Course Project

CS5453: Internet of Things

Smart Heating, Ventilation and Air Conditioning(HVAC) System

भारतीय प्रौद्योगिकी संस्थान हैदराबाद

Indian Institute of Technology Hyderabad

Submitted by

Tahir Ahmed Shaik, CS20MTECH14007
Pratik Madhukar Lahase,CS20MTECH11003
Ashish Kishor, CS20MTECH11002

Date : 01/05/2021

Index

1. Introduction

1.1 The HVAC Controller System

The **HVAC** system is a collaborative system of devices whose main objective is to provide a uniform, thermal stability and ensure proper ventilation levels in an indoor or outdoor environment. The HVAC helps in stabilizing the temperatures through a constant system processing. This system regulates the thermal levels and maintains the equilibrium in the environment,

The HVAC systems are employed almost everywhere ranging from the indoor to outdoor environments such as residential complexes, apartments, vehicles , hotels, and hospitals. The Air Conditioning in the HVAC system is the most essential part of the system that controls the temperature regulation and also takes care of the humidity levels.

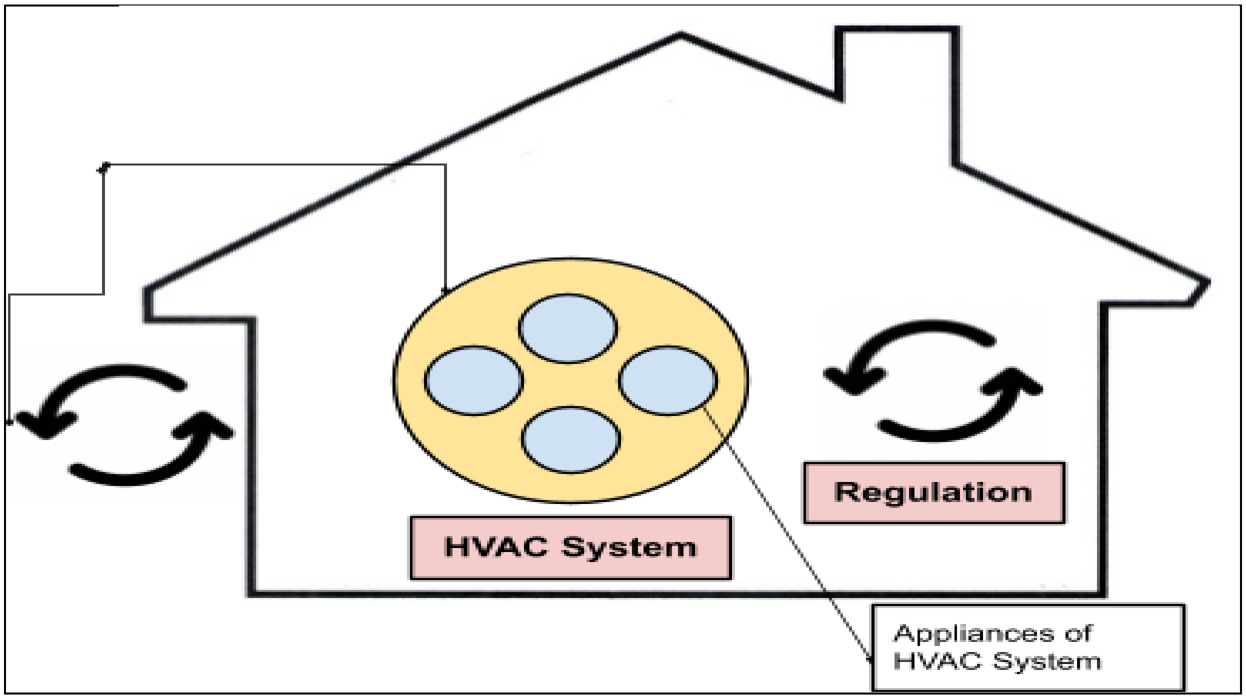


Fig 1 : Conceptual HVAC System

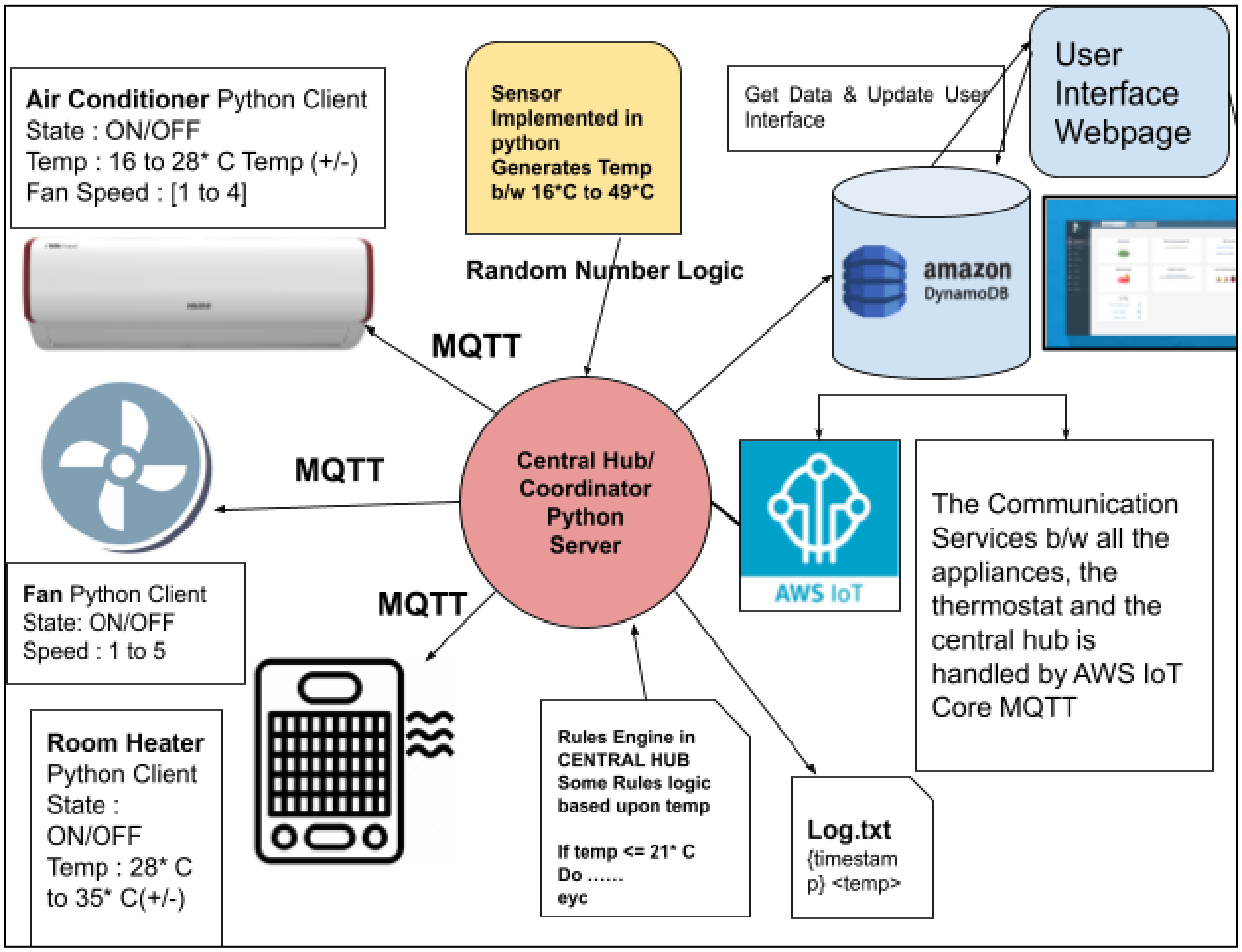
1.2 Core components of the HVAC System :

The core components of the general HVAC system include the following components:

- Thermostat
- Air Conditioner
- Air Handler (including Evaporator Coil and Blower)
- etc

2. Project Design

We implement the HVAC system by emulating the system in real time , we emulate the various devices and the thermostat sensor that senses the ambient temperature in the indoor environment. Our system's focus is to maintain the equilibrium thermal stability in the environment by regulating the temperatures using appliances such as the Air Conditioner, Fan and the Room Heater. All the appliances in our project are centrally co-ordinated by a higher level device called the ***"Central Hub"*** or the gateway that takes the decisions for temperature regulations and also establishes the communication among the devices. The proposed system architecture is shown below in fig 2.



Air Conditioner Python Client
State : ON/OFF
Temp : 16 to 28* C Temp (+/-)
Fan Speed : [1 to 4]
Sensor Implemented in python Generates Temp b/w 16*C to 49*C
Get Data & Update User Interface
User Interface Webpage
amazon DynamoDB
Random Number Logic
MQTT
MQTT
MQTT
Central Hub/ Coordinator Python Server
AWS IoT
The Communication Services b/w all the appliances, the thermostat and the central hub is handled by AWS IoT Core MQTT
Fan Python Client
State: ON/OFF
Speed : 1 to 5
Room Heater Python Client
State : ON/OFF
Temp : 28* C to 35* C(+/-)
Rules Engine in CENTRAL HUB
Some Rules logic based upon temp
If temp <= 21* C
Do ......
eyc
Log.txt
{timestamp} <temp>

Fig 2 : System Architecture

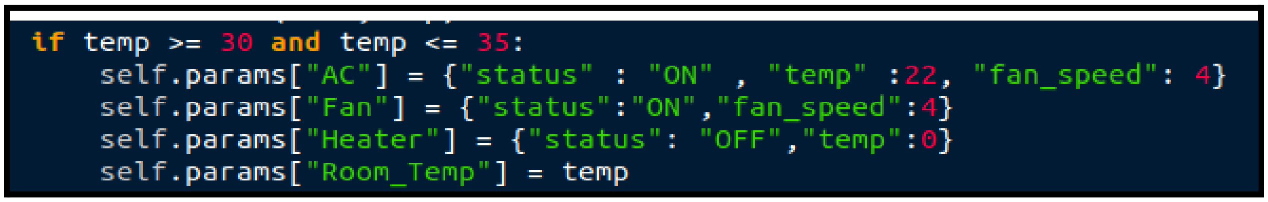
2.1 Working of the System

2.1.1 HVAC Appliances and the Central Hub logic

As illustrated above the entire project simulation happens through a series of processes as defined. The Central Hub/ Gateway device is a co-ordinator device that has high level logic, can process and communicate. The central hub in a communication scenario is a server that handles the communications and performs or processes the information. Each appliance (AC, Fan and

Room Heater) is an emulated python client which displays the series of information that is sent to them. The ***thermostat which has the temperature sensor*** is also a python client that based on a random number generator logic generates a temperature in a realtime temperature range (16* C to 49* C) to simulate the ambient temperature. This temperature is measured at an interval of every 5 minutes by the sensor. The temperature is then received by the central hub or the gateway device which has a rule engine running on it which handles the core decision making logic based on temperature, it sets the various parameters of the appliances based on the temperature such as (ac_temperature, ac_fanspeed, ceiling fan speed, room_heater temperature etc). All the communication between these clients happen using the MQTT communication standard provided by the ***AWS IoT core service.***

The ***Rule Engine*** is a specific class structure that runs on the central hub which has a predefined set of event driven logic decisions based on the ambient temperature as received from the temperature sensor. It sets the HVAC appliances parameters based on the ambient temperature. One of the rule engine temperature logic is as shown below in the following figure.

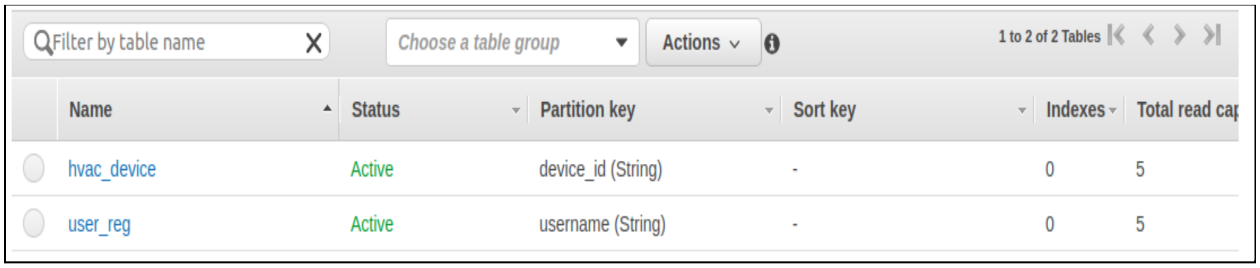


```
if temp >= 30 and temp <= 35:
    self.params["AC"] = {"status" : "ON" , "temp" :22, "fan_speed": 4}
    self.params["Fan"] = {"status":"ON","fan_speed":4}
    self.params["Heater"] = {"status": "OFF","temp":0}
    self.params["Room_Temp"] = temp
```

Fig 3: A Rule Engine Logic

2.1.2 Interface and Backend logic

The project also has a user interface, which is an interactive web-interface. The interface shows all the specific information including the HVAC system statistics, regulated average temperatures, live system logging information, status of appliances for monitoring. The back-end of this interface is handled by the NoSQL cloud database provided by the **AWS DynamoDB** service. The AWS DynamoDB is configured with tables for authentication of the user that is used to store the user level details for user registration and the authentication during logging up to the webpage, this table is called **"user_reg"**. The other database **"hvac_device"** is used by the interface to fetch and display the dashboard using the various system level information including the current status , parameters of the appliances etc at each interval.



Name	Status	Partition key	Sort key	Indexes	Total read cap
hvac_device	Active	device_id (String)	-	0	5
user_reg	Active	username (String)	-	0	5

Fig 4: Backend tables on DynamoDB

3. Program Design

As described in the above section, the project consists of a set of appliances or devices and a user Interface web-page. All the appliances including the AC(Air Conditioner) , Fan , Room Heater and the Central_Hub are designed in python , which use the MQTT communication for communicating the information. All the appliances interfaces are displayed using the Tkinter GUI python library. The temperature sensor works on the random number generator logic, generating the ambient temperature levels at every interval of 5 minutes. All these components use the AWS IoT Core for the communication exchange. Before they can be run, these devices are initialized on the AWS IoT core which provide a unique digital certificate to each component that is later used by each of them to connect to the AWS MQTT endpoint and communicate with each other. The AC, Fan and the Room Heater module is implemented by the code "Air_Conditioner.py", "Fan.py" and "Room_Heater.py" respectively. Each of these devices load the certificates provided by the AWS IoT core and the Amazon root CA certificate for verification while connecting to the AWS MQTT broker endpoint. The temperature sensor is realised in the "Temp_sensor.py" which uses the random number logic to simulate real time ambient temperatures in every time interval of 5 minutes. This temperature by the sensor is published on the topic ***"hvac/room/sensor"*** which is subscribed by the Central Hub device. The ***"Central Hub"*** is also conceptually another MQTT client. This is more sophisticated client that has a class called ***"Rule_Engine"*** that defines the event driven logic that based on the received temperature from the sensor sets the appliance parameters and the central hub publishes it to the topic ***"hvac/room/hub/params"*** which is subscribed by each appliance. Based on the parameters received by the central hub each device sets their parameters states accordingly. Also the Central-Hub pushes the same data to the AWS DynamoDB "hvac_device" table which is later used to render the dashboard on the interface. The Central Hub also logs the messages in the file called **"logs.txt"** in the same directory at each timestamp.

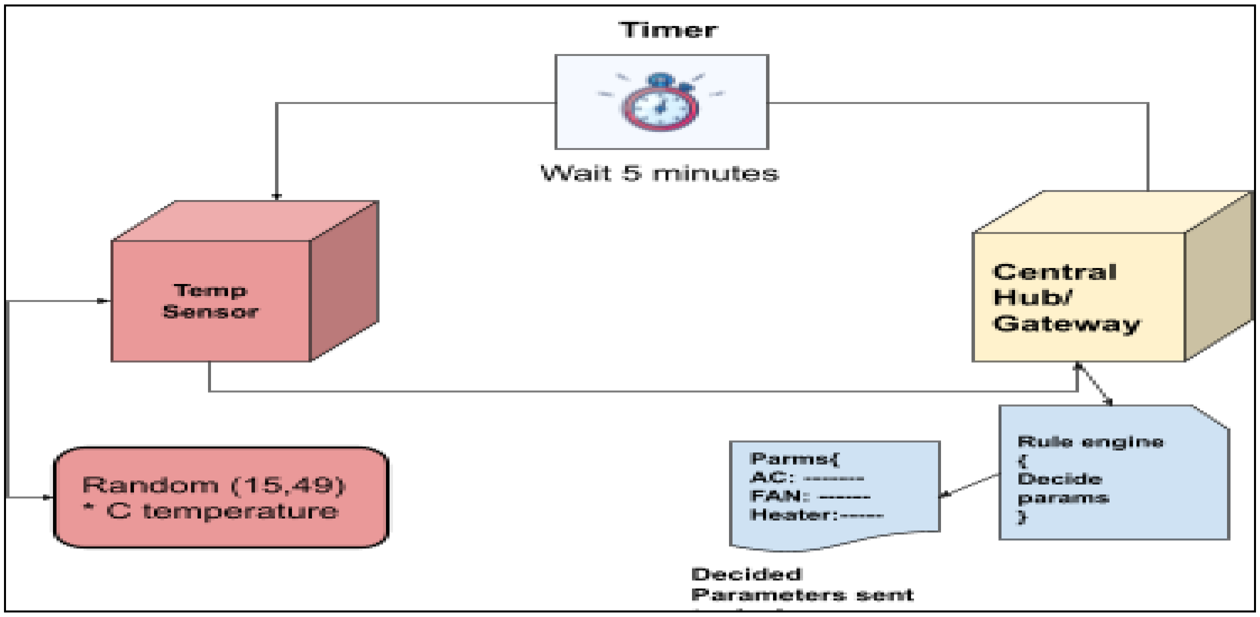

Timer
Wait 5 minutes
Temp Sensor
Central Hub/ Gateway
Random (15,49) * C temperature
Parms{ AC: ------- FAN: ------ Heater:------
Rule engine { Decide params }
Decided Parameters sent

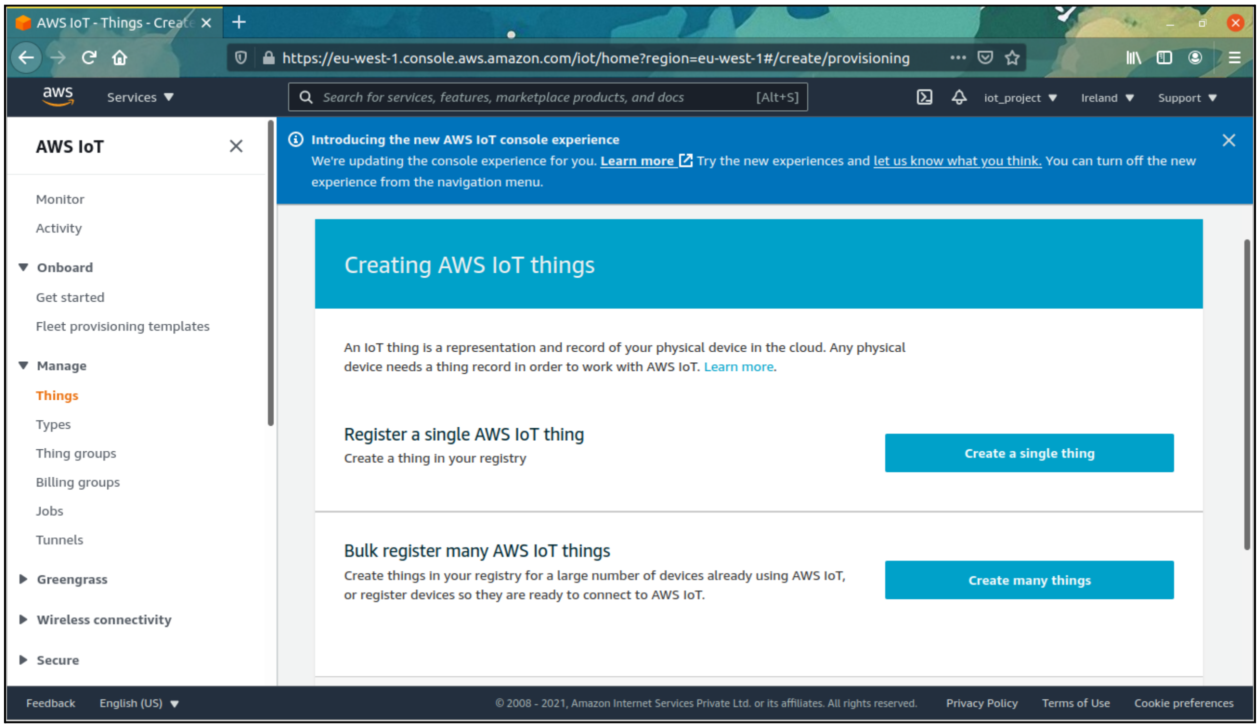
Fig 5 : Sensor and Rule Engine

The Web Interface can be launched in any HTML5 supported browser by launching the "index.html" file. The new user needs to register with the central hub using a specific device ID. This is to mimic a real time user - device connection. For this project we have fixed a certain random device ID to the central hub as **"19AX532021".** The next sections describe all the initial setup and the configuration and running of the system.

4. Running the Project

4.1 Initial Configuration and setup

Before all the devices can communicate with each other on the AWS MQTT on the AWS IoT core they need to be initialized on the AWS IoT core and generate a unique digital certificate for each device.
First each device is added to the **"Thing"** in the **"Manage"** section of the AWS IoT core service.



Introducing the new AWS IoT console experience
We're updating the console experience for you. Learn more Try the new experiences and let us know what you think. You can turn off the new experience from the navigation menu.

Creating AWS IoT things

An IoT thing is a representation and record of your physical device in the cloud. Any physical device needs a thing record in order to work with AWS IoT. Learn more.

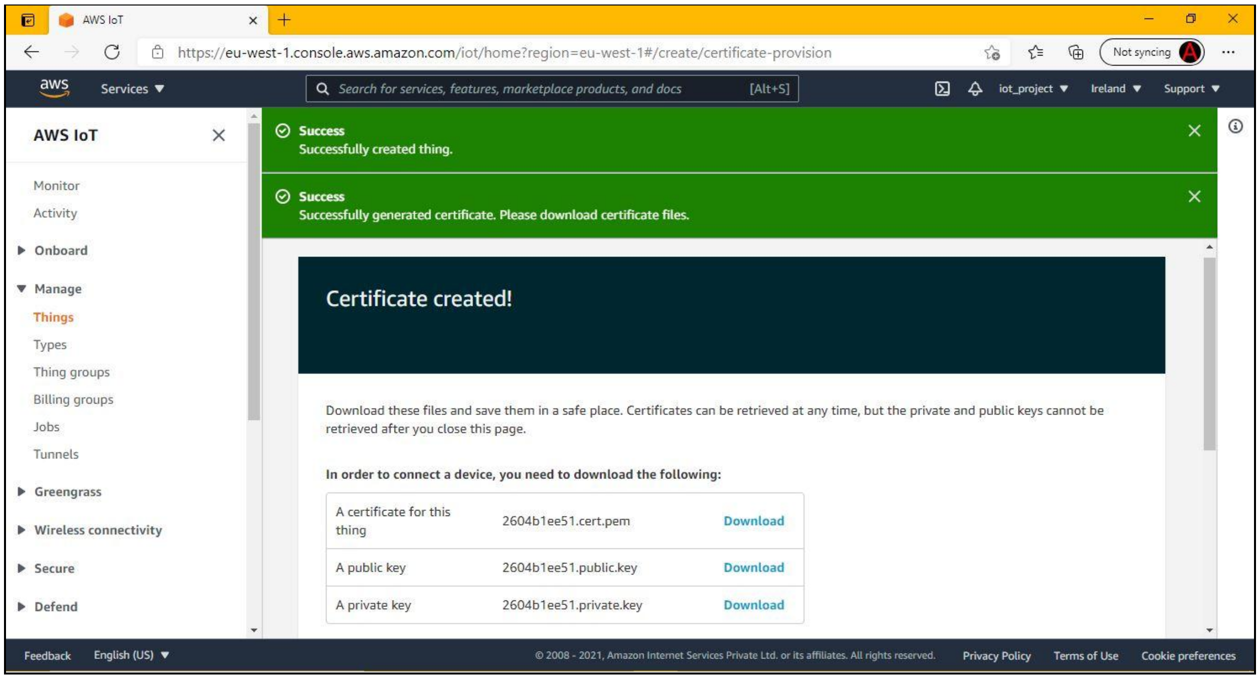
Register a single AWS IoT thing
Create a thing in your registry

Bulk register many AWS IoT things
Create things in your registry for a large number of devices already using AWS IoT, or register devices so they are ready to connect to AWS IoT.

AWS IoT - Things - Create
https://eu-west-1.console.aws.amazon.com/iot/home?region=eu-west-1#/create/provisioning
Services
Search for services, features, marketplace products, and docs [Alt+S]
iot_project
Ireland
Support
AWS IoT
Monitor
Activity
Onboard
Get started
Fleet provisioning templates
Manage
Things
Types
Thing groups
Billing groups
Jobs
Tunnels
Greengrass
Wireless connectivity
Secure
Create a single thing
Create many things
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

Fig 6 : Creating a new ***thing***

Then each thing has a certificate generated automatically by the AWS and downloaded locally to be used during connecting to the MQTT broker.


AWS IoT
https://eu-west-1.console.aws.amazon.com/iot/home?region=eu-west-1#/create/certificate-provision
Not syncing
aws
Services
Search for services, features, marketplace products, and docs [Alt+S]
iot_project
Ireland
Support
AWS IoT
Monitor
Activity
Onboard
Manage
Things
Types
Thing groups
Billing groups
Jobs
Tunnels
Greengrass
Wireless connectivity
Secure
Defend
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

Success
Successfully created thing.

Success
Successfully generated certificate. Please download certificate files.

Certificate created!

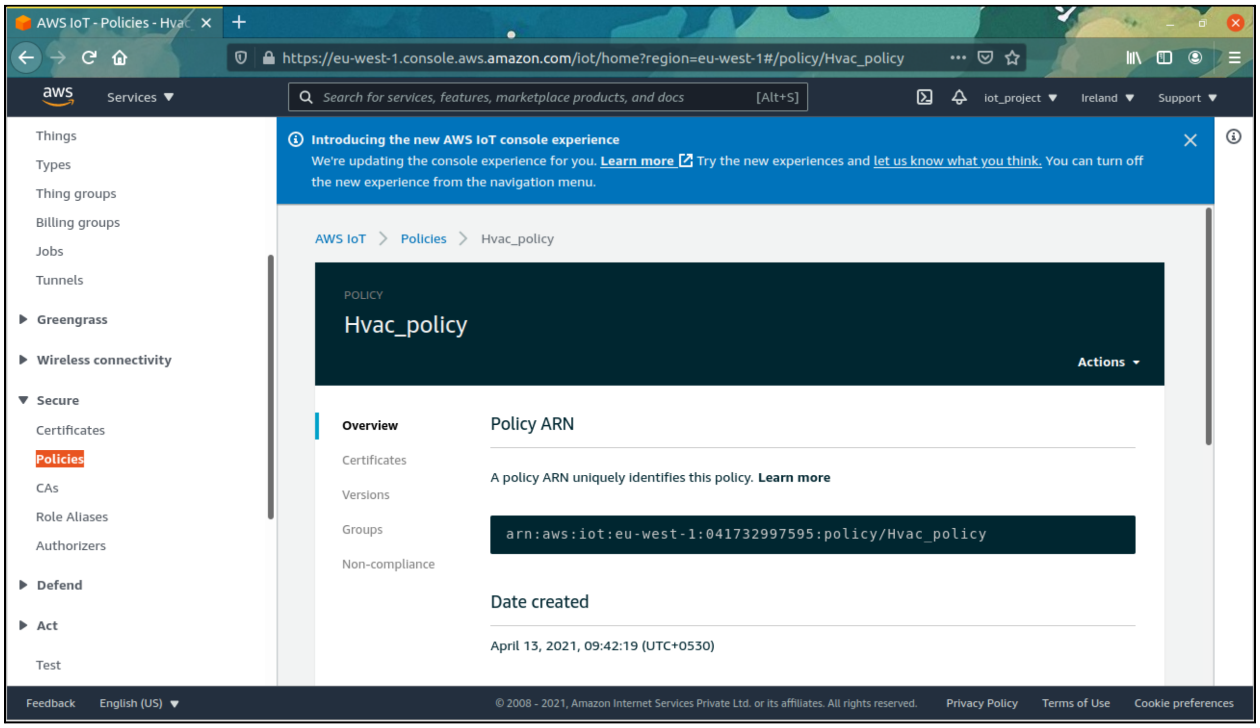
Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

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

A certificate for this thing	2604b1ee51.cert.pem	Download
A public key	2604b1ee51.public.key	Download
A private key	2604b1ee51.private.key	Download

Fig 7 : Certificate Generation for the device

A new security policy is also created and attached to each certificate to allow all types of connections such as publish (), connect() etc:


AWS IoT - Policies - Hvac
https://eu-west-1.console.aws.amazon.com/iot/home?region=eu-west-1#/policy/Hvac_policy
aws
Services
Search for services, features, marketplace products, and docs [Alt+S]
iot_project
Ireland
Support
Things
Types
Thing groups
Billing groups
Jobs
Tunnels
Greengrass
Wireless connectivity
Secure
Certificates
Policies
CAs
Role Aliases
Authorizers
Defend
Act
Test
AWS IoT > Policies > Hvac_policy
Actions
Overview
Certificates
Versions
Groups
Non-compliance
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

Introducing the new AWS IoT console experience
We're updating the console experience for you. Learn more Try the new experiences and let us know what you think. You can turn off the new experience from the navigation menu.

POLICY
Hvac_policy

Policy ARN

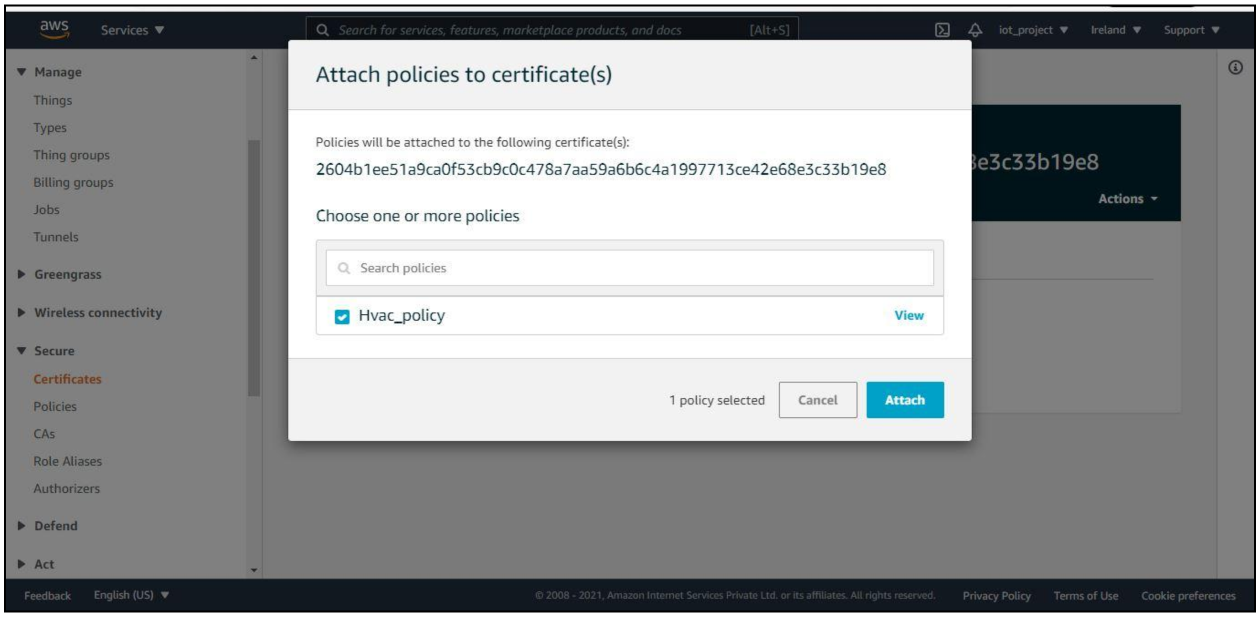
A policy ARN uniquely identifies this policy. Learn more

```
arn:aws:iot:eu-west-1:041732997595:policy/Hvac_policy
```

Date created

April 13, 2021, 09:42:19 (UTC+0530)

Fig 8 : Security Policy for each device



Attach policies to certificate(s)

Policies will be attached to the following certificate(s):

2604b1ee51a9ca0f53cb9c0c478a7aa59a6b6c4a1997713ce42e68e3c33b19e8

Choose one or more policies

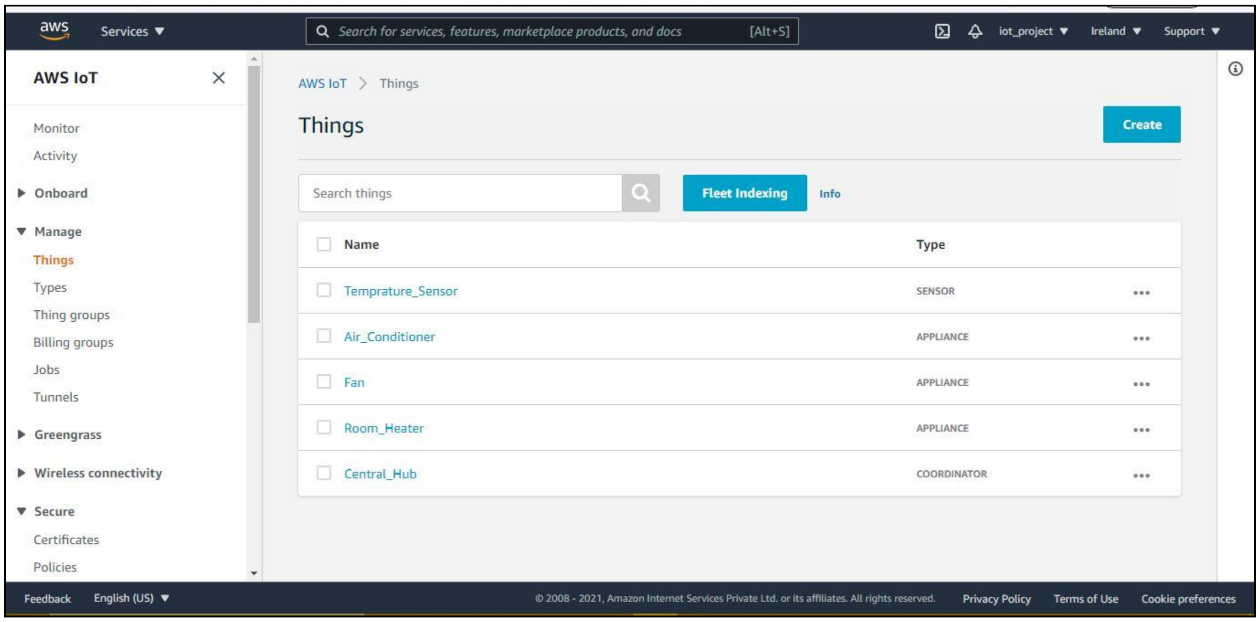
Search policies

[✓] Hvac_policy View

1 policy selected Cancel Attach

Fig 9 : Attaching Security Policy

All the devices are configured, set up and added up on the system as follows:

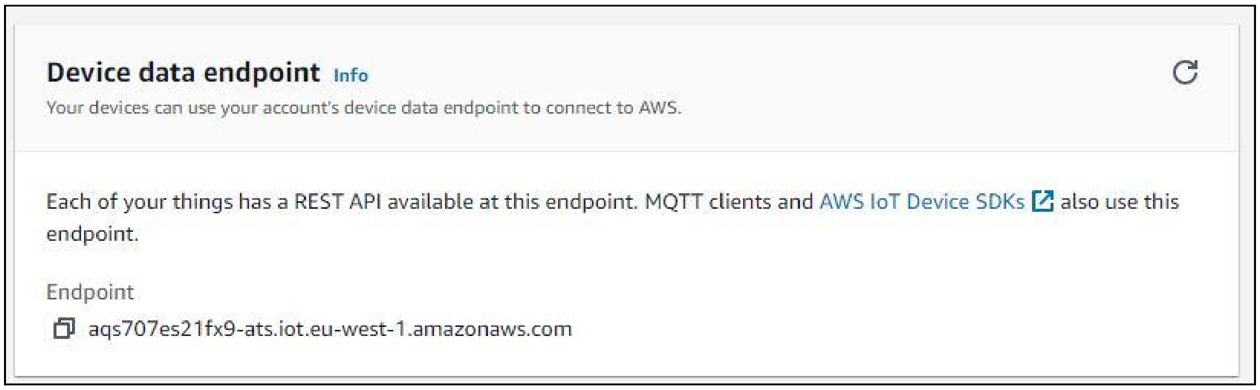


Things

Name	Type
Temprature_Sensor	SENSOR
Air_Conditioner	APPLIANCE
Fan	APPLIANCE
Room_Heater	APPLIANCE
Central_Hub	COORDINATOR

Fig 10 : All things configured

The MQTT Endpoint that all devices can connect and communicate through is as follows:


Device data endpoint Info

Your devices can use your account's device data endpoint to connect to AWS.

Each of your things has a REST API available at this endpoint. MQTT clients and AWS IoT Device SDKs also use this endpoint.

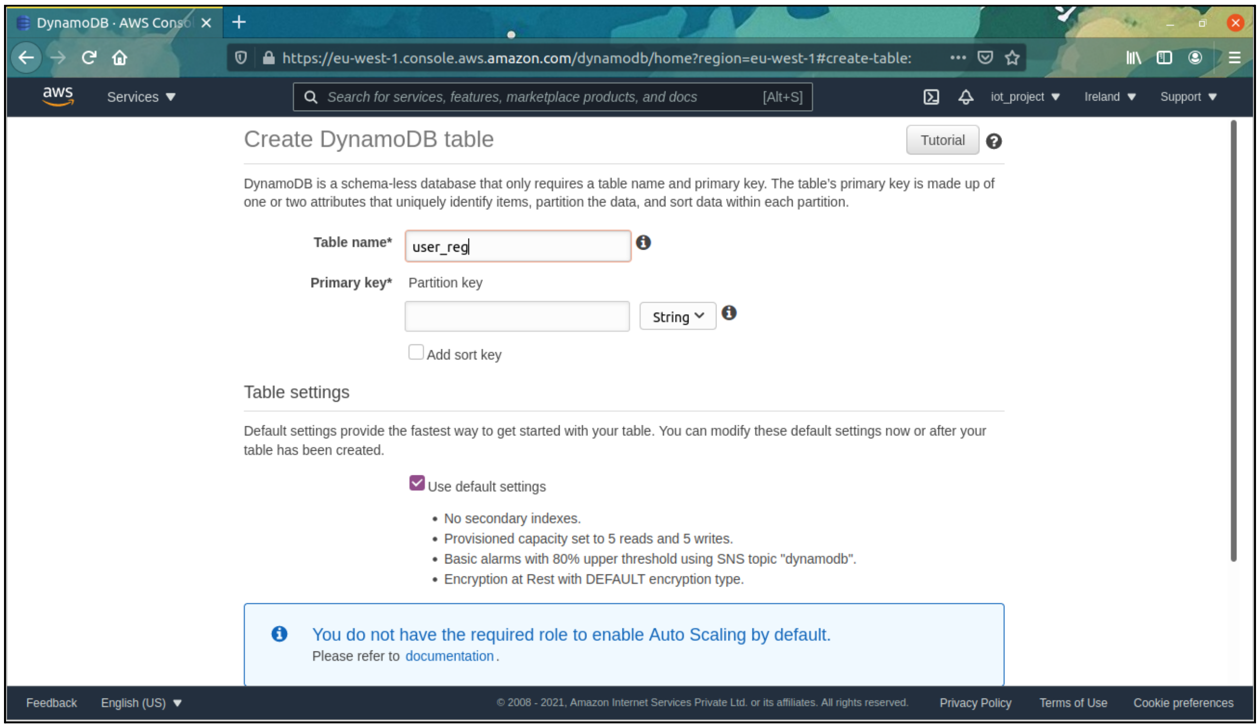
Endpoint

aqs707es21fx9-ats.iot.eu-west-1.amazonaws.com

Fig 11 : AWS MQTT Endpoint

Setting up tables on AWS DynamoDB:

The NoSQL tables “user_reg” and “hvac_device” are created for user authentication and hvac device data communication to the interface as follows:


DynamoDB · AWS Conso
https://eu-west-1.console.aws.amazon.com/dynamodb/home?region=eu-west-1#create-table:
aws
Services
Search for services, features, marketplace products, and docs [Alt+S]
iot_project
Ireland
Support
Tutorial
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

Create DynamoDB table

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* user_reg

Primary key* Partition key

String

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]

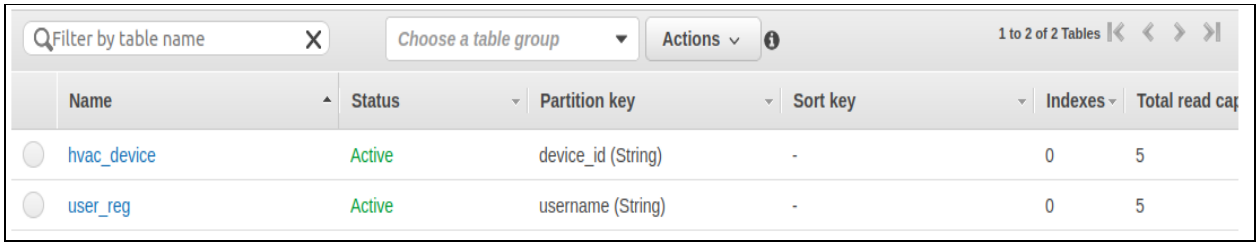
- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".
- Encryption at Rest with DEFAULT encryption type.

You do not have the required role to enable Auto Scaling by default.

Please refer to documentation.

Fig 12 : Creating user_reg table

Similarly the “hvac_dev” table is created and configured. The following tables are created on the DynamoDB service.



Name	Status	Partition key	Sort key	Indexes	Total read cap
hvac_device	Active	device_id (String)	-	0	5
user_reg	Active	username (String)	-	0	5

Fig 13 : Created Tables on DynamoDB

Running the system :

After all the setup is done, we now launch the system and run the project. To run the project for linux the shell script "run_iot_project.sh" is executed.

Running On Linux / Ubuntu

```
$ sudo bash .\run_iot_project.sh
```

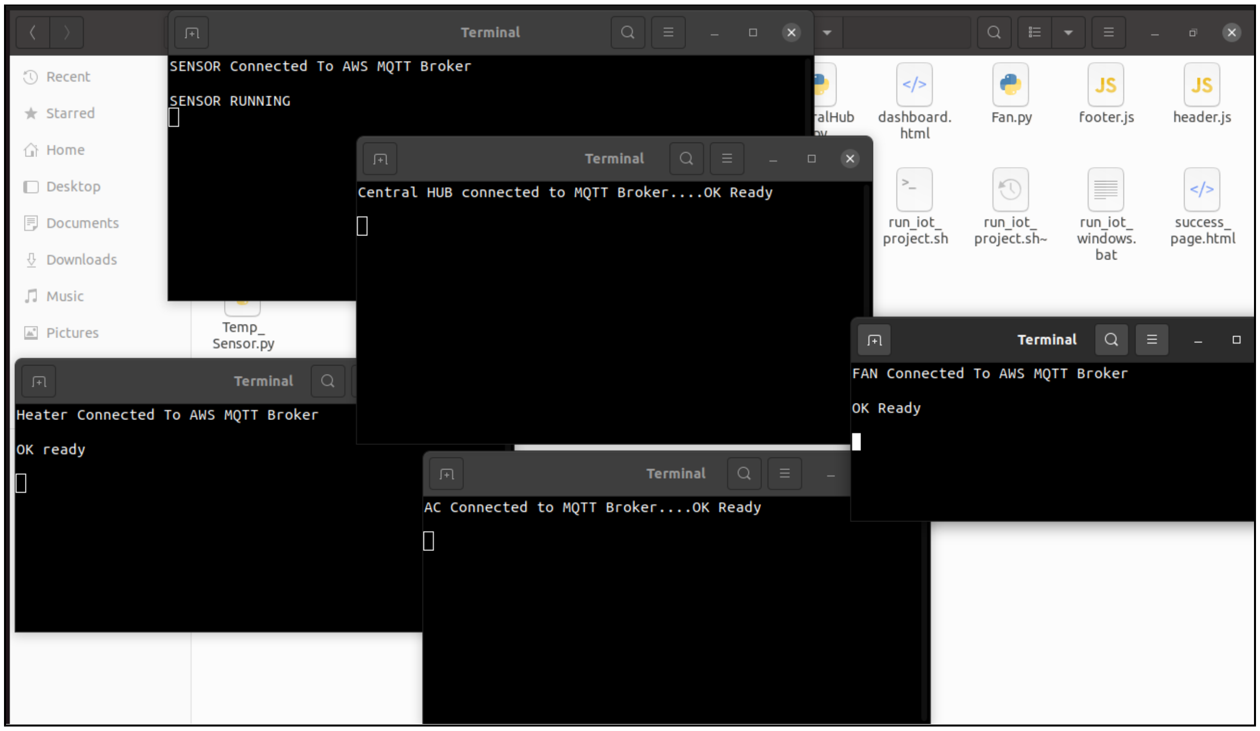
NOTE: The above script, tries also to install some required python libraries if not available such as boto3 (AWS SDK), paho-mqtt (MQTT Client)

Running on Windows

```
> .\run_iot_windows.bat
```

NOTE: The above script, tries also to install some required python libraries if not available such as boto3 (AWS SDK), paho-mqtt (MQTT Client)

The Device Interfaces, central hub and the temperature sensor launched :



```
SENSOR Connected To AWS MQTT Broker

SENSOR RUNNING
```

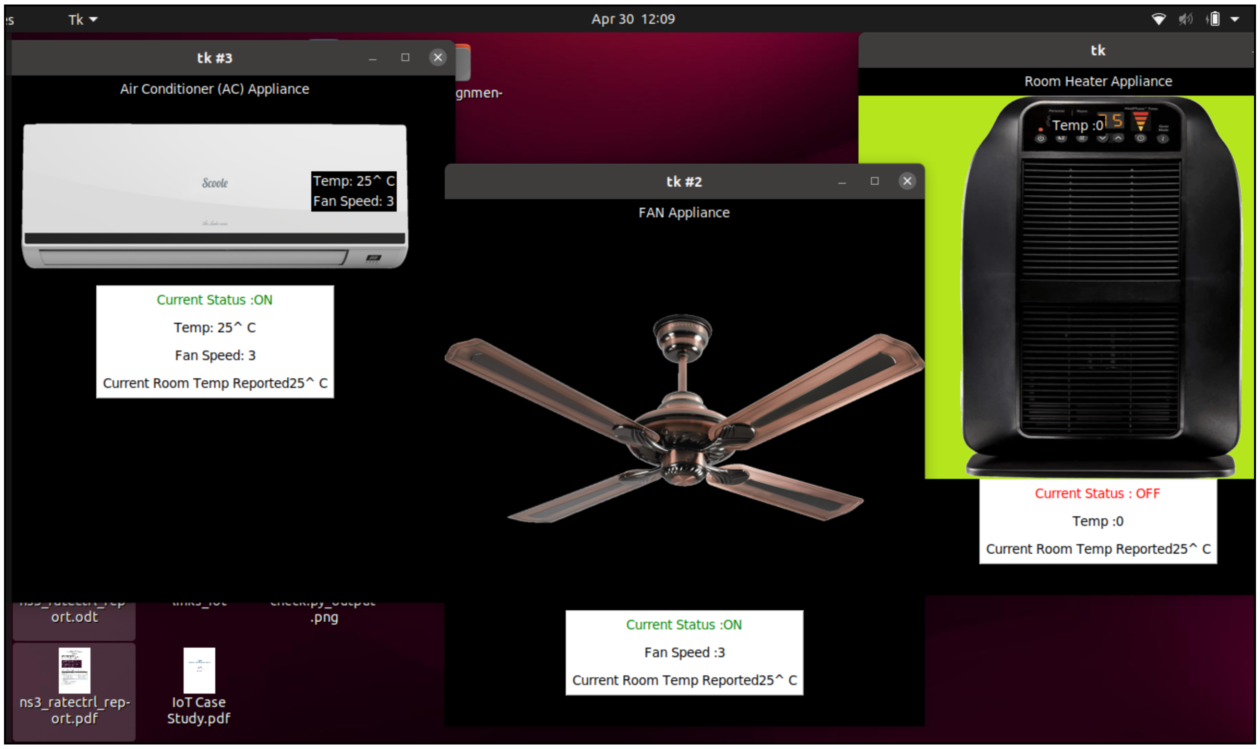
```
Central HUB connected to MQTT Broker....OK Ready
```

```
Heater Connected To AWS MQTT Broker

OK ready
```

```
FAN Connected To AWS MQTT Broker

OK Ready
```

```
AC Connected to MQTT Broker....OK Ready
```

Fig 14 : Terminals for all devices launched



Current Status :ON
Temp: 25^ C
Fan Speed: 3
Current Room Temp Reported25^ C

Current Status :ON
Fan Speed :3
Current Room Temp Reported25^ C

Current Status : OFF
Temp :0
Current Room Temp Reported25^ C

Fig 15 : Interfaces of the Appliances Modules

Running the user Interface :
Logging into the dashboard

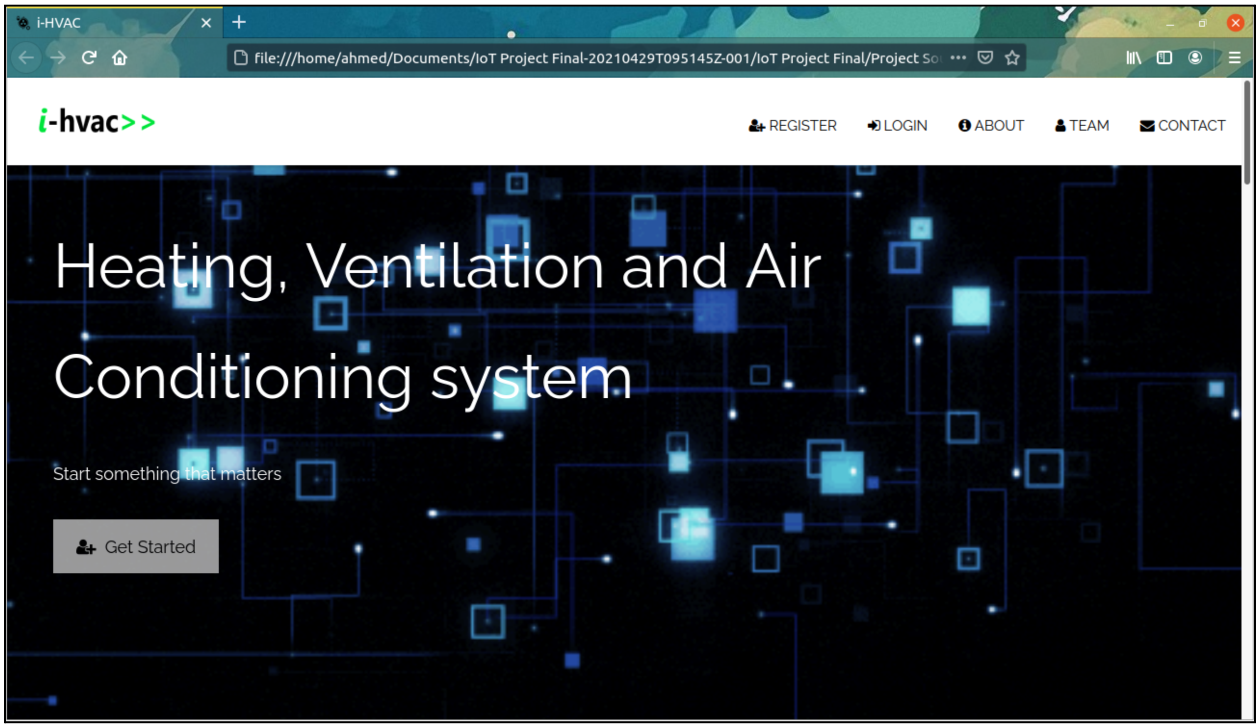
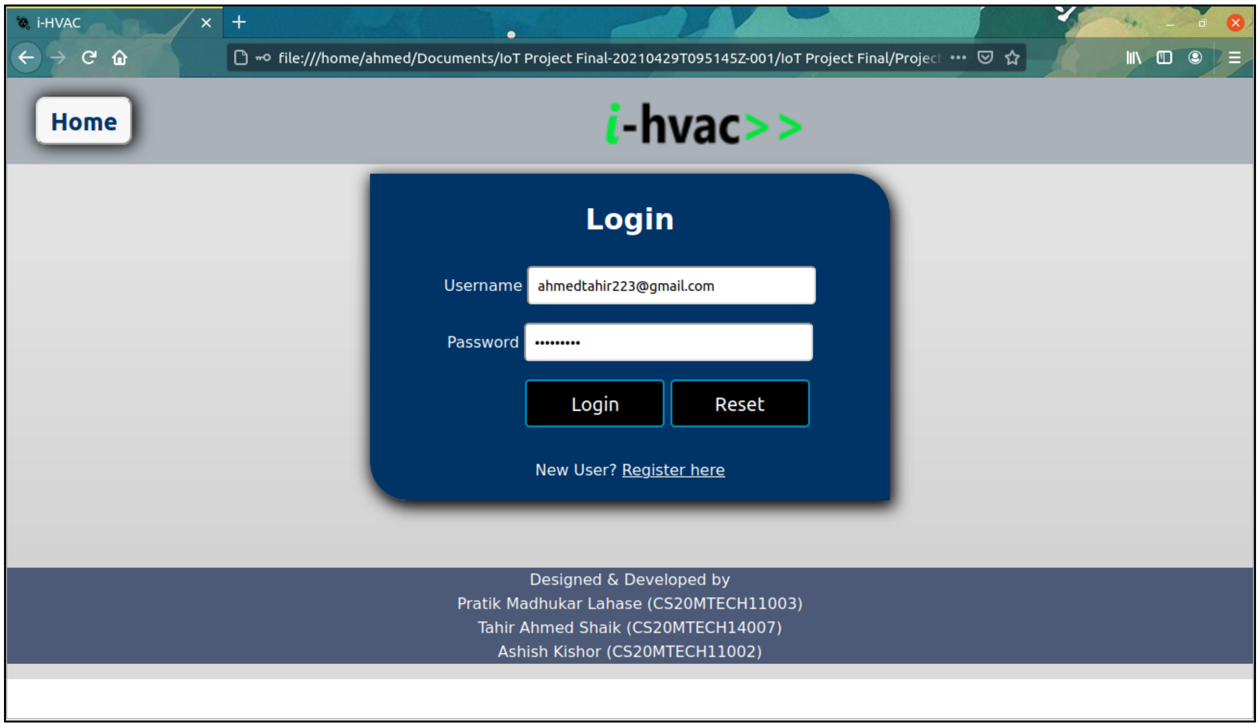

i-HVAC
file:///home/ahmed/Documents/IoT Project Final-20210429T095145Z-001/IoT Project Final/Project So
i-hvac>>
REGISTER
LOGIN
ABOUT
TEAM
CONTACT
Heating, Ventilation and Air Conditioning system
Start something that matters
Get Started

Fig 16 : Main User Interface Page


i-HVAC
file:///home/ahmed/Documents/IoT Project Final-20210429T095145Z-001/IoT Project Final/Project
Home
i-hvac>>
Login
Login
Reset

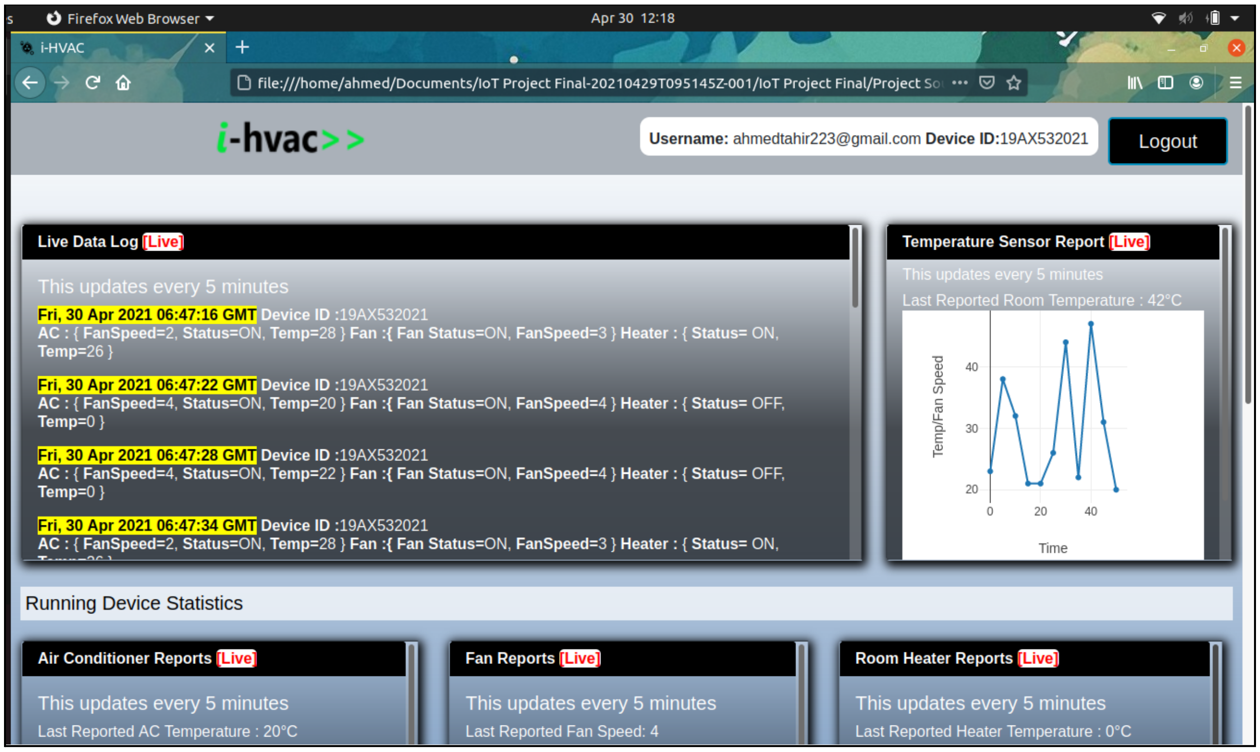
Username: ahmedtahir223@gmail.com

Password: •••••••••

New User? Register here

Designed & Developed by
Pratik Madhukar Lahase (CS20MTECH11003)
Tahir Ahmed Shaik (CS20MTECH14007)
Ashish Kishor (CS20MTECH11002)

Fig 17 : Logging in to the system


Firefox Web Browser
Apr 30 12:18
i-HVAC
file:///home/ahmed/Documents/IoT Project Final-20210429T095145Z-001/IoT Project Final/Project So
i-hvac>>
Logout
Temp/Fan Speed
40
30
20
0
20
40
Time

Username: ahmedtahir223@gmail.com **Device ID:**19AX532021

Live Data Log Live

This updates every 5 minutes

Fri, 30 Apr 2021 06:47:16 GMT **Device ID** :19AX532021
AC : { **FanSpeed**=2, **Status**=ON, **Temp**=28 } **Fan** :{ **Fan Status**=ON, **FanSpeed**=3 } **Heater** : { **Status**= ON, **Temp**=26 }

Fri, 30 Apr 2021 06:47:22 GMT **Device ID** :19AX532021
AC : { **FanSpeed**=4, **Status**=ON, **Temp**=20 } **Fan** :{ **Fan Status**=ON, **FanSpeed**=4 } **Heater** : { **Status**= OFF, **Temp**=0 }

Fri, 30 Apr 2021 06:47:28 GMT **Device ID** :19AX532021
AC : { **FanSpeed**=4, **Status**=ON, **Temp**=22 } **Fan** :{ **Fan Status**=ON, **FanSpeed**=4 } **Heater** : { **Status**= OFF, **Temp**=0 }

Fri, 30 Apr 2021 06:47:34 GMT **Device ID** :19AX532021
AC : { **FanSpeed**=2, **Status**=ON, **Temp**=28 } **Fan** :{ **Fan Status**=ON, **FanSpeed**=3 } **Heater** : { **Status**= ON,

Temperature Sensor Report Live

This updates every 5 minutes

Last Reported Room Temperature : 42°C

Running Device Statistics

Air Conditioner Reports Live

This updates every 5 minutes

Last Reported AC Temperature : 20°C

Fan Reports Live

This updates every 5 minutes

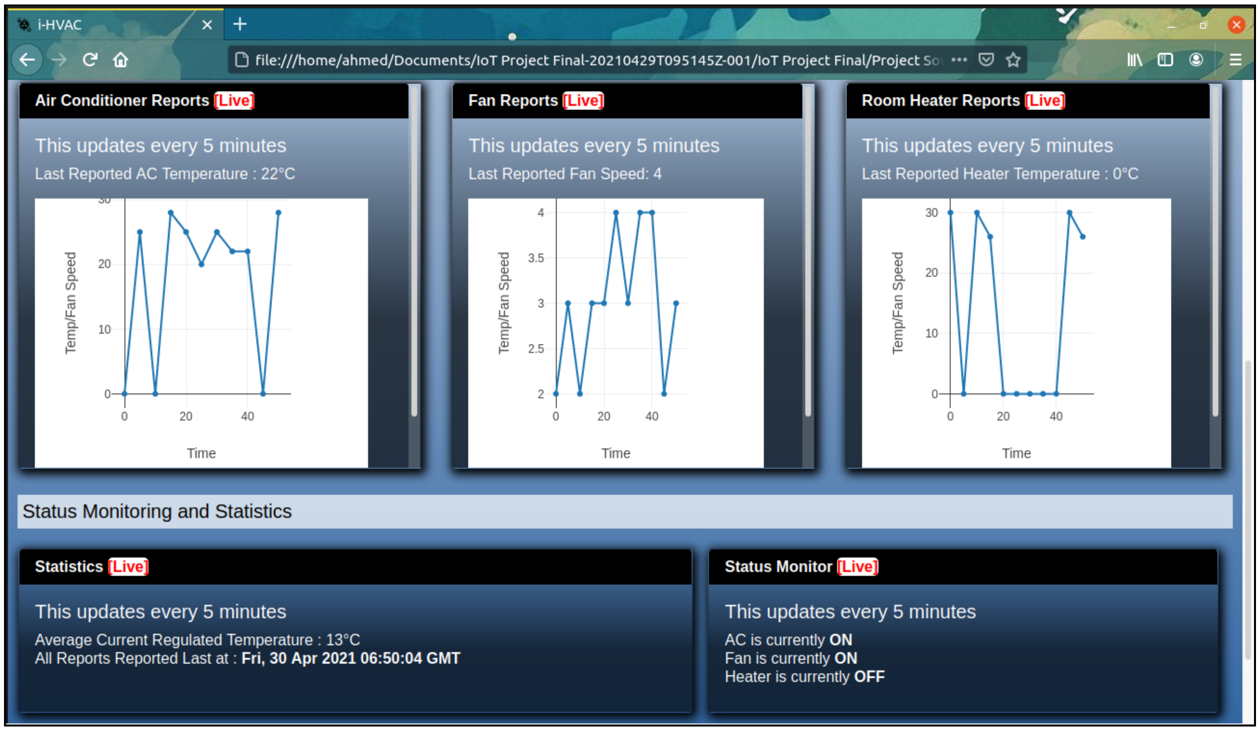
Last Reported Fan Speed: 4

Room Heater Reports Live

This updates every 5 minutes

Last Reported Heater Temperature : 0°C

Fig 18 : Interface Dashboard



i-HVAC
file:///home/ahmed/Documents/IoT Project Final-20210429T095145Z-001/IoT Project Final/Project So
Temp/Fan Speed
Time

Air Conditioner Reports Live

This updates every 5 minutes

Last Reported AC Temperature : 22°C

Fan Reports Live

This updates every 5 minutes

Last Reported Fan Speed: 4

Room Heater Reports Live

This updates every 5 minutes

Last Reported Heater Temperature : 0°C

Status Monitoring and Statistics

Statistics Live

This updates every 5 minutes

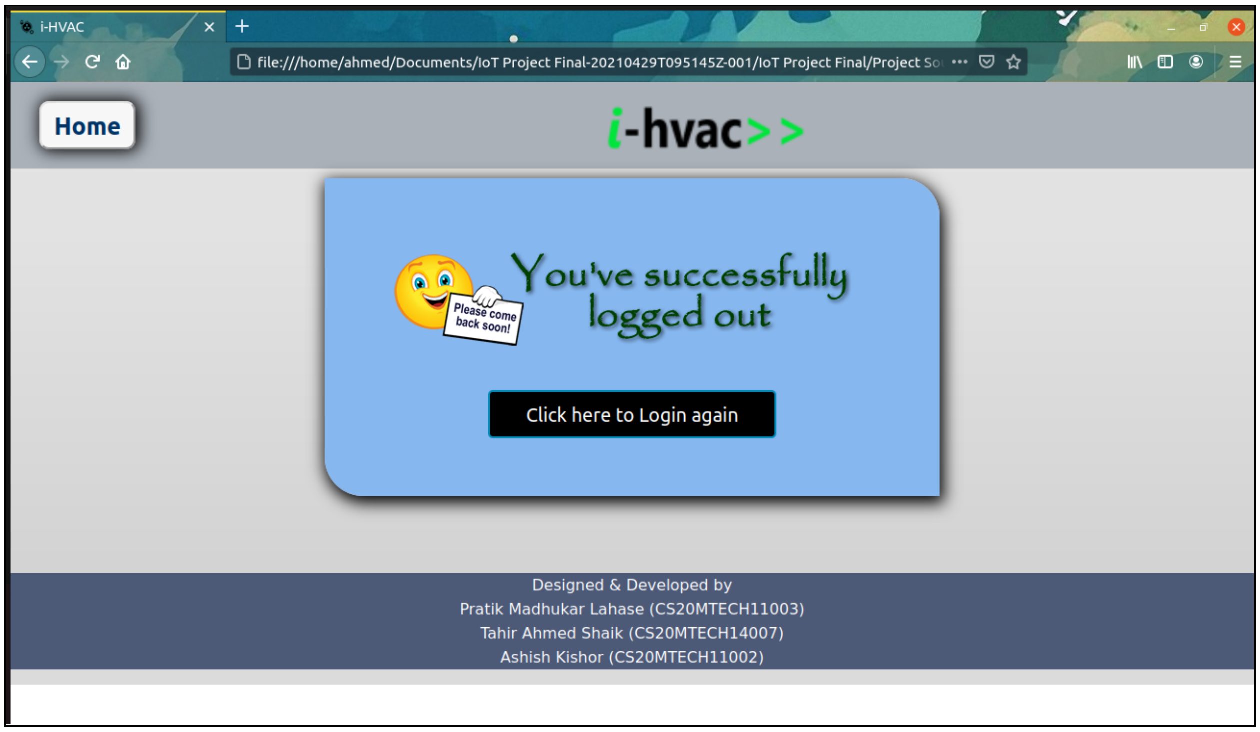
Average Current Regulated Temperature : 13°C
All Reports Reported Last at : **Fri, 30 Apr 2021 06:50:04 GMT**

Status Monitor Live

This updates every 5 minutes

AC is currently **ON**
Fan is currently **ON**
Heater is currently **OFF**

Fig 19 : Interface Dashboard



i-HVAC
file:///home/ahmed/Documents/IoT Project Final-20210429T095145Z-001/IoT Project Final/Project So
Home
i-hvac>>
Please come back soon!
Click here to Login again

You've successfully logged out

Designed & Developed by
Pratik Madhukar Lahase (CS20MTECH11003)
Tahir Ahmed Shaik (CS20MTECH14007)
Ashish Kishor (CS20MTECH11002)

Fig 20 : Logged out of system

A few system plots for the system :

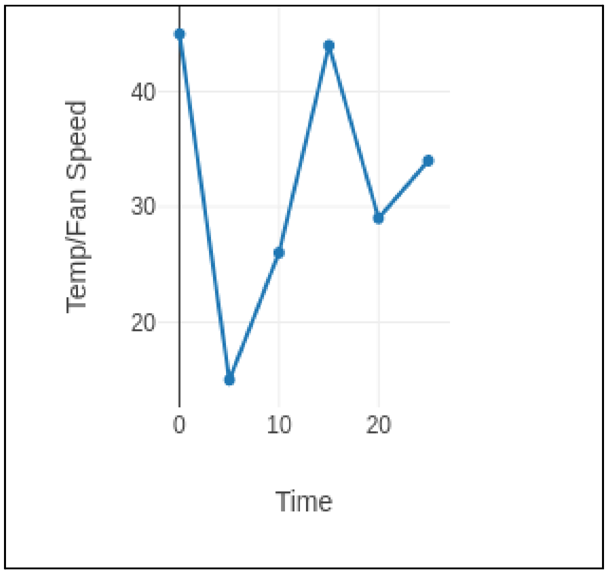


Fig 21 : Temperature Sensor Plot

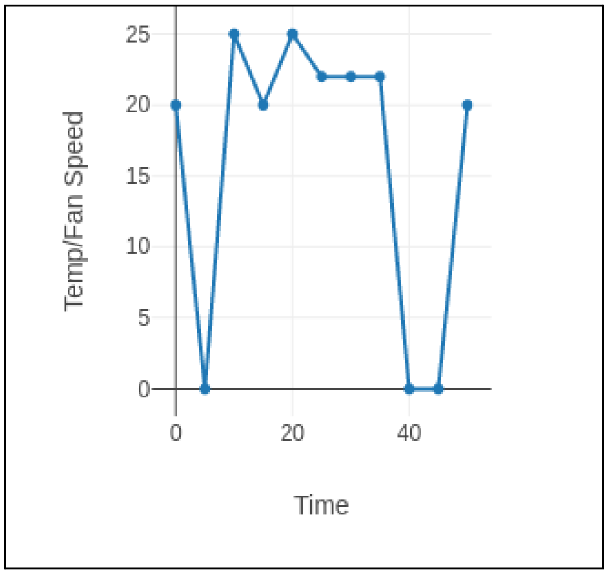


Fig 22 : AC Sensor Plot

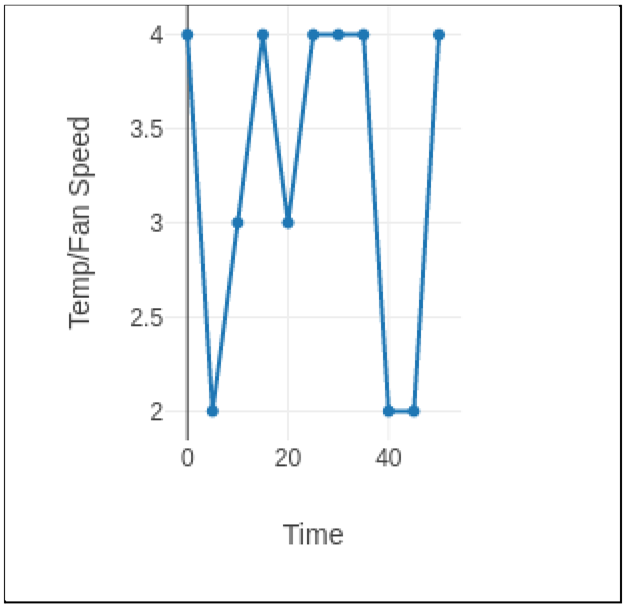


Fig 23 : Fan Plot

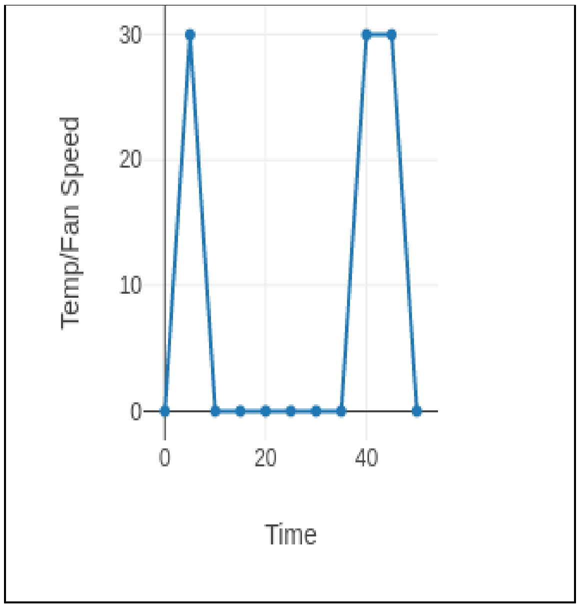
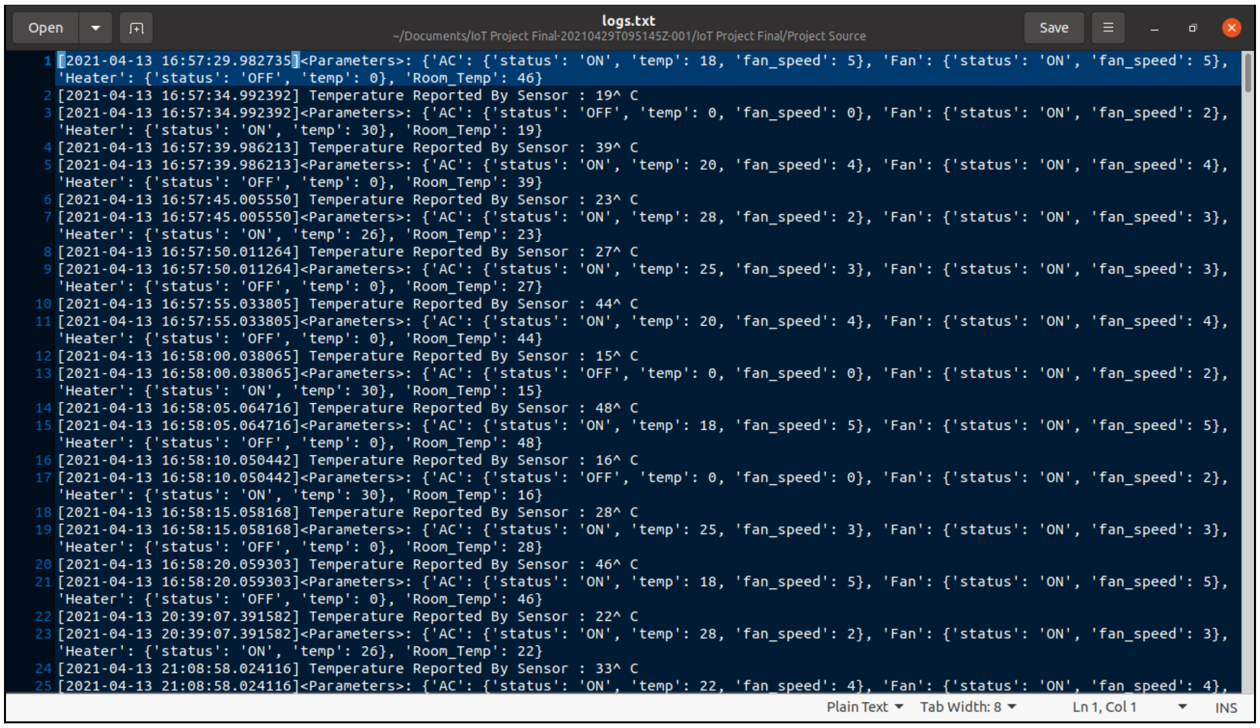


Fig 24 : Room Heater Plot

The Message Logs saved on the system are as follows :



```
[2021-04-13 16:57:29.982735]<Parameters>: {'AC': {'status': 'ON', 'temp': 18, 'fan_speed': 5}, 'Fan': {'status': 'ON', 'fan_speed': 5},
'Heater': {'status': 'OFF', 'temp': 0}, 'Room_Temp': 46}
[2021-04-13 16:57:34.992392] Temperature Reported By Sensor : 19^ C
[2021-04-13 16:57:34.992392]<Parameters>: {'AC': {'status': 'OFF', 'temp': 0, 'fan_speed': 0}, 'Fan': {'status': 'ON', 'fan_speed': 2},
'Heater': {'status': 'ON', 'temp': 30}, 'Room_Temp': 19}
[2021-04-13 16:57:39.986213] Temperature Reported By Sensor : 39^ C
[2021-04-13 16:57:39.986213]<Parameters>: {'AC': {'status': 'ON', 'temp': 20, 'fan_speed': 4}, 'Fan': {'status': 'ON', 'fan_speed': 4},
'Heater': {'status': 'OFF', 'temp': 0}, 'Room_Temp': 39}
[2021-04-13 16:57:45.005550] Temperature Reported By Sensor : 23^ C
[2021-04-13 16:57:45.005550]<Parameters>: {'AC': {'status': 'ON', 'temp': 28, 'fan_speed': 2}, 'Fan': {'status': 'ON', 'fan_speed': 3},
'Heater': {'status': 'ON', 'temp': 26}, 'Room_Temp': 23}
[2021-04-13 16:57:50.011264] Temperature Reported By Sensor : 27^ C
[2021-04-13 16:57:50.011264]<Parameters>: {'AC': {'status': 'ON', 'temp': 25, 'fan_speed': 3}, 'Fan': {'status': 'ON', 'fan_speed': 3},
'Heater': {'status': 'OFF', 'temp': 0}, 'Room_Temp': 27}
[2021-04-13 16:57:55.033805] Temperature Reported By Sensor : 44^ C
[2021-04-13 16:57:55.033805]<Parameters>: {'AC': {'status': 'ON', 'temp': 20, 'fan_speed': 4}, 'Fan': {'status': 'ON', 'fan_speed': 4},
'Heater': {'status': 'OFF', 'temp': 0}, 'Room_Temp': 44}
[2021-04-13 16:58:00.038065] Temperature Reported By Sensor : 15^ C
[2021-04-13 16:58:00.038065]<Parameters>: {'AC': {'status': 'OFF', 'temp': 0, 'fan_speed': 0}, 'Fan': {'status': 'ON', 'fan_speed': 2},
'Heater': {'status': 'ON', 'temp': 30}, 'Room_Temp': 15}
[2021-04-13 16:58:05.064716] Temperature Reported By Sensor : 48^ C
[2021-04-13 16:58:05.064716]<Parameters>: {'AC': {'status': 'ON', 'temp': 18, 'fan_speed': 5}, 'Fan': {'status': 'ON', 'fan_speed': 5},
'Heater': {'status': 'OFF', 'temp': 0}, 'Room_Temp': 48}
[2021-04-13 16:58:10.050442] Temperature Reported By Sensor : 16^ C
[2021-04-13 16:58:10.050442]<Parameters>: {'AC': {'status': 'OFF', 'temp': 0, 'fan_speed': 0}, 'Fan': {'status': 'ON', 'fan_speed': 2},
'Heater': {'status': 'ON', 'temp': 30}, 'Room_Temp': 16}
[2021-04-13 16:58:15.058168] Temperature Reported By Sensor : 28^ C
[2021-04-13 16:58:15.058168]<Parameters>: {'AC': {'status': 'ON', 'temp': 25, 'fan_speed': 3}, 'Fan': {'status': 'ON', 'fan_speed': 3},
'Heater': {'status': 'OFF', 'temp': 0}, 'Room_Temp': 28}
[2021-04-13 16:58:20.059303] Temperature Reported By Sensor : 46^ C
[2021-04-13 16:58:20.059303]<Parameters>: {'AC': {'status': 'ON', 'temp': 18, 'fan_speed': 5}, 'Fan': {'status': 'ON', 'fan_speed': 5},
'Heater': {'status': 'OFF', 'temp': 0}, 'Room_Temp': 46}
[2021-04-13 20:39:07.391582] Temperature Reported By Sensor : 22^ C
[2021-04-13 20:39:07.391582]<Parameters>: {'AC': {'status': 'ON', 'temp': 28, 'fan_speed': 2}, 'Fan': {'status': 'ON', 'fan_speed': 3},
'Heater': {'status': 'ON', 'temp': 26}, 'Room_Temp': 22}
[2021-04-13 21:08:58.024116] Temperature Reported By Sensor : 33^ C
[2021-04-13 21:08:58.024116]<Parameters>: {'AC': {'status': 'ON', 'temp': 22, 'fan_speed': 4}, 'Fan': {'status': 'ON', 'fan_speed': 4},
```

Open
logs.txt
~/Documents/IoT Project Final-20210429T095145Z-001/IoT Project Final/Project Source
Save
Plain Text
Tab Width: 8
Ln 1, Col 1
INS

Fig 25 : Message logs saved on system

5. Technical Specifications

Item	**Specification**
Operating System	Linux Ubuntu LTS 20.04
Programming Language	Python 3.8
Interface Modules	● Air_Conditioner.py (AC Module) ● Fan.py (Fan Module) ● Room_Heater.py (Room Heater) ● Central_Hub.py(Central Hub Module) ● Temp_Sensor.py (Thermostat/Temperature sensor module)
	Required Libraries ● boto3 (AWS SDK Library) ● ssl (For SSL verification of certificates) ● Paho-mqtt (MQTT Client Library)

Web-Interface	**HTML, CSS, JS** ● index .html - Main page of the interface ● Login.html - Login page of interface ● Logout.html - log out page ● Registration.html - User registration page ● dashboard .html - dashboard page of interface ● Success_page.html - success page
	Required Scripts ● Bootstarp.js ● AWS-SDK 2.17 - AWS SDK javascript lib ● Plotly.js - For graph plotting in interface
Services Used	AWS IoT Core AWS DynamoDB

6. Conclusion

Through this project, we have understood the working of an HVAC system, the various components involved and how the different components exchange information and communicate leveraging the IoT standards and technologies such as the MQTT communication standard, The cloud computing storage technology for data storage and processing, We get to know the working of IoT devices and the task of making the processes smart, efficient and self-reliant with little human intervention.

7. References:

[1] https://en.wikipedia.org/wiki/Heating,_ventilation,_and_air_conditioning
[2] https://en.wikipedia.org/wiki/Thermostat
[3] https://www.explainthatstuff.com/thermostats.html
[4] M. Fukuta, K. Matsui, M. Ito and H. Nishi, "Proposal for home energy management system to survey individual thermal comfort range for HVAC control with little contribution from users," 2015 IEEE 13th International Conference on Industrial Informatics (INDIN), 2015, pp. 658-663, doi: 10.1109/INDIN.2015.7281813.

Appendix

Deliverables Enclosed:
Project Structure:

Cs20mtech14007|cs20mtech11003|cs20mtech11002.zip

```
          |
          |
          |----------------------Project Source
          |------README.txt     |
                                |---Certificates (Directory Consisting of all device certificates
                                |
                                |---css (Directory of CSS Files)
                                |---res (All resource images, logos used)
                                |---screenshots (All screenshots of working system, logs etc)
                                |
                                |---Air_Conditioner.py (AC Module)
                                |---Fan.py (FAN Module)
                                |---Room_Heater.py ( Room Heater Module)
                                |---Central_Hub.py (Central Hub Module)
                                |---Temp_Sensor.py (Temperature Sensor Module)
                                |
                                |
                                |---logs.txt (Log file consisting of logs till date)
                                |---index.html, registration.html, login.html, logout.html
                                |    (Web-Interface Files)
                                |
                                |---run_iot_project.sh (Script for running project on linux)
                                |---run_iot_windows.bat (Script to run project on Windows)
```

File	Description
Project Source/	Project source directory consisting of all code base files (Device Modules, Sensor Script, Interface files, log files, screenshots)
README.txt	Readme file for the project
Report.pdf	Project Report file