RV College of Engineering®

Go, change the world

IOT and APPLICATIONS
IS224AI-UG 4th sem
2022 scheme

Unit-V

Reference Book: Internet Of Things With Raspberry Pi And Arduino, Rajesh Singh, Anita Gehlot, Lovi Raj Gupta, Bhupendra Singh, and Mahendra Swain, CRC Press, Taylor & Francis Group, 2020, ISBN: 13: 978-0-367-24821-5

By,
Dr. G S Mamatha
Professor & Associate Dean(PG Studies)
Department of ISE
R V College of Engineering

- The objective of this project is to capture the real-time data of DHT11 by Raspberry Pi and upload to the cloud. DHT11 is a temperature and humidity sensor.
- ThingSpeak is used as a cloud server for data forecasting.
- The process involves three parts: setting up Raspberry Pi, configuring Adafruit library, and then connecting to ThingSpeak.
- The system is comprised of a Raspberry Pi, a power supply, and a DHT11 sensor.
- Connect pin (Vcc) of DHT11 to +5V and ground, respectively.
- Connect pin(OUT) of DHT11 to GPIO17 of the Raspberry Pi

DHT11-Pi4 code


```
import sys                          # import sys library
importRPi.GPIO as GPIO   # import GPIO library
import time as wait              # add time library
Import Adafruit_DHT as GPIO_DHT
import urllib2                      #import urllib library
my_API = 'F1MAEF943TLMVTC1'
def  sensor_data():
HUM, TEMP=
GPIO_DHT.read_retry(GPIO_DHT.DHT11, 21)
return (str(HUM), str(TEMP))
def   main_function():
base_URL =
'https://api.thingspeak.com/update?api_key=
%s' % my_API
while True:                                  # infinite loop
try:
HUM, TEMP = sensor_data()    # read value of
                                                Temperature and Humidity
f = urllib2.urlopen(base_URL + '&field1=%s
&field2=%s' % (HUM,
TEMP))
printf.read()                                # print function value
f.close()                                    # close function
wait.sleep(15)                               # delay of 15 Sec
except:
print 'exit from program' # print string on terminal
break
# calling main function
if __name__ == '__main_function__':
main_function()
```

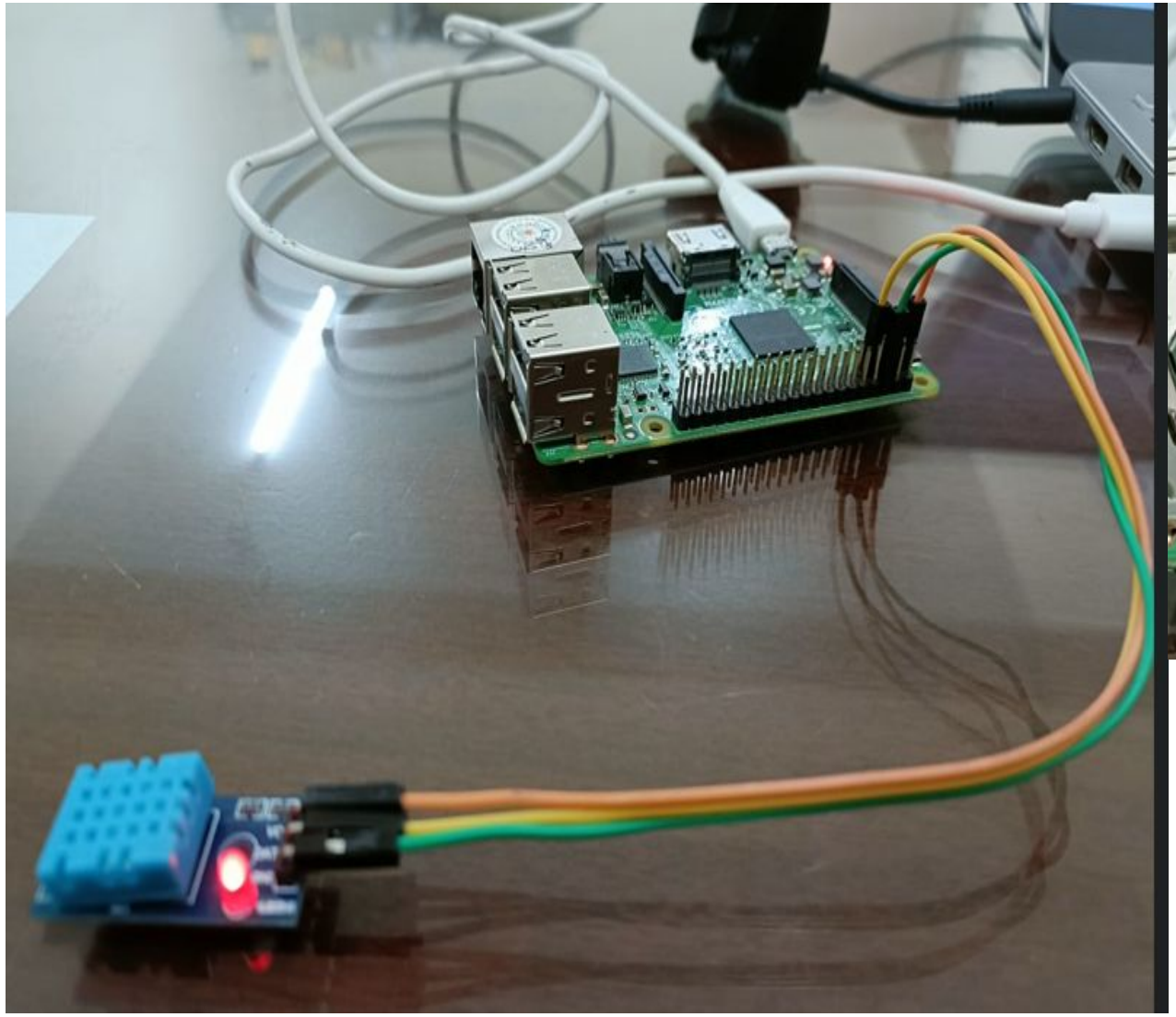
DHT11 with Thingspeak Cloud-Pi3 code


```
# uses classes  for http client side
import http.client
#parses URL string and uses http url scheme
import urllib.parse
import time
import RPi.GPIO as GPIO
import Adafruit_DHT


#GPIO.setmode(GPIO.BOARD)
#GPIO.setup(7,GPIO.IN)
GPIO.setwarnings(False)
key = "TCEHHDU8JBPN2ZNM"  # Put your API Key here


def dht():
        #senses light density and switch on/off with buzzer ring
    #printing the value from ldr module
  while True:
        humidity, temperature = Adafruit_DHT.read_retry(11, 4)
        print('Temp: {0:0.1f}C Humidity:{1:0.1f}%'.format(temperature,
humidity))
 params = urllib.parse.urlencode({'field1':humidity,
'field2':temperature,'key':key })
headers = {"Content-typZZe":
"application/x-www-form-urlencoded","Accept": "text/plain"}
      #create instances that connect to the HTTP server at the
same host and port
          conn =
http.client.HTTPConnection("api.thingspeak.com:80")
          try:
              conn.request("POST", "/update", params, headers)
              response = conn.getresponse()
              #print("temperature, humidity")
              print(response.status, response.reason)
              data = response.read()
              conn.close()
          except:
              print("connection failed")
          break
if __name__ == "__main__":
      while True:
          dht()
```

- The URL parsing functions focus on splitting a URL string into its components, or on combining URL components into a URL string (addressing scheme, network location, path etc.), to combine the components back into a URL string, and to convert a “relative URL” to an absolute URL given a “base URL.”
- urlencode() to encode a query with multiple parameters at **once**. **Call urllib. parse**. urlencode(query) with query as a dictionary of key-value pairs or a sequence of two-element tuples to return query as a percent-encoded ASCII text string.
- headers = {"Content-typZZe": "application/x-www-form-urlencoded","Accept": "text/plain"}

- `application/x-www-form-urlencoded`: the keys and values are encoded in key-value tuples separated by `'&'`, with a `'='` between the key and the value. Non-alphanumeric characters in both keys and values are percent encoded: this is the reason why this type is not suitable to use with binary data (use `multipart/form-data` instead)
- `multipart/form-data`: each value is sent as a block of data ("body part"), with a user agent-defined delimiter ("boundary") separating each part. The keys are given in the `Content-Disposition` header of each part.

Circuit



GND- Pin 6
Vcc- Pin1
Data- Pin 7 (GPIO 4)

RV College of Engineering®

Thingspeak cloud Data Logger

Go, change the world

ThingSpeak™ Channels Apps Devices Support Commercial Use How to Buy MG

DHTCloud

Channel ID: **2613928**
Author: mamathags
Access: Private

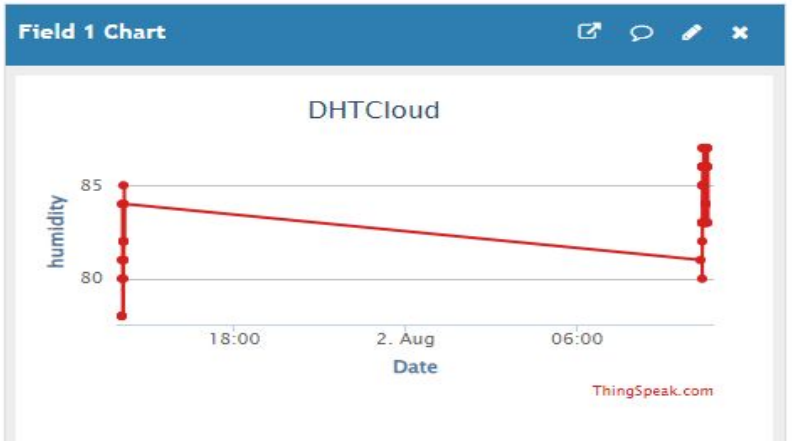
Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export

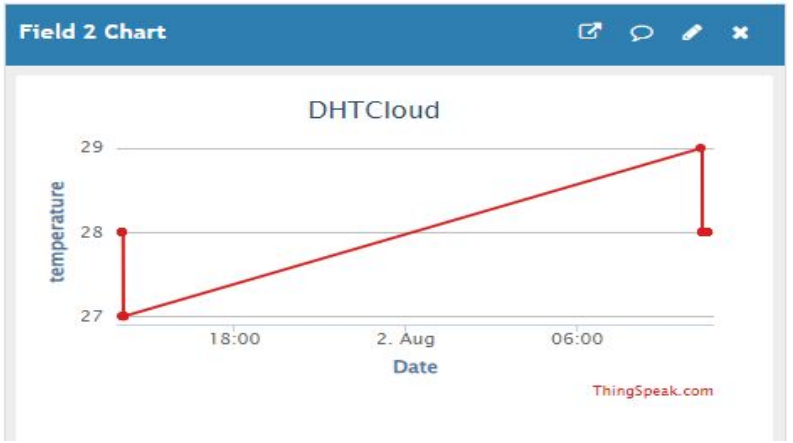
Add Visualizations | Add Widgets | Export recent data | MATLAB Analysis | MATLAB Visualization

Channel 2 of 3 < >

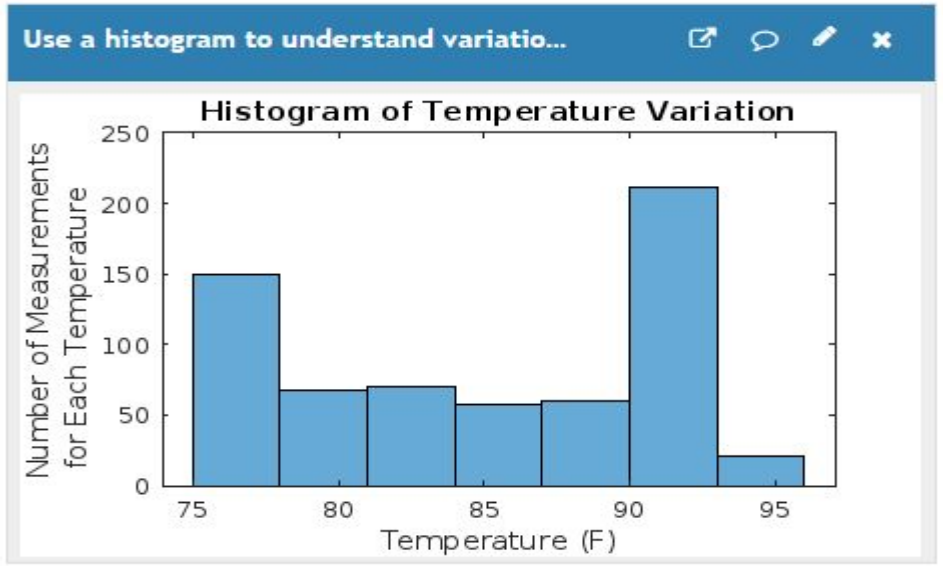
Channel Stats

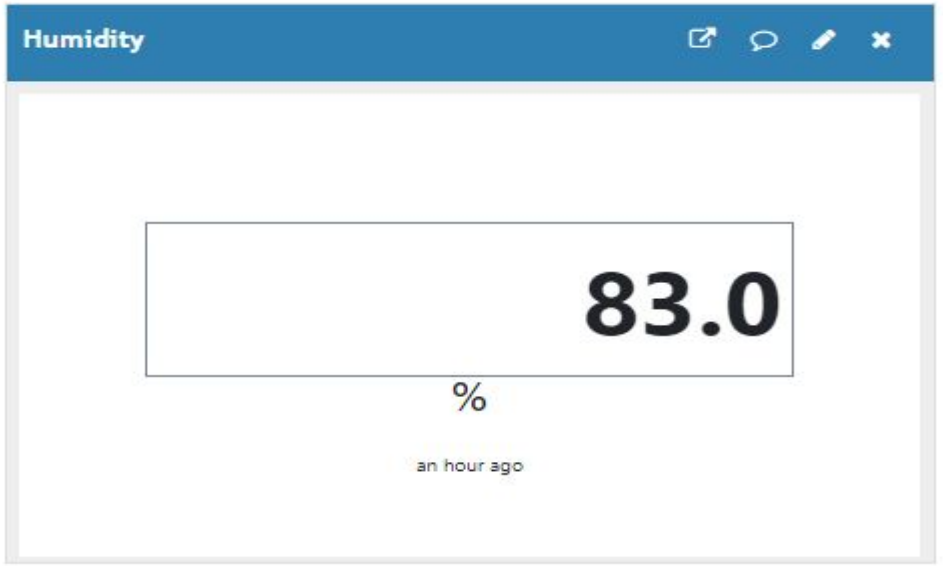
Created: a day ago
Last entry: about an hour ago
Entries: 125

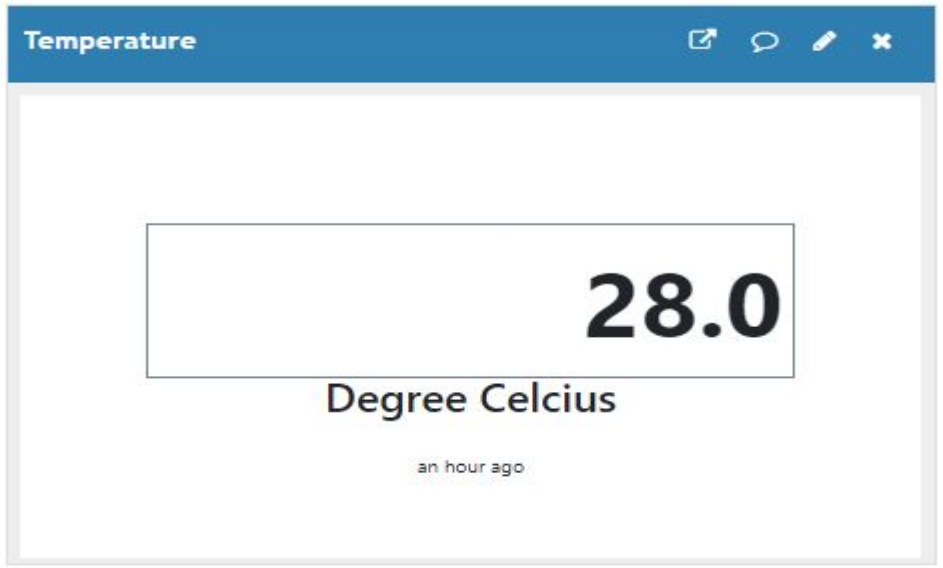




Widgets created

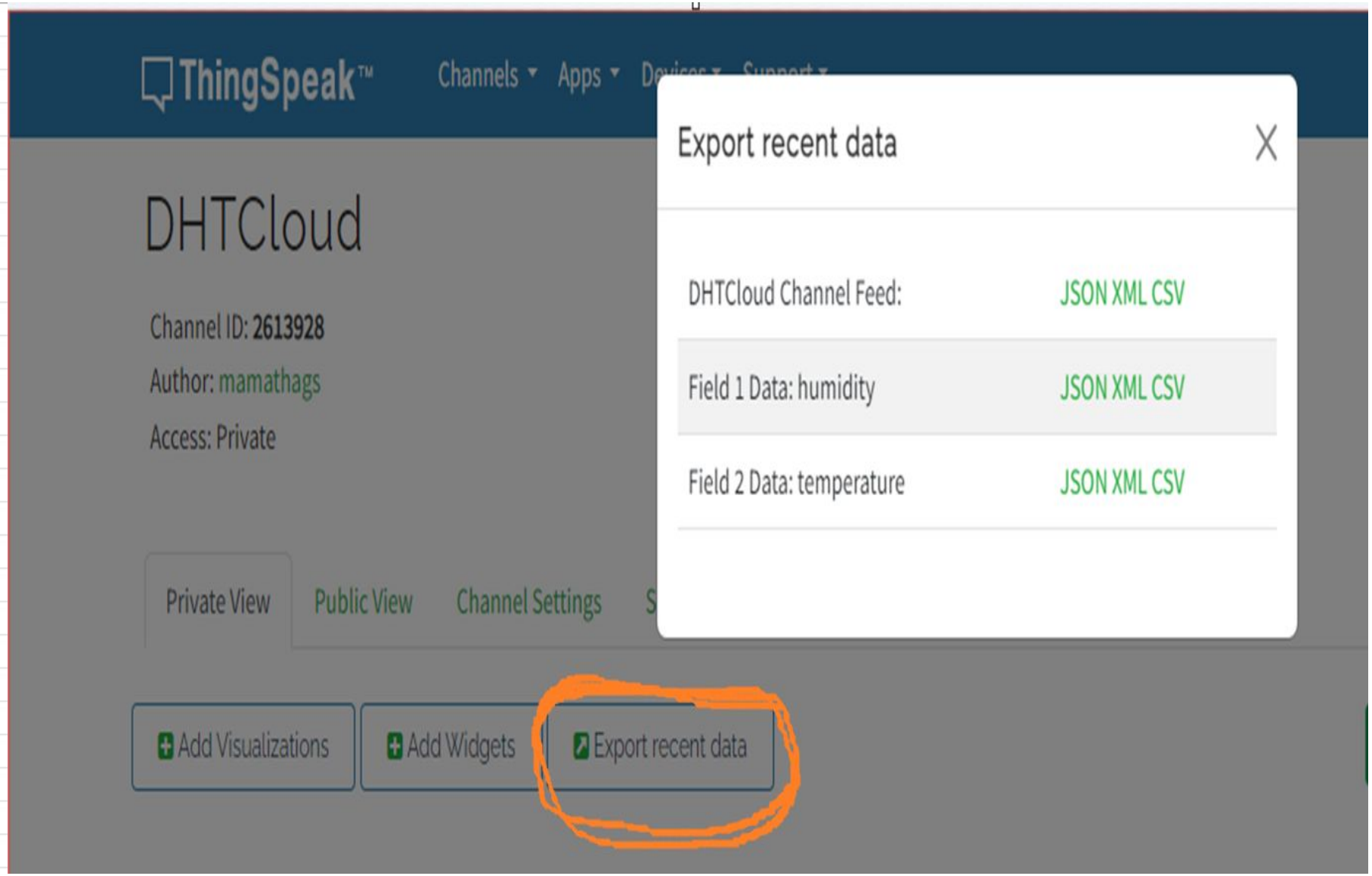




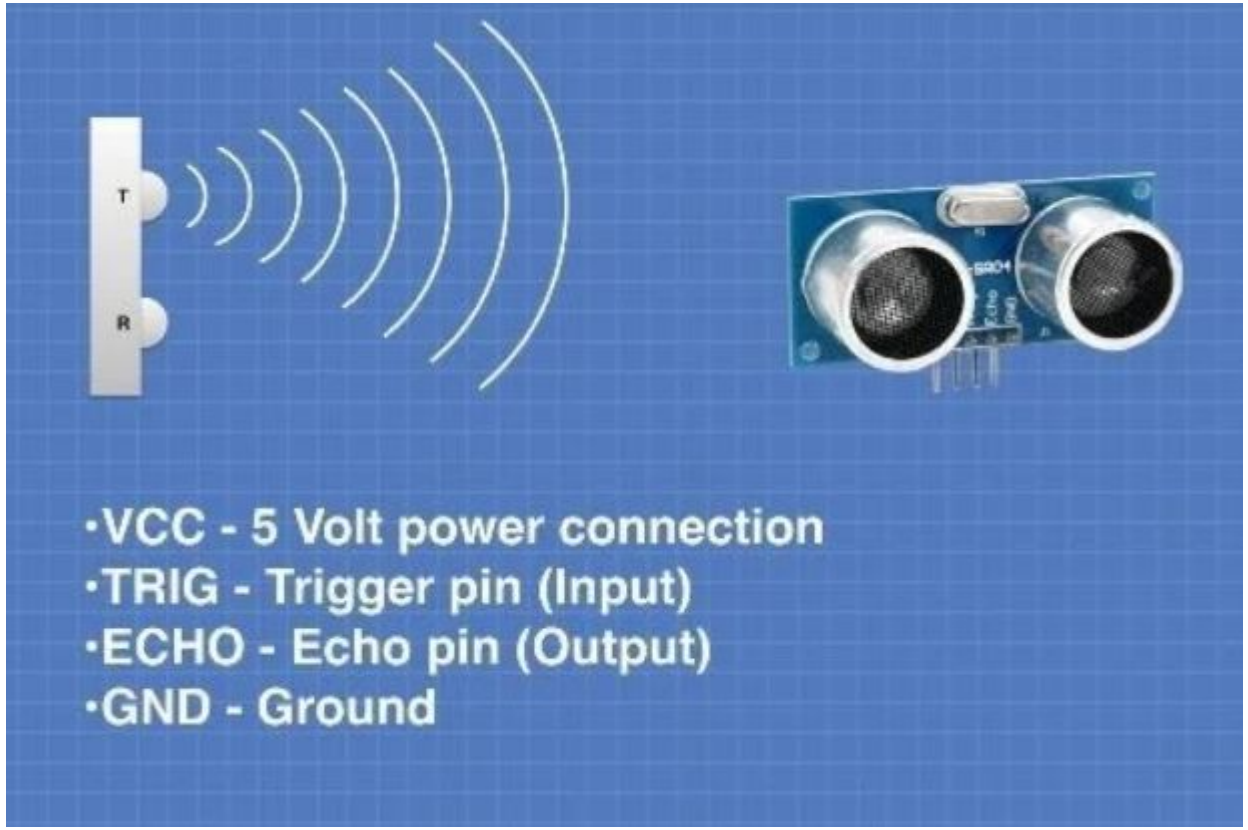


Data logged

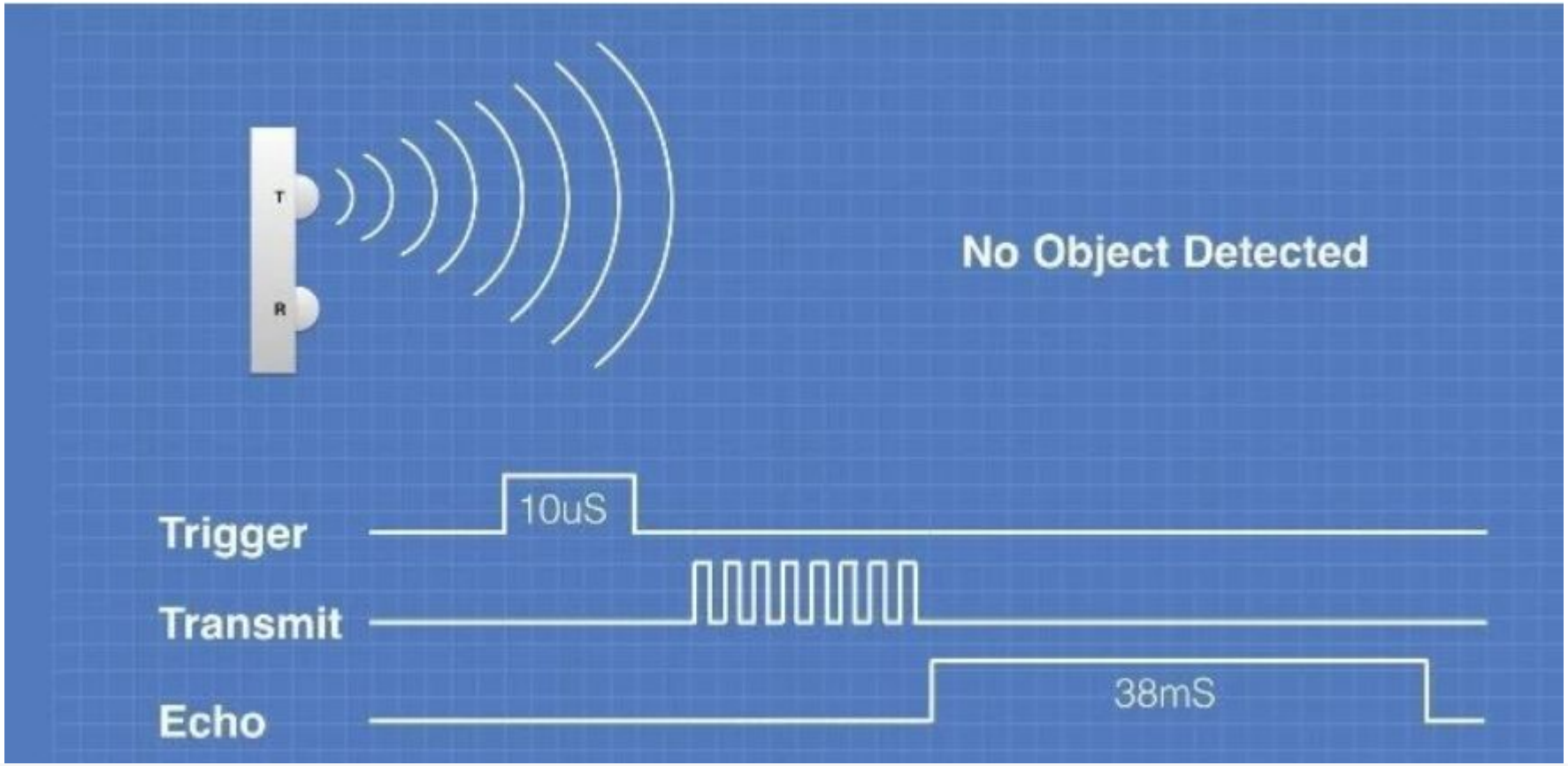
created_at	entry_id	field1
2024-08-0	26	84
2024-08-0	27	84
2024-08-0	28	81
2024-08-0	29	84
2024-08-0	30	81
2024-08-0	31	85
2024-08-0	32	84
2024-08-0	33	81
2024-08-0	34	81
2024-08-0	35	84
2024-08-0	36	81
2024-08-0	37	85
2024-08-0	38	84
2024-08-0	39	83
2024-08-0	40	82
2024-08-0	41	82
2024-08-0	42	80
2024-08-0	43	82
2024-08-0	44	81
2024-08-0	45	85
2024-08-0	46	84
2024-08-0	47	84
2024-08-0	48	83
2024-08-0	49	84
2024-08-0	50	84



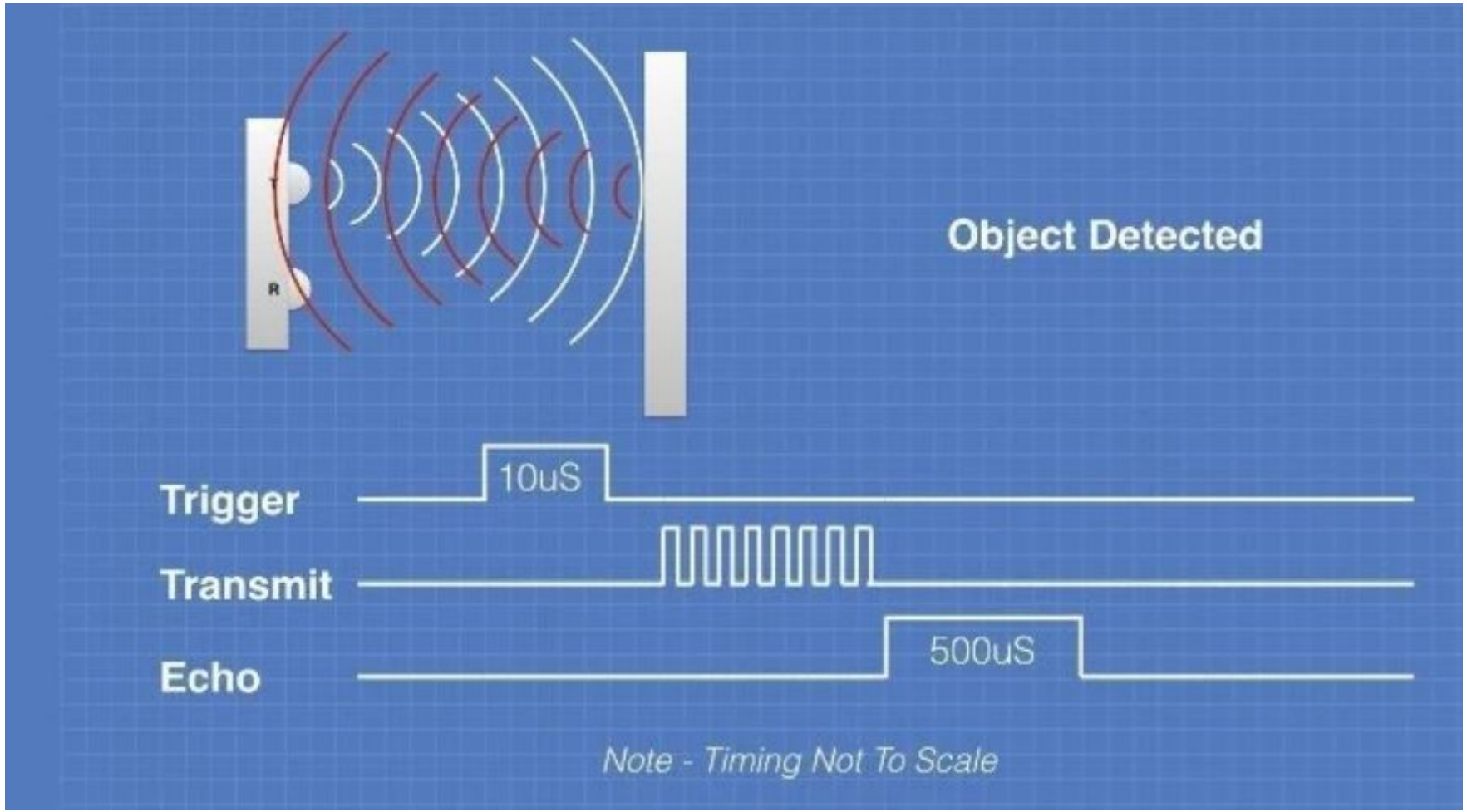
Ultrasonic sensor



- VCC pin: For Power supply.
- TRIG pin: This pin is the input pin and transmits the waves.
- ECHO pin: This pin is the output pin and it detects the reflected wave.
- GND pin: This pin is the ground pin.



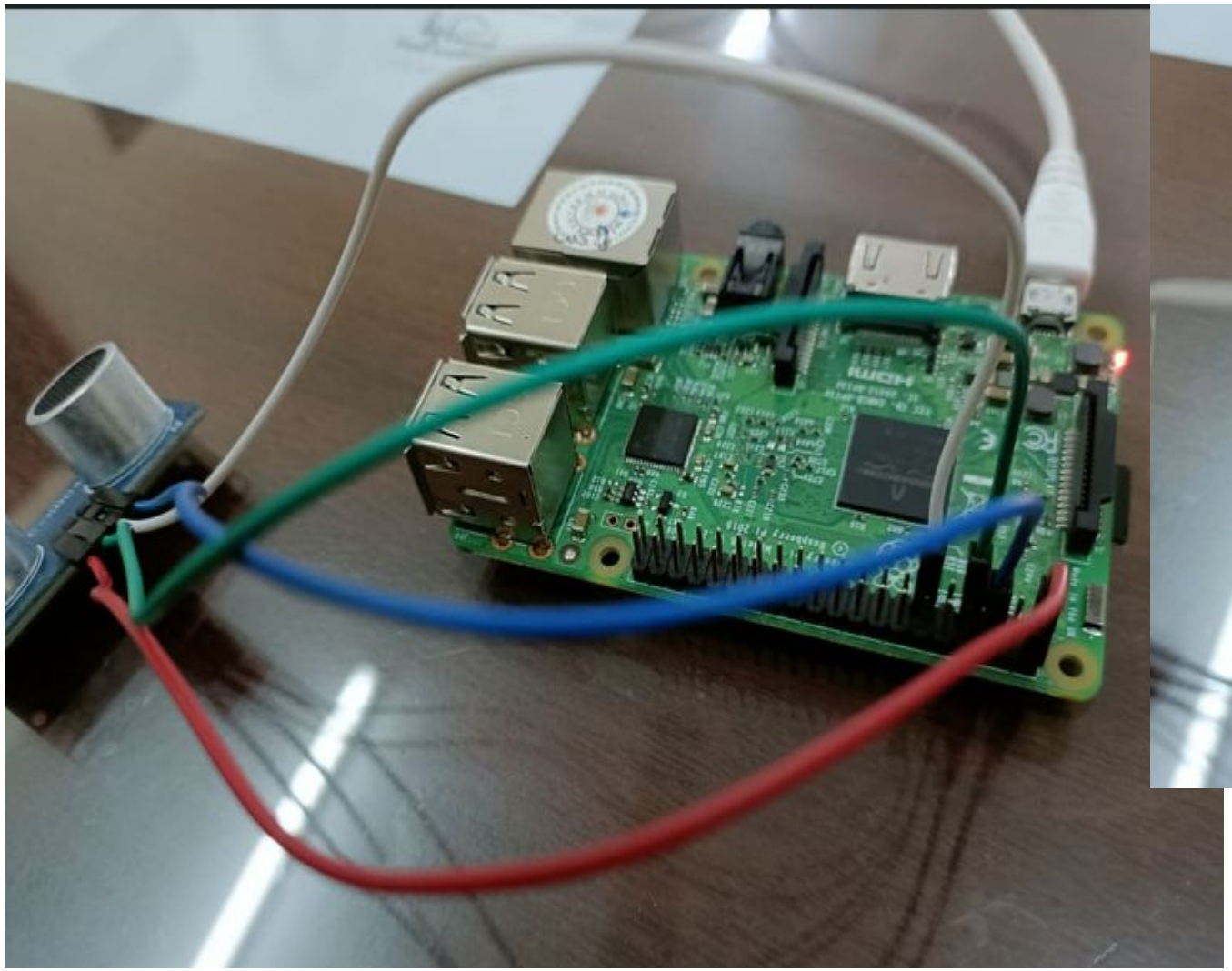
- The ECHO pin goes low after 38ms if the wave does not get reflected.
- If the transmitted wave gets reflected, then the ECHO pin will immediately go low. And the width of the output pulse will be anywhere from 150 microseconds to 25 milliseconds.
- The width of the pulse from the ECHO pin is used to calculate the distance of the object from the sensor.

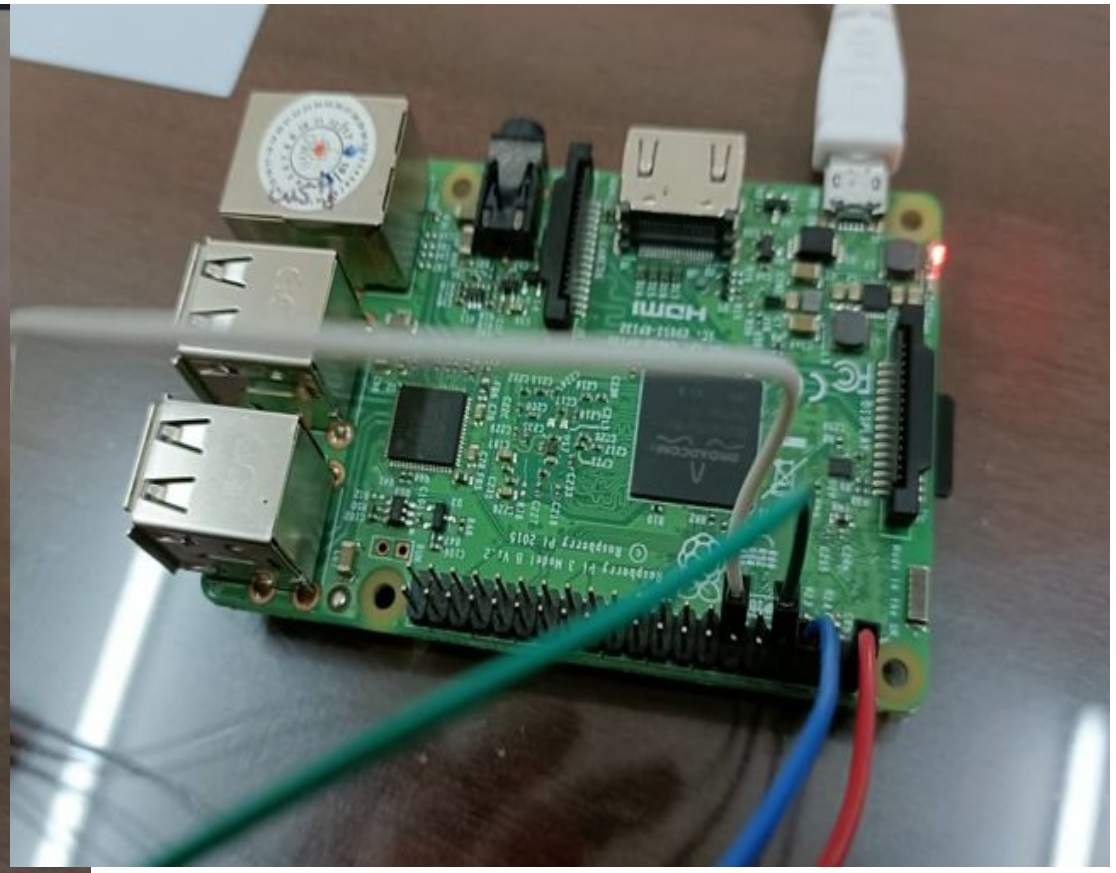

R
Object Detected
Trigger
10uS
Transmit
Echo
500uS
Note - Timing Not To Scale

Ultrasonic with cloud connectivity


```
# uses classes  for http client side
import http.client
#parses URL string and uses http url scheme
import urllib.parse
import time
import RPi.GPIO as GPIO
from gpiozero import DistanceSensor


#GPIO.setmode(GPIO.BOARD)
#GPIO.setup(7,GPIO.IN)
GPIO.setwarnings(False)
key = "MKQ13DILO94GR23J"  # Put your API Key here


def ultra():
        #senses light density and switch on/off with buzzer ring
    #printing the value from ldr module
  while True:
        ultrasonic = DistanceSensor(echo=17, trigger=4)
        print(ultrasonic.distance)
Params=urllib.parse.urlencode({'field1':ultrasonic.distance,'
key':key })
headers = {"Content-typZZe":
"application/x-www-form-urlencoded","Accept": "text/plain"}
      #create instances that connect to the HTTP server at
the same host and port
conn =
http.client.HTTPConnection("api.thingspeak.com:80")
          try:
              conn.request("POST", "/update", params,
headers)
              response = conn.getresponse()
              #print("temperature, humidity")
              print(response.status, response.reason)
              data = response.read()
              conn.close()
          except:
              print("connection failed")
          break
if __name__ == "__main__":
      while True:
          ultra()
```

Circuit





GND- pin 2
Vcc- pin 6
TRIG- pin 7 (GPIO4)
Echo- pin 11 (GPIO 17)

Data logger and visualization

ThingSpeak™ Channels Apps Devices Support

ULTRASONICLOUD

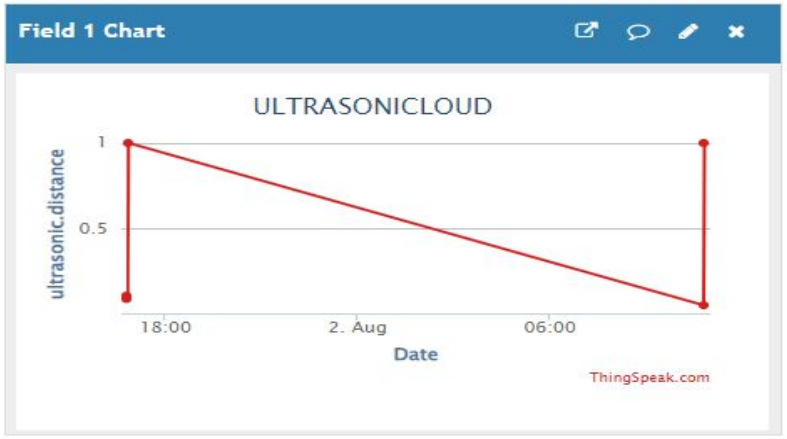
Channel ID: **2614255**
Author: mamathags
Access: Private

Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export

Add Visualizations | Add Widgets | Export recent data

Channel Stats

Created: about 21 hours ago
Last entry: about an hour ago
Entries: 9



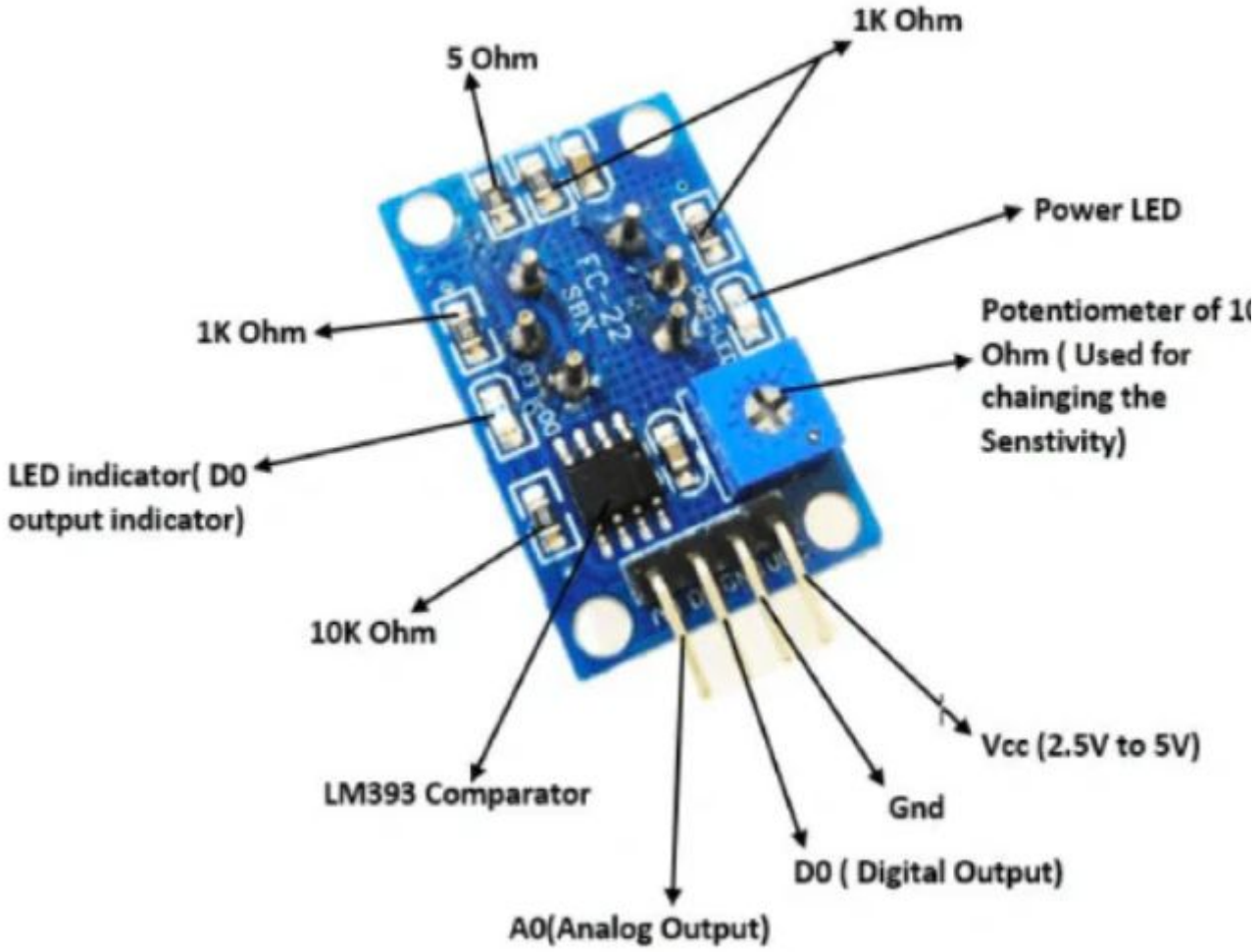
Air Quality sensor-MQ135

AQI Classification

AQI Range	AQI Category
0–50	Excellent
51–100	Good
100–150	Lightly polluted
151–200	Moderately polluted
201–250	Heavily polluted
251–300	Severely polluted

- The **MQ-135 Gas sensor** can detect gases like Ammonia (NH3), sulfur (S), Benzene (C6H6), CO2, and other harmful gases and smoke.
- Similar to other MQ series gas sensor, this sensor also has a digital and analog output pin.
- When the level of these gases go beyond a threshold limit in the air the digital pin goes high.
- This threshold value can be set by using the on-board potentiometer.
- The analog output pin, outputs an analog voltage which can be used to approximate the level of these gases in the atmosphere.
- The MQ135 air quality sensor module operates at 5V and consumes around 150mA.
- It requires some pre-heating before it could actually give accurate results

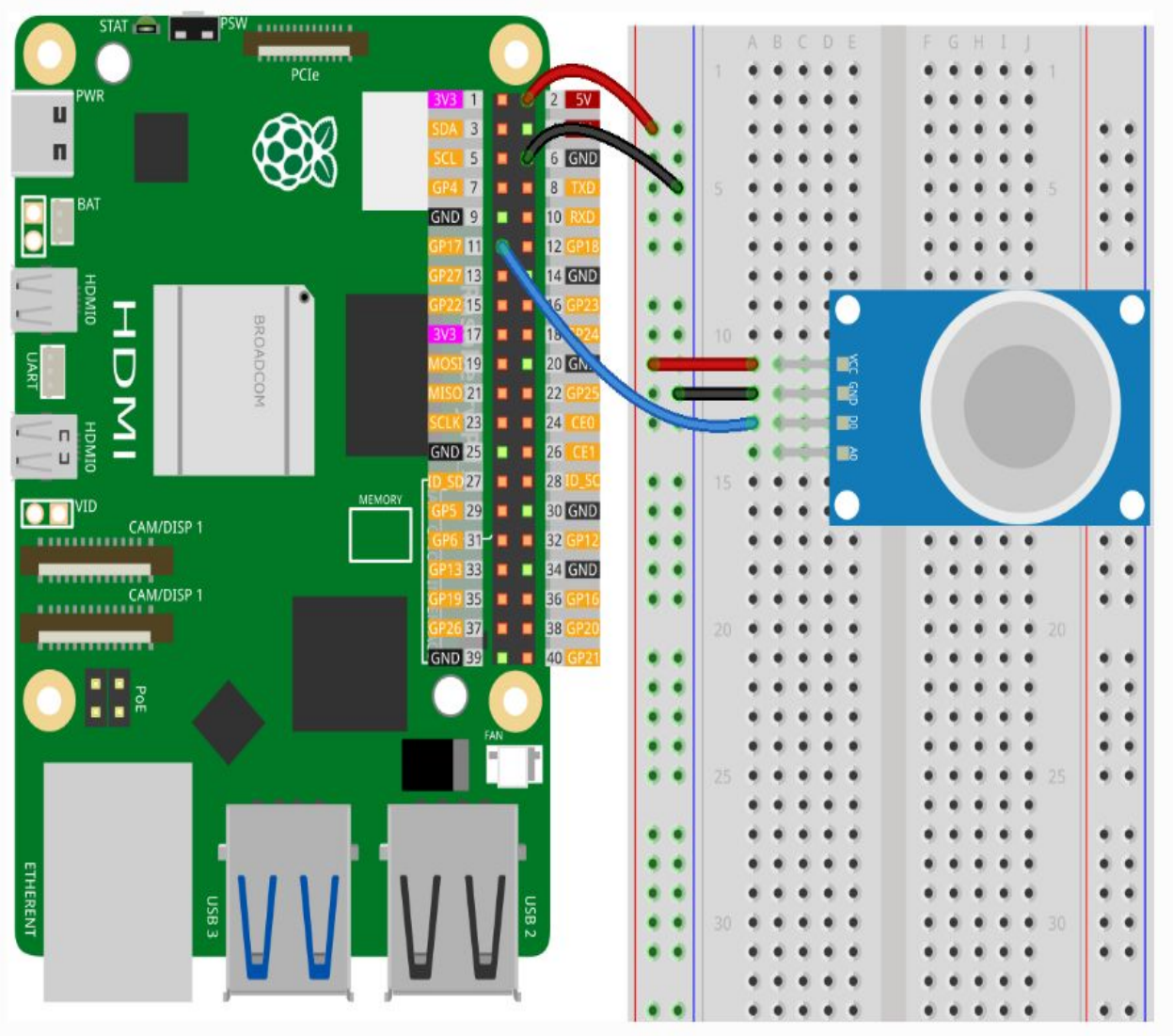
Sensor details



Technical Specifications of MQ135 Gas Sensor

- Operating Voltage: 2.5V to 5.0V
- Power consumption: 150mA
- Detect/Measure: NH3, Nox, CO2, Alcohol, Benzene, Smoke
- Typical operating Voltage: 5V
- Digital Output: 0V to 5V (TTL Logic) @ 5V Vcc
- Analog Output: 0-5V @ 5V Vcc

Circuit



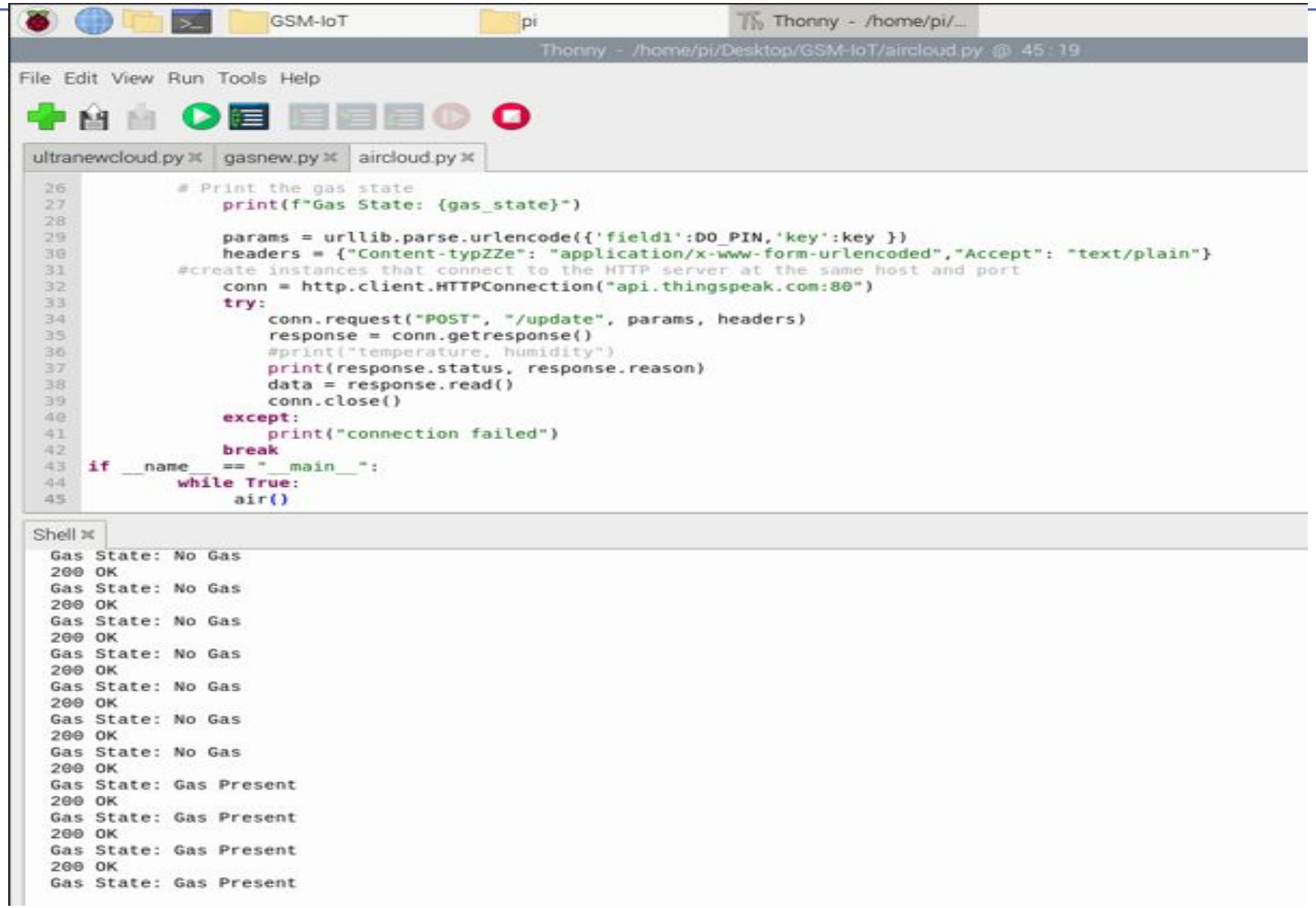
Vcc- pin 2
GND- pin 6
Do- pin 26 (GPIO 7)

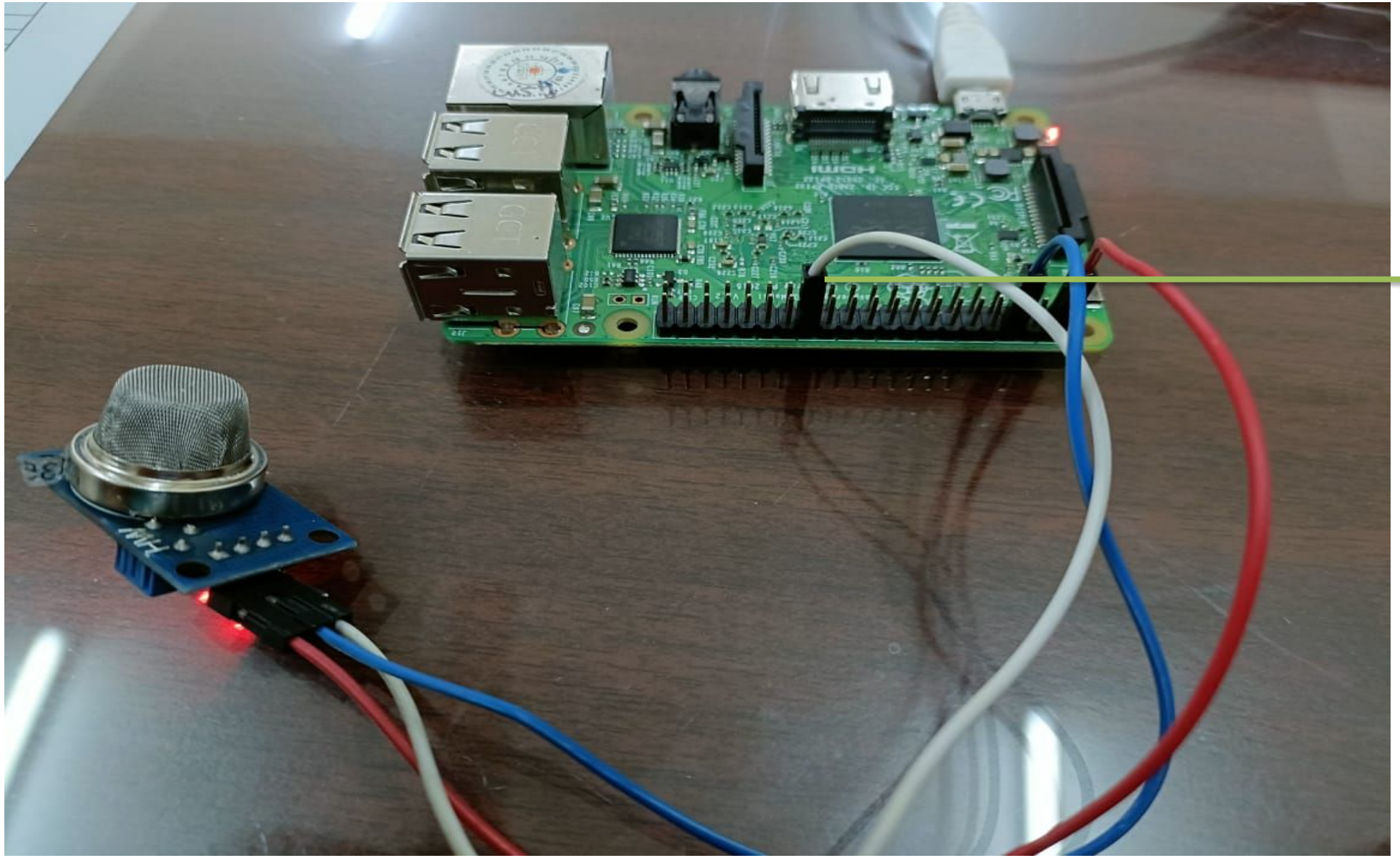
- **DO**: Digital output. It indicates the presence of combustible gases. When the gas concentration exceeds the threshold value (as set by the potentiometer), D0 becomes LOW; otherwise, it is HIGH.
- **AO**: Analog output. It produces an analog output voltage proportional to gas concentration, so a higher concentration results in a higher voltage and a lower concentration results in a lower voltage.

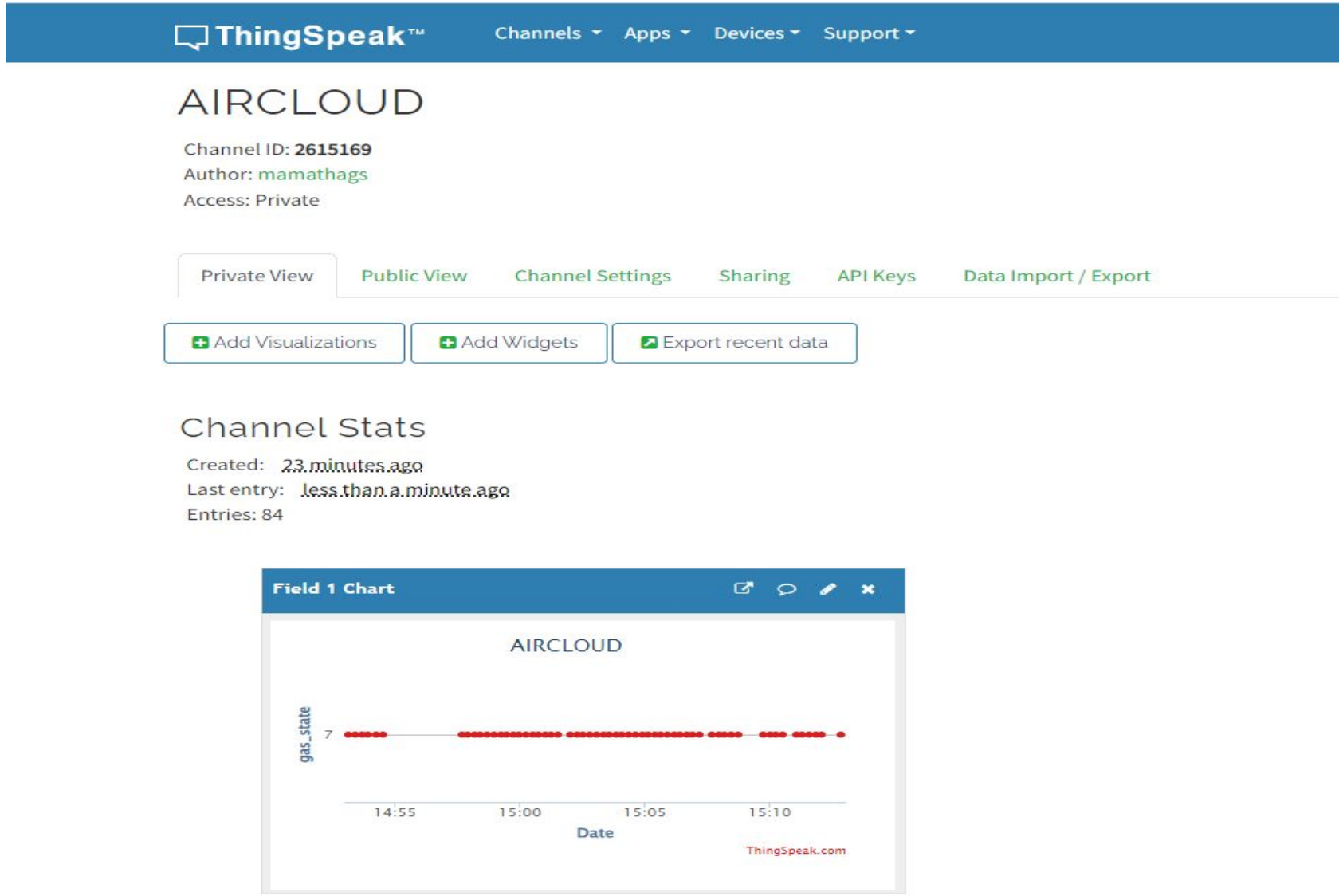
```
# uses classes  for http client side
import http.client
import urllib.parse
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
key = "HY5941TNS5UP3GZ8"        # Put your API Key here
DO_PIN = 7                      # Replace with the actual GPIO
pin number-
                                    pin 26
GPIO.setup(DO_PIN, GPIO.IN)
def air():
    while True:
        gas_present = GPIO.input(DO_PIN)
      # Print the gas state
        print(f"Gas State: {gas_state}")
```

```
  # Determine if gas is present or not
            if gas_present == GPIO.LOW:
                gas_state = "Gas Present"
            else:
                gas_state = "No Gas"
params = urllib.parse.urlencode({'field1':DO_PIN,'key':key })
            headers = {"Content-typZZe":
"application/x-www-form-urlencoded","Accept": "text/plain"}
        #create instances that connect to the HTTP server at the
same host and port
            conn =
http.client.HTTPConnection("api.thingspeak.com:80")
            try:
                conn.request("POST", "/update", params, headers)
                response = conn.getresponse()
                #print("temperature, humidity")
                print(response.status, response.reason)
                data = response.read()
                conn.close()
            except:
                print("connection failed")
            break
if __name__ == "__main__":
        while True:
            air()
```

Output

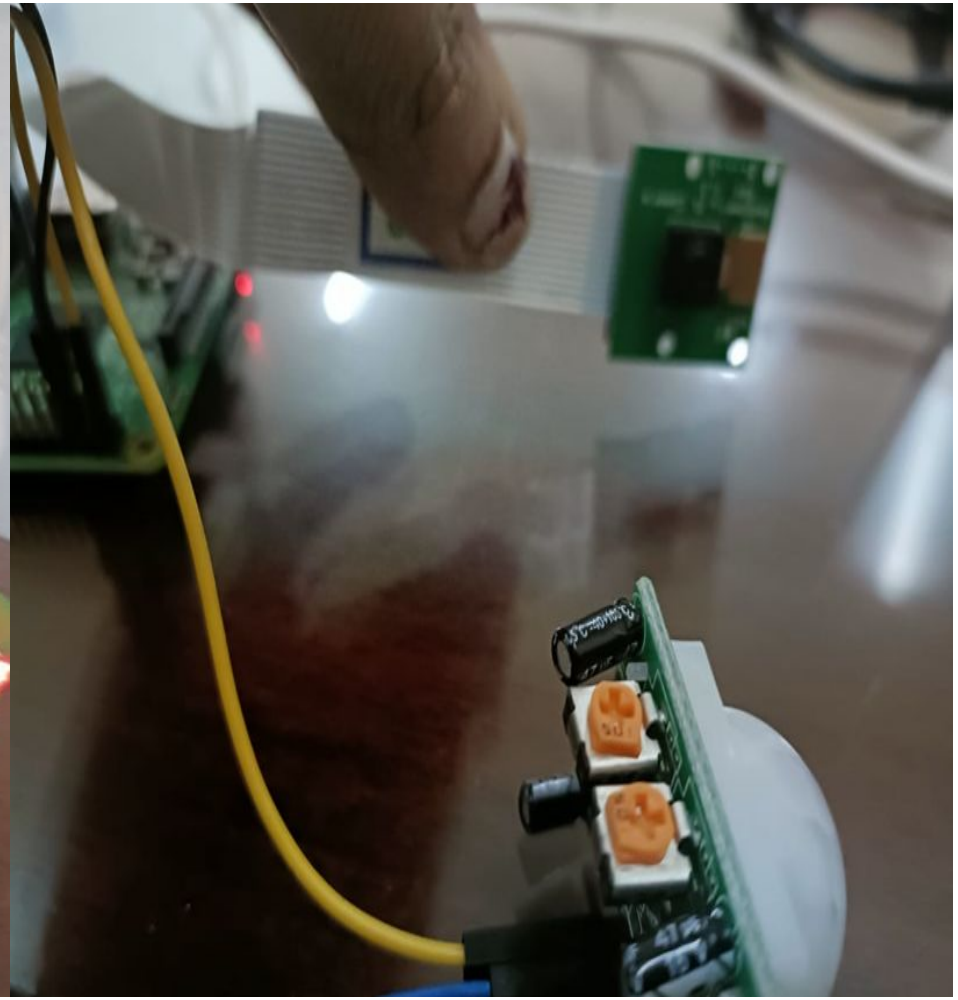

GSM-IoT
pi
Thonny - /home/pi/...
Thonny - /home/pi/Desktop/GSM-IoT/aircloud.py @ 45 : 19
File Edit View Run Tools Help
ultranewcloud.py
gasnew.py
aircloud.py
# Print the gas state
print(f"Gas State: {gas_state}")
params = urllib.parse.urlencode({'field1':DO_PIN,'key':key })
headers = {"Content-typZZe": "application/x-www-form-urlencoded","Accept": "text/plain"}
#create instances that connect to the HTTP server at the same host and port
conn = http.client.HTTPConnection("api.thingspeak.com:80")
try:
conn.request("POST", "/update", params, headers)
response = conn.getresponse()
#print("temperature, humidity")
print(response.status, response.reason)
data = response.read()
conn.close()
except:
print("connection failed")
break
if __name__ == "__main__":
while True:
air()
Shell
Gas State: No Gas
200 OK
Gas State: No Gas
200 OK
Gas State: No Gas
200 OK
Gas State: No Gas
200 OK
Gas State: No Gas
200 OK
Gas State: No Gas
200 OK
Gas State: No Gas
200 OK
Gas State: Gas Present
200 OK
Gas State: Gas Present
200 OK
Gas State: Gas Present
200 OK
Gas State: Gas Present

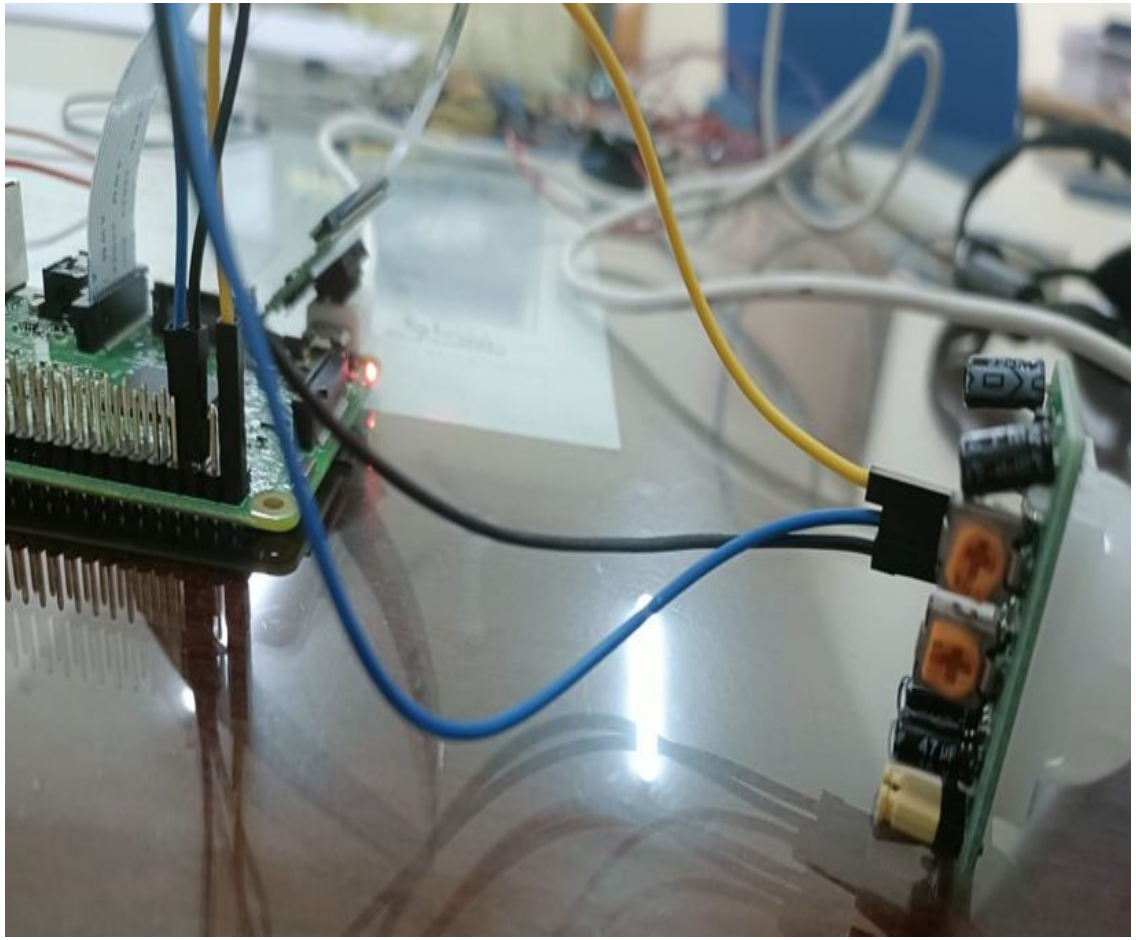

Do-
26

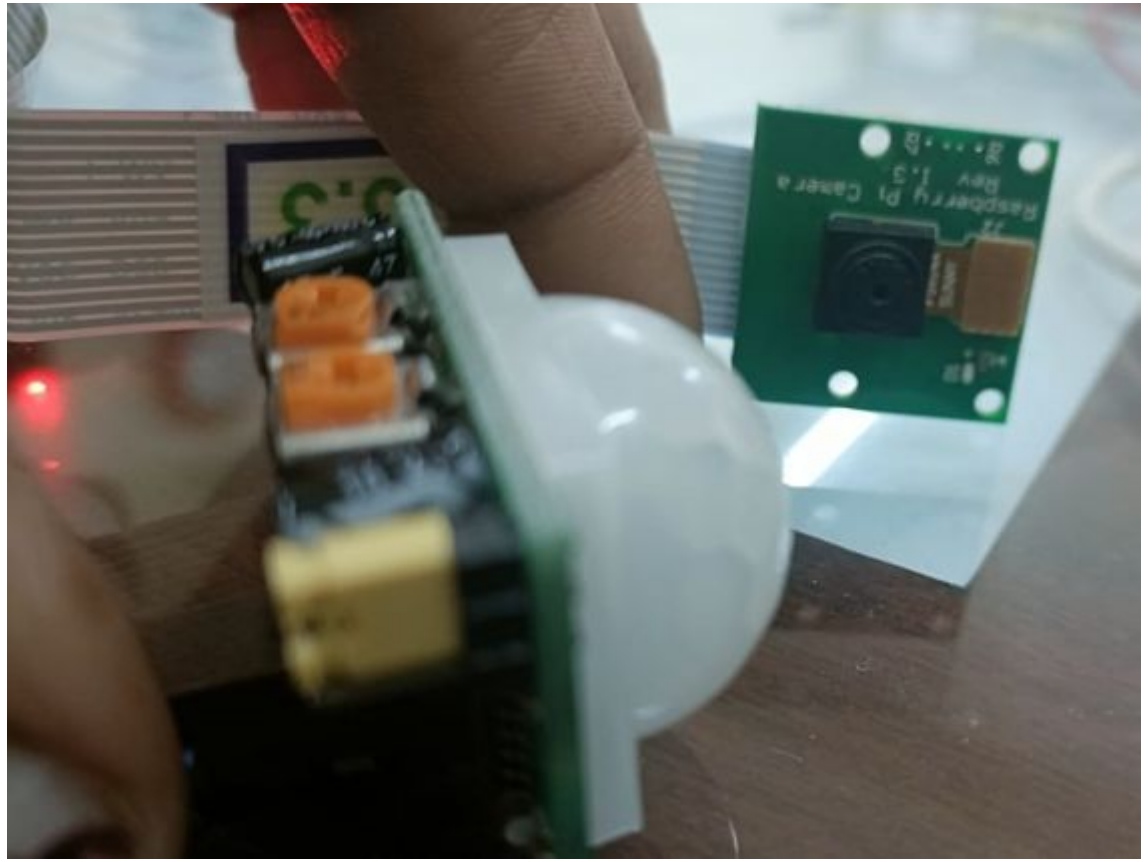

ThingSpeak™
Channels
Apps
Devices
Support
AIRCLOUD
Channel ID: 2615169
Author: mamathags
Access: Private
Private View
Public View
Channel Settings
Sharing
API Keys
Data Import / Export
Add Visualizations
Add Widgets
Export recent data
Channel Stats
Created: 23 minutes ago
Last entry: less than a minute ago
Entries: 84
Field 1 Chart
AIRCLOUD
gas_state
7
14:55
15:00
15:05
15:10
Date
ThingSpeak.com

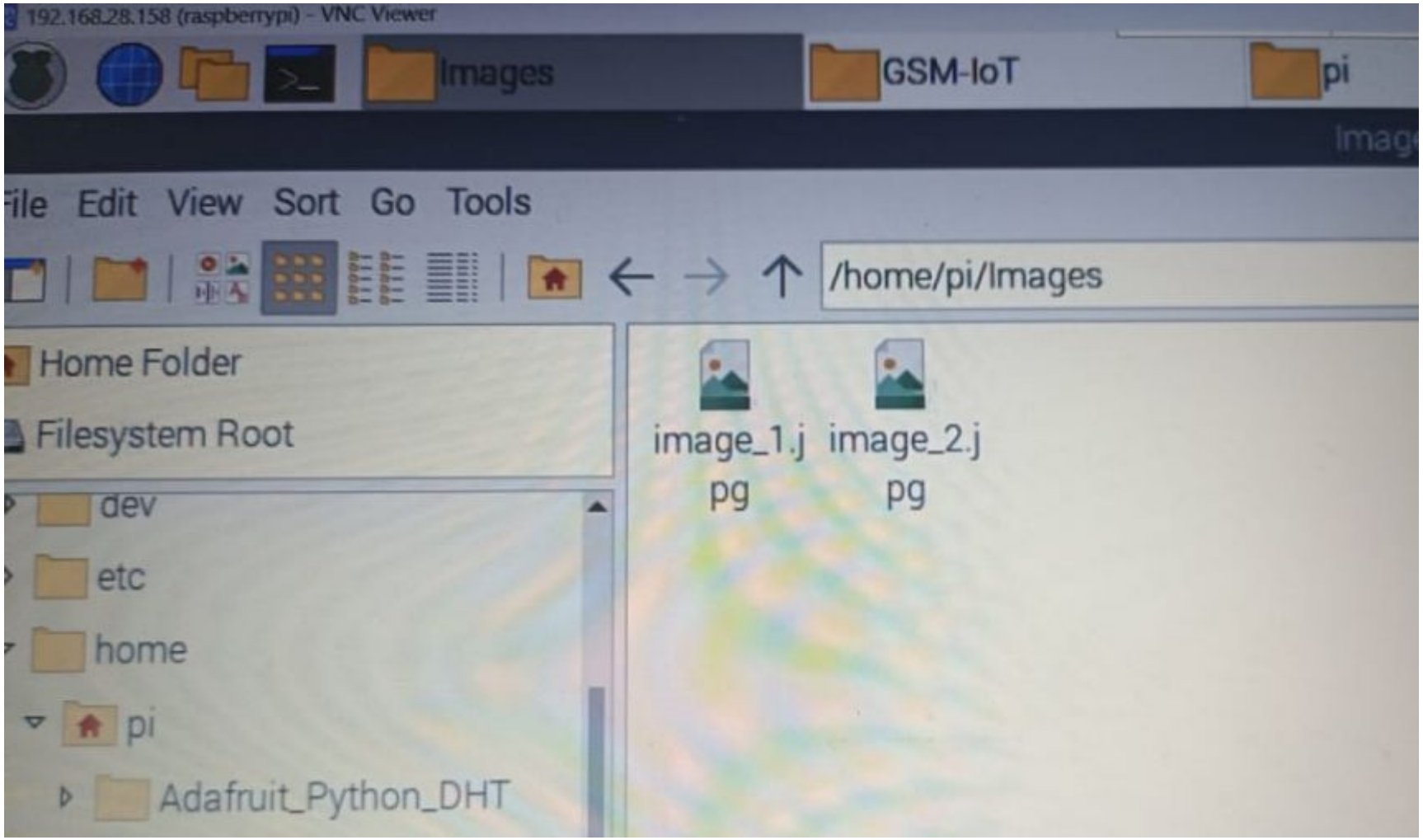
Motion with Camera IoT system


```
from gpiozero import Button, MotionSensor
from picamera import PiCamera
from time import sleep
from signal import pause

#create objects that refer to a button,
#a motion, sensor, and the PiCamera
#button = Button(2)
pir = MotionSensor(4)
camera = PiCamera()

#start the camera
camera.rotation = 180
camera.start_preview()
#create image names
i = 0

#take a photo when motion is detected
def take_photo():
    global i
    i = i + 1

camera.capture('/home/pi/Images/image_%s.jpg' % i)
    print('A photo has been taken')
    sleep(10)

#assign a function that runs when motion is detected
pir.when_motion = take_photo
pause()
```






Raspberry Pi Camera
Rev 1.3


192.168.28.158 (raspberrypi) - VNC Viewer
Images
GSM-IoT
pi
File Edit View Sort Go Tools
/home/pi/Images
Home Folder
Filesystem Root
dev
etc
home
pi
Adafruit_Python_DHT
image_1.jpg
image_2.jpg

DHT11 data to ThingSpeak-extra


```
import time
import board
import adafruit_dht
import thingspeak
from rpi_lcd import LCD

# Define your ThingSpeak channel parameters
channel_id = 2595470
write_key = '86CNU3LZRKBSRJW9'

# Initialize the ThingSpeak channel
channel = thingspeak.Channel(id=channel_id, api_key=write_key)

# Initialize the LCD
lcd = LCD()

# Initialize the sensor (DHT11 connected to GPIO 4)
sensor = adafruit_dht.DHT11(board.D4, use_pulseio=False)
```

```
while True:
    try:
        # Read the sensor data
        temperature_c = sensor.temperature
        temperature_f = temperature_c * (9 / 5) + 32
        humidity = sensor.humidity

        # Print the values to the serial port
        print("Temp={0:0.1f}C, Temp={1:0.1f}F, Humidity={2:0.1f}%".format(temperature_c, temperature_f,
humidity))

        # Display the values on the LCD
        lcd.text("Temp={0}C".format(temperature_c), 1)
        lcd.text("Humi={0}%".format(humidity), 2)

        # Send the data to ThingSpeak
        response = channel.update({'field1': temperature_c, 'field2': humidity})
        print("Data sent to ThingSpeak. Response:", response)
```

```
except RuntimeError as error:
    # Errors happen fairly often with DHT sensors, just keep going
    print(error.args[0])
    time.sleep(2.0)
    continue
except Exception as error:
    sensor.exit()
    raise error

# Wait before taking the next reading
time.sleep(15)
```

Steps to install Arduino in Pi

Play with Digital Sensor

Program for digital sensor

Play with Analog Sensor

Play with Actuators

- **L293D and DC motor connection**
  - Connect L293D pin 3 to +ve pin of DC motor1.
  - Connect L293D pin 6 to –ve pin of DC motor1.
  - Connect L293D pin 11 to +ve pin of DC motor2.
  - Connect L293D pin 14 to +ve pin of DC motor2.
- **L293D connection**
  - Connect Arduino GND to pins 4, 5, 12, 13 of L293D.
  - Connect Arduino +5 V to pins 1, 9, 16 of L293D.
  - Connect Arduino pin 7 to pin 2 of L293D.
  - Connect Arduino pin 6 to pin 7 of L293D.
  - Connect Arduino pin 5 to pin 10 of L293D.
  - Connect Arduino pin 4 to pin 15 of L293D.
  - Connect L293D pin 8 to +ve of 12V battery
- **LED connection**
  - Connect Arduino pin 7 to anode of LED1.
  - Connect Arduino pin 6 to anode of LED2.
  - Connect Arduino pin 5 to anode of LED3.
  - Connect Arduino pin 4 to anode of LED4.
  - Connect cathode of all LEDs to ground.

LCD connection

- Connect Arduino digital pin 13 to RS pin(4) of LCD.
- Connect Arduino digital pin GND to RW pin(5) of LCD.
- Connect Arduino digital pin 12 to E pin(6) of LCD.
- Connect Arduino digital pin 11 to D4 pin(11) of LCD.
- Connect Arduino digital pin 10 to D5 pin(12) of LCD.
- Connect Arduino digital pin 9 to D6 pin(13) of LCD.
- Connect Arduino digital pin 8 to D7 pin(14) of LCD.

6.4.1.2

```
int directionPin = 12;
int pwmPin = 3;
int brakePin = 9;

//uncomment if using channel B, and remove above definitions
//int directionPin = 13;
//int pwmPin = 11;
//int brakePin = 8;

//boolean to switch direction
bool directionState;

void setup() {

//define pins
pinMode(directionPin, OUTPUT);
pinMode(pwmPin, OUTPUT);
pinMode(brakePin, OUTPUT);

}

void loop() {

//change direction every loop()
directionState = !directionState;

//write a low state to the direction pin (13)
if(directionState == false){
 digitalWrite(directionPin, LOW);
//write a high state to the direction pin (13)
else{
 digitalWrite(directionPin, HIGH);
}

//release breaks
digitalWrite(brakePin, LOW);

//set work duty for the motor
analogWrite(pwmPin, 30);

delay(2000);

//activate breaks
digitalWrite(brakePin, HIGH);

//set work duty for the motor to 0 (off)
analogWrite(pwmPin, 0);

delay(2000);
}
```

Sketch

```
#include <LiquidCrystal.h> // include library of LCD
LiquidCrystallcd(13, 12, 11, 10,9, 8); // attach LCD pin RS, E, D4, D5, D6,
D7 to the given pins
int MPIN1= 7; // assign pin 7 as MPIN1
int MPIN2= 6; // assign pin 6 as MPIN2
int MPIN3= 5; // assign pin 5 as MPIN3
int MPIN4= 4; // assign pin 4 as MPIN4
void setup()
{
pinMode(MPIN1, OUTPUT); // make MPIN1 as an output
pinMode(MPIN2, OUTPUT); // make MPIN2 as an output
pinMode(MPIN3, OUTPUT); // make MPIN3 as an output
pinMode(MPIN4, OUTPUT); // make MPIN4 as an output
lcd.begin(20,4); // initialise LCD
lcd.setCursor(0, 0); // set cursor on LCD
lcd.print("DC Motor direction"); // print string on LCD
lcd.setCursor(0, 1); // set cursor on LCD
lcd.print("control system..."); // print string on LCD
delay(1000); // delay of 1000 mS
lcd.clear(); // clear the contents of LCD
}
void loop() // infinite loop
{
digitalWrite(MPIN1, HIGH); // make MPIN1 to HIGH
```



```
digitalWrite(MPIN2, LOW); // make MPIN2 to LOW
digitalWrite(MPIN3, HIGH); // make MPIN3 to HIGH
digitalWrite(MPIN4, LOW); // make MPIN4 to LOW
lcd.setCursor(0, 2); // set cursor on LCD
lcd.print("CLOCKWISE"); // print string on LCD
delay(2000); // delay of 2 sec
lcd.clear(); // clear the contents of LCD
digitalWrite(MPIN1, LOW); //make MPIN1 to LOW
digitalWrite(MPIN2, HIGH); //make MPIN2 to HIGH
digitalWrite(MPIN3, LOW); //make MPIN3 to LOW
digitalWrite(MPIN4, HIGH); //make MPIN4 to HIGH
lcd.setCursor(0, 2); // set cursor on LCD
lcd.print("ANTI-CLOCKWISE"); // print string on LCD
delay(2000); // delay of 2 Sec
lcd.clear(); // clear the contents of LCD
digitalWrite(MPIN1, LOW); // make MPIN1 to LOW
digitalWrite(MPIN2, LOW); //make MPIN2 to LOW
digitalWrite(MPIN3, HIGH); //make MPIN3 to HIGH
digitalWrite(MPIN4, LOW); //make MPIN4 to LOW
lcd.setCursor(0, 2); // set cursor on LCD
lcd.print("LEFT"); // print string on LCD
delay(2000); // delay of 2 Sec
lcd.clear(); // clear the contents of LCD
digitalWrite(MPIN1, HIGH); //make MPIN1 to HIGH
digitalWrite(MPIN2, LOW); //make MPIN2 to LOW
digitalWrite(MPIN3, LOW); //make MPIN3 to LOW
digitalWrite(MPIN4, LOW); //make MPIN4 to LOW
lcd.setCursor(0, 2); // set cursor on LCD
lcd.print("RIGHT"); // print string on LCD
delay(2000); // delay of 2 Sec
lcd.clear(); // clear the contents of LCD
}
```

Servo Motor

- A servo motor is a rotary actuator used for precise control of the angular position.
- It is comprised of a motor coupled with a sensor for position feedback.
- It also requires a servo drive. The drive uses the feedback sensor to control the rotary position of the motor precisely.
- This is called a closed-loop operation. The high torque standard servo motor with metal gears and 360° rotation can provide 11 kg/cm at 4.8 V, 13.5 kg/cm at 6 V, and 16 kg/cm at 7.2 V.
- Figure shows the block diagram to interface the servo motor with Arduino.
- It is comprised of an Arduino Uno, a power supply, a liquid crystal display, a potentiometer (POT), and a servo motor.
- The system is designed to control the angle of the servo motor with the potentiometer.

Servo connection

- Connect Arduino GND to GND pin of servo motor.
- Connect Arduino +5 V to “+” terminal of servo motor.
- Connect Arduino pin(3) to PWM pin of servo motor.

POT connection

- Connect Arduino GND to GND pin of POT.
- Connect Arduino +5 V to “+” terminal of POT.
- Connect Arduino A0 pin to data out pin of POT.

LCD connection

- Connect Arduino digital pin (13) to RS pin(4) of LCD.
- Connect Arduino digital pin (GND) to RW pin(5) of LCD.
- Connect Arduino digital pin (12) to E pin(6) of LCD.
- Connect Arduino digital pin (11) to D4 pin(11) of LCD.
- Connect Arduino digital pin (10) to D5 pin(12) of LCD.
- Connect Arduino digital pin (9) to D6 pin(13) of LCD.
- Connect Arduino digital pin (8) to D7 pin(14) of LCD.

Sketch

```
#include <LiquidCrystal.h> // include library of LCD
LiquidCrystallcd(13, 12, 11, 10, 9, 8); // attach LCD pin RS,E,D4,D5,D6,D7
to the given pins
Servo myservo; // create servo object to control a servo
int POT_PIN = A0; // analog pin used to connect the potentiometer
int POT_PIN_ADC_LEVEL; // variable to read the value from the analog pin
```

- void setup()
- {
- myservo.attach(3); // attaches the servo on pin 9 to the servo object
- lcd.begin(20,4); // initialise LCD
- lcd.setCursor(0, 0); // set cursor on LCD
- lcd.print("Servo ANALOG write "); // print string on LCD
- lcd.setCursor(0, 1);// set cursor on LCD
- lcd.print("system at LPU....");// print string on LCD
- }

```
void loop()
{
POT_PIN_ADC_LEVEL = analogRead(POT_PIN); //
reads POT value
in the form of levels
POT_PIN_ADC_LEVEL = map(POT_PIN_ADC_LEVEL, 0,
1023, 0, 179);
// map the value //between 0 to 180 degree for servo
myservo.write(POT_PIN_ADC_LEVEL); // sets the servo
position
according to the scaled value
lcd.setCursor(0, 2); // set cursor on LCD
lcd.print("ANGLE:"); // print string on LCD
lcd.print(POT_PIN_ADC_LEVEL); // print value on LCD
delay(15); // delay of 15 mSec
}
```