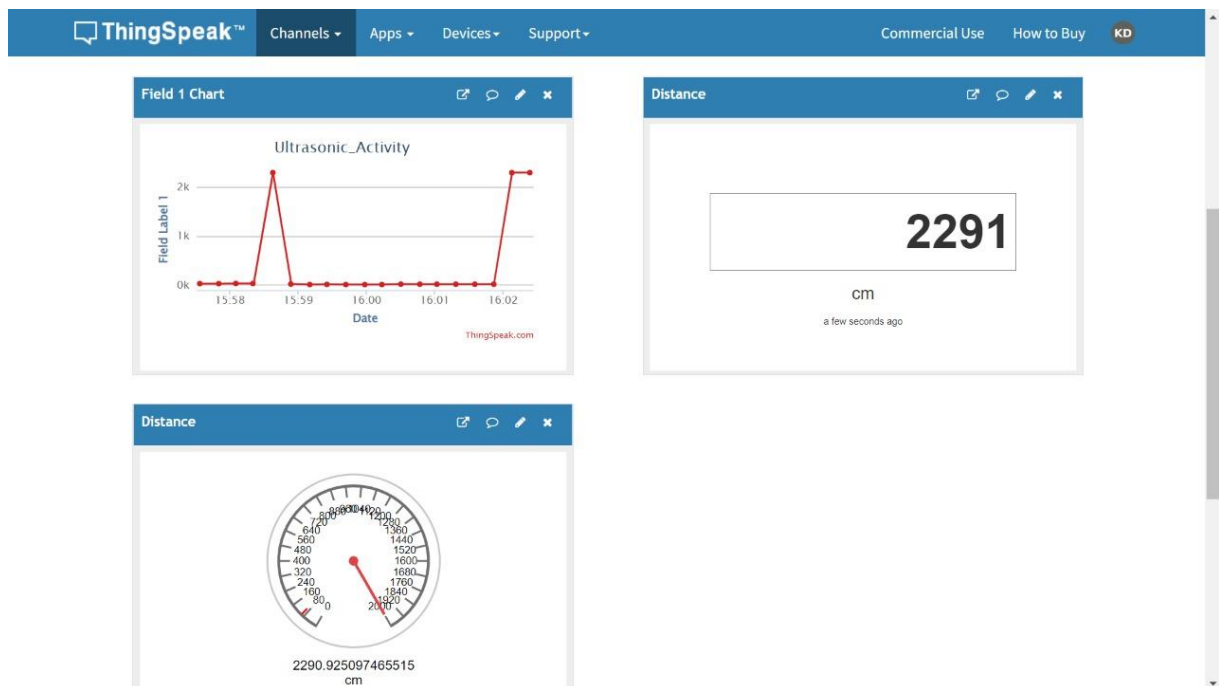
Measured Distance = 9.8 cm
Measured Distance = 9.9 cm
Measured Distance = 26.1 cm
Measured Distance = 11.4 cm
Measured Distance = 13.3 cm
Measured Distance = 9.3 cm
Measured Distance = 14.1 cm
Measured Distance = 14.6 cm
Measured Distance = 11.1 cm

Output on ThingSpeak Server

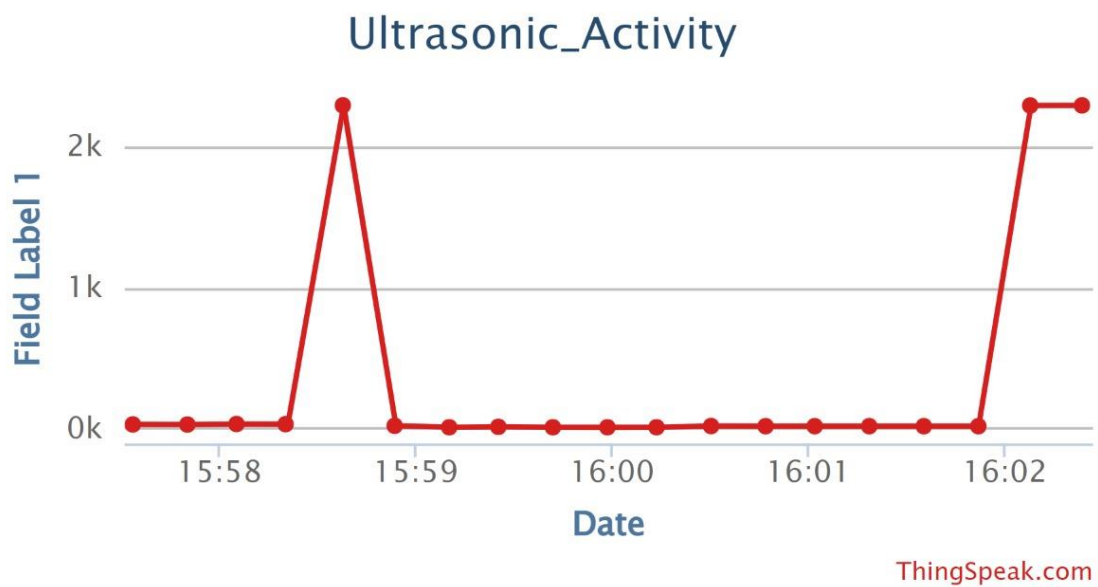
When object was in front of Sensor



When there was no object in front of sensor



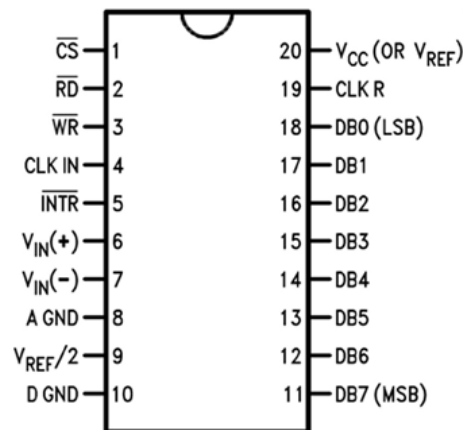
Graph of all the readings sent to Server



INTERFACE LM35 SENSOR WITH RASPBERRY PI USING ADC AND SHOW THE VALUE ON SERIAL MONITOR

ADC0804 AND RASPBERRY PI:

[ADC0804](#) is a chip designed to convert analog signal into 8 bit digital data. This chip is one of the popular series of ADC. It's an 8bit conversion unit, so we have values from 0 to 255 values. The resolution of this chip changes based on the reference voltage we choose, we will talk more about it later. Below is the **Pinout of ADC0804**:

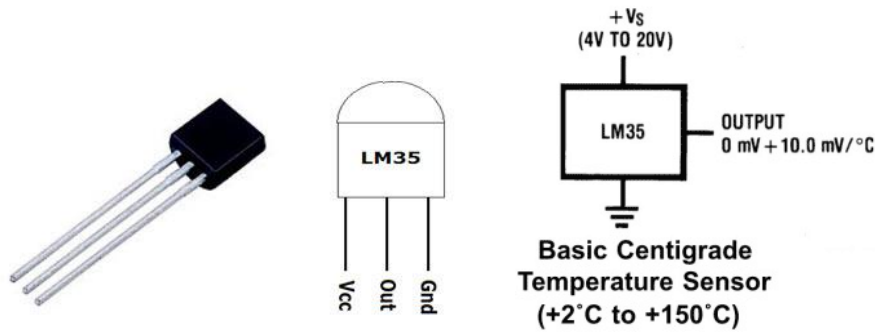


Now another important thing here is, the **ADC0804 operates at 5V** and so it provides output in 5V logic signal. In 8 pin output (representing 8bits), every pin provides +5V output to represent logic '1'. So the problem is the PI logic is of +3.3v, so you cannot give +5V logic to the +3.3V GPIO pin of PI. If you give +5V to any GPIO pin of PI, the board gets damaged.

So to step-down logic level from +5V, we will be using voltage divider circuit. We have discussed Voltage Divider Circuit previously look into it for further clarification. What we will do is, we use two resistors to divide +5V logic into 2*2.5V logics. So after division we will give +2.5v logic to PI. So, whenever logic '1' is presented by ADC0804 we will see +2.5V at the PI GPIO Pin, instead of +5V.

LM35 Temperature Sensor:

Now for **Reading Temperature of Room**, we need a sensor. Here we are going to use **LM35 Temperature Sensor**. Temperature is usually measured in "Centigrade" or "Fahrenheit". "LM35" sensor provides output in degree Centigrade.



As shown in figure, LM35 is a three pin transistor like device. The pins are numbered as,

PIN1= Vcc - Power (Connected to +5V)

PIN2= Signal or Output (connected to ADC chip)

PIN3 = Ground (Connected to ground)

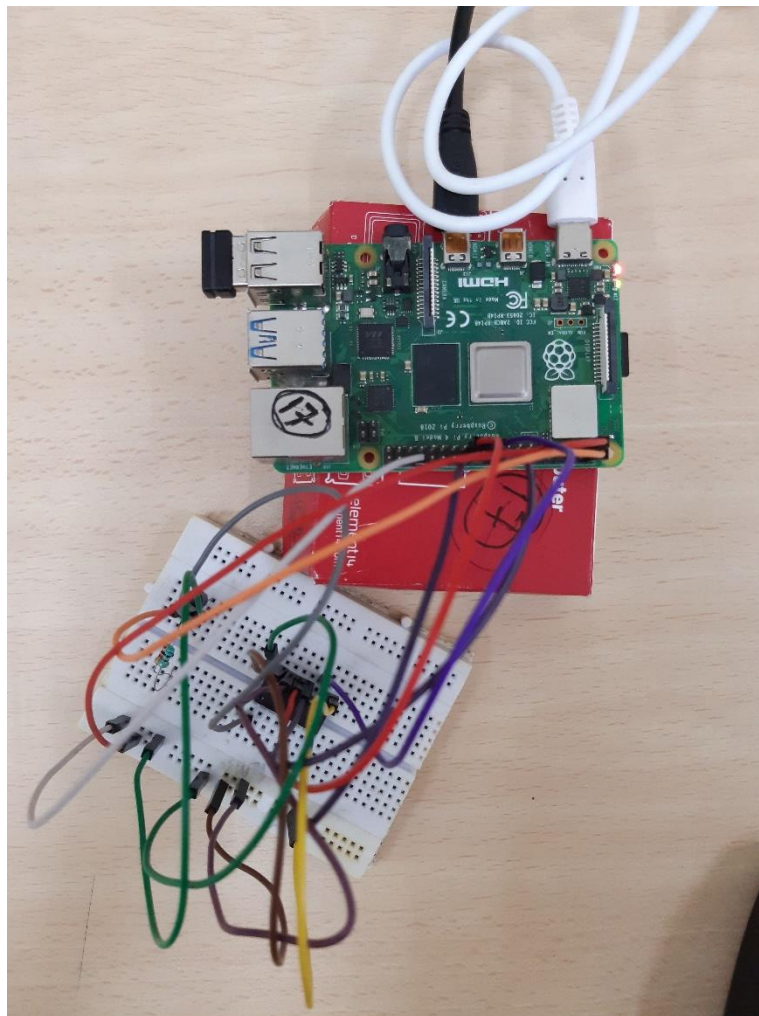
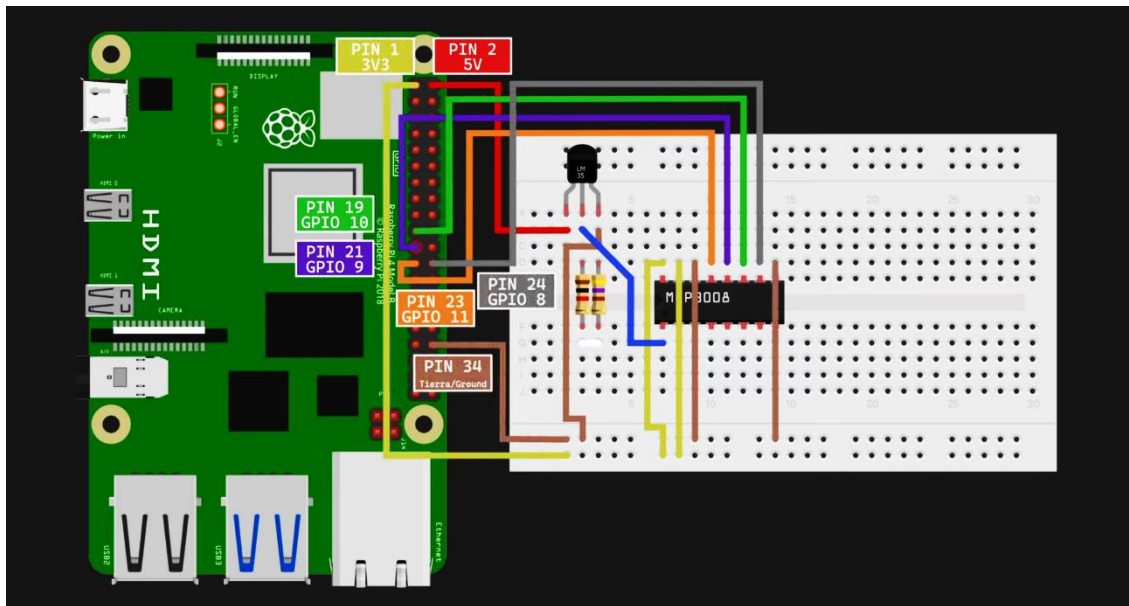
This sensor provides variable voltage at the output, based on temperature. For every +1 centigrade rise in temperature there will be +10mV higher voltage at the output pin. So if the temperature is 0° centigrade the output of sensor will be 0V, if the temperature is 10° centigrade the output of sensor will be +100mV, if the temperature is 25° centigrade the output of sensor will be +250mV.

COMPONENTS REQUIRED

Here we are using **Raspberry Pi 2 Model B with Raspbian Jessie OS**. All the basic Hardware and Software requirements are previously discussed, you can look it up in the Raspberry Pi Introduction, other than that we need:

- Connecting pins
- 10K pot
- ADC0804 IC
- LM35 Temperature Sensor
- Bread Board

CIRCUIT



CODE

```
1 #-*- coding: utf-8 -*-
2 from gpiozero import MCP3008
3 from time import sleep
4
5 # Set up channel number and SPI chip select device
6 reading = MCP3008(channel=0)
7
8 while True:
9     # Converts ACD voltage to temperature in Celsius
10    temp_c = round((reading.value * 3.3) * 100, 2)
11
12    # Convert Celsius degrees to Farenheit
13    temp_f = round(temp_c * 1.8 + 32, 2)
14
15    # Print both temperatures
16    print('Temp: {}°C  {}°F'.format(temp_c, temp_f))
17
18    sleep(1.5) # Wait 1.5 seconds for the next read    StopTime = time.time()
19
20
```

OUTPUT

