

IOT SECURITY AND PRIVACY

ASSIGNMENT 6 – AWS IOT

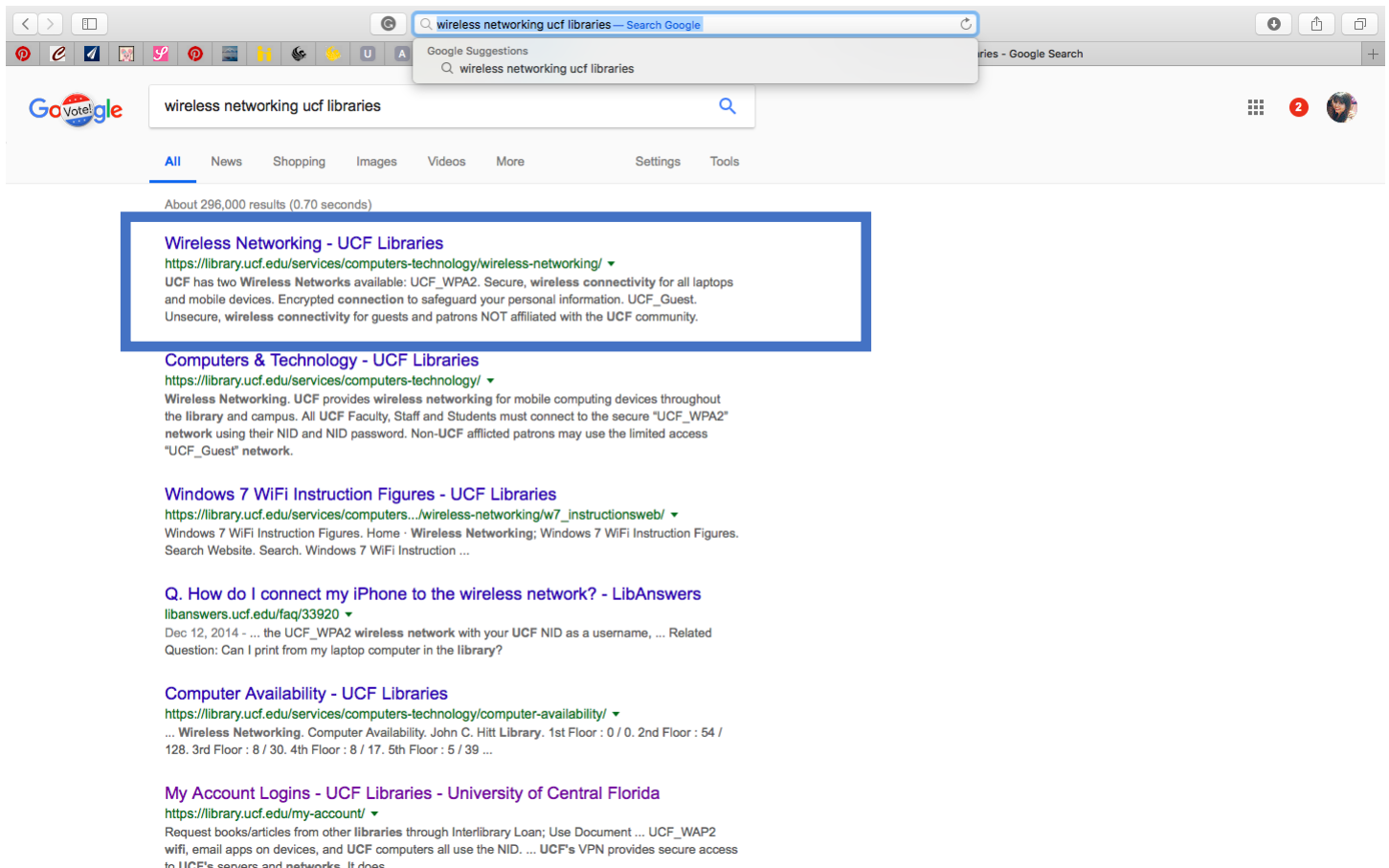
By:- Anmol Sureshkumar Panchal

Questions:

In this assignment, students are required to connect ESP32 through UCF_WPA2 or eduroam to Amazon AWS IoT and update the state of the connected sensor (DHT11) continuously with the AWS IoT thing shadow. The data such as the state generated by the sensor should be written into Amazon DynamoDB. Note: be careful not to exceed the free account quota of data for AWS IoT. Please refer to the references [1][2][3][4][5][6][7][8][9], particularly [1] [2], if necessary for this assignment.

Requirements:

1. Document in detail the procedures connecting ESP32 through *UCF_WPA2* or *eduroam* to Amazon AWS IoT, including configuring Amazon AWS IoT and writing code connecting to Amazon AWS IoT. (8 points)
- Go to <https://library.ucf.edu/services/computers-technology/wireless-networking/> and download the certificate.



Pages

- Computer Availability
- Computer Policy
- Computer Specifications
- LibTech
- Software
- Study Areas with Tech
- Technology Help
- Technology Lending
- Wireless Networking

Wireless Networking

UCF has two Wireless Networks available:

- **UCF_WPA2**
 - Secure, wireless connectivity for all laptops and mobile devices
 - Encrypted connection to safeguard your personal information.
 - Uses your NID & NID Password for authentication
 - Faculty, Staff, and UCF Students should always connect to UCF_WPA2.
- **UCF_Guest**
 - Unsecure, wireless connectivity for guests and patrons NOT affiliated with the UCF community.
 - No encryption; Do NOT use for web browsing that requires submitting personal information.
 - Requires authentication via Acceptable Use Policy page before fully connected.
 - Only university guest may connect to UCF_Guest.

Use the tabs below to view directions on how to connect your device to the UCF_WPA2 network.

Additional wireless networking support can be found at wireless.ucf.edu

Windows 7Windows 8 / 8.1 / 10Mac OS XiOS (iPhone/iPad)Android

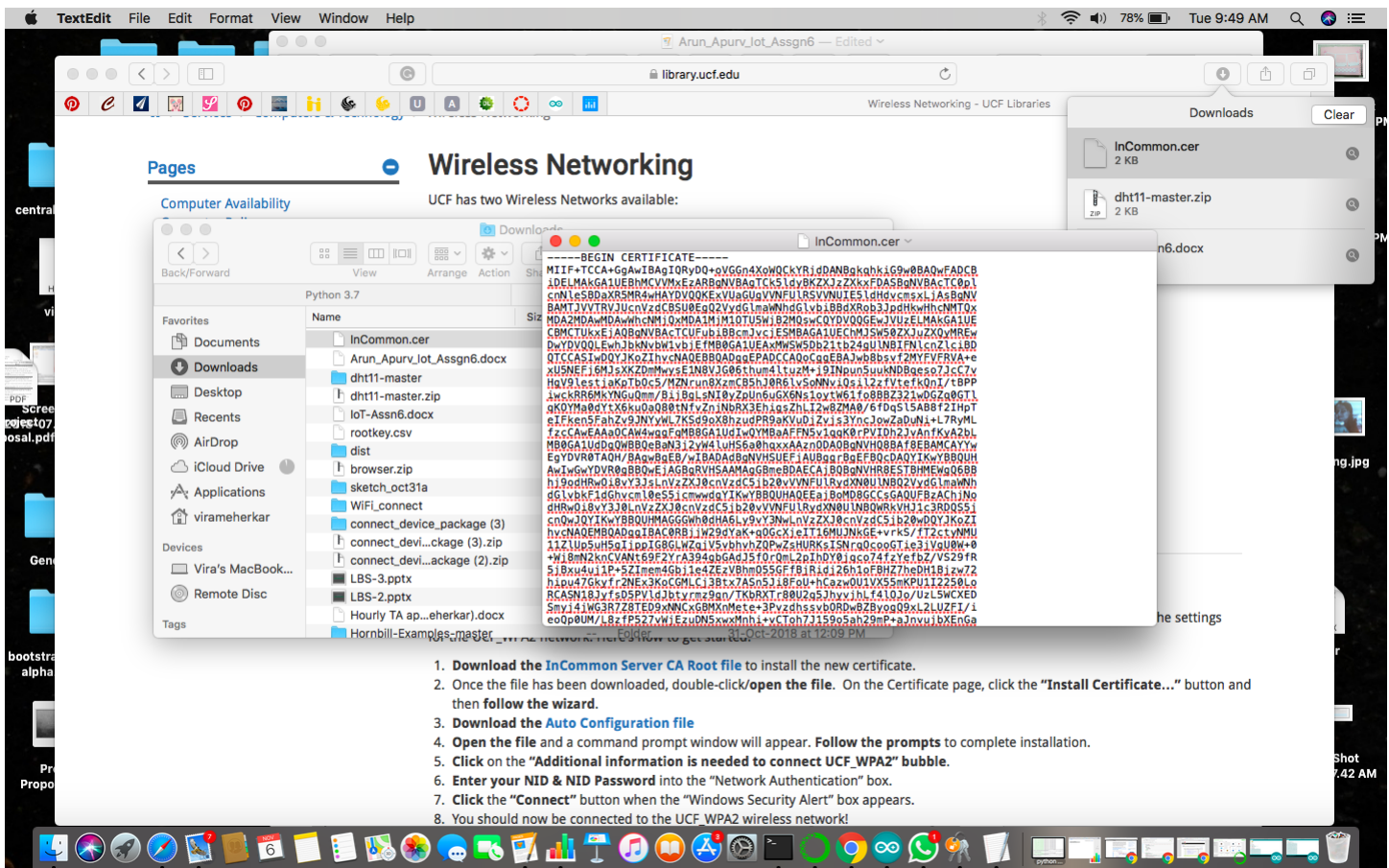
Windows 7

Windows 7 requires you to first install a security certificate and then run a special script that automatically sets up all of the settings for the UCF_WPA2 network. Here's how to get started:

1. **Download the InCommon Server CA Root file** to install the new certificate.
2. Once the file has been downloaded, double-click/**open the file**. On the Certificate page, click the **"Install Certificate..."** button and then **follow the wizard**.
3. **Download the Auto Configuration file**
4. **Open the file** and a command prompt window will appear. **Follow the prompts** to complete installation.
5. Click on the **"Additional information is needed to connect UCF_WPA2"** bubble.
6. **Enter your NID & NID Password** into the "Network Authentication" box.
7. Click the **"Connect"** button when the "Windows Security Alert" box appears.
8. You should now be connected to the UCF_WPA2 wireless network!

Still having trouble? Stop by the LibTech desk on the 3rd floor for friendly assistance.

- Once downloaded the certificate open it in text Editor and copy paste the certificate into the code as a



array of character.

- Then use the WPA2Enterprise library to configure the certificate with you UCFID and Password.

```

Arduino File Edit Sketch Tools Help
WiFi_connect | Arduino 1.8.7

WiFi_connect
#include "esp_wpa2.h"
#include <WiFi.h>
String line; //variable for response
const char* ssid = "UCF_WPA2"; // WPA2 SSID
const char* host = "arduino.php5.sk"; //external server domain
#define EAP_IDENTITY "vi975487" //NID
#define EAP_PASSWORD "Alpa203948" //Password
const char* server_ca_cert = "-----BEGIN CERTIFICATE-----\n"
"MIIF-TCCA-GgwwIBAgIQryDQ+oVGGn4XowWCKYRjdANBgkqhkiG9w0BAQwFADCB\n"
"iDELMAkGA1UEBHMCMYmEzARBgNVAgTCK51dyBkZXJzZXkxPDA5BgNVBACTQ0p1\n"
"cnNleS80aXRSMR4wYAYDVQKExVUadugVNFULRSVVNUE5Ldhdvcm5xLjAsBgNV\n"
"BAUTJVVTRVJUCnVzCDBSU0EgQ2VydGlnaW9uWm91dGlnaW9uWm91dGlnaW9u\n"
"MDA2MDAwMDAwMjQxMjQxMjQxMjQxMjQxMjQxMjQxMjQxMjQxMjQxMjQxMjQx\n"
"CBMCTUkxJAGBgNVBACTQ0p1cnNleS80aXRSMR4wYAYDVQKExVUadugVNFULRSVVNUE5Ldhdvcm5xLjAsBgNV\n"
"BAUTJVVTRVJUCnVzCDBSU0EgQ2VydGlnaW9uWm91dGlnaW9uWm91dGlnaW9u\n"
"QTCASINzQxMjQxMjQxMjQxMjQxMjQxMjQxMjQxMjQxMjQxMjQxMjQxMjQx\n"
"XUSNEFjg6KjSxKZDmMysE1N8VJG06thum41tuZM+j9INpunSuukND8qeso7JcC7V\n"
"tBgV91est-jakpTb0c5/MZNrun8XzmCBShJ0R61vSoNnV1Qs1L2zfVtefkQnI/tBPP\n"
"vN"
"iwcRR0MkYNguQmm/B1j8LsN10yZpln6uGxGNS1oytW61Fo88BZ321wD6Gzq0GTL\n"
"vN"
"qK0YMa0dYEX6kuQdQ80tFvZnjNBX3EH1gsZHLI2w8ZMA0/6fDqS1SAB8fZIHpT\n"
"vN"
"eIFken5FrahZv9JNYWL7K5dS0xJ8hZudPR9akVuDjZvjs3YncJowZaduNI+L7RyML\n"
"vN"
"ZzcAAEAAQCAW4ggfQMB8GA1UdIwQYMBAAFNF5vLqQK0PvIDh2JvAnfKyA2bL\n"
"vN"
"MB0GA1UdGQNB8QeBaN3j2yH41uH56a0hxxAAzn0DA0B8gNVHQ8BAF8EBAMCAYY\n"
"vN"
"EgYDVR0TAQH/BAGwBgEB/wIBADA8BgNVHSUEFjAUBgggrBgEFBQcDAQYIKwYBBQUH\n"
"Vn"
"AwIwGwYDVR0gBBQwEjAGBgRVHSAAMAgGBmeBDAECAjBQBgNVHR8ESTBqMEwG\n"
"vN"
"hnj9odHRwOi8vY3J3J3J3LnVzZXJ0cnVzdC5jb20vVWVNFULRydXN0U1NBQ2VydGlnaW9u\n"
"vN"
"dG1vbKf1dghvcm10eS5JcmwmdgYIKwYBBQUHAQEaJBoMD8GCGCsGAQUFBzAChJ\n"
"vN"
"dhRw018vY3J0LnVzZXJ0cnVzdC5jb20vVWVNFULRydXN0U1NBQ2VydGlnaW9u\n"
"vN"
"cnQwJQYIKwYBBQUHMAAGG0h0dHA6Ly9vY3NwLnVzZXJ0cnVzdC5jb20vVWVNFULRydXN0U1NBQ2VydGlnaW9u\n"
"vN"
"hnvNAQEMBQAGGIBAG0R8jJW29dYak+gQGCXjeIT16MUJNKGe+vrK5/FTZctyM\n"
"vN"
"11Z1Up5uH5gIJppIG8GLWZqjV5vvhvZQfWZsHURKSISNrq0cooGT1e3jVgU0W+0\n"
"vN"
"+Wj8mN2kncVANT69F2YrA394gbGadJ5f0rQmL2pIHdY0jaco74fzYefbZ/V529FR\n"
"vN"
"5jBxu4uj1P+5ZImem4GbJ1e4ZeZvBhm055GFfBjR1dJ26h1ofBHZ7heDH1Bjzw72\n"
"vN"
"htpu47Gkyfr2NEx3KocGMlCj38tx7ASn5318FoU+hCazw0U1VX55mKPU1I2Z50Lo\n"
"vN"
"RCASN18Jyfr5D5PVLJbtyrmz9gn/TKbRATr80Uzq3JhyvJhLf410Jo/UzL5WCXED\n"
"vN"
"SmYj4jWG3R7Z8TED9xNMCxGBMxNmete+3FvzhssvbORDwBZByogQ9xL2L2UZFI/L\n"
"vN"
"eoQp0UM/L8zfP527/WjEzuDN5wxMhhi+vcTtoh7J159o5ah29mPa+JnvuJbXEnGa\n"
"vN"
"nrNhxZzu+AG0ePV8hwrGGG7hOICPDQWkuYwZn/xT291Lp/cqf9ZEtKGCc1ImH3b\n"
"vN"
"oJ8LfsCnSbu0GB9L06Yqh7LcyvKDTeads1IaeSEINxh02Y1fmcYFX/Fqrrp1WnhHv\n"
"vN"

Done uploading.
/Users/virameherkar/Downloads/WiFi_connect/WiFi_connect.ino
/Users/virameherkar/Downloads/WiFi_connect/WiFi_connect.ino
/Users/virameherkar/Downloads/sketch_oct31a.ino

43 Heltec_WiFi_Kit_32, 80MHz, 921600 on /dev/cu.SLAB_USBtoUART

```

- Create a free AWS account, from the AWS console go to IoT Core and then on the left side bar click on Manage.



Root user sign in ⓘ

Email: 20394meherkarvira@gmail.com

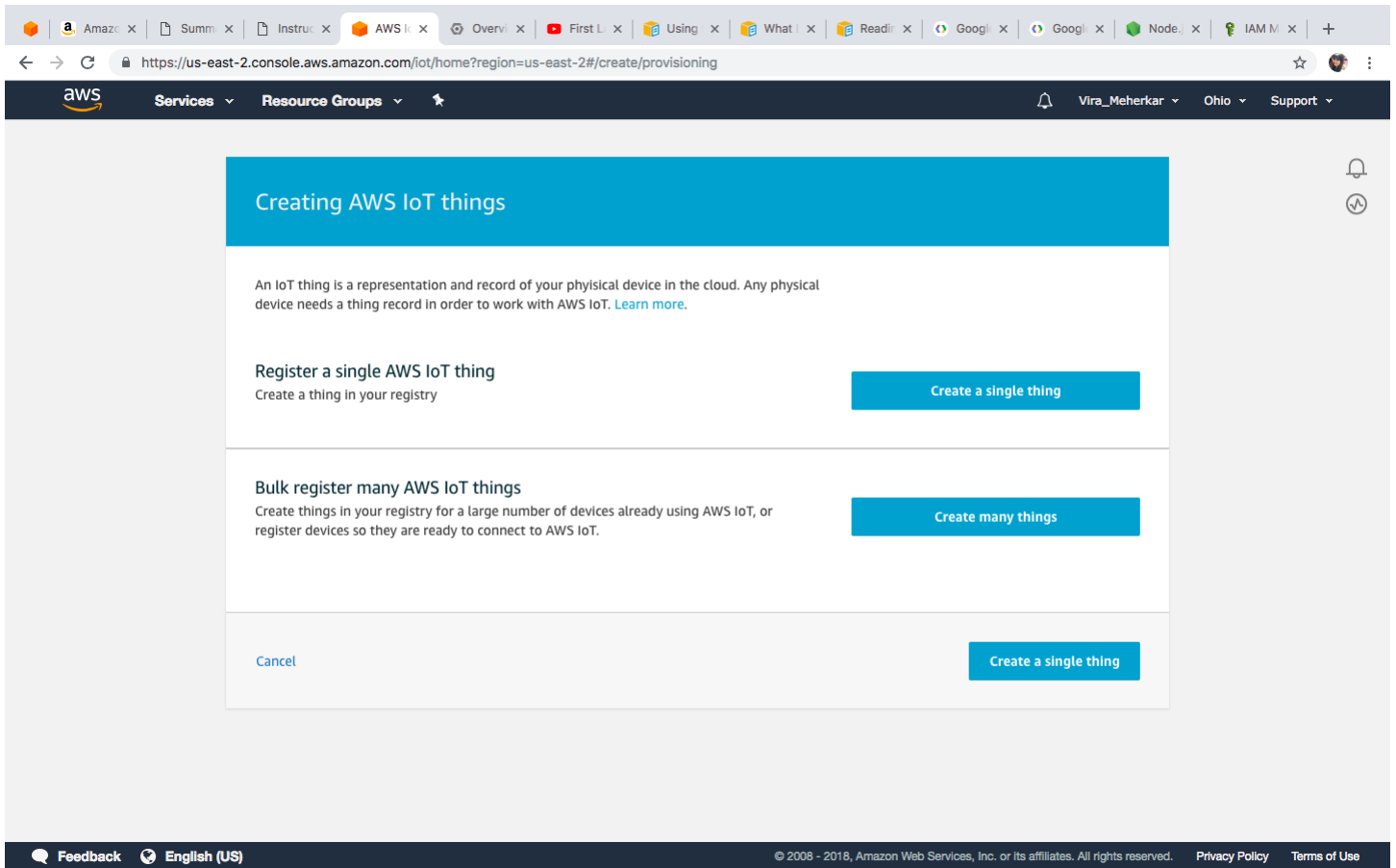
Password

[Forgot password?](#)

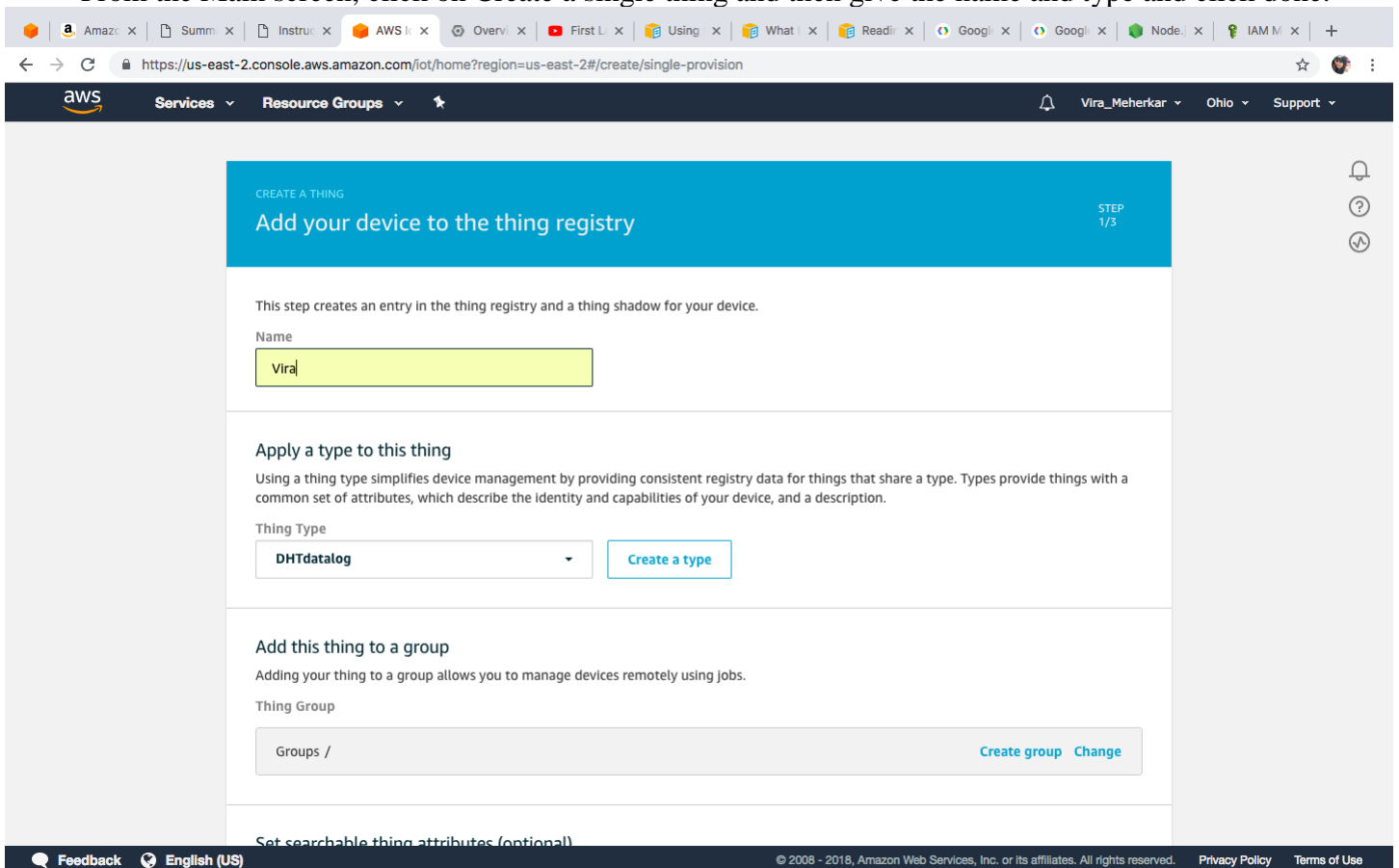
Sign in

[Sign in to a different account](#)

[Create a new AWS account](#)



- From the Main screen, click on Create a single thing and then give the name and type and click done.



Browser tabs: Amazon, Summ, Instru, AWS Io, Overvi, First L, Using, What, Read, Googl, Googl, Node, IAM M, +

URL: <https://us-east-2.console.aws.amazon.com/iot/home?region=us-east-2#/certificatehub>

Navigation: Services, Resource Groups, Vira_Meherkar, Ohio, Support

Create a certificate

A certificate is used to authenticate your device's connection to AWS IoT.

One-click certificate creation (recommended)
This will generate a certificate, public key, and private key using AWS IoT's certificate authority.

Create certificate

Create with CSR
Upload your own certificate signing request (CSR) based on a private key you own.

Create with CSR

Use my certificate
Register your CA certificate and use your own certificates for one or many devices.

Get started

Feedback English (US) © 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

- Once the thing is created click on it and from there side bar click on security and then create certificate.

Browser tabs: Amazon, Summ, Instru, AWS Io, Overvi, First L, Using, What, Read, Googl, Googl, Node, IAM M, +

URL: <https://us-east-2.console.aws.amazon.com/iot/home?region=us-east-2#/certificatehub>

Navigation: Services, Resource Groups, Vira_Meherkar, Ohio, Support

Certificate created!

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	ded137212f.cert.pem	Download
A public key	ded137212f.public.key	Download
A private key	ded137212f.private.key	Download

You also need to download a root CA for AWS IoT:
A root CA for AWS IoT [Download](#)

Deactivate

Cancel Done Attach a policy

Feedback English (US) © 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Browser tabs: Amazon, Summ, Instru, AWS, Overvi, First L, Using, What, Readir, Googl, Googl, Node, IAM M, +

URL: <https://us-east-2.console.aws.amazon.com/iot/home?region=us-east-2#/create/policy>

Create a policy

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the [AWS IoT Policies documentation page](#).

Name
Virapolicy

Add statements

Policy statements define the types of actions that can be performed by a resource. **Advanced mode**

Action
DHTdataLog

Resource ARN
arn:aws:iot:us-east-2:421334515349:topic/replaceWithATopic

Effect
☒ Allow ☐ Deny

Remove

Add statement

Feedback English (US) © 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

- After the creation of certificate, Create new policy attach a policy to the certificate.

Browser tabs: Amazon, Summ, Instru, AWS, Overvi, First L, Using, What, Readir, Googl, Googl, Node, IAM M, +

URL: <https://us-east-2.console.aws.amazon.com/iot/home?region=us-east-2#/policy/Virapolicy>

Policies > Virapolicy

POLICY

Virapolicy

Actions ▾

Overview

Certificates

Versions

Groups

Non-compliance

Certificates

7cb382fbdded3357d... ***

Feedback English (US) © 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

- Now from dynamo DB create a table to insert data from the IOT thing.

Create DynamoDB table Tutorial ?

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name* ⓘ

Primary key* Partition key

ⓘ

☒ Add sort key

ⓘ

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

☒ Use default settings

- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".

ⓘ You do not have the required role to enable Auto Scaling by default.
Please refer to [documentation](#).

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

Cancel Create

[Feedback](#) [English \(US\)](#) © 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

- From the Side Bar click on ACT and then create rule for the insertion action to select the data from the IOT thing and insert in the specific table.

Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name

Data_Insert

Description

enter humidity and temperature

Rule query statement

Indicate the source of the messages you want to process with this rule.

Using SQL version

2016-03-23



You don't have any rules yet

Rules give your things the ability to interact with AWS and other web services. Rules are analyzed and actions are performed based on the messages sent by your things.

[Learn more](#)

[Create a rule](#)

Resource Groups

Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*.required)

Add action

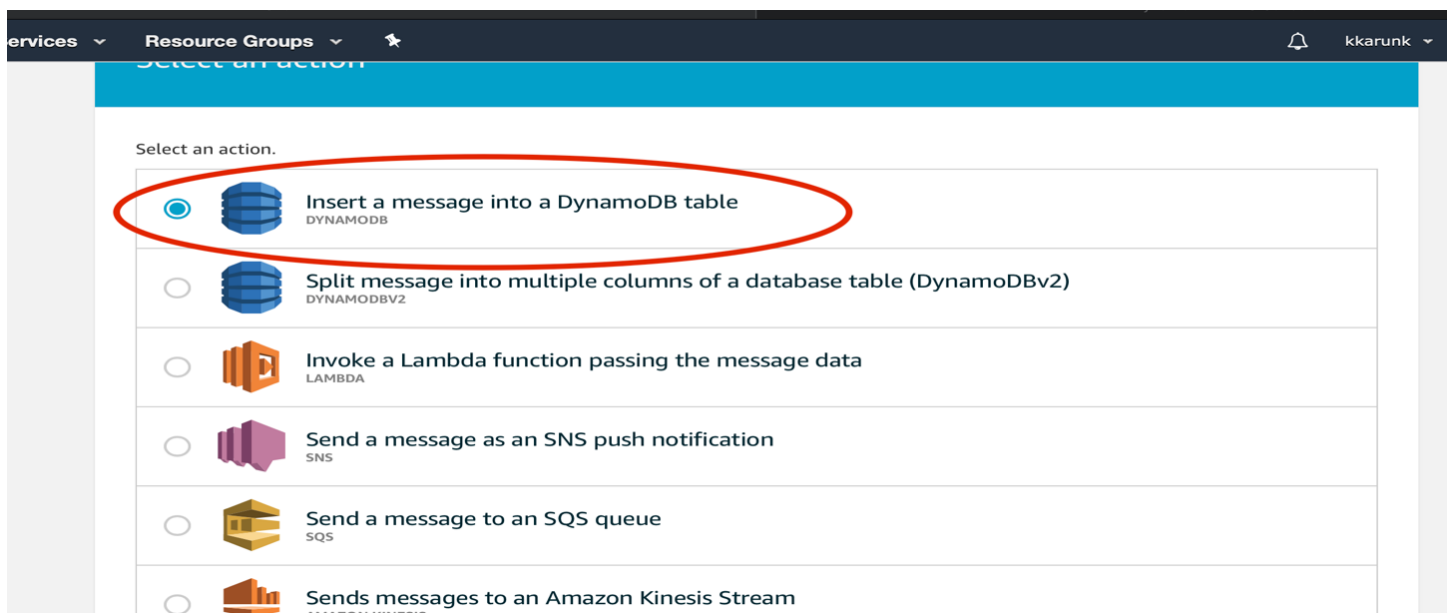
Error action

Optionally set an action that will be executed when something goes wrong with processing your rule.

Add action

Cancel

Create rule



- Select the table name in you dynamo DB and the IOT Thing name for the insertion operation.

Configure action

Insert a message into a DynamoDB table
DYNAMODB

The table must contain Hash and Range keys.

*Table name
Choose a resource

Create a new resource

*Hash key
Required field does not exist

*Hash key type
Required field does not exist

*Hash key value

Range key
Optional field does not exist

Range key type
Optional field does not exist

Range key value

Write message data to this column

Operation

Choose or create a role to grant AWS IoT access to perform this action.

*IAM role name
Choose a role

Cancel Add action

Values from IOT
THING

DHTDataInsert
ENABLED

Overview

Description
insert data into db

Cancel Update

Rule query statement
Using SQL version
2016-03-23

Rule query statement
To learn more about the SQL statement, see AWS IoT SQL Reference.

1 SELECT * FROM 'IOT_THING'

Text

Cancel Update

Actions
Actions that happen when a rule is triggered. [Learn more](#)

Insert a message into a DynamoDB table
IOTDataLog

Remove Edit

Add action

Error action
Optionally set an action that will be executed when something goes wrong with processing your rule.

Add action

Inserted dynamo DB with the table

- In the ESP32 code, we use the AWS IOT library to connect to the AWS, which uses the HTTPS AWS

THING

Vira

NO TYPE

Actions

Details

Security

Groups

Shadow

Interact

Activity

Jobs

Violations

Thing ARN

Edit

A thing Amazon Resource Name uniquely identifies this thing.

arn:aws:iot:us-east-2:421334515349:thing/Vira

Type

No type

endpoint to connect to the IOT

- For the Publish and Subscribe it uses the MQTT shadow updates, also we should JSON format to ensure

The screenshot displays the AWS IoT console interface. The left-hand navigation pane lists various management actions: Monitor, Onboard, Manage, Secure, Defend, Act, Test (which is currently selected), Software, Settings, and Learn. The main panel is titled 'Subscriptions' and is divided into two primary functional areas. The upper area, 'Subscribe to a topic', provides configuration for receiving MQTT messages. It features a text input for the 'Subscription topic' containing the text 'Vira', a 'Max message capture' input set to '100', and 'Quality of Service' (QoS) settings. The QoS section shows two options: '0 - This client will not acknowledge to the Device Gateway that messages are received' (which is selected) and '1 - This client will acknowledge to the Device Gateway that messages are received'. Below this, the 'MQTT payload display' section offers three choices: 'Auto-format JSON payloads (improves readability)' (selected), 'Display payloads as strings (more accurate)', and 'Display raw payloads (in hexadecimal)'. The lower area, 'Publish', is for sending messages. It includes a label 'Specify a topic and a message to publish with a QoS of 0.' and a text input field containing 'Specify a topic to publish to, e.g. myTopic/1'. A 'Publish to topic' button is positioned to the right of this input. The bottom of the console features a 'Feedback' button, a language dropdown set to 'English (US)', and a footer with copyright information and links to 'Privacy Policy' and 'Terms of Use'.

the data read and insertion in the Dynamo DB table, that we have created.

- Also we need to update the certificates and key in a 'C' file to ensure connection to the IOT.

```
aws_iot_certificates.c

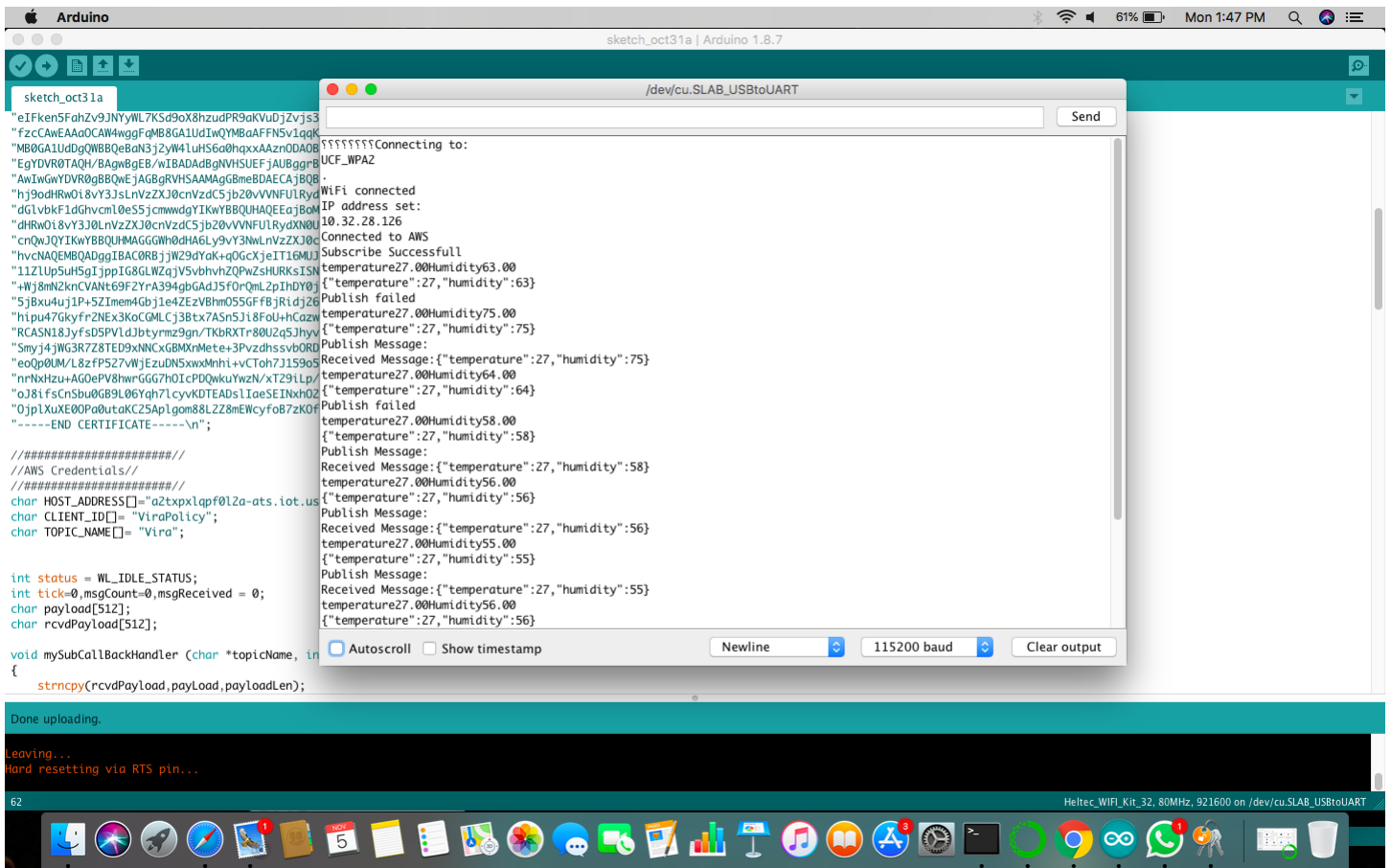
const char aws_root_ca_pem[] = {"-----BEGIN CERTIFICATE-----\n
MIIDOTCCAimgAwIBAgITBmyfz5m/iAo54yB4ikPmliZbyiANBqkqhkiG9w0BAQsF\n
ADA5MQswCQYDVQ0GEwJUVzEPMA0GA1UEChMGMGQW1hem9uMRkwFwYDVQ0DEwBB\n
b24qUm9vdCB0QSAxMB4XDTE1MDUyNjE1MDUyNjE1MDUyNjE1MDUyNjE1MDUyNjE1\n
MAkGA1UEBhMCMVVMxOzANBgNVBAoTBkFtYXpvcjEzMDEGA1UEAxMOMGQW1hem9u\n
b3QqOGEtMCCASiW0QYJkZiHvcNAOE8B8QDggEPAPCAQoCggEBAJ4gHHKcNj\n
ca9HgFB0fW7Y14h29Jl091ghYPl0hAEvRAItht0g03p0sqTQNR0Bvo3b5MgHFzZM\n
906IIBc+6zf1tRn4SWiw3te5diqdY26k/oI2peVKuRF4fn9t8b6dNqcmzU5L/qw\n
IFAGbHrGgKLn+a/sRxpPUdGh3KH0Vi4utWp+UhnM1bulHheb4m1UcAwhmahRwa6\n
V0Ujw5HSNz/0eawLX0tdHA114q957EwW67c4cX8jJGKLbD+rcdqsq08p8kdi1L\n
93FcXmV/6pUCyziKrlA4b9v7LWibxcceV0F34GfID5yHI9Y/OCB/IIDEGw+0y0m\n
JgSubJrIgg0CAwEAANCMCAwDwYDVR0TAQH/BAUwAwEB/zA0BgNVH08BA8E8BAMC\n
AYYwHQYDVRR0BBYEFiQYzIU07LwMlJ0uCFmcx7I0TaoIMA0GCSqGSIb3D0E8CwUA\n
A4IBAQC8Y8daQZChG5V2U5qgNiM0ruY0u6r4LK5IpD8/G/wkijUu0yKX9rbxenD\n
U5PMCCijmCXPI6T53iHTfIUJrU6adTrCC2qJhZEXRhlbi18iit/msv0tadQ1wUs\n
N+qD563pYaCbvXy8MWy7Vu33PaUXHeeE6V/Ug2V8yiT096LXfYKw1JbYK8U90v\n
o/ufQJVMVT80tPHR8BjrdkPSHCa2XV4cdFyQzR1bldZwgJcJmApzyMZFO6I06XU\n
5MsI+YMR0+hDKXJioaldXgUkK642M4UwtBV8ob2xJND02ZhwLno0deXeGADbkpy\n
rRqRfboQnoZsG45WTP46850vxyG5\n
-----END CERTIFICATE-----\n"};

const char certificate_pem_crt[] = {"-----BEGIN CERTIFICATE-----\n
MIIDwCCAKGAwIBAgIWA03E0r6Ma3X531YLzwDrTFYLoADMA0GCSqGSIb3D0E8Cw\n
CwUAME0xSzBjBGNVBA8M0kFtYXpvcjEzMDEGA1UEChMGMGQW1hem9uMRkwFwYDVQ0DEwBB\n
b24qUm9vdCB0QSAxMB4XDTE1MDUyNjE1MDUyNjE1MDUyNjE1MDUyNjE1MDUyNjE1\n
MAkGA1UEBhMCMVVMxOzANBgNVBAoTBkFtYXpvcjEzMDEGA1UEAxMOMGQW1hem9u\n
b3QqOGEtMCCASiW0QYJkZiHvcNAOE8B8QDggEPAPCAQoCggEBAJ4gHHKcNj\n
ca9HgFB0fW7Y14h29Jl091ghYPl0hAEvRAItht0g03p0sqTQNR0Bvo3b5MgHFzZM\n
906IIBc+6zf1tRn4SWiw3te5diqdY26k/oI2peVKuRF4fn9t8b6dNqcmzU5L/qw\n
IFAGbHrGgKLn+a/sRxpPUdGh3KH0Vi4utWp+UhnM1bulHheb4m1UcAwhmahRwa6\n
V0Ujw5HSNz/0eawLX0tdHA114q957EwW67c4cX8jJGKLbD+rcdqsq08p8kdi1L\n
93FcXmV/6pUCyziKrlA4b9v7LWibxcceV0F34GfID5yHI9Y/OCB/IIDEGw+0y0m\n
JgSubJrIgg0CAwEAANCMCAwDwYDVR0TAQH/BAUwAwEB/zA0BgNVH08BA8E8BAMC\n
AYYwHQYDVRR0BBYEFiQYzIU07LwMlJ0uCFmcx7I0TaoIMA0GCSqGSIb3D0E8CwUA\n
A4IBAQC8Y8daQZChG5V2U5qgNiM0ruY0u6r4LK5IpD8/G/wkijUu0yKX9rbxenD\n
U5PMCCijmCXPI6T53iHTfIUJrU6adTrCC2qJhZEXRhlbi18iit/msv0tadQ1wUs\n
N+qD563pYaCbvXy8MWy7Vu33PaUXHeeE6V/Ug2V8yiT096LXfYKw1JbYK8U90v\n
o/ufQJVMVT80tPHR8BjrdkPSHCa2XV4cdFyQzR1bldZwgJcJmApzyMZFO6I06XU\n
5MsI+YMR0+hDKXJioaldXgUkK642M4UwtBV8ob2xJND02ZhwLno0deXeGADbkpy\n
rRqRfboQnoZsG45WTP46850vxyG5\n
-----END CERTIFICATE-----\n"};

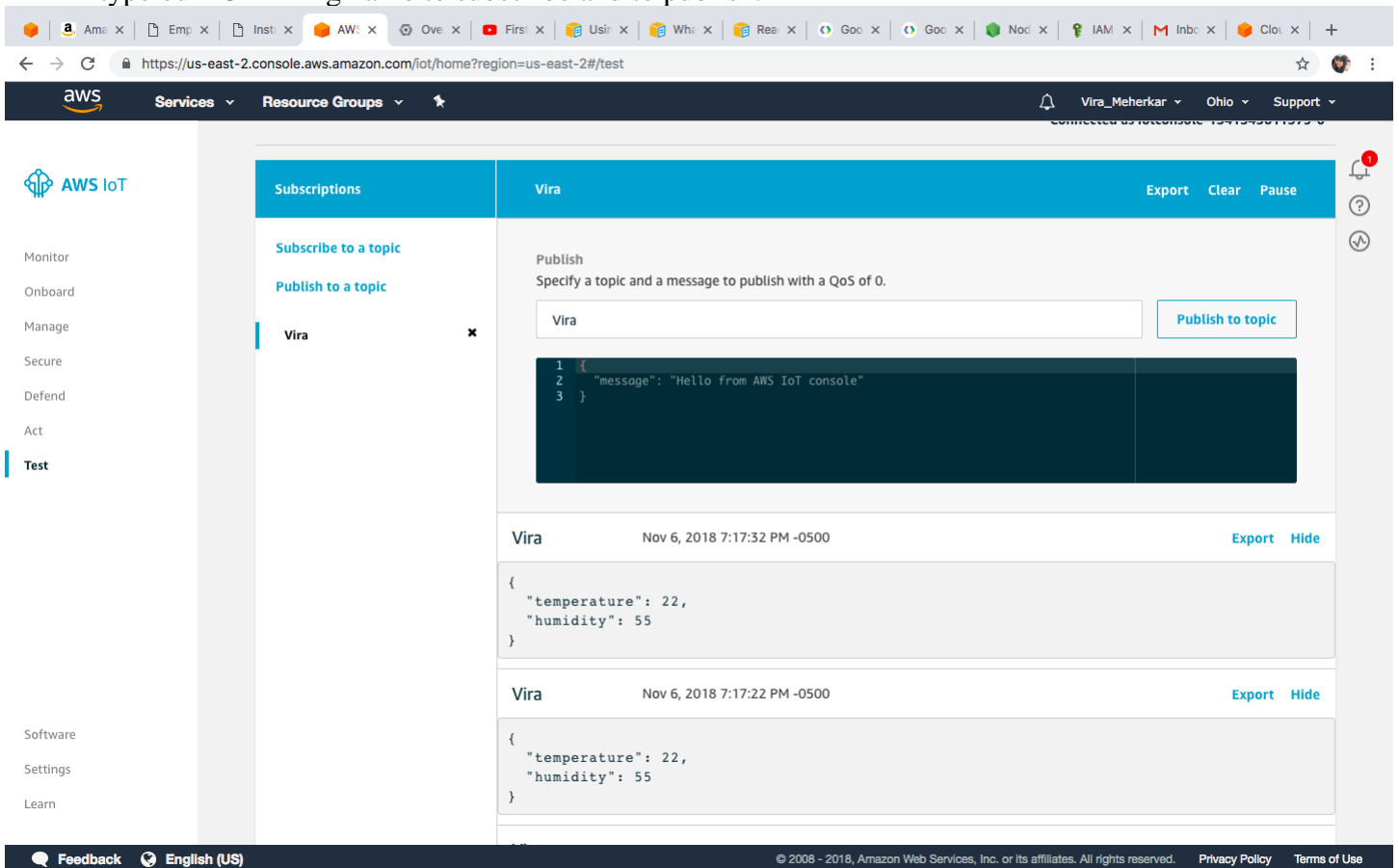
const char private_pem_key[] = {"-----BEGIN RSA PRIVATE KEY-----\n
MIIEpAIBAAKCAQEA76QyAtwKrbUdBYIr/SkGiA5AshEitaMM1GfUJMcbdlKX1S6\n
1qLP3ExE06jH6X8vU+VqTv400yTbSvel1yV+Lh1E6dh21866fDFg+S7pRoe6uA\n
6t2/yKNfd/h20dTziS7u6tkHwtNQiriWUUVxv0TC15ie0amMo8MQ+SAJLSQIKcG0\n
Eeq4mBH6iABhmVTyEwMVCM39rxuqA0z0fe06a7UNT2F0rzUjJN41qLnJNsdn62L\n
BXASnWHphU3Z7/InqL70TqY9M1U1qxio620u0MhapZJCARJF2XF03iY+GumYce7W\n
6ZHCqf3Jra6rGEZ0bk0eiYrKC0Dx0ULbvErwIDAQABAAIBAEbJnBdMeeY4ir8R\n
FmbFqANSAWHZITVU74ruwRu1mtHocP820pZb56Q1TITqSAL+XsCBf+I36w+rQ\n
UR3EBE9f9HE6bMuxZma6cRh0PehEr8U7bXkfoq57ASXs11KuIMkuHIYlW03du\n
brSe/ZKIL/vcfSPUZDDyV7LLP7fYcYPSbenmjeCKFhLO0U058rkVJA/vkRcGhmZ\n
wXYxn8b2D9q9Bii/pM7g0ScCSbNItmiv50opL3D5STdy63pAd18ruoV1REzSXWBS\n
2weeLVxtkYUCs10ag8h/n10DNF19wVx+02Ffhl+xRbMo+vsVd7cpAzB0YvaWC1\n
NvBXNIECqYEA4x1qsadEEYg+/0/aCnfpIam6soduHzPbsDHxwt09VvdQK1l9AYa\n
Z0h42JWfVbDpk8ZS3RqXiFVP//DnFdwpiwps0Qq8CKHJH17KoUDNixM7YC/LqT\n
WhwPd+wrL3gSuRVh+V6as3EL7Z7kfr3DIXiJGMCA3JVAz0VJci80BkCqYEAyprS\n
qtNq8hhey76IIAUTDUMaaV0+8dvsMdRt9LYL7EY9VBwt57cIOEJFC56QtfUSHCZ\n
WpPfx7TyCkpdM2herx+wbhVeL1P3EEDWD9RtmBaQ0DA/MbFc+LNHP/QVDz5Uk7/\n
6IX90bo/Qaj41I3xHvRi0MfCqNHLH48veE5VNAcCqYBX9Embd9S0g0swqW06wW2s\n
10N1H5SCk71U0csg60MXvE0Ds1G1E8TnTv5t0r1T7b4Z7wP11ahdFw67X\n
-----END RSA PRIVATE KEY-----\n"};
```

```
#####
//AWS Credentials//
#####
char HOST_ADDRESS[]="a2txp1qpf0l2a-ats.iot.us-east-2.amazonaws.com";
char CLIENT_ID[]="ViraPolicy";
char TOPIC_NAME[]="Vira";
```


- To test the operation of publish and subscribe, we can go to Test section in the Side Bar and We can



type our IOT Thing name to subscribe and to publish.



- From the above screen shot we can see that the Values are same in Dynamo DB as the Value from the ESP32 DHT sensor read.
- Once we can see the data in the PubSub Test page, we head to the dynamo DB Table and From the Item

The screenshot shows the AWS DynamoDB console interface. The left sidebar contains the navigation menu. The main content area displays the 'Items' tab for the 'DHT11' table. The table has a primary key 'Temperature' and a secondary key 'Humidity'. The 'Items' tab shows a list of 9 items, each with a 'payload' field containing a JSON object representing sensor data. The bottom of the console shows the footer with 'Feedback', 'English (US)', and copyright information.

Temperature	Humidity	payload
26	52	{ "humidity": { "N": "52" }, "temperature": { "N": "26" } }
26	53	{ "humidity": { "N": "53" }, "temperature": { "N": "26" } }
26	54	{ "humidity": { "N": "54" }, "temperature": { "N": "26" } }
27	50	{ "humidity": { "N": "50" }, "temperature": { "N": "27" } }
27	51	{ "humidity": { "N": "51" }, "temperature": { "N": "27" } }
27	52	{ "humidity": { "N": "52" }, "temperature": { "N": "27" } }
27	53	{ "humidity": { "N": "53" }, "temperature": { "N": "27" } }
27	54	{ "humidity": { "N": "54" }, "temperature": { "N": "27" } }
27	55	{ "humidity": { "N": "55" }, "temperature": { "N": "27" } }

Tab we can see the data there on the table.

Q.5:- For processing real time data

Download and install required libraries for GPS to work in Arduino IDE

Connect Arduino/ESP32 with GPS sensor and store the latitude longitude along like we stored temperature and humidity in aws in DynamoDB table.

Write the code for the same and upload it to Arduino? ESP32 board and let it run.

You can build a variety of real-time data processing systems using AWS Lambda, Amazon Kinesis, Amazon S3, and Amazon DynamoDB. We will create a web page where we will use Google Map API key generated to embed the webpage using coordinates from DynamoDB table. And we will be able to show the data in real time then on Google Maps.

ESP32 CODE

```
#include <AWS_IOT.h>
#include <WiFi.h>
#include <ArduinoJson.h>
#include "esp_wpa2.h"
AWS_IOT hornbill;

#include "DHTesp.h"
#include "Ticker.h"
DHTesp dht;

//#####//
//UCF login credentials//
//#####//
const char* ssid ="UCF_WPA2";
const char* host = "arduino.php5.sk";
#define EAP_IDENTITY "virameherkar@knights.ucf.edu"
#define EAP_PASSWORD "Alpa20394#"
#define EAP_USERNAME "vi975487"

//#####//
// UCF SERVER CERTIFICATE //
//#####//
const char* UCF_server_ca_cert = "-----BEGIN CERTIFICATE-----\n"
"MIIF+TCCA+GgAwIBAgIQRYdQ+oVGGn4XoWQCKYRjdDANBgkqhkiG9w0BAQwFADCB\n"
"iDELMAkGA1UEBhMCVVMxEzARBgNVBAgTCk5ldyBKZXJzZXkxFDASBgNVBAcTC0pl\n"
"cnNleSBDaXR5MR4wHAYDVQQKEzVUaGUgVGVNFUIRSVVNUIE5ldHdvcmxLjAsBgNV\n"
"BAMTJVVTRVJUcnVzdCBSU0EgQ2VydGlmaWNhdGlubiBBdXR0b3JpdHkwHhcNMjQx\n"
"MDA2MDAwMDAwWHcnMjQxMDA1MjM1OTU5WjB2MQswCQYDVQQGEwJVUzELMAkGA1UE\n"
"CBMCTUkxEjAQBgNVBAcTCUFubiBBcmJvcjESMBAGA1UEChMJSW50ZXJuZXQyMREw\n"
"DwYDVQQLEwhJbkNvbWw1bWVjEjEgMB0GA1UEAxMWSW50ZXJuZXQyMREwYDQw\n"
"QTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAlw8bsvf2MYFVFRVA+e\n"
"xU5NEFj6MJsXKZDmMwysE1N8VJG06thum4ltuzM+j9INpun5uukNDBqeso7JcC7v\n"
"HgV9lestjaKpTbOc5/MZNrun8XzmCB5hJ0R6lvSoNNviQsil2zfVtefkQnl/tBPP\n"
"iwckRR6MkYNGuQmm/BijBgLsNI0yZpUn6uGX6Ns1oytW61fo8BBZ321wDGZq0GTI\n"
"qKOYMa0dYtX6kuOaQ80tNfvZnjNbRX3EhigsZhLI2w8ZMA0/6fDqSl5AB8f2IHpT\n"
"elFken5FahZv9JNYyWL7KSd9oX8hzudPR9aKVuDjZvjs3YncJowZaDuNi+L7RyML\n"
"fzcCAwEAaAOCaW4wggFqMB8GA1UdIwQYMBaAFFN5v1qqK0rPVIDh2JvAnfKyA2bL\n"
"MB0GA1UdDgQWBBQeBaN3j2yW4luHS6a0hqxxAAznODAOBgNVHQ8BAf8EBAMCAYYw\n"
"EgYDVROTAQH/BAGwBgEB/wIBADAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUH\n"
"AwIwGwYDVROgBBQwEjAGBgRVHSAAMAgGBmeBDAECAjBQBgNVHR8ESTBHMEWgQ6BB\n"
"hj9odHRwOi8vY3J0LnVzZXJ0cnVzdC5jb20vVVNFUIRydXN0UINBQ2VydGlmaWNh\n"
"dGlvbKf1dGhvcml0eS5jcmwwdG9yYkYBBQUHAQEeAjBoMD8GCCsGAQUFBzACHjNo\n"
"dHRwOi8vY3J0LnVzZXJ0cnVzdC5jb20vVVNFUIRydXN0UINBQWRkVHJ1c3RDQS5j\n"
"cnQwJQYIKwYBBQUHMAGGGWh0dHA6Ly9vY3NwLnVzZXJ0cnVzdC5jb20wDQYJKoZI\n"
```

```
"hvcNAQEMBQADggIBAC0RBjjW29dYaK+qOGcXjeIT16MUJNkGE+vrkS/ft2ctyNMU\n"
"11ZlUp5uH5gljppIG8GLWZqjV5vbhvhZQPwZsHURKsISNrQOcooGTie3jVgU0W+0\n"
"+Wj8mN2knCVANt69F2YrA394gbGAdJ5fOrQmL2plhDY0jqco74fzYefbZ/VS29fR\n"
"5jBxu4uj1P+5Zlmem4Gb1e4ZEzVBhmO55GFfBjRidj26h1oFBHZ7heDH1Bjzw72\n"
"hipu47Gkyfr2NEx3KoCGMLCj3Btx7ASn5Ji8FoU+hCazwOU1VX55mKPU1I2250Lo\n"
"RCASN18JyfsD5PVldJbtyrmz9gn/TKbRXTr80U2q5JhyvjhLf4lOJo/UzL5WCXED\n"
"Smyj4jWG3R7Z8TED9xNNCxGBMXnMete+3PvzdhssvbORDwBZByogQ9xL2LUZFI/i\n"
"eoQp0UM/L8zfP527vWjEzuDN5xwxMnhi+vCToh7J159o5ah29mP+aJnvujbXEnGa\n"
"nrNxHzu+AGOePV8hwrGGG7hOlcPDQwkuYwzN/xT29iLp/cqf9ZhEtkGcQclmH3b\n"
"oJ8ifsCnSbu0GB9L06Yqh7lcyvKDTEADsllaeSEINxhO2Y1fmcYFX/Fqrrp1WnhH\n"
"OjplXuXE0OPaOutaKC25Aplgom88L2Z8mEWcyfoB7zKOfD759AN7JKZWCYwk\n"
"-----END CERTIFICATE-----\n";
```

```
//#####//
```

```
//AWS Credentials//
```

```
//#####//
```

```
char HOST_ADDRESS[]="a2txpxlqpf0l2a-ats.iot.us-east-2.amazonaws.com";
```

```
char CLIENT_ID[]="ViraPolicy";
```

```
char TOPIC_NAME[]="Vira";
```

```
int status = WL_IDLE_STATUS;
```

```
int tick=0,msgCount=0,msgReceived = 0;
```

```
char payload[512];
```

```
char rcvdPayload[512];
```

```
void mySubCallbackHandler (char *topicName, int payloadLen, char *payload)
```

```
{
    strncpy(rcvdPayload,payload,payloadLen);
    rcvdPayload[payloadLen] = 0;
    msgReceived = 1;
}
```

```
void setup()
```

```
{
    Serial.begin(115200);
    int dhtPin = 17;
    dht.setup(dhtPin, DHTesp::DHT11);
    Serial.println();
}
```

```
//#####//
```

```
// WIFI CONNECTION STARTS HERE //
```

```
//#####//
```

```
byte error = 0;
```

```
Serial.begin(115200);
```

```

delay(10);
Serial.println("Connecting to: ");
Serial.println(ssid);
WiFi.disconnect(true); //disconnect from wifi to set new wifi connection
WiFi.mode(WIFI_STA);
error += esp_wifi_sta_wpa2_ent_set_ca_cert((const unsigned char*)UCF_server_ca_cert,
strlen(UCF_server_ca_cert));
error += esp_wifi_sta_wpa2_ent_set_identity((uint8_t *)EAP_USERNAME, strlen(EAP_USERNAME));
error += esp_wifi_sta_wpa2_ent_set_username((uint8_t *)EAP_USERNAME, strlen(EAP_USERNAME));
error += esp_wifi_sta_wpa2_ent_set_password((uint8_t *)EAP_PASSWORD, strlen(EAP_PASSWORD));
if (error != 0)
{
    Serial.println("Error setting WPA properties.");
}
WiFi.enableSTA(true);
esp_wpa2_config_t config = WPA2_CONFIG_INIT_DEFAULT();
if (esp_wifi_sta_wpa2_ent_enable(&config) != ESP_OK)
{
    Serial.println("WPA2 Settings Not OK");
}
WiFi.begin(ssid); //connect to Eduroam function
WiFi.setHostname("RandomHostname"); //set Hostname for your device - not necessary
while (WiFi.status() != WL_CONNECTED)
{
    delay(5000);
    Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address set: ");
Serial.println(WiFi.localIP()); //print LAN IP

//#####//
// WIFI CONNECTION ENDS HERE //
//#####//

//#####//
// AWS CONNECTION STARTS HERE //
//#####//

if (hornbill.connect(HOST_ADDRESS,CLIENT_ID)== 0)
{
    Serial.println("Connected to AWS");
    delay(1000);

    if(0==hornbill.subscribe(TOPIC_NAME,mySubCallBackHandler))
    {

```

```

        Serial.println("Subscribe Successfull");
    }
    else
    {
        Serial.println("Subscribe Failed, Check the Thing Name and Certificates");
        while(1);
    }
}
else
{
    Serial.println("AWS connection failed, Check the HOST Address");
    while(1);
}

delay(2000);

}
//#####//
// AWS CONNECTION ENDS HERE //
//#####//
void loop()
{

//#####//
// DHT DATA READ SATRTS HERE //
//#####//
    delay(1000);
    TempAndHumidity newValues = dht.getTempAndHumidity();

    float heatIndex = dht.computeHeatIndex(newValues.temperature, newValues.humidity);
    float dewPoint = dht.computeDewPoint(newValues.temperature, newValues.humidity);

//#####//
// DHT DATA READ ENDS HERE //
//#####//

//#####//
// AWS PUBLISH AND RECEIVE STARTS HERE //
//#####//
    if(msgReceived == 1)
    {
        msgReceived = 0;
        Serial.print("Received Message:");
        Serial.println(rcvdPayload);
    }
}

```



```
if(tick >= 5) // publish to topic every 5seconds
{
    tick=0;
    StaticJsonBuffer<300> JSONbuffer;
    JsonObject& JSONencoder = JSONbuffer.createObject();
    JSONencoder["temperature"] = newValues.temperature;
    JSONencoder["humidity"] = newValues.humidity;
    Serial.println("temperature" + String(newValues.temperature) + "Humidity" +
String(newValues.humidity));
```

```
char JSONmessageBuffer[100];
JSONencoder.printTo(JSONmessageBuffer, sizeof(JSONmessageBuffer));
Serial.println(JSONmessageBuffer);
```

```
if(hornbill.publish(TOPIC_NAME,JSONmessageBuffer) == 0)
{
    Serial.print("Publish Message:");
    Serial.println(payload);
}
else
{
    Serial.println("Publish failed");
}

}
vTaskDelay(1000 / portTICK_RATE_MS);
tick++;
}
```
