Assignment-1
IoT Lab

By:

Gurram Vamshi Kumar

MSc (MI); Roll No: 211113

Table of Contents

Get Values for IR, LDR, PIR, UltraSonic, Rain, and Sound sensors using Raspberry.

Interfacing LDR sensor with the Raspberry Pi

Introduction:

LDR (Light Dependent Resistor) is a type of resistor whose resistance varies according to the surrounding light. Resistance will increase with the increase in light intensity. LDR also known as Photoresistor, are available in 5mm, 7mm, 10mm, and many other sizes.

In this experiment we connect the LDR sensor to Raspberry Pi and run the python code

Then we will get the status of the day whether its dark or light.

Required Components:

IR Sensor x 1

Breadboard x 1

Raspberry Pi x 1

Jumper Wires

Source Code:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
LIGHT_PIN = 24
GPIO.setup(LIGHT_PIN, GPIO.IN)
lOld =  not GPIO.input(LIGHT_PIN)
print('Starting up the LIGHT Module (click on STOP to exit)')
time.sleep(0.2)
while True:
  if GPIO.input(LIGHT_PIN) <= 10:
    if GPIO.input(LIGHT_PIN):
      print ('\u263e')
    else:
      print ('\u263c')
  lOld = GPIO.input(LIGHT_PIN)
  time.sleep(0.2)
```

Interfacing IR sensor with the Raspberry Pi

Introduction:

IR (Infrared sensor) Sensor Module. As it is a Reflective type IR Sensor, whenever an object is placed in front of the sensor, the Infrared light from the IR LED gets reflected back after hitting the object and falls on the Photo Diode. The photo diode then starts conducting. As a result, the voltage at the non-inverting input of the LM358 will be greater than that at the inverting input.

In this experiment we connect the IR sensor to the Raspberry Pi and the run the cod given below. Whenever we find any object /obstacle in front it, it will detect.

Required Components:

IR Sensor x 1

Breadboard x 1

Raspberry Pi x 1

Jumper Wires

Source Code:

```
import RPi.GPIO as GPIO
import time

sensor = 16
#buzzer = 18

GPIO.setmode(GPIO.BCM)
GPIO.setup(sensor,GPIO.IN)
#GPIO.setup(buzzer,GPIO.OUT)

#GPIO.output(buzzer,False)
print("IR Sensor Ready.....")
print(" ")


while True:
  if not GPIO.input(sensor):
     #print(GPIO.input(sensor))
     #GPIO.output(buzzer,True)

     print("Object Detected")
     time.sleep(0.5)
  else:
       print("Object not Deteceted")
```

```
    time.sleep(0.5)
```

Interfacing Ultrasonic sensor with the Raspberry Pi

Introduction:

Ultrasonic Sensors, particularly HC-SR04 Ultrasonic Sensor, are very popular among electronic hobbyists and are frequently used in a variety of projects like Obstacle Avoiding Robot, Distance Measurement, Proximity Detection and so forth.

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. 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.

In this experiment we connect the Ultrasonic sensor to the Raspberry Pi and the run the cod given below. Whenever we find any object /obstacle in front it, it return the how far it is from the sensor.

Required Components:

Ultrasonic Sensor

Breadboard x 1

Raspberry Pi x 1

Jumper Wires

Source Code:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO_TRIGGER = 18
GPIO_ECHO = 24

GPIO.setup(GPIO_TRIGGER , GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)

def distance():
```

```
    GPIO.output(GPIO_TRIGGER, True )
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False )

    starttime = time.time()
    stoptime = time.time()

    while GPIO.input(GPIO_ECHO)==0:
        starttime = time.time()

    while GPIO.input(GPIO_ECHO)==1:
        stoptime = time.time()

    timeescaped = stoptime - starttime

    distance = (timeescaped * 34300) /2
    return distance

if __name__ == '__main__':
    try:
        while True:
            dist = distance()
            print(f'Measured distance = {round(dist)}')
            time.sleep(0.2)

    except KeyboardInterrupt:
        print('why you stopped ??????????????????')
        GPIO.cleanup()
```

Interfacing PIR module with the Raspberry Pi

Introduction:

If the PIR Sensor detects any human movement, it raises its Data Pin to HIGH.

In this experiment we connect the PIR sensor to the Raspberry Pi and the run the cod given below. Whenever any motion happens it will detect it.

Required Components:

PIR Sensor Module x 1

Breadboard x 1

Raspberry Pi x 1

Jumper Wires

Source Code:

```
import RPi.GPIO as GPIO
 import time


 sensor = 16
 buzzer = 18


 GPIO.setmode(GPIO.BOARD)
 GPIO.setup(sensor,GPIO.IN)
 GPIO.setup(buzzer,GPIO.OUT)


 GPIO.output(buzzer,False)
 Print("Initialzing PIR Sensor.....")
 time.sleep(12)
 Print("PIR Ready...")
 Print(" ")


 try:
   while True:
     if GPIO.input(sensor):
        GPIO.output(buzzer,True)
        Print("Motion Detected" )
        while GPIO.input(sensor):
           time.sleep(0.2)
     else:
        GPIO.output(buzzer,False)




 except KeyboardInterrupt:
    GPIO.cleanup()
```

Interfacing Rain sensor module with the Raspberry Pi

Introduction:

When no raindrops are on the sensor, the sensor controller's DO (digital out) pin is HIGH (3.3V in our case). When raindrops are detected this changes to LOW.

In this experiment we connected Raspberry Pi to Rain sensor and run the code. If sensor finds any water or the rain drops then it will print rain detected.

Required Components:

Rain Sensor x 1

Breadboard x 1

Raspberry Pi x 1

Jumper Wires

Source Code:

```
from time import sleep
from gpiozero import Buzzer, InputDevice

#buzz = Buzzer(13)
no_rain = InputDevice(18)

def buzz_now(iterations):
  for x in range(iterations):
      #buzz.on()
      sleep(0.1)
      buzz.off()
      sleep(0.1)

while True:
  if not no_rain.is_active:
      print("It's raining - get the washing in!")
      #buzz_now(5)
      # insert your other code or functions here
      # e.g. tweet, SMS, email, take a photo etc.
  sleep(1)
```

Interfacing Sound sensor module with the Raspberry Pi

Detects the noise in the environment and eleminates the background noise from there it gives any noise greater than the ambient noise.

Source Code:

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

#GPIO SETUP
channel = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(channel, GPIO.IN)

def callback(channel):
    if GPIO.input(channel):
            print "Sound Detected!"
    else:
            print "Sound Detected!"

GPIO.add_event_detect(channel, GPIO.BOTH, bouncetime=300)  # let us know when the pin goes HIGH or LOW
GPIO.add_event_callback(channel, callback)  # assign function to GPIO PIN, Run function on change

# infinite loop
while True:
        time.sleep(1)
```

Interfacing 16 x 2 LCD display with Raspberry pi

Introduction:

Interfacing 16 x 2 display with raspberry pi and print data on the display. For this experiment we have used user input which will be displayed in the 16 x 2 LCD display for 5 second and after that LCD will be cleared.

Required Components:

16x2 LCD display x 1

Breadboard x 1

Raspberry Pi x 1

Jumper Wires

Source Code:

```
#!/usr/bin/python
# Example using a character LCD connected to a Raspberry Pi
import time
import Adafruit_CharLCD as LCD

# Raspberry Pi pin setup
lcd_rs = 25
lcd_en = 24
lcd_d4 = 23
lcd_d5 = 17
lcd_d6 = 18
lcd_d7 = 22
lcd_backlight = 2

# Define LCD column and row size for 16x2 LCD.
lcd_columns = 16
lcd_rows = 2

lcd = LCD.Adafruit_CharLCD(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, lcd_columns, lcd_rows
, lcd_backlight)

lcd.message('Hello\nworld!')
# Wait 5 seconds

time.sleep(5.0)
lcd.clear()
text = raw_input("Type Something to be displayed: ")
lcd.message(text)

# Wait 5 seconds
time.sleep(5.0)
lcd.clear()
lcd.message('Goodbye\nWorld!')

time.sleep(5.0)
lcd.clear()
```

Uploading values of IoT sensors in the Cloud

Introduction:

In this experiment we have interfaced the Ultrasonic sensor with the Raspberry Pi. We made a flask web page that will take outputs of the ultrasonics sensor code and displays it.

Required Components:

Ultrasonic sensor x 1

Breadboard x 1

Raspberry Pi x 1

Jumper Wires

Source Code:

Flask code:

```
from flask import Flask,request,jsonify,render_template

app = Flask(__name__)
gloabl_variable = 0

@app.route("/")
def hello_world():
    return render_template('auto_update.html')

@app.route("/get_data")
def get_data():
    return jsonify({'distance': gloabl_variable})

@app.route('/set_data',methods=['GET','POST'])
def set_data():
    print(request.args.to_dict())
    global gloabl_variable
    gloabl_variable = request.args.get('distance')
    return 'thank you'

if __name__=='__main__':
    app.run(host='0.0.0.0',debug= True)
```

##

Ultrasonic Sensor Code which needs to send to the cloud:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO_TRIGGER = 18
GPIO_ECHO = 24

GPIO.setup(GPIO_TRIGGER , GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)

def distance():
    GPIO.output(GPIO_TRIGGER, True )
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False )

    starttime = time.time()
    stoptime = time.time()

    while GPIO.input(GPIO_ECHO)==0:
        starttime = time.time()

    while GPIO.input(GPIO_ECHO)==1:
        stoptime = time.time()

    timeescaped = stoptime - starttime

    distance = (timeescaped * 34300) /2
    return distance

if __name__ == '__main__':
    try:
        while True:
            dist = distance()
            print(f'Measured distance = {round(dist)}')
            # add the requets code here + imported
            time.sleep(0.2)

    except KeyboardInterrupt:
        print('why you stopped ??????????????????')
        GPIO.cleanup()
```

Establish wireless communication between 2 Raspberry pi using NRF and Bluetooth modules.

Introduction:

The NRF24L01 module is a low-cost (less than$3) ultra-low power, bidirectional transceiver module. It is designed to operate within the 2.4GHz ISM band which means it can be used for projects with industrial, scientific and medical applications. The module can achieve data rates as high as 2Mbits! and uses a high-speed SPI interface in order to communicate with the Arduino and other kind of microcontroller and development boards.

For this experiment we took two 2 Raspberry Pi s (one is sender and other is receiver) and connected each of them with the NRF module and transferred the data.

Required Components:

NRF Bluetooth Module x 2

Breadboard x 1

Raspberry Pi x 2

Jumper Wires

Source Code:

```
import RPi.GPIO as GPIO # import gpio
import time
#import time library
import spidev
from lib_nrf24 import NRF24
#import NRF24 library
# set the gpio mode
GPIO.setmode (GPIO.BCM)
# set the pipe address. this address shoeld be entered on the receiver alo
pipes = [[0xE0, 0xE0, 0xF1, 0xF1, 0xE0], [0xF1, 0xF1, 0xF0, 0xF0, 0xE0]]
radio NRF24 (GPIO, spidev. SpiDev())
# use the gpio pins
radio.begin(0, 25) # start the radio and set the ce, csn pin ce= GPI008, csn= GPI025
radio.setPayloadSize (32) # set the payload size as 32 bytes
radio.setChannel (0x76) # set the channel as 76 hex
radio.setDataRate (NRF24.BR_1MBPS) # set radio data rate
radio.setPALevel (NRF24.PA_MIN)
# set PA level
# set acknowledgement as true
radio.setAutoAck(True) radio.enableDynamicPayloads() radio.enableAckPayload() radio.openWritin
gPipe (pipes[0])
radio.printDetails()
```

```
sendMessage =
# open the defined pipe for writing
# print basic detals of radio
list("Hi..Arduino UNO") #the message to be sent
while len(sendMessage) < 32:
sendMessage.append(0)

while True:
start time.time()
radio.write(sendMessage)
#start the time for checking delivery time
# just write the message to radio
print("Sent the message: {}".format(sendMessage))
radio.startListening()
while not radio. available (0):
time.sleep(1/100)
# print a message after succesfull send # Start listening the radio
if time.time() - start > 2:
-
print("Timed out.") # print errror message if radio disconnected or not functioning anymore
break
radio.stoplistening()
# close radio
time.sleep(3) # give delay of 3 seconds
```

Bluetooth to receive data from Phone Bluetooth console

Introduction:

For this experiment we have used an Arduino to interface a Bluetooth device with it. after that connect the Bluetooth to mobile and in Bluetooth console mobile app, send string data. which will be shown in serial console in Arduino.

Required Components:

Arduino x 1

Mobile phone

Breadboard x 1

Bluetooth module x 1

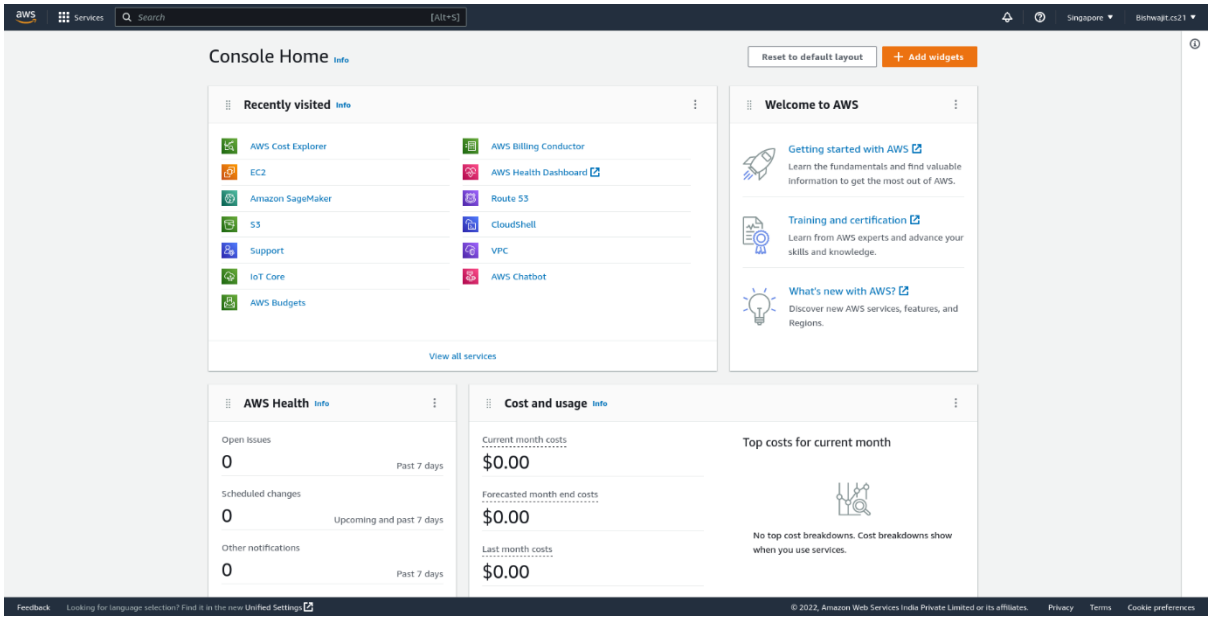
Jumper Wires

Source Code:

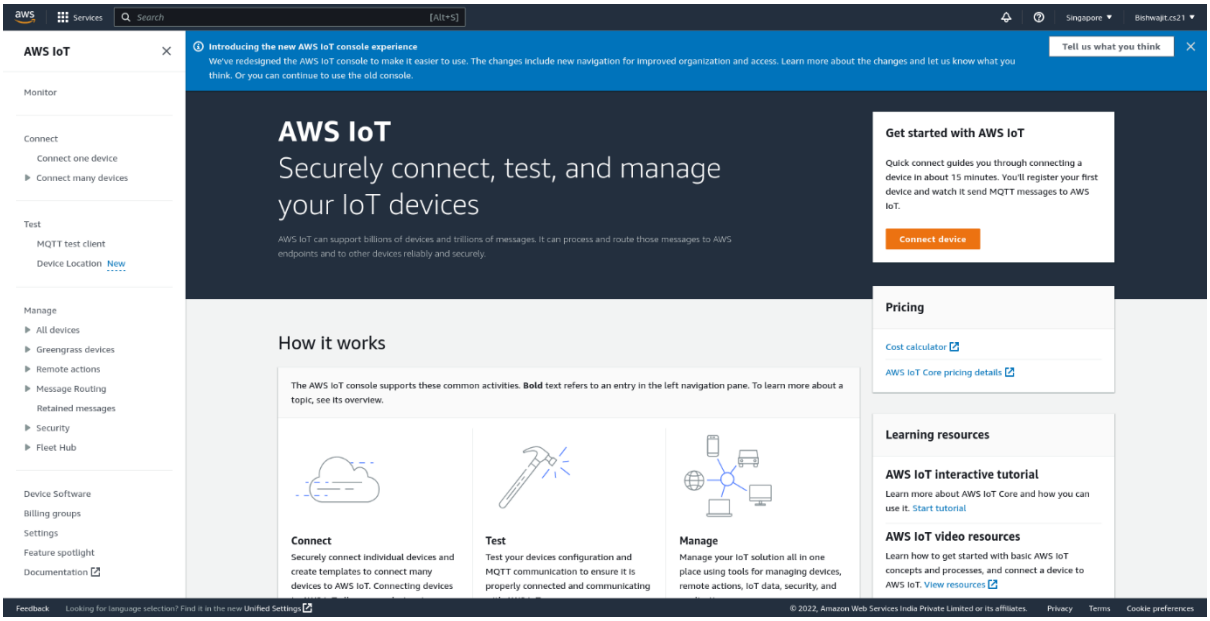
```
#include <SoftwareSerial.h>
SoftwareSerial EEBlue(10, 11); // RX | TX
void setup()
{

  Serial.begin(9600);
  EEBlue.begin(9600);  //Default Baud for comm, it may be different for your Module.
  Serial.println("The bluetooth gates are open.\n Connect to HC-
05 from any other bluetooth device with 1234 as pairing key!.");

}

void loop()
{

   // Feed any data from bluetooth to Terminal.
  if (EEBlue.available())
  Serial.write(EEBlue.read());

  // Feed all data from termial to bluetooth
  if (Serial.available())
  EEBlue.write(Serial.read());
}
```

Implement IoT Core service to deploy sensor values on AWS Cloud

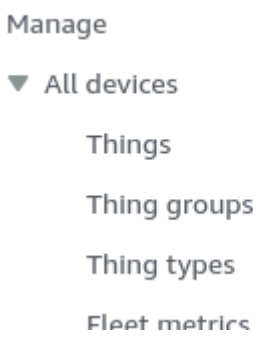
- For this experiment we'll be using MQTT. MQTT is a lightweight, publish-subscribe, machine to machine network protocol for Message queue/Message queuing service. It is designed for connections with remote locations that have devices with resource constraints or limited network bandwidth.
- AWS cloud give IoT core option and MQTT service with free SSL encryption support.
- First go to aws.amazon.com



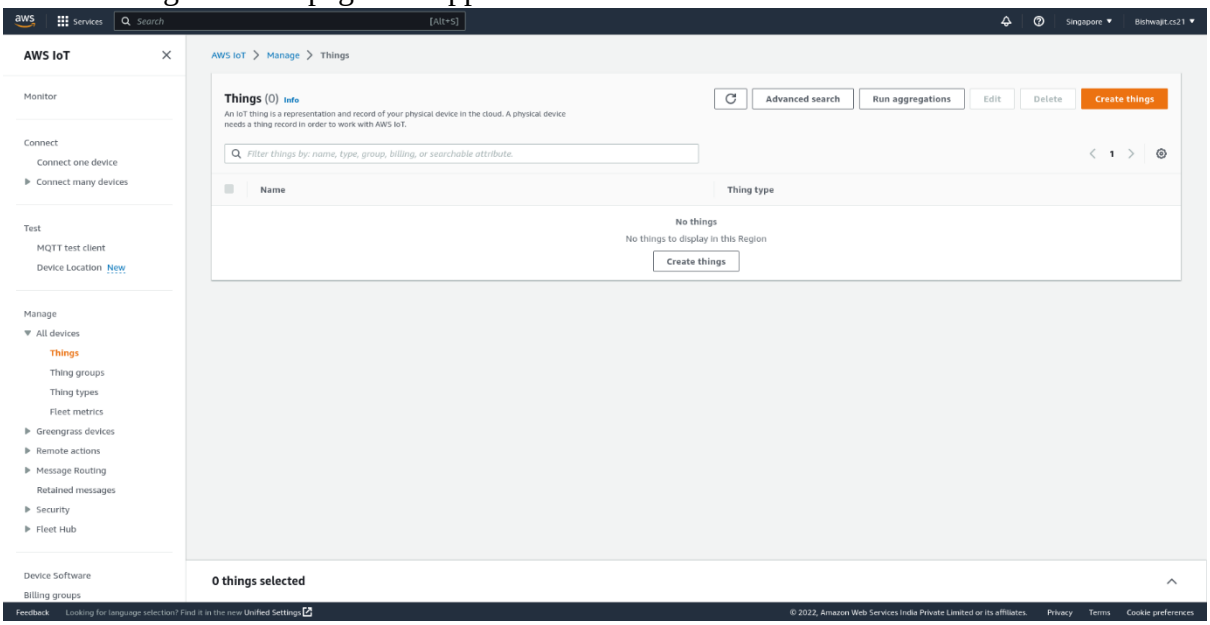
- From here go to IoT Core



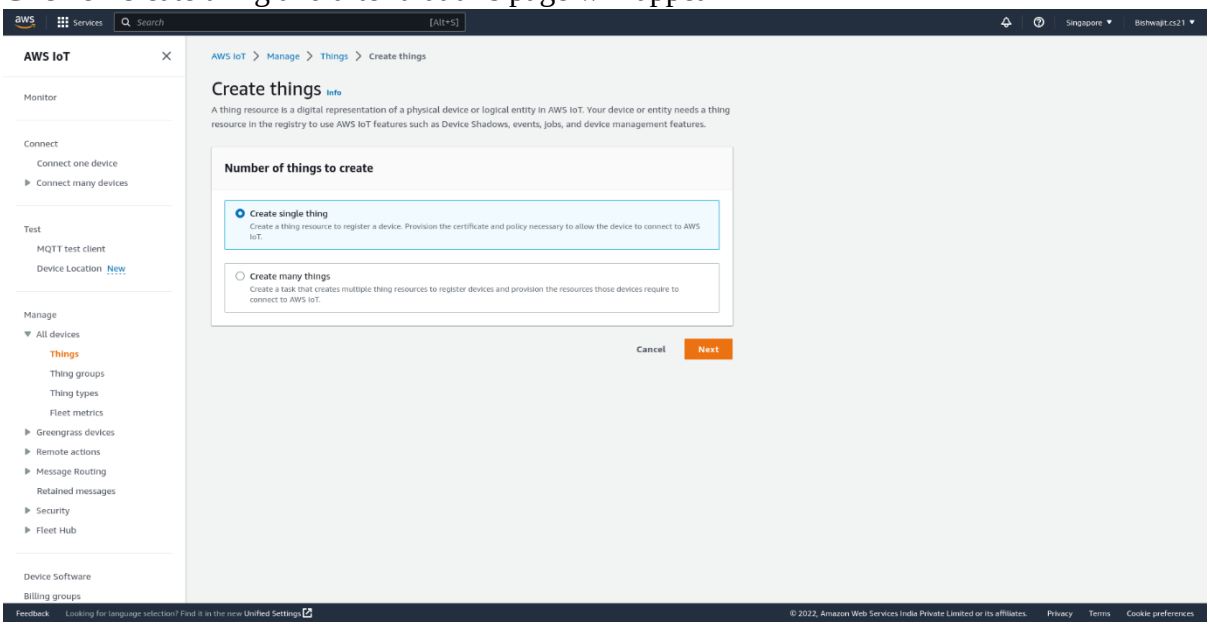
- From here go to All device > Things



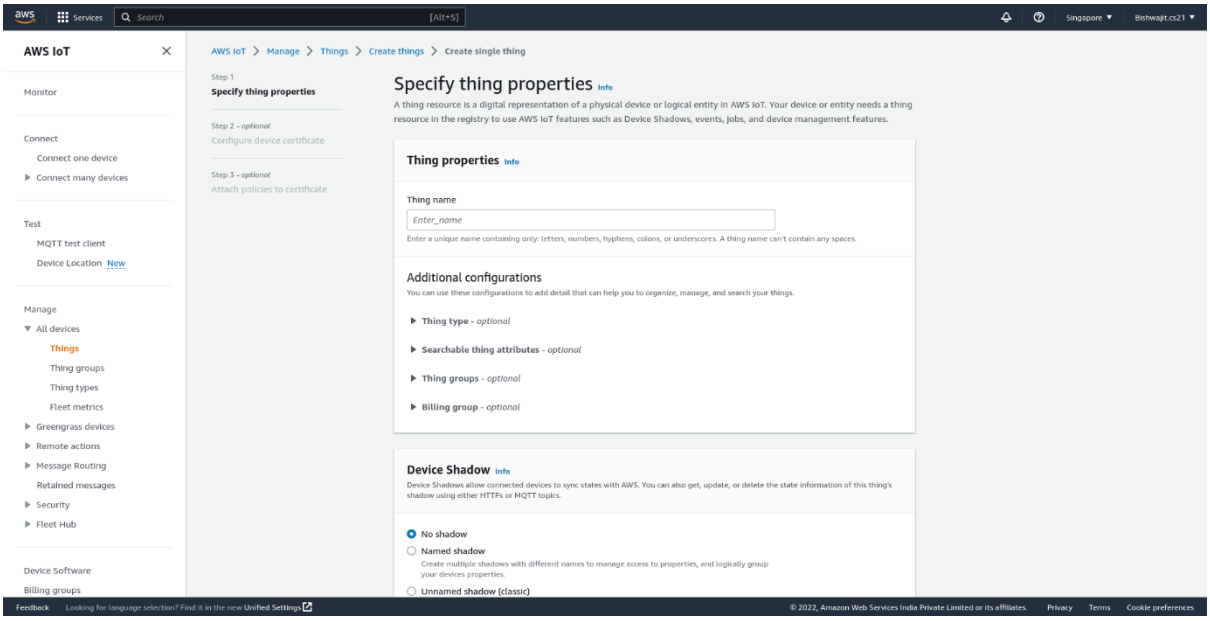
- click on things and this page will appear



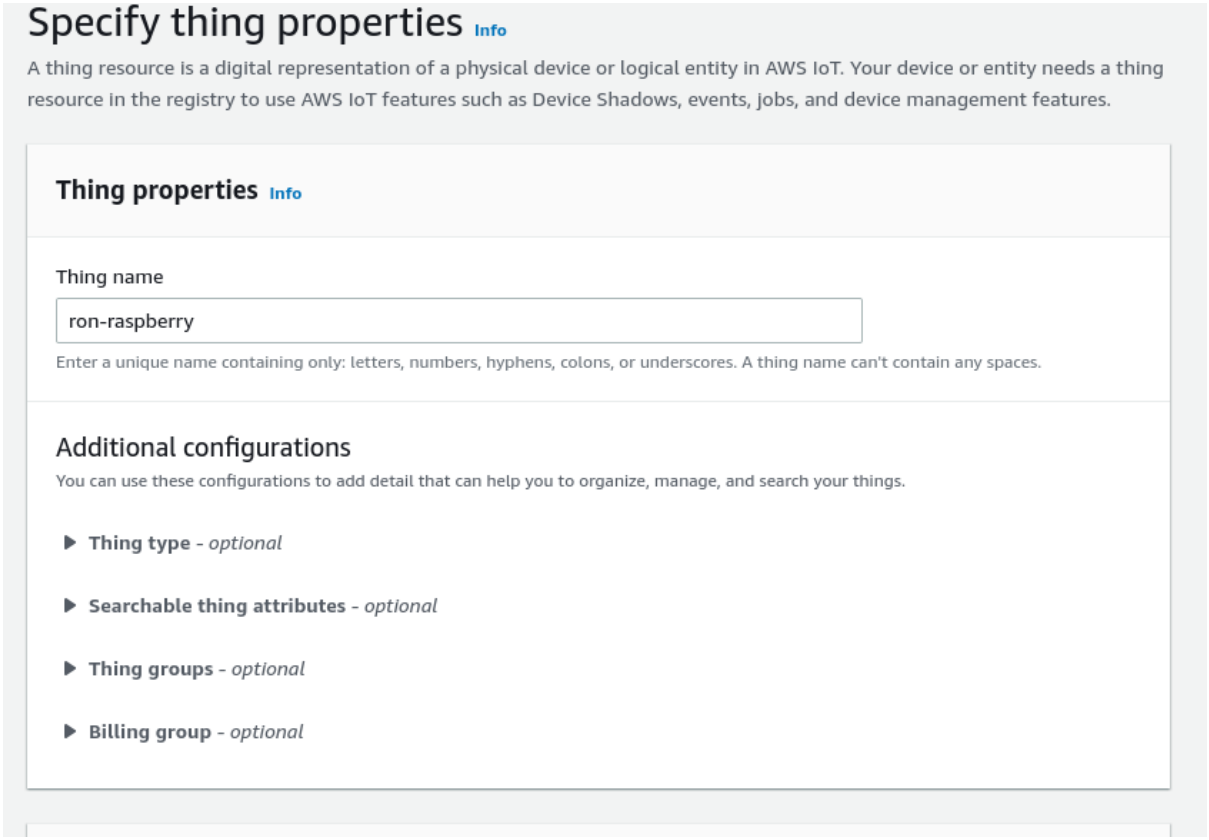
- Click on create thing and after that this page will appear



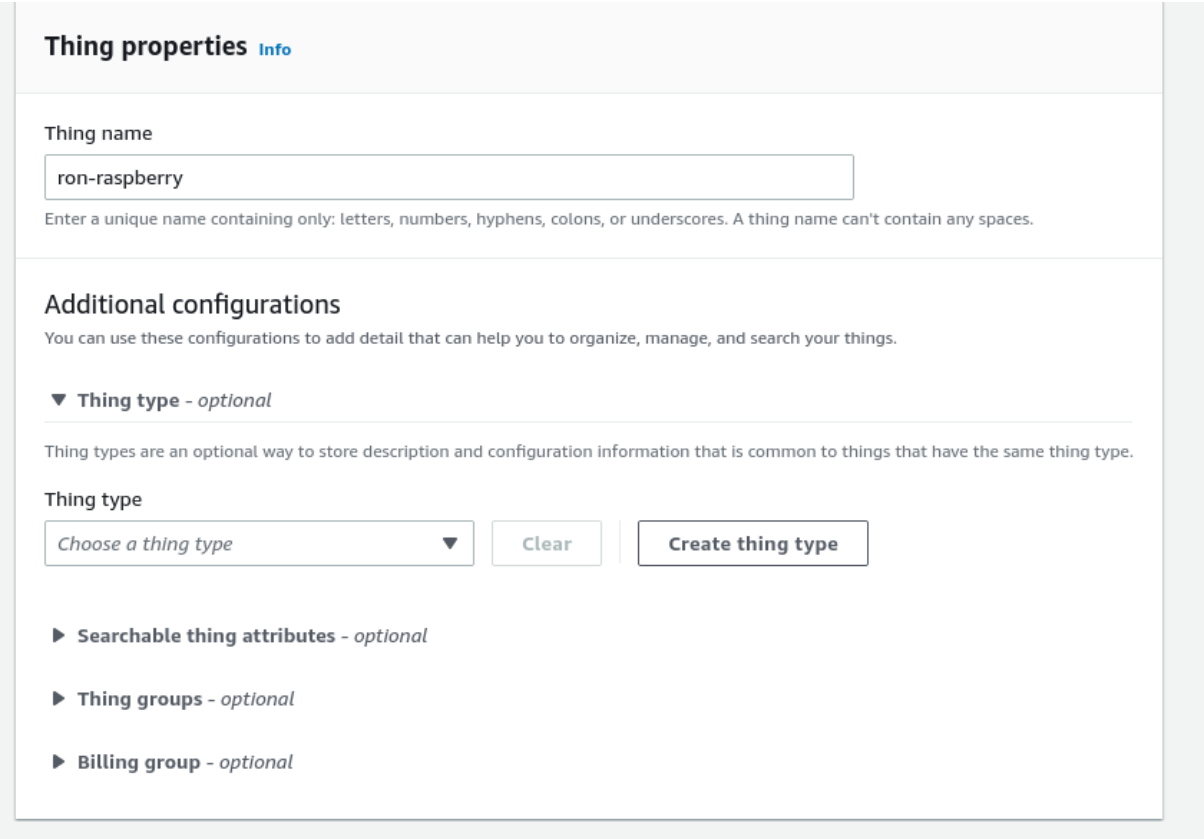
- Click on next and specify the name



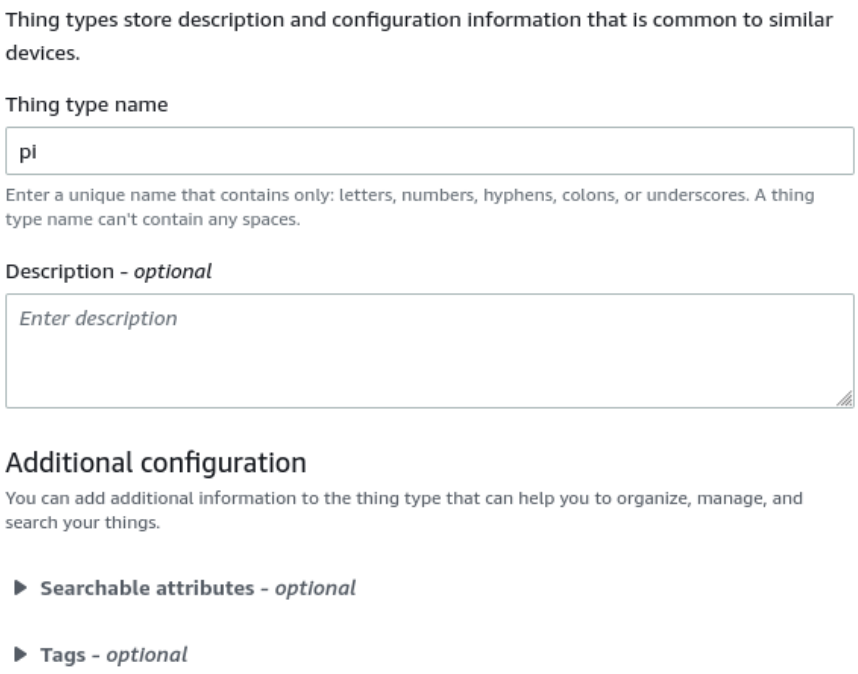
- For my Case I am using ron-raspberry



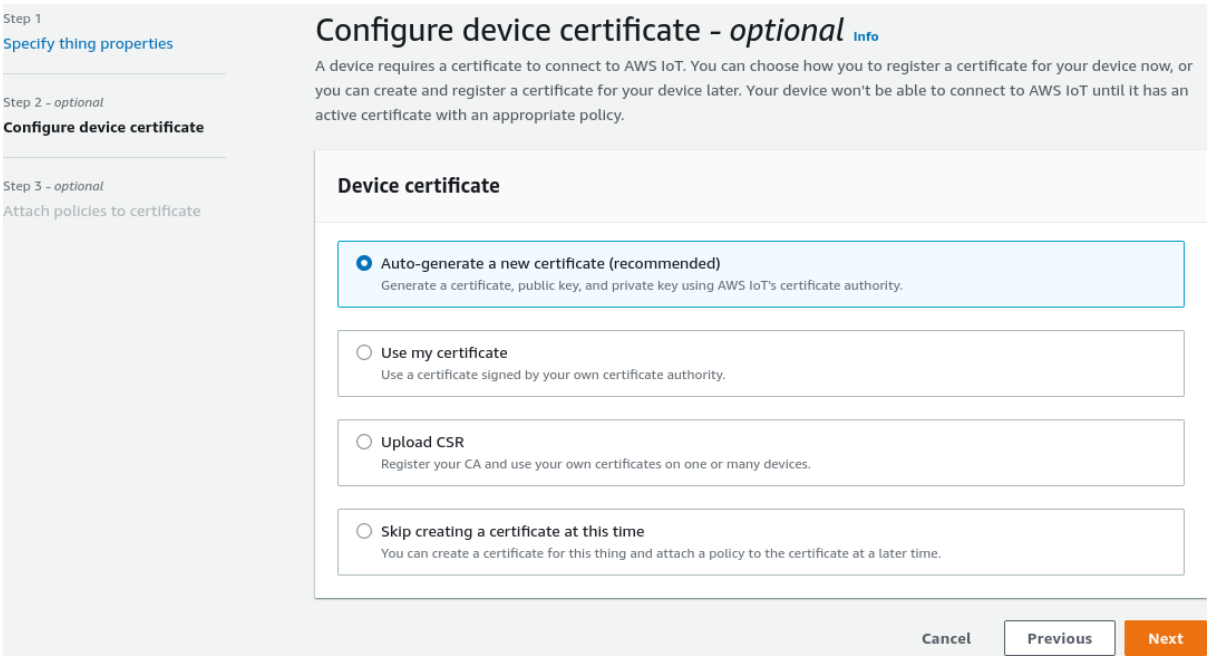
- Now in the thing type
    - create a thing type



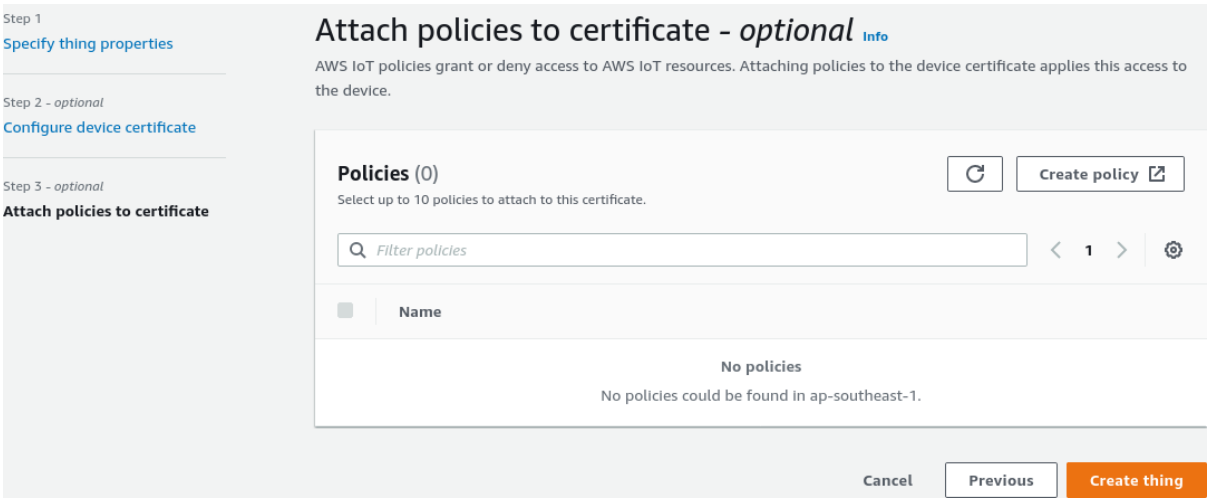
- specify the name as pi
  Change the thing type if needed in this case we are working with raspberry pi model 4 B



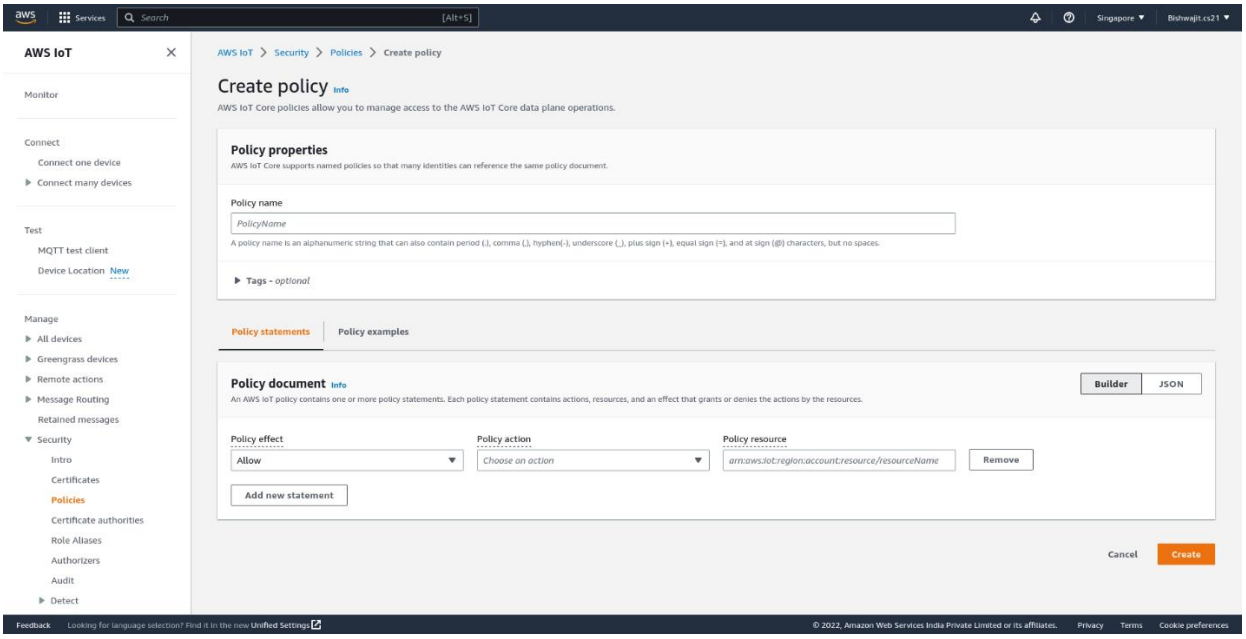
- Go to next and you will get a certification generation gateway



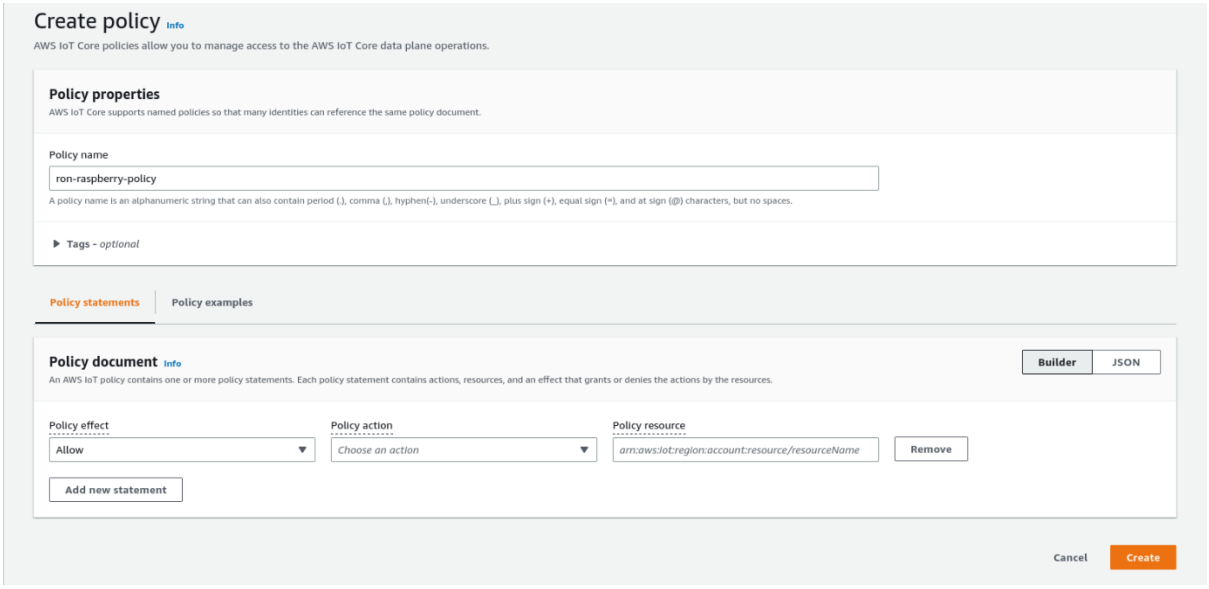
- Click on auto generate and go to next



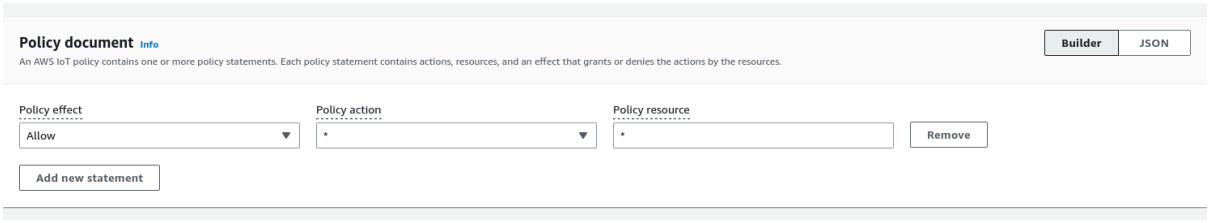
- Go to create policy where a new tab will open where you can create the policy and attactch with raspberry pi



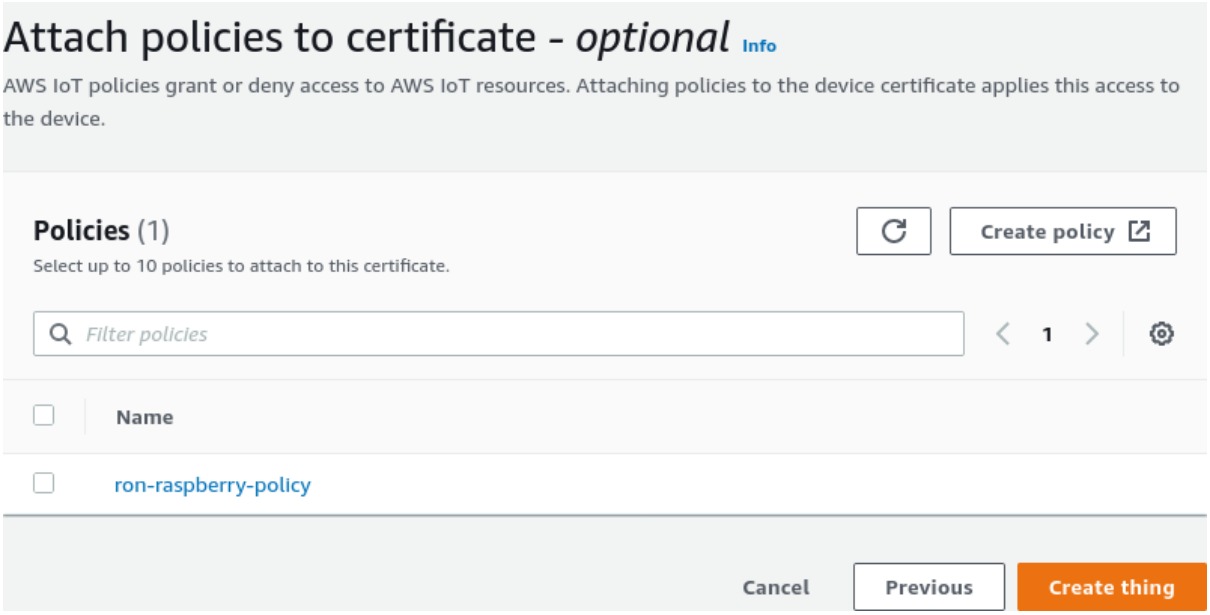
- Give a name for my case I am using ron-raspberry-policy



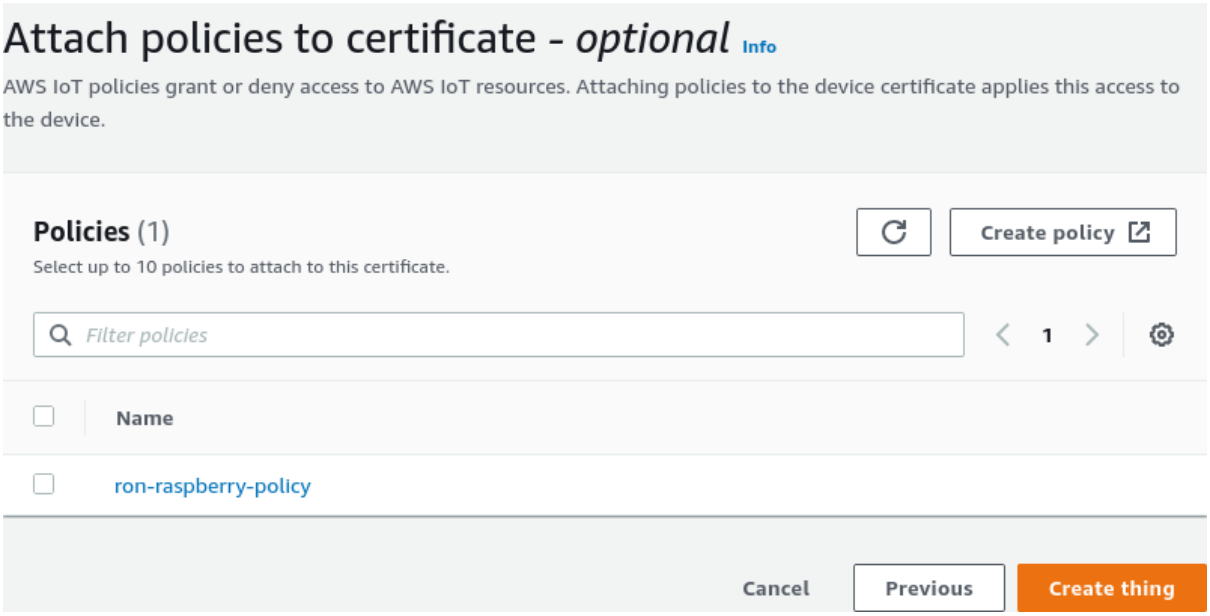
- after that set the polcy action to * and policy resource to *



- now close the TAB and go to previous tab where you will find this page



- now close the TAB and go to previous tab where you will find this page



- select ron-raspberry-policy and then create thing and you will get this screen

Download certificates and keys

Download certificate and key files to install on your device so that it can connect to AWS.

Device certificate

You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.

Device certificate

28a0b8ae553...te.pem.crt

Deactivate certificate | Download

Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

⚠ This is the only time you can download the key files for this certificate.

Public key file

28a0b8ae5536ddb6125f552...88ad0f9-public.pem.key

Download

Key downloaded

Private key file

28a0b8ae5536ddb6125f552...8ad0f9-private.pem.key

Download

Key downloaded

Root CA certificates

Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

Amazon trust services endpoint

RSA 2048 bit key: Amazon Root CA 1

Download

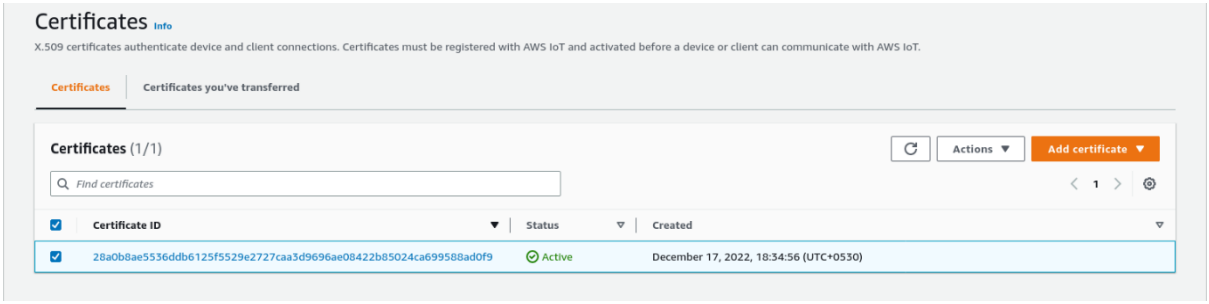
Amazon trust services endpoint

ECC 256 bit key: Amazon Root CA 3

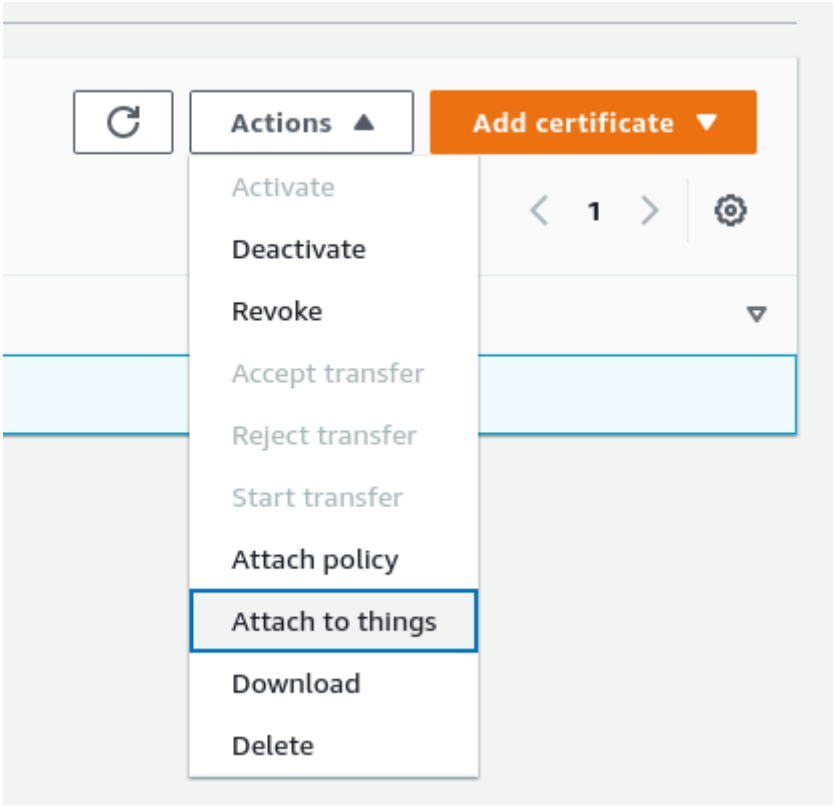
Download

If you don't see the root CA certificate that you need here, AWS IoT supports additional root CA certificates. These root CA certificates and others are available in our developer guides. Learn more

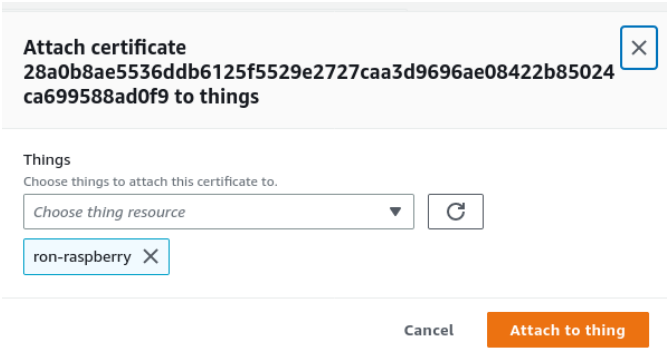
- Now as shown download device certificate + Public key + private key
- Then click on Done
- Go to certificates and attach to thing you created.



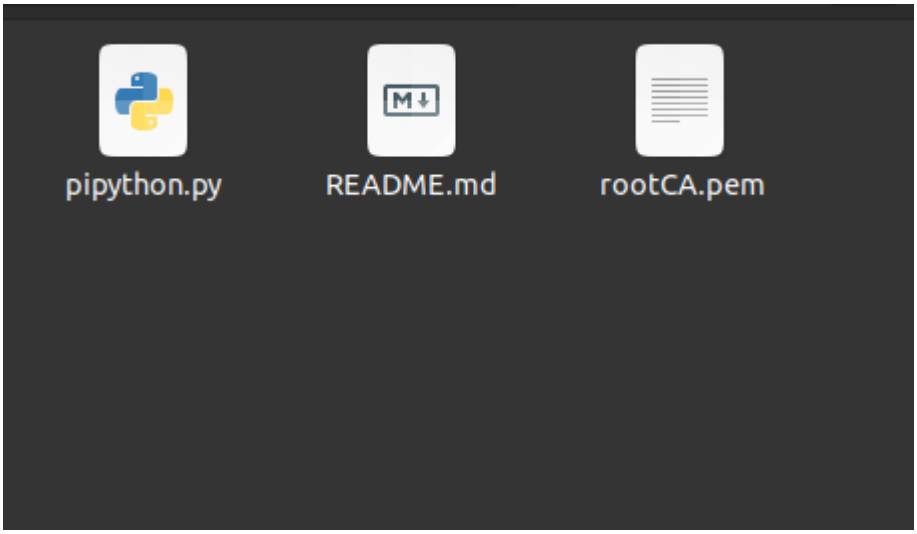
  - Select action and then select Attatch to a thing



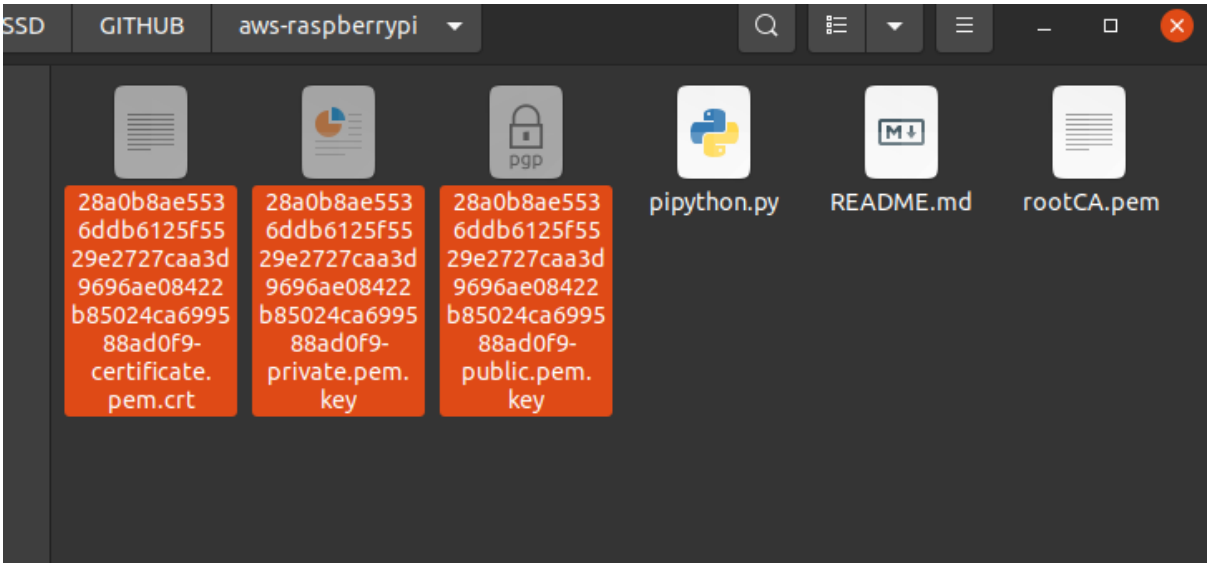
  - Type your thing name select and click on attatch to thing



- Clone the repo https://github.com/binaryupdates/aws-raspberrypi



- The repo consists of above files
- now COPY the three files here

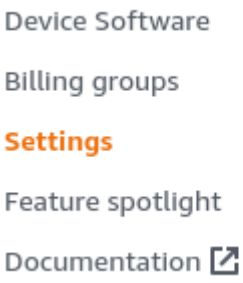


- Now change the piython.py to this code where change file of certificate and change the of private key

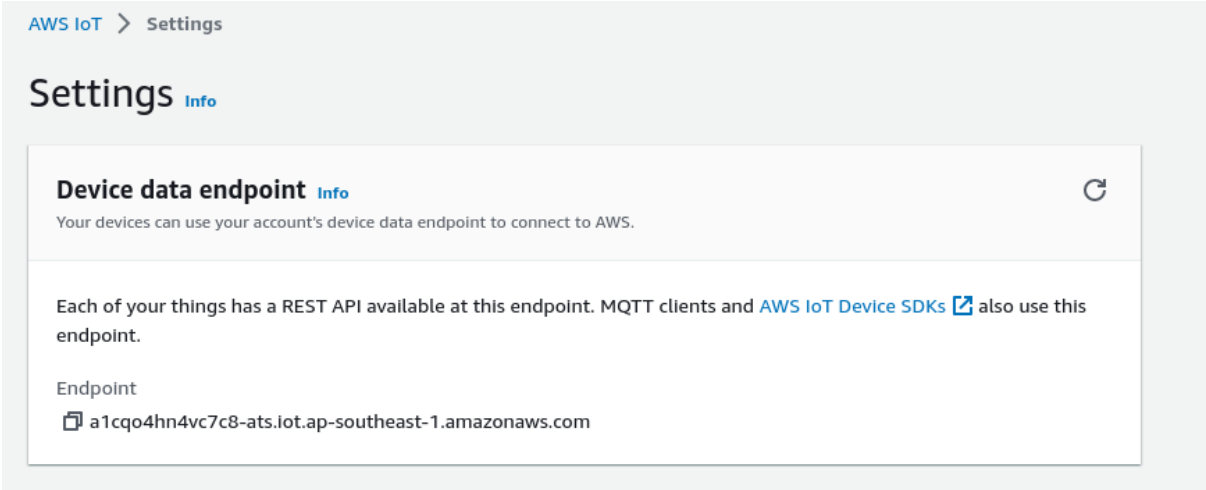
Source code

```
import time
import paho.mqtt.client as mqtt
import ssl
import json
import _thread
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setup(21, GPIO.OUT)

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))


client = mqtt.Client()
client.on_connect = on_connect
client.tls_set(ca_certs='./rootCA.pem', certfile='./28a0b8ae5536ddb6125f5529e2727caa3d9696ae08422b85024ca699588ad0f9-certificate.pem.crt', keyfile='./28a0b8ae5536ddb6125f5529e2727caa3d9696ae08422b85024ca699588ad0f9-private.pem.key', tls_version=ssl.PROTOCOL_SSLv23)
client.tls_insecure_set(True)
client.connect("a1cqo4hn4vc7c8-ats.iot.ap-southeast-1.amazonaws.com", 8883, 60) #Taken from REST API endpoint - Use your own.


def intrusionDetector(Dummy):
    while(1):
        x = GPIO.input(21)
        if(x==0):
            print("Just Awesome")
            client.publish("device/data", payload="Hello from BinaryUpdates!!" , qos=0, retain=False)
        time.sleep(5)

_thread.start_new_thread(intrusionDetector,("Create intrusion Thread",))

client.loop_forever()
```

- Here the client.connect URL you can find in the aws its different from others



- Go to settings in aws and in the enpoint you will find the url of yours aws settings for endpoint
    - From here you will see this appear



- Copy the end point and paste in
client.connect("a1cqo4hn4vc7c8-ats.iot.ap-southeast-1.amazonaws.com", 8883, 60) #Taken from REST API endpoint - Use your own.

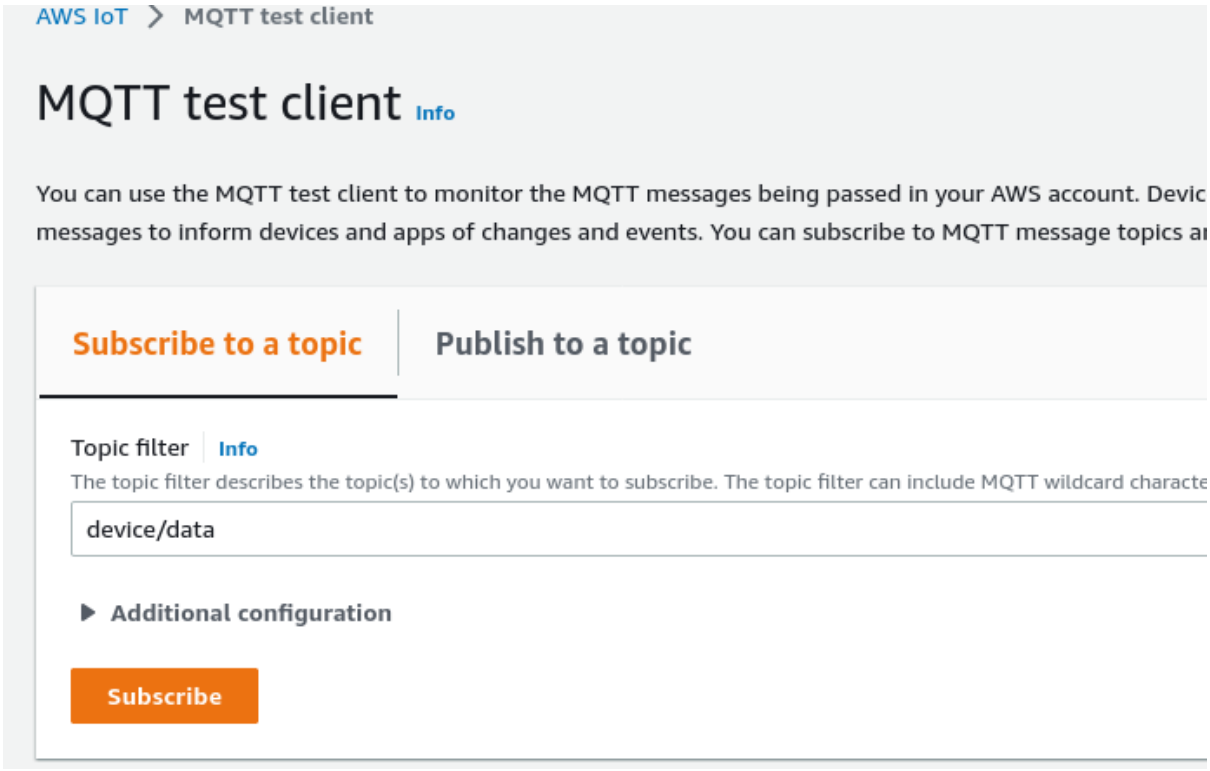
- you need to install paho-mqtt pypi
pip install paho-mqtt

- run the code and you will see this kind of output

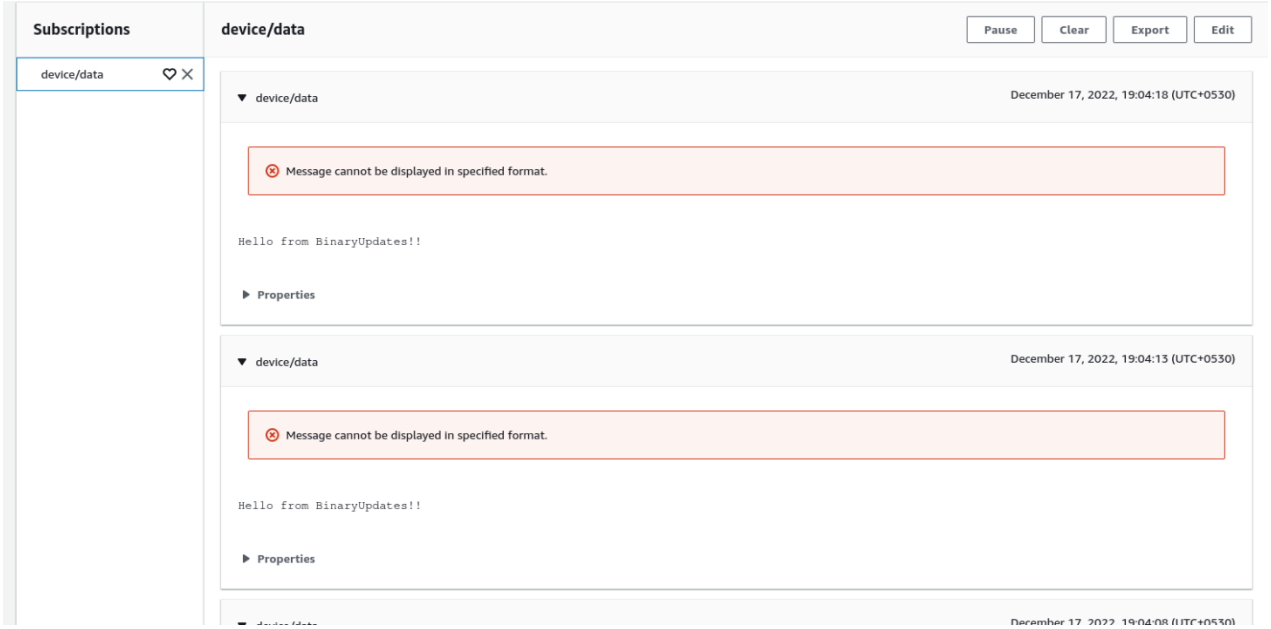
```
(pt_venv) ron@ron-linux:~/SECONDARY_SSD/GITHUB/aws-
raspberrypi$ python3 pipython.py
Just Awesome
Connected with result code 0
Just Awesome
Just Awesome
Just Awesome
Just Awesome
Just Awesome
Just Awesome
Just Awesome
```

```
Just Awesome
Just Awesome
Just Awesome
```

- TO if its working go to AWS and click on MQTT Test Client and type the topic name device/data and click on



- Click on subscribe
  The output will be like this



- IOT message is publishing

Establishing client server communication

By using request module in the client side we can establish the connection between server. For this experiment we can use the client side requests.py. Server situated in another place.

Source code:

```
import requests
import random
import time
for i in range(100):
    x = requests.get(f'http://10.10.27.169:5000/set_data?distance={random.randint(0,99)}')
    print(x)
    time.sleep(1)
```

Sniffing messages communicated between client and server

Install scapy (using pi in your python environment) in your system and run following code

Source Code:

```
#!/usr/bin/env python
import scapy.all as scapy
import argparse
from scapy.layers import http
def get_interface():
    parser = argparse.ArgumentParser()
    parser.add_argument("-i", "--interface", dest="interface", help="Specify interface on which to sniff
packets")
    arguments = parser.parse_args()
    return arguments.interface

def sniff(iface=None):
    scapy.sniff(store=False, prn=process_packet)

def process_packet(packet):
```

```
    if packet.haslayer(http.HTTPRequest):
        print(f"[+] Http Request >>   {packet[http.HTTPRequest].Host} to destination
{packet[scapy.IP].src}  port number {packet[scapy.TCP].sport} location :
{packet[http.HTTPRequest].Path}")
        if packet.haslayer(scapy.Raw):
            load = bytes(packet[scapy.Raw].load)
            print(load)
            keys = ["username", "password", "pass", "email"]
            # for key in keys:
            #     if key in load:
            #         print("\n\n\n[+] Possible password/username >> " + load + "\n\n\n")
            #         break

# iface = get_interface()
sniff()
```

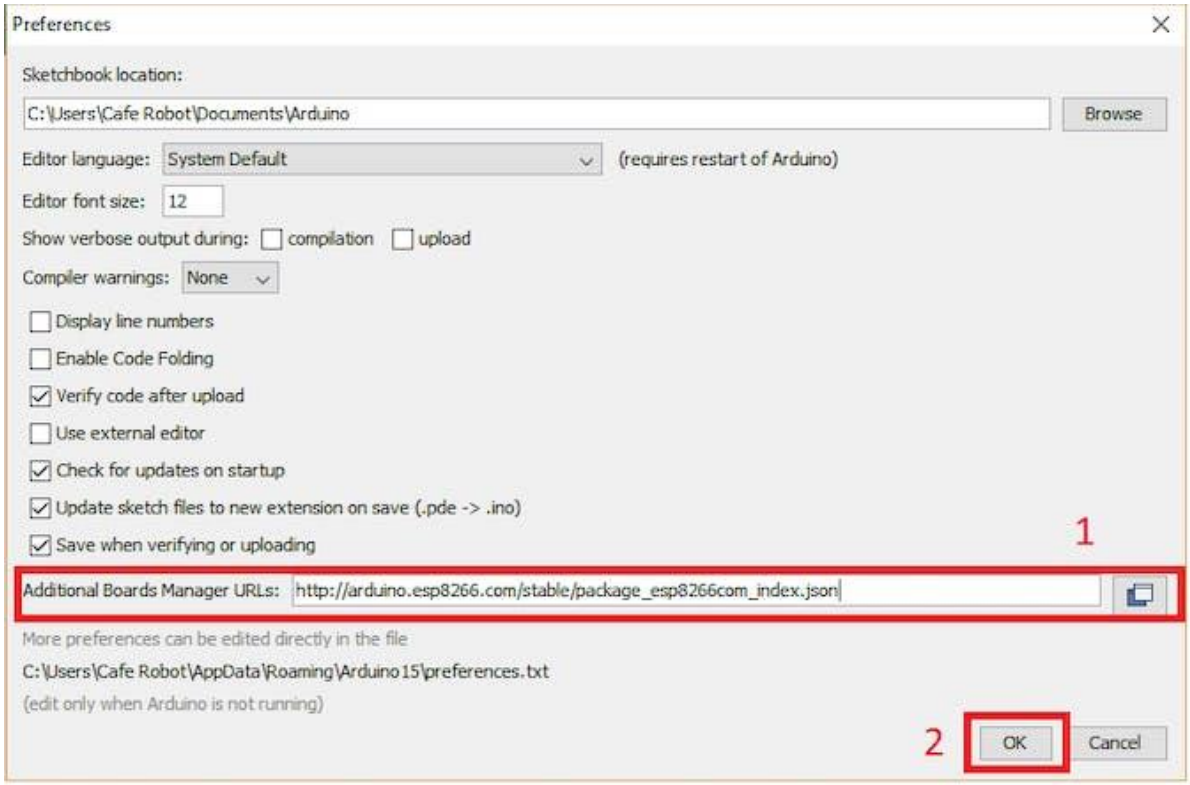
Configure ESP8266 using Aurdino IDE and configure GPIOs to Run the Hellp World Program (Using LED)

- The ESP8266 board comes with built in LED. Which can be used to blink with a delay of 250ms.

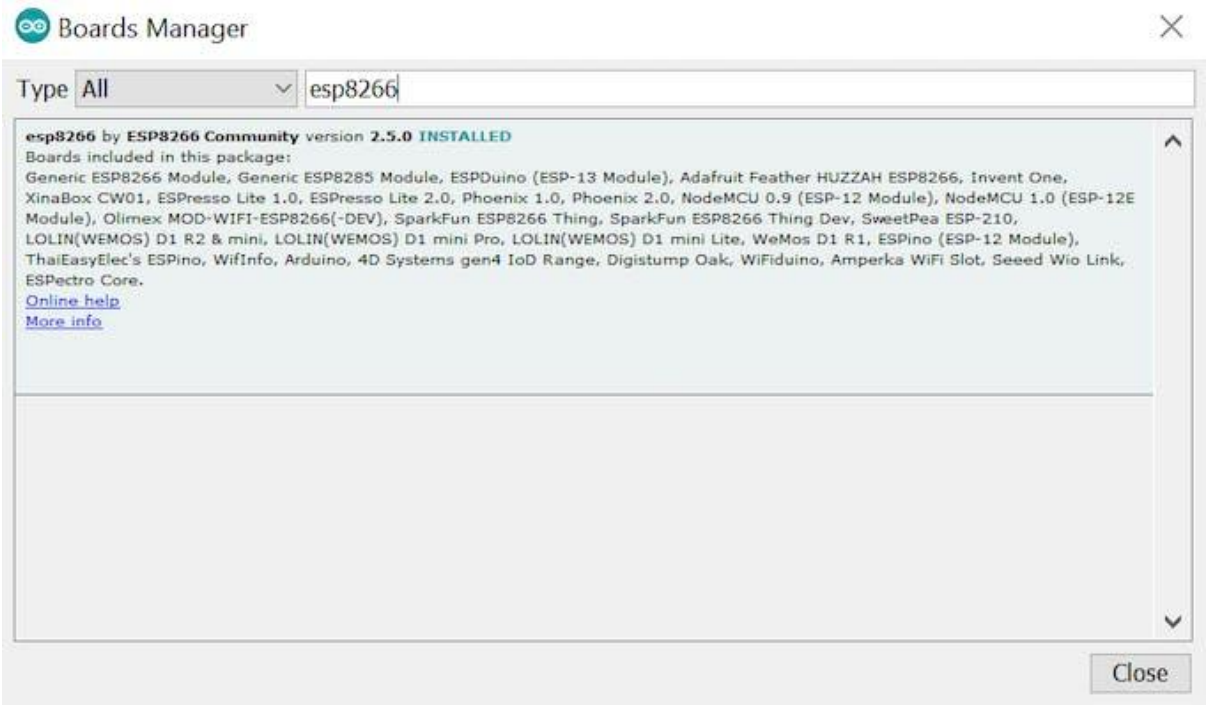
- After connecting the ESP8266 to we need to import/download ESP8266 binaries so that machine can compile the code compatible with the binaries in the ESP8266, below are the instructions to do that.
- The steps the bellow / if already done not needed

- COPY the link bellow
    - http://arduino.esp8266.com/stable/package_esp8266com_index.json

      Choose Preferences in the File menu and enter the copied code in Additional Board Manager URLs part. Then press OK.

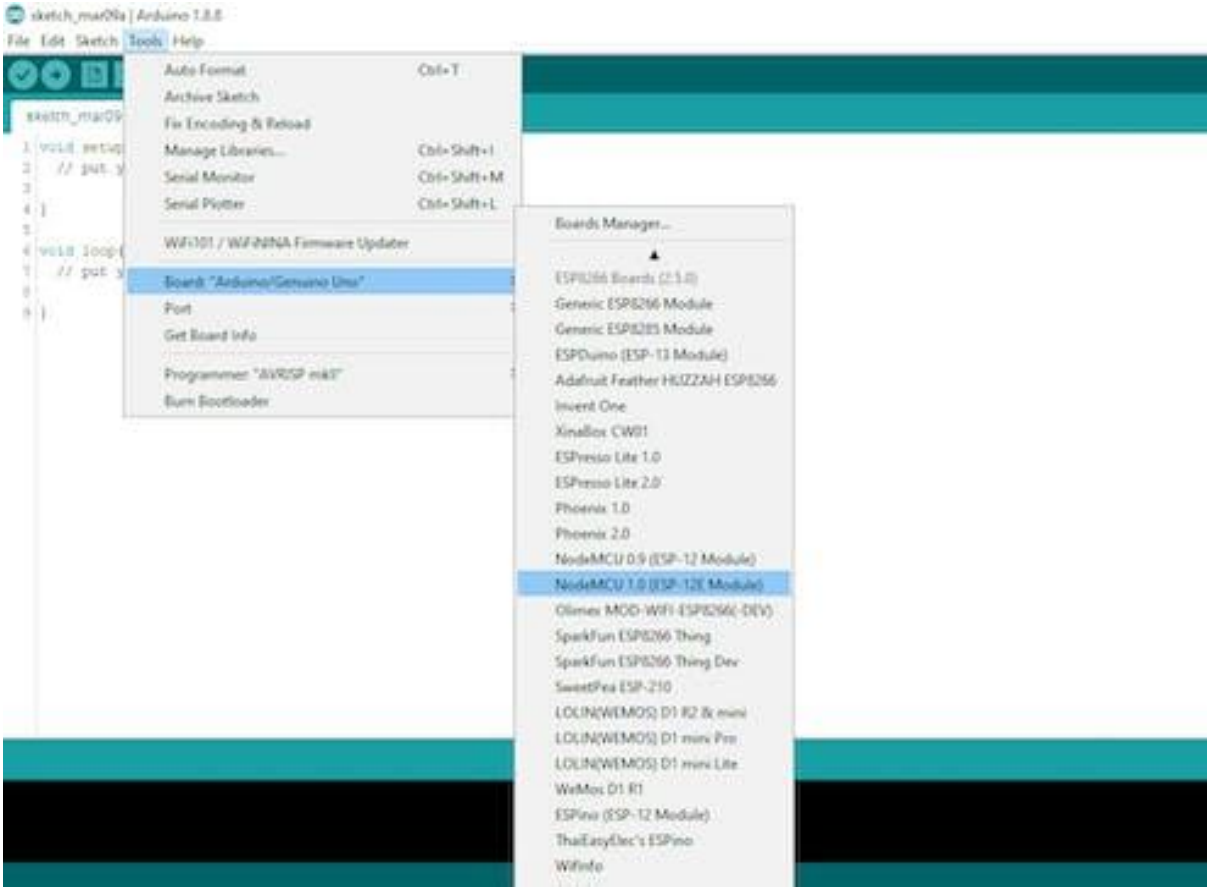


- Search the word ESP8266 in Boards>boards manager from Tools menu. Then install ESP8266 boards. After complete installation, you will see the INSTALLED label on ESP8266 boards.



- After these two steps, you can see ESP8266 based boards such as NodeMCU in your Arduino IDE boards list, and you can choose your desired board to upload the code.

  *

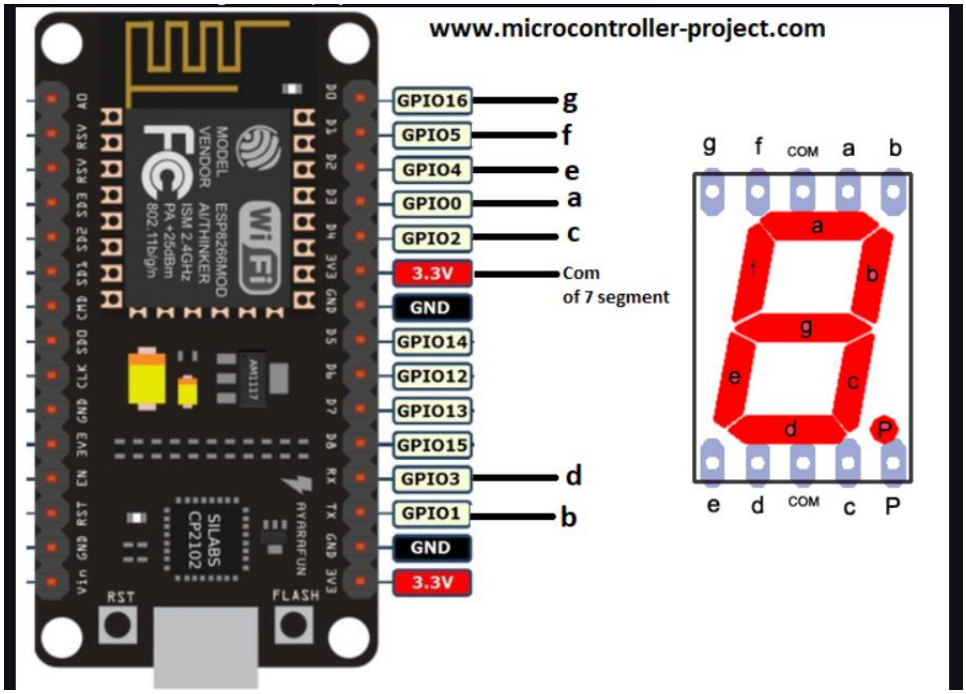


- Copy and run this simple code in your audrino IDE

```
void setup() {
// initialize digital pin LED_BUILTIN as an output.
pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
delay(250);                       // wait for a second
digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
delay(250);                       // wait for a second
}
```

Configure ESP8266 using Aurdino IDE and display characters in the Seven Segment Display.

- Connect the seven segment display in this manner



- Run the following code:
  Source Code:

```
#include <ESP8266WiFi.h>

//const char* ssid = "Your SSID";
//const char* password = "Your Wifi Password";

//Seven segment pins attachecd with nodemcu pins
int a = 0;  //Gpio-0 with a of 7 segment display
int b = 1;  //Gpio-1 with b of 7 segment display
int c = 2;  //Gpio-2 with c of 7 segment display
int d = 3;  //Gpio-3 with d of 7 segment display
int e = 4;  //Gpio-4 with e of 7 segment display
int f = 5;  //Gpio-5 with f of 7 segment display
int g = 16; //Gpio-16 with g of 7 segment display

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(a, OUTPUT);
  pinMode(b, OUTPUT);
```

```
  pinMode(c, OUTPUT);
  pinMode(d, OUTPUT);
  pinMode(e, OUTPUT);
  pinMode(f, OUTPUT);
  pinMode(g, OUTPUT);
}

void loop() {
  digitalWrite(a, LOW);
  delay(250);
  digitalWrite(b, LOW);
  delay(250);
  digitalWrite(c, LOW);
  delay(250);
  digitalWrite(d, LOW);
  delay(250);
  digitalWrite(e, LOW);
  delay(250);
  digitalWrite(f, LOW);
  delay(250);
  digitalWrite(g, LOW);
  delay(250);
  digitalWrite(a,HIGH);
  digitalWrite(b,HIGH);
  digitalWrite(c,HIGH);
  digitalWrite(d,HIGH);
  digitalWrite(e,HIGH);
  digitalWrite(f,HIGH);
  digitalWrite(g,HIGH);
  delay(250);
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, HIGH);
  digitalWrite(d, HIGH);       //Displaying 1
  digitalWrite(e, LOW);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
  delay(250);
  digitalWrite(a, LOW);
  digitalWrite(b, LOW);
  digitalWrite(c, HIGH);      //Displaying 2
  digitalWrite(d, LOW);
  digitalWrite(e, LOW);
  digitalWrite(f, HIGH);
  digitalWrite(g, LOW);
  delay(250);
  digitalWrite(a, LOW);
  digitalWrite(b, LOW);
```

```
digitalWrite(c, HIGH);      //Displaying 2
digitalWrite(d, LOW);
digitalWrite(e, LOW);
digitalWrite(f, HIGH);
digitalWrite(g, LOW);
delay(250);
digitalWrite(a, LOW);
digitalWrite(b, LOW);
digitalWrite(c, LOW);
digitalWrite(d, LOW);      //Displaying 3
digitalWrite(e, HIGH);
digitalWrite(f, HIGH);
digitalWrite(g, LOW);
delay(250);
digitalWrite(a, HIGH);
digitalWrite(b, LOW);
digitalWrite(c, LOW);
digitalWrite(d, HIGH);
digitalWrite(e, HIGH);     //Displaying 4
digitalWrite(f, LOW);
digitalWrite(g, LOW);
delay(250);
digitalWrite(a, LOW);
digitalWrite(b, HIGH);
digitalWrite(c, LOW);
digitalWrite(d, LOW);
digitalWrite(e, HIGH);     //Displaying 5
digitalWrite(f, LOW);
digitalWrite(g, LOW);
delay(250);
 digitalWrite(a, LOW);
digitalWrite(b, HIGH);
digitalWrite(c, LOW);
digitalWrite(d, LOW);    //Displaying 6
digitalWrite(e, LOW);
digitalWrite(f, LOW);
digitalWrite(g, LOW);
delay(250);
digitalWrite(a, LOW);
digitalWrite(b, LOW);
digitalWrite(c, LOW);
digitalWrite(d, HIGH);
digitalWrite(e, HIGH);   //Displaying 7
digitalWrite(f, HIGH);
digitalWrite(g, HIGH);
delay(250);
digitalWrite(a, LOW);
digitalWrite(b, LOW);
```

```
  digitalWrite(c, LOW);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);   //Displaying 7
  digitalWrite(f, HIGH);
  digitalWrite(g, HIGH);
  delay(250);
  digitalWrite(a, LOW);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);    //Displaying 9
  digitalWrite(f, LOW);
  digitalWrite(g, LOW);
  delay(250);
  digitalWrite(a,HIGH);
  digitalWrite(b,HIGH);
  digitalWrite(c,HIGH);
  digitalWrite(d,HIGH);
  digitalWrite(e,HIGH);
  digitalWrite(f,HIGH);
  digitalWrite(g,HIGH);
  delay(250);

}
```