

CALIFORNIA STATE UNIVERSITY SAN MARCOS

PROJECT SIGNATURE PAGE

PROJECT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE

MASTER OF SCIENCE

COMPUTER SCIENCE

PROJECT TITLE: SENSOR DATA VISUALIZATION ON GOOGLE MAPS USING AWS, IOT
AND STM32L475 DISCOVERY KIT BOARD

AUTHOR: Vishakha Subhash Supekar

DATE OF SUCCESSFUL DEFENSE: 04/28/2020

THE PROJECT HAS BEEN ACCEPTED BY THE PROJECT COMMITTEE IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF
SCIENCE IN COMPUTER SCIENCE.

Dr Ali Ahmadiania
PROJECT COMMITTEE CHAIR



SIGNATURE

4/28/2020

DATE

Dr Ahmad Hadaegh
PROJECT COMMITTEE MEMBER



April 28, 2020

DATE

Sensor data Visualization on Google Maps using AWS, IOT and STM32L475 discovery Kit board

Submitted in Partial fulfillment of the requirements for
the degree of Master of Science

By

Vishakha Subhash Supekar

Under the guidance of

Dr. Ali Ahmadinia

Abstract

The changing environment plays a vital role in the health of humans as well as animals. Due to the grave impacts of pollution and increased temperature on human health, it is of utmost importance to monitor the environment parameters at every step. With the recent developments in the Internet of Things (IoT), monitoring these parameters in real-time has become possible and cost-effective. In this project, I have designed and implemented a web application to demonstrate an intelligent temperature and humidity reporting system using IoT and cloud. This application aims to provide a visual map for the users to analyze the temperatures in different areas to make an informed decision. This system can be used as a prototype in strengthening real-time temperature data and humidity data in many applications such as Nest, activating farm sprinklers based on weather data, and helping patients sensitive to high temperature and high humidity to take prompt action upon real-time notification on change in temperature or humidity. The proposed system provides a new solution by utilizing the sensor activity on various applications as it is represented using Amazon Web Services IoT, which is an emerging area of research.

Acknowledgment

I would like to express my profound gratitude to my project advisor and mentor Dr. Ali Ahmadinia for giving me the opportunity to work on this project. I am grateful for his constant guidance and support throughout the completion of this project. I have learned a lot while working on this project under his supervision. I would also like to thank Dr. Ahmad Hadaegh for agreeing to be on the project committee and providing valuable feedback about my work.

Contents

Introduction	7
1.1 Motivation	7
1.2 Goal	8
2. Related Work	9
2.1 Aeroqual	9
2.2 PurpleAir	9
2.3 AirSense	11
2.4 IoT based Weather Station	12
2.5 Mobile sensor unit for online air quality monitoring	13
3. Background	14
3.1 What is a Microcontroller?	14
3.2 Overview of STM32L475 Discovery Kit	14
3.2.1 Key Features	15
3.3 Amazon FreeRTOS	16
3.4 AWS IOT Things	17
3.5 AWS DynamoDB (NoSql Database)	18
4. Project Architecture, Implementation and Results	19
4.1 High Level Architecture	19
4.2 Flow chart	20
4.3 Board configuration	21
4.4 Setup a serial communication	23
4.5 Verify the data on AWS	23
4.6 Code snippet to Query the DynamoDB table	25
4.7 Client-Side programming code snippet	27
5. Applications	33
5.1 Regulating the Air Conditioner(AC) temperature of a room	33
5.2 Monitoring humidity in parks	33
5.3 Smart Farming	34
6. Conclusion and Future Enhancements	35
7. References	36
APPENDIX - Technical terminologies	390

List of Figures

Figure 1. PurpleAir map reflecting all the devices around the globe [27]	10
Figure 2. AirSense Smartphone Application [22]	12
Figure 3. MCU Ecosystem [28]	14
Figure 4. STM32 IoT Discovery kit with Sensor placement and Features [24]	15
Figure 5. Amazon FreeRTOS High-level Architecture [29]	17
Figure 6. AWS IoT High-level Architecture [30]	17
Figure 7. High-level architecture diagram of the project	19
Figure 8. Flow chart.....	20
Figure 9. Screenshot of serial terminal showing successful connection to MQTT and sensor data	23
Figure 10. Subscribe to MQTT topic on AWS.....	24
Figure 11. Sensor data sent from board to subscribed topic on AWS.....	24
Figure 12. Screenshot of the DynamoDB table data.....	25
Figure 13. Screenshot of the Map with data points in satellite view	29
Figure 14. Screenshot of the Map with data points in map view	29
Figure 15. Screenshot of the Map with Data information on that location	30
Figure 16. Screenshot of the map showing popup for lowest temperature value.....	30
Figure 17. Temperature Chart	31
Figure 18. Humidity chart	31
Figure 19. Email notification screenshot	32

List of Tables

1. Cost comparison of PurpleAir and Aeroqual products with the STM32 board.....	11
2. Software requirements.....	21

1. Introduction

In today's world, the increase in temperature has been a grave concern for human survival. The environmental parameters not only affect humans but also animals and plants. In recent years, with the revolutionization of IoT, many research and studies were conducted to monitor the environmental parameters in real-time in almost every industry [5]. In the agricultural industry, the use of sensors to monitor plant growth and taking plantation decisions based on the collected data has been proposed [16,17]. Similarly, in the health industry studies have concluded the adverse effects of increased temperature, humidity and air pollutants on the health of heart patients and patients with respiratory diseases [18]. Keeping in mind the need for and vast areas of application, I have designed and developed a web application which will collect the sensor data from multiple sensors and visualize that data on google maps.

1.1 Motivation

Nowadays, IoT has become an integral part of our life. Its influence on our day to day activities has significantly increased over the past few years. For instance, turning on the home air conditioner while sitting in the office has only made possible with the evolution of IoT. From inside the home to right across society, the IoT is a rebellion that guarantees to change people's lives and lead to an increased quality of life.

Today, humans need everything at their disposal and with just a few clicks away. Keeping in mind the need for making things smarter and accessible in real-time has fueled the development of things, technologies, and applications connected to the cloud. The usage of such smart devices is not just limited to the software industry but is also a boom for the medical, food, agriculture industry, etc. The sensor technologies can sense the surroundings in many ways such as monitoring temperature, pressure, air pollutants and many more.

In recent years, studies have shown the effects of increased temperature and air pollutants on the health of patients. Many companies have heavily invested in the IoT healthcare sector to improve the quality of people's life [21]. In this project, I decided to build a web application that could help people know about the temperatures at various locations in real-time. The STM32 board that I am using has embedded Wi-Fi on it which motivated me to go forward with this project idea. The board sends data directly to the cloud where it is stored and could be further used for analysis purposes. The gathering of the data opens up

multiple possibilities for its use in real-time. The real-time feature of the project forms the basis for the project idea. Also, with the low-cost power consumption of the board and the ease of building things cloud-based fueled the project idea further.

1.2 Goal

The prime aim of this project is to store the real-time data on the cloud and manipulate that data in the form of map visualization. The real-time aspect of the project makes it different from other temperature sensor projects out there. Another goal of the project is to make it feasible enough so that it can be used as a prototype for building other web applications. With the implementation of this project, I have tried to address some of the issues or missing links from other similar projects. Similar projects and related work are discussed in Section 2.

2. Related Work

Due to widespread awareness amongst people regarding the impact of changing environmental parameters and pollution, there are a variety of air quality monitoring products available in the market.

2.1 Aeroqual

Aeroqual is a New Zealand based company that provides hardware and software tools to monitor air quality and help professionals, researchers, government and individuals to learn more about the air they breathe. Aeroqual offers a number of indoor and outdoor air quality monitoring portable and handheld products. Some of the features that these products offer are as follows [25].

- Air quality sensor design, fabrication, and calibration
- Signal processing and algorithms
- Instrument design and development
- Application and support of instruments in the field
- Cloud software design and development
- Data capture and visualization
- Remote network management
- Atmospheric science

It also offers a wide range of products to monitor the ozone impact. Though Aeroqual provides graphs to study the fluctuations among the specified parameters, it does not have the map feature which this master project has implemented.

2.2 PurpleAir

One of the other products available in the market for air quality monitoring is PurpleAir. It offers 3 air quality Arduino based sensors

- a) PurpleAir PA-II
- b) PurpleAir PA-II-SD and
- c) PurpleAir PA-I-Indoor.

These sensors are often used for residential, commercial, or industrial use. PurpleAir sensors include a BME280 pressure, temperature, and humidity sensor powered by 5v USB power source along with WiFi

and cloud storage capabilities. It uses HighCharts for generating maps and Thinkspeak for cloud storage. It also provides a map interface to visualize the various devices in use around the globe [26].

Some of the features provided by PurpleAir are:

- PurpleAir uses PMS5003 and PMS1003 laser particle counters
- ESP8266 chip and Arduino with remote firmware updates
- Thinkspeak and Highcharts for cloud and graph purposes
- Mapbox for Map interface. Figure 1 shows a map for PurpleAir reflecting all the devices around the globe.
- BME280 pressure, temperature and humidity sensors
- 5v USB power supply

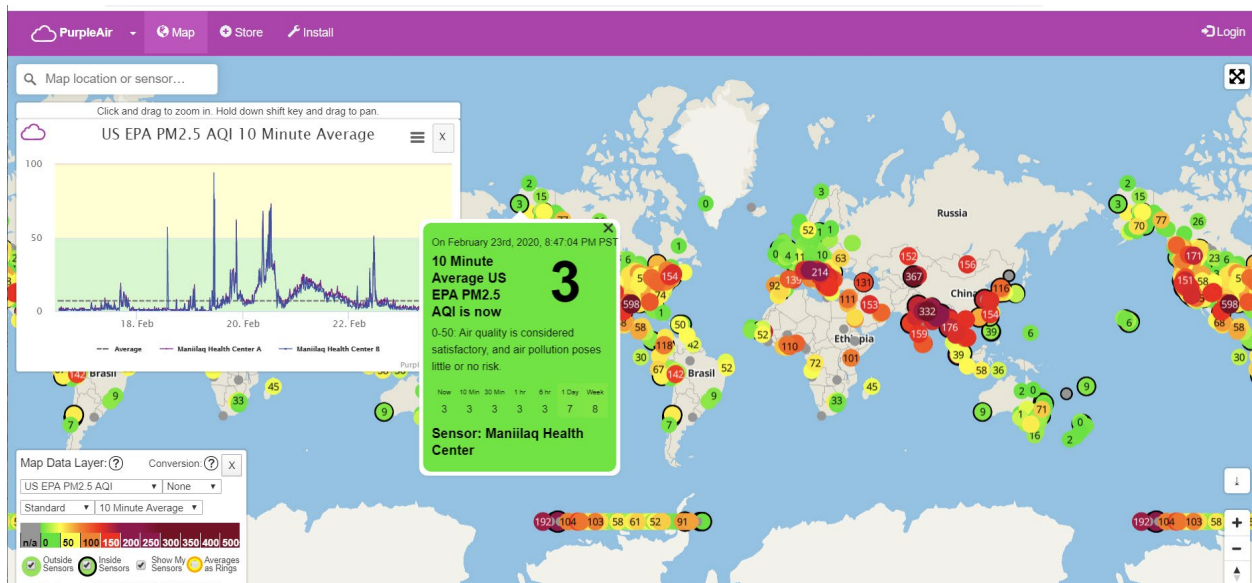


Figure 1. PurpleAir map reflecting all the devices around the globe [27]

Table 1 below shows the cost comparison between **STM32L475 board** with PurpleAir and AeroQual products. Although Aeroqual has a wide range of products the table shows only the cost of the relevant indoor products and not for all. Aeroqual products are costly because of the wide number of sensors and other features that it offers

Cost Comparison			
	Product	Cost(per unit)	Description
PurpleAir	PurpleAir PA-II-SD	\$259.00	Mostly used for residential, commercial or industrial use. Comes with storage capacity and limited Wi-Fi access
	PurpleAir PA-II	\$229.00	Mostly used for residential, commercial or industrial use and equipped with Wi-Fi and laser particle sensors.
	PurpleAir PA-I-Indoor	\$179.00	A smaller device suitable for home or office use and direct Wi-Fi access to PurpleAir quality maps.
Aeroqual	PM10 / PM2.5 Portable Particulate Monitor (series 200,300,500)	\$1500 - \$ 1750 range	Mostly suitable for Indoor use and offers a wide range of features and sensors
STM32L475 discovery Kit board	B-L475E-IOT01A1	\$52 + \$5(AWS cloud storage cost)	The board is equipped with temperature, humidity, etc. sensors and Wi-Fi connectivity

Table 1. Cost comparison of PurpleAir and Aeroqual products with the STM32 board

2.3 AirSense

AirSense is a mobile application developed to monitor the indoor air pollutants of a residence and alert the users of the pollution levels and also predicts the future air pollutant levels based on the gathered data [22]. Figure 2. Shows the Airsense Smartphone application Screenshots

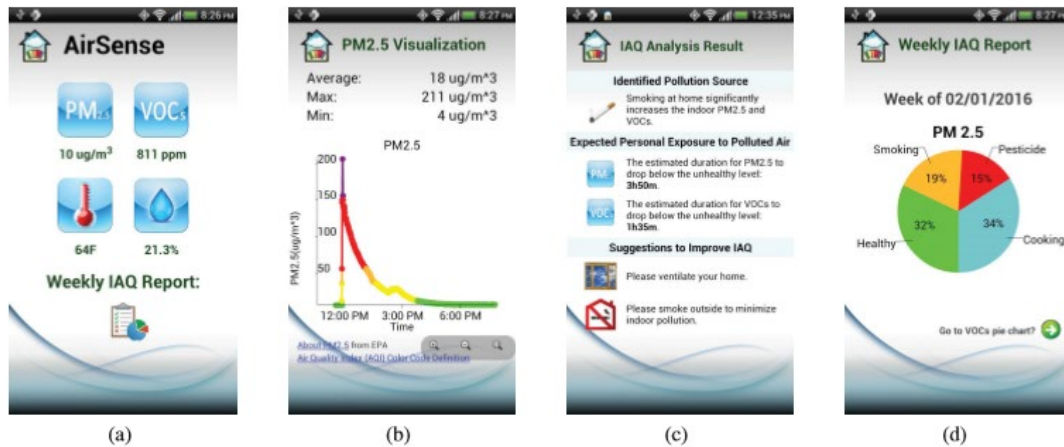


Figure 2. AirSense Smartphone Application [22]

One of the hottest topics in the IoT field researched is related to making all connected devices smarter, by computing relevant information locally and integrating data received from other sensors over a local network [13]. Also reducing the environmental footprint by efficiently reducing the energy consumption of the smart devices without affecting the user experience. The paper "Adaptive and Personalized Energy Saving Suggestions for Occupants in Smart Buildings." suggests the same [23].

Similarly, as described in the article "Indoor Temperature prediction in an IoT scenario", it focuses on the feasibility of a home refrigerator forecasting temperature, using the information provided by both external and internal sensors. This scenario is reviewed for its potential applications [12].

This master project goes beyond the above prototype by creating a distributed system with multiple devices connected to the cloud and the addition of more features to it.

2.4 IoT based Weather Station

A weather station is nothing but a device, which constantly provides the details of the weather in our surrounding environment. For example, it can provide us with details about the neighboring temperature, humidity, barometric pressure, etc. In order to extend this application further, it can be used to monitor the humidity or temperature of a particular place/room. The author has also talked about using a sensor to monitor the atmospheric pressure of a particular room. This sensor can also monitor the rain value [3,9].

The proposed system is developed only using one board and four sensor boards mounted on it along with the Wi-Fi shield. I have further extended the proposed design by connecting multiple devices to the cloud

and visualizing the data on the map. Also, sending the location information in the notification to the user. In addition, the board that I am using has embedded Wi-Fi and sensors which need to be configured with AWS cloud to store the gathered data. This collected data can be further used by environmental professionals for study and monitoring.

2.5 Mobile sensor unit for online air quality monitoring

The article “mobile sensor unit for online air quality monitoring” talks about a comprehensive system of air pollution measuring mobile sensor units in urban areas. The system is equipped with a mobile unit containing small electrochemical sensors and a Low Energy communication Bluetooth module. This Pollution Gathering application is developed for receiving processing data from the unit on a mobile device. It uses a smartphone's data connectivity to connect to a server in order to send the data. The server persists the data in the environmental database and provides a restful API interface [2]. The application users and public authorities can view data on the map. The mobile sensor network is capable of monitoring a very large area for air quality [6].

The above prototype is built for a mobile app (Android only) using the mobile device's sensor and W-Fi to get the sensor data. To get to the environment parameters one must be physically present at the location. In my project, this dependency has been removed by using the stationary devices connected at a particular location and sending data through its in-built Wi-Fi. This way, a user can get the exact temperature and humidity sensor values at that time at a particular location beforehand. This will help people for example cyclists or hikers to choose a location preferable to them with suitable conditions.

3. Background

Before we jump into the actual implementation of the project, it is important to have an overview of some of the services and keywords used to build it.

3.1 What is a Microcontroller?

A microcontroller (MCU) is a small chip with a simple process that can be found in many devices, including sensors, appliances, fitness trackers, automobiles, and industrial automation. Many of such devices could benefit from cloud connectivity to other devices. For example, smart electricity meters will have to connect to the cloud for usage reporting and building security systems have to communicate locally in order to unlock the door when you badge in [11]. MCU's compute power and memory capacity is limited so that it can perform simple, easy functional tasks [10]. Microcontrollers usually run operating systems that do not have in-built functionality to connect the cloud or to local networks, challenging IoT applications. Amazon FreeRTOS solves this problem by providing both the software libraries and core operating system (to run the edge device) which makes it easy to connect securely to the cloud (or other different edge devices) so data can be collected from them for IoT applications in order to take action. Figure 3 shows the MCU ecosystem.

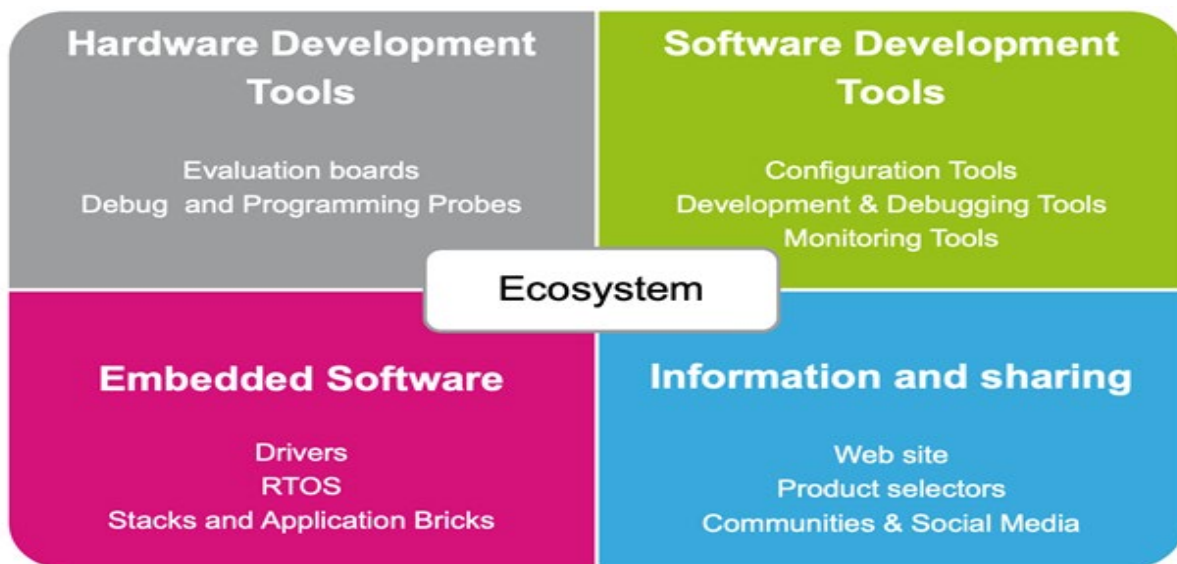


Figure 3. MCU Ecosystem [28]

3.2 Overview of STM32L475 Discovery Kit

The STM32L475 Discovery kit IoT node enables users to develop cloud applications supporting direct connection to servers running on AWS infrastructures. The Discovery kit enables various applications depending on low-power communication and multiway sensing and M4-core based series features [4]. While the board has a plethora of features, the features that are pertinent to this project are the B-L475E-IOT01A1 sensor which is the heart of the project and the wireless connectivity using Wi-Fi.

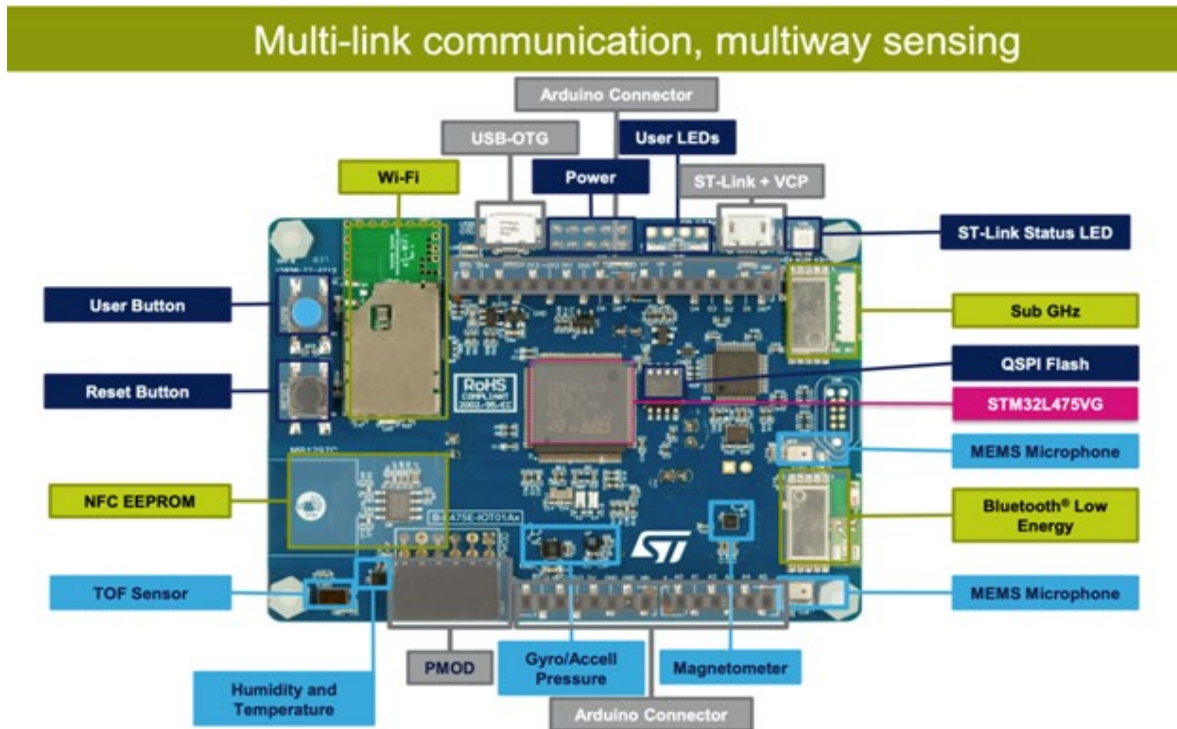


Figure 4. STM32 IoT Discovery kit with Sensor placement and Features [24]

Figure 4 shows the sensor placement on the board along with the buttons and other hardware parts.

3.2.1 Key Features

As shown in Figure 4 the STM32 provides rich set of features as follows

- Ultra-low power STM32L4 Series MCUs.
- 64 Mb Quad-SPI Flash memory
- Bluetooth v4.1 module
- Sub GHz (868 or 915 MHz) low-power-programmable RF module
- 802.11 b/g/n compliant Wi-Fi module.

-
- M24SR based Dynamic NFC
 - Digital one directional microphone
 - Digital sensor for humidity and temperature
 - Three-axis magnetometers
 - 3D gyroscope and accelerometer
 - Digital barometer
 - Gesture-detection and time of flight sensor
 - 2 push buttons (reset, user)
 - USB Micro-AB connector
 - Expansion connectors
 - Power supply options: ST-LINK USB V Bus
 - On-board ST-LINK/V2-1 debugger and programmer
 - Free software HAL comprehensive library
 - Various IDEs including IAR™, Keil®, Arm® Mbed Enabled™
 - Arm Mbed

3.3 Amazon FreeRTOS

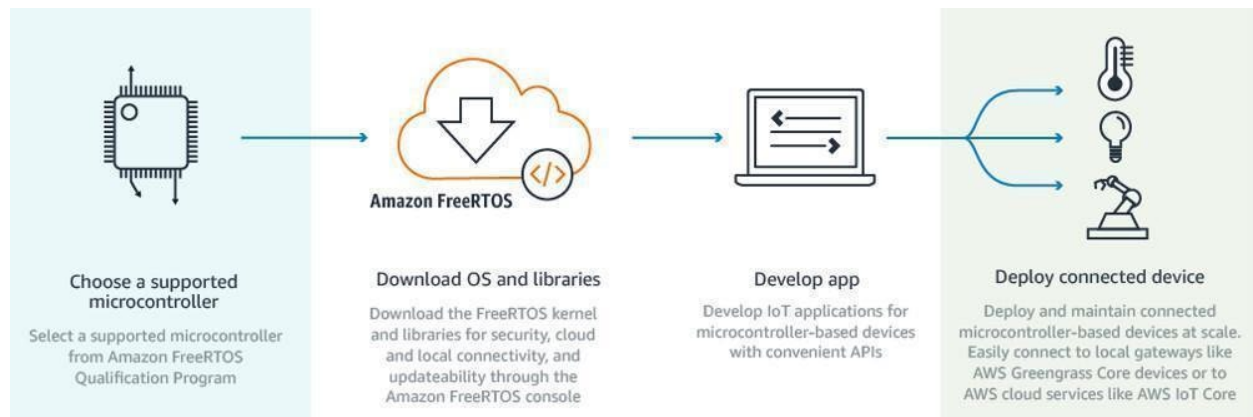


Figure 5. Amazon FreeRTOS High-level Architecture [29]

Amazon FreeRTOS is an open-source OS used for microcontrollers that enable low-power and small edge devices to connect to AWS cloud making it easy to deploy, secure, program and manage [1]. Amazon FreeRTOS is based on FreeRTOS kernel, an open-source OS used for microcontrollers, with many software libraries making it easy to securely connect small, low power devices to AWS cloud services like AWS IoT Core or AWS IoT Greengrass over Wi-Fi.

Figure 5 above Demonstrates the high-level steps of connecting your Microcontroller board to AWS using FreeRTOS.

3.4 AWS IOT Things

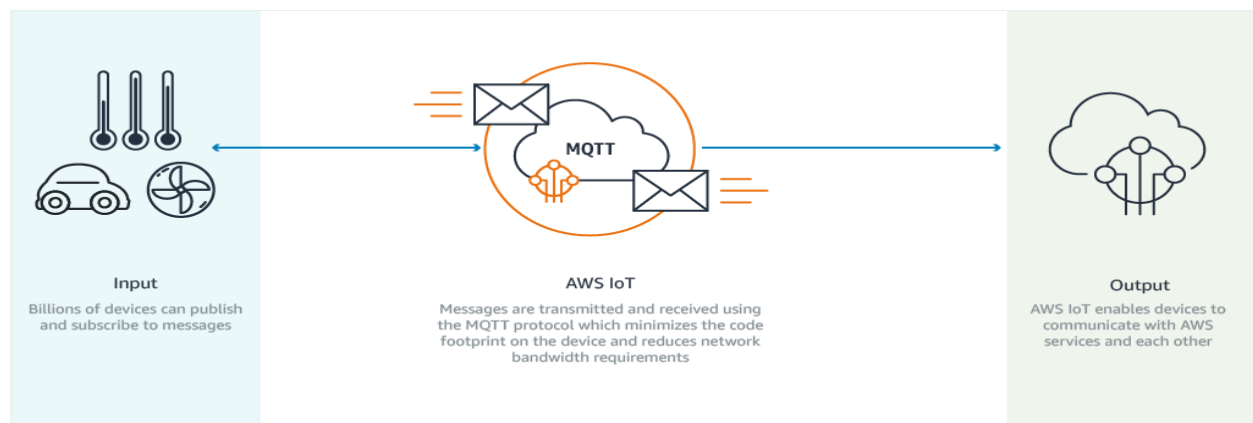


Figure 6. AWS IoT High-level Architecture [30]

AWS IoT offers a service that easily allows us to visually connect web services and different devices to develop IoT applications. Figure 6 above depicts the high level overview of AWS IoT.

More and more IoT applications are being developed today using a variety of web services and devices in order to automate tasks for a wide range of use cases, such as energy management, industrial automation, and smart homes [8]. Since there are not any standards that are widely adopted, developers find it difficult today to get generic devices from many manufacturers to connect to each other and also connect to web services. This practice forces many developers to write extra code to wire together all of the web services and devices which they need for their IoT application. AWS IoT Things provides a visual drag and drop interface for coordinating and connecting interactions between web services and devices, so you can develop IoT applications very quickly [20]. For example, in many commercial agriculture applications, you can define interactions between temperature, humidity, and sensors responsible for sprinklers with weather data web services in the cloud to automate watering. You represent services and devices using reusable pre-built components, known as models, which hides low-level details, such as interfaces and protocols, and easy integration to create sophisticated workflows.

3.5 AWS DynamoDB (NoSql Database)

Amazon DynamoDB is a key-value and document database that is highly scalable. It's a fully managed, durable database that is multiregional and multimaster, with in-built support for backup, security, and restoration. Additionally, it supports in-memory caching for internet-scale applications. It can handle more than 1 trillion requests per day and can support more than 20 million requests per second during peak time [19].

Many of the world's popular businesses such as Lyft, Amazon, and Uber as well as enterprises such as Apple, Facebook depend on the performance and scale of DynamoDB to support their mission-critical workloads as their key-value and document database for web, mobile, gaming, IoT, and other applications that need data access at any scale with low latency [20].

4. Project Architecture, Implementation and Results

4.1 High Level Architecture

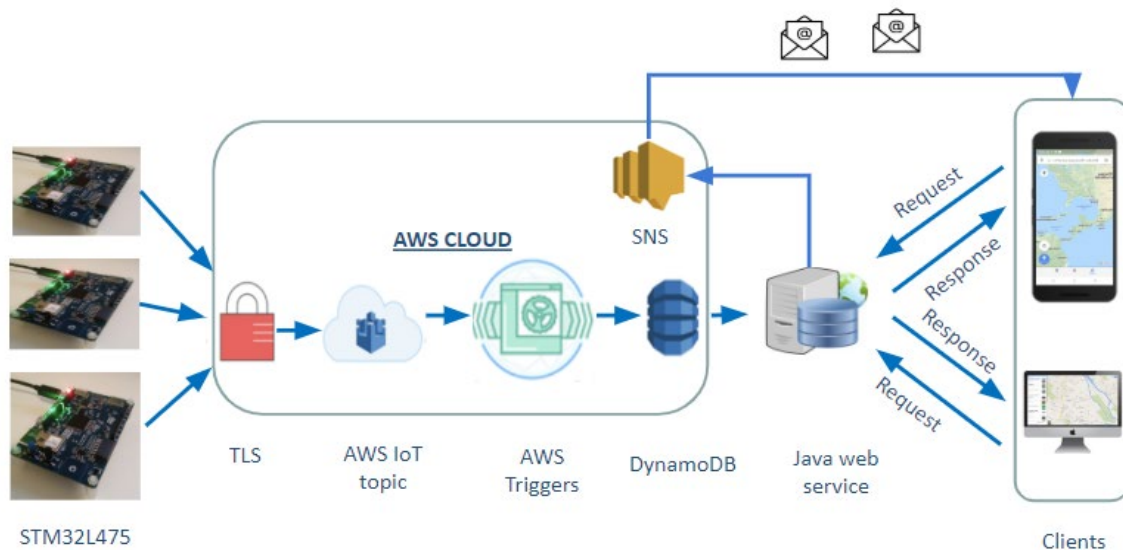


Figure 7. High-level architecture diagram of the project

Figure 7 exhibits the end to end flow of this application involving all the components required to design and develop this IoT based cloud application.

The basic technical flow of the application is as follows:

- The boards are connected to AWS cloud over Wi-Fi.
- The sensor data payload is sent to AWS IoT MQTT topic
- This MQTT topic data is then sent to DynamoDB for persistent storage and data manipulation.
- The triggered AWS Service executes an action by writing received sensor data into Dynamodb
- The device state can be read, stored and set with Device Shadows
- The Spring boot RESTful service provides a rest endpoint for communication with the database.
- The client makes a call to the rest endpoint and displays the data on google maps.

Figure 8 (Flow chart) below elaborates the High level architecture (showed in figure 7) to represent the internal data flow of the project.

4.2 Flow chart

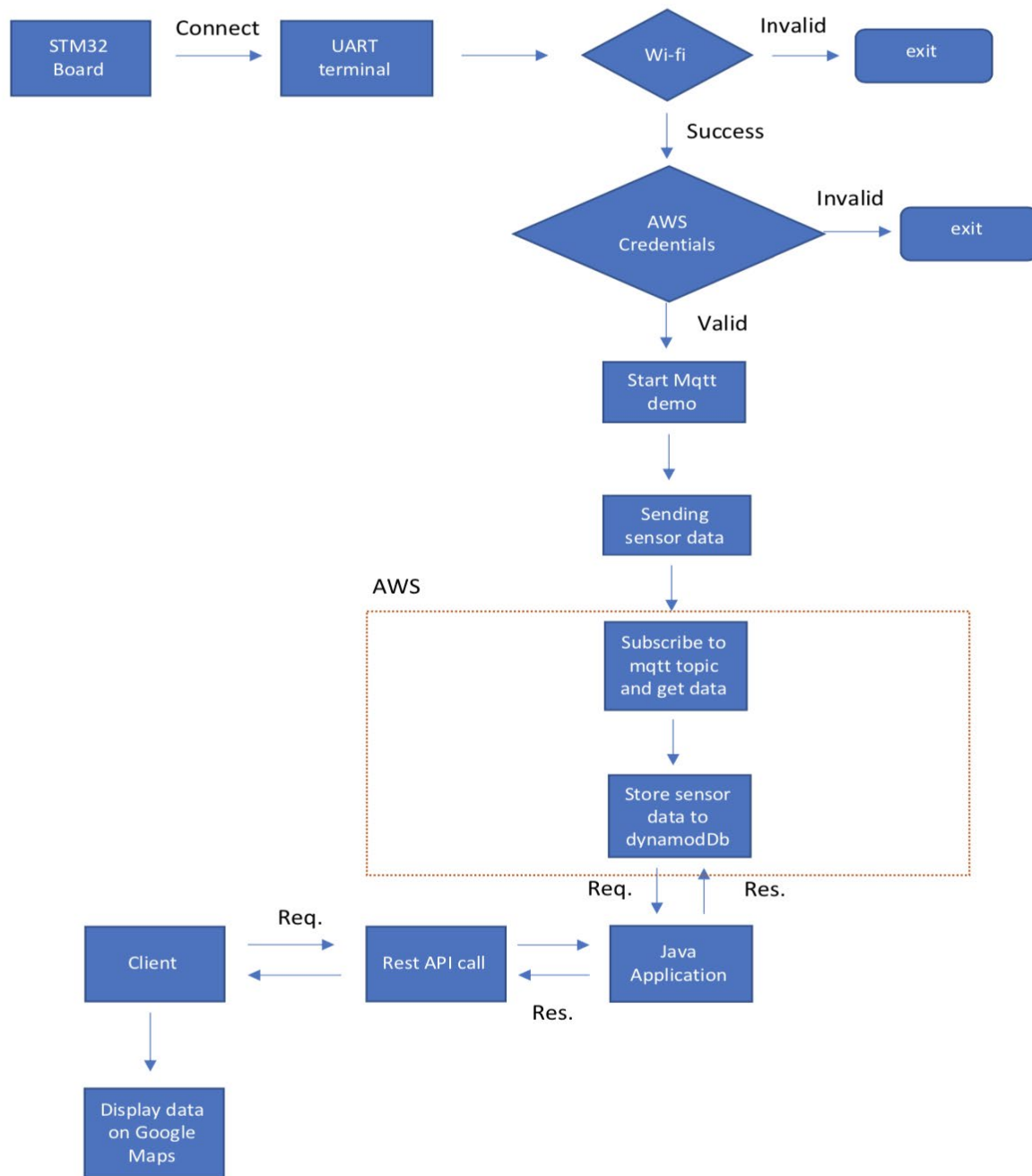


Figure 8. Flow chart

Figure 8 shows the flow of the data in the project and the possible exception scenarios. For instance, if the Wi-Fi connectivity is not established then the application halts. Similar goes for the AWS credentials.

The board is connected to the system through the **UART terminal**. The **Wifi credentials** are part of the code and verified at the time of initialization. The Wifi connection makes sure the sensor data is sent to the cloud (AWS). Then a MQTT connection is established and the sensor data is published to the topic.

On the **AWS side**, we need to **subscribe to the MQTT topic**. Once subscribed to the topic, the connection is active as long as we don't disconnect the board. At the same time the data is **stored on DynamoDB** for persistent storage and map visualization. This step also does not require manual intervention as it is a rule which will be automatically triggered.

The **Java application** is a spring boot web application which has RESTful web services used to request and receive the data requested by **Client** (in the project browser is the client).

Each step of the flow chart is explained in detail with screenshots in the following subsections.

Table 2 shows the Software technologies and tools used in this project.

OS	Windows 10 (64 bit)/ Linux
Programming Languages	Java8, C, Javascript
Framework	Spring boot MVC
Cloud technologies	AWS IoT, SNS, Triggers
Database	DynamoDB
IDE and tools	Intellij, TeraTerm, Maven, Git, System Workbench for STM32

Table 2. Software requirements

4.3 Board configuration

I have built and deployed the code snippet below onto the STM32 board as an executable. The code snippet below shows the part of the code which reads the sensor data and sends it to AWS topic "freertos-mqtt-temp" via MQTT.

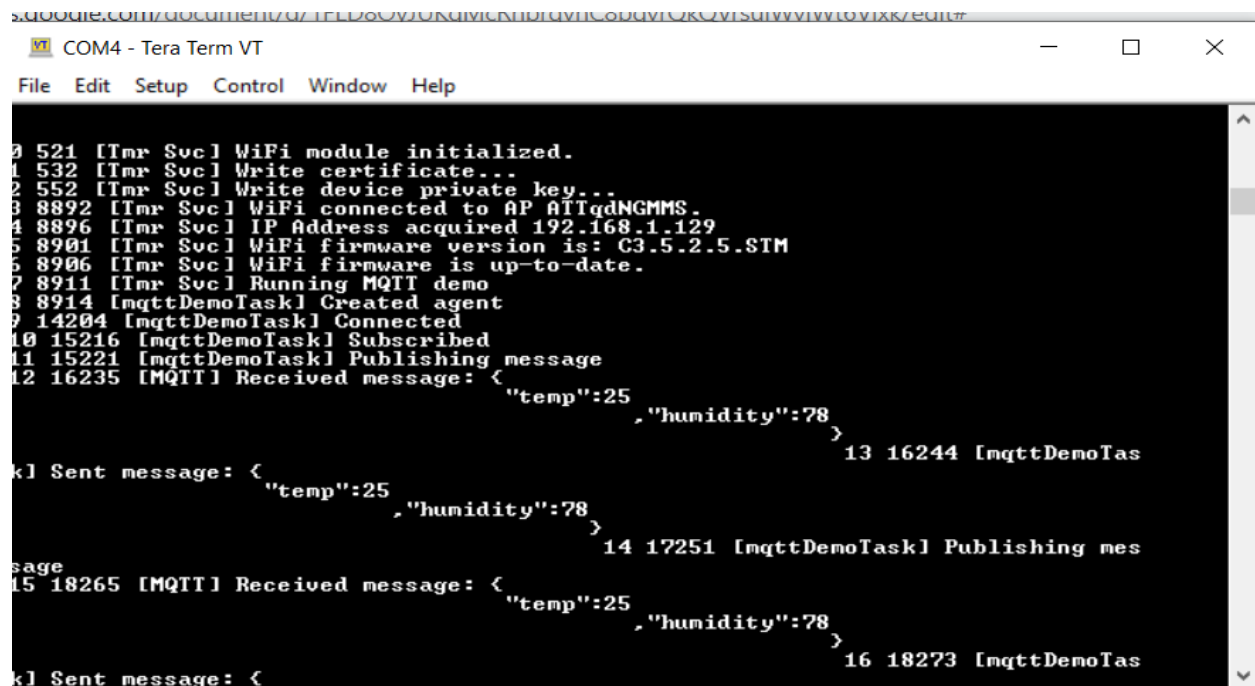
```

MQTT_AGENT_Subscribe(.xMQTTHandle,
→ . → → → → .....&xSubscribeParams,
→ → → → → → ..pdMS_TO_TICKS(.2500.));
↵
....configPRINTF("Subscribed\r\n");
↵
→ for.(int.i=.0;.i.<.100;.i++).{
→ ....char.cDataBuffer[.mqtMAX_DATA_LENGTH.];
↵
→ ....float.temp.=.BSP_TSENSOR_ReadTemp();
→ ....float.humidity.=.BSP_HSENSOR_ReadHumidity();
→ ....int.deviceId.=.2;
↵
→ ....sprintf(cDataBuffer,.mqtMAX_DATA_LENGTH,
"{\n\"deviceId\":%d\n,\"temp\":%d\n,\"humidity\":%d\n}",deviceId,(int)temp,
(int)humidity);
↵
// → ....sprintf(cDataBuffer,.mqtMAX_DATA_LENGTH,
"{\n\"temp\":%d\n,\"humidity\":%d\n}",(int)temp,(int)humidity);
↵
↵
→ ....MQTTAgentPublishParams_t.xPublishParameters;
→ ....memset(&(xPublishParameters.),0x00,sizeof(xPublishParameters.
).);
→ ....xPublishParameters.pucTopic.=.mqtTOPIC_NAME;
→ ....xPublishParameters.usTopicLength.=.(uint16_t).strlen(.(const.
char.*).mqtTOPIC_NAME.);
→ ....xPublishParameters.pvData.=.cDataBuffer;
→ ....xPublishParameters.ulDataLength.=.(uint32_t).strlen(
cDataBuffer.);
→ ....xPublishParameters.xQoS.=.eMQTTQoS1;
↵
→ ....configPRINTF("Publishing message\r\n");
↵
→ ....MQTT_AGENT_Publish(.xMQTTHandle,
.....&(xPublishParameters.),
→ → → → → → pdMS_TO_TICKS(.2500.));
↵
→ ....configPRINTF("Sent message: %s\n",cDataBuffer);
↵
→ ....vTaskDelay(.pdMS_TO_TICKS(.1000UL.));
→ }

```

4.4 Setup a serial communication

In order to connect the board to a Desktop/Laptop, TeraTerm is installed in it for serial communication. Once the configuration is done, the board connects to Wi-Fi and AWS starts sending sensor data via MQTT to the cloud (AWS). Figure 9 below shows the temperature and humidity sensor data that is published to the MQTT successfully.



```
COM4 - Tera Term VT
File Edit Setup Control Window Help

0 521 [Tmr Svc] WiFi module initialized.
1 532 [Tmr Svc] Write certificate...
2 552 [Tmr Svc] Write device private key...
3 8892 [Tmr Svc] WiFi connected to AP AllQdNGMMS.
4 8896 [Tmr Svc] IP Address acquired 192.168.1.129
5 8901 [Tmr Svc] WiFi firmware version is: C3.5.2.5.STM
6 8906 [Tmr Svc] WiFi firmware is up-to-date.
7 8911 [Tmr Svc] Running MQTT demo
8 8914 [mqttDemoTask] Created agent
9 14204 [mqttDemoTask] Connected
10 15216 [mqttDemoTask] Subscribed
11 15221 [mqttDemoTask] Publishing message
12 16235 [MQTT] Received message: {
    "temp":25
    "humidity":78
  }
13 16244 [mqttDemoTask] Sent message: {
    "temp":25
    "humidity":78
  }
14 17251 [mqttDemoTask] Publishing message
15 18265 [MQTT] Received message: {
    "temp":25
    "humidity":78
  }
16 18273 [mqttDemoTask] Sent message: {
```

Figure 9. Screenshot of serial terminal showing successful connection to MQTT and sensor data

4.5 Verify the data on AWS

The screenshot shows the AWS IoT console's 'Subscriptions' page. On the left is a navigation menu with options: Monitor, Onboard, Manage, Greengrass, Secure, Defend, Act, **Test** (highlighted), Software, Settings, and Learn. The main content area has a blue header 'Subscriptions' and two links: 'Subscribe to a topic' and 'Publish to a topic'. The 'Subscribe' section contains a description: 'Subscribe. Devices publish MQTT messages on topics. You can use this client to subscribe to a topic and receive these messages.' Below this is a 'Subscription topic' input field with the text 'freertos/mqtt/temp' and a 'Subscribe to topic' button. Further down are settings for 'Max message capture' (set to 100), 'Quality of Service' (set to 0 with a description: '0 - This client will not acknowledge to the Device Gateway that messages are received'), and 'MQTT payload display' (set to 'Auto-format JSON payloads (improves readability)').

Figure 10. Subscribe to MQTT topic on AWS

After subscribing to the topic (as shown in Figure 10) the sensor data is now on AWS MQTT.

The screenshot shows the AWS IoT console with the 'Test' tab selected. At the top, a terminal window displays a JSON message:

```
1 {
2   "message": "Hello from AWS IoT console"
3 }
```

. Below the terminal is a table of received messages. The table has columns for the topic name, the timestamp, and actions (Export, Hide). The data rows show the topic 'freertos/mqtt/temp' and timestamps from Jan 21, 2020 1:15:47 PM -0800. The JSON payloads are:

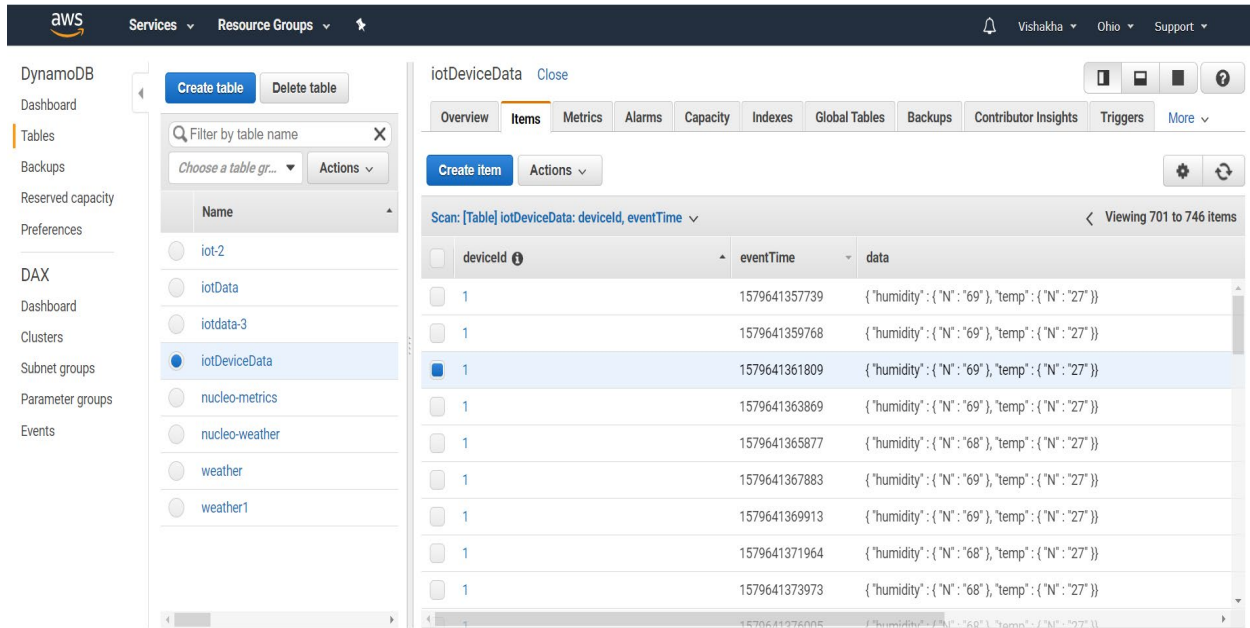
```
{
  "temp": 27,
  "humidity": 69
}
```

Topic	Time	Message	Actions
freertos/mqtt/temp	Jan 21, 2020 1:15:47 PM -0800	<pre>{ "temp": 27, "humidity": 69 }</pre>	Export Hide
freertos/mqtt/temp	Jan 21, 2020 1:15:47 PM -0800	<pre>{ "temp": 27, "humidity": 69 }</pre>	Export Hide
freertos/mqtt/temp	Jan 21, 2020 1:15:43 PM -0800	<pre>{</pre>	Export Hide

Figure 11. Sensor data sent from board to subscribed topic on AWS

Figure 11 above shows the JSON data received from the terminal along with the time it was received. Now, this data needs to be stored persistently for visualization purposes. Hence, it is stored in DynamoDB along with the timestamp as the sort key.

The Primary key is the device Id and the sort key is the Timestamp in Epoch time. Figure 12 below shows the table data where the highlighted row shows device Id = 1, the timestamp and the temperature, humidity sensor data in JSON format under the data column of the table.



The screenshot shows the AWS DynamoDB console interface. On the left, the 'DynamoDB' sidebar is visible with options like 'Create table', 'Delete table', and 'Filter by table name'. The main panel displays the 'iotDeviceData' table. The table has a primary key 'deviceId' and a sort key 'eventTime'. The data is displayed in a table with columns: deviceId, eventTime, and data. The data is in JSON format, showing humidity and temperature for device 1.

deviceId	eventTime	data
1	1579641357739	{ "humidity": { "N": "69" }, "temp": { "N": "27" } }
1	1579641359768	{ "humidity": { "N": "69" }, "temp": { "N": "27" } }
1	1579641361809	{ "humidity": { "N": "69" }, "temp": { "N": "27" } }
1	1579641363869	{ "humidity": { "N": "69" }, "temp": { "N": "27" } }
1	1579641365877	{ "humidity": { "N": "68" }, "temp": { "N": "27" } }
1	1579641367883	{ "humidity": { "N": "69" }, "temp": { "N": "27" } }
1	1579641369913	{ "humidity": { "N": "69" }, "temp": { "N": "27" } }
1	1579641371964	{ "humidity": { "N": "68" }, "temp": { "N": "27" } }
1	1579641373973	{ "humidity": { "N": "68" }, "temp": { "N": "27" } }

Figure 12. Screenshot of the DynamoDB table data

4.6 Code snippet to Query the DynamoDB table

Below is a part of the Spring boot application code which will be running in the background every time a GET request is made from the client (the browser) to fetch the sensor data on google map.

```

public List<AvgHumidity> getDeviceDataByDate(String startDate, String endDate) throws
ParseException {
    try {
        DateFormat utcFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        Date date1 = utcFormat.parse(startDate);
        Date date2 = utcFormat.parse(endDate);
        long l1 = date1.getTime();
        long l2 = date2.getTime();
        List<IoTData> result = new ArrayList<>();
        Map<String, AttributeValue> eav = new HashMap<>();
        eav.put(":val1", new AttributeValue().withS("1"));
        eav.put(":val2", new AttributeValue().withS(Long.toString(l1)));
        eav.put(":val3", new AttributeValue().withS(Long.toString(l2)));
        DynamoDBQueryExpression<IoTData> queryExpression = new
        DynamoDBQueryExpression<IoTData>().withKeyConditionExpression("deviceId = :val1 and eventTime
        between :val2 and :val3").withExpressionAttributeValues(eav);
        List<IoTData> betweenDevice1Data = mapper.query(iotData.class, queryExpression);
        if(betweenDevice1Data != null){
            result.addAll(betweenDevice1Data);
        }
        LinkedHashMap<LocalDate, AvgDate> heatMapData = new LinkedHashMap<>();
        for(IoTData a : result) {
            long dateTime = Long.parseLong(a.getEventTime());
            LocalDate date =
            Instant.ofEpochMilli(dateTime).atZone(ZoneId.systemDefault()).toLocalDate();
            if(date.getYear() < 2019){
                System.out.println(date);
            }
            if (heatMapData.containsKey(date)) {
                AvgDate avgDate = heatMapData.get(date);
                if(a.getData()!=null) {
                    avgDate.tempTotal = avgDate.tempTotal + a.getData().getTemp();
                    avgDate.tempCount = avgDate.tempCount + 1;
                    avgDate.humidityCount = avgDate.humidityCount + 1;
                    avgDate.humidityTotal = avgDate.humidityTotal + a.getData().getHumidity();
                }
            } else {
                AvgDate avgDate = new AvgDate(a.getData().getTemp(), 1,
                a.getData().getHumidity(),1);
                heatMapData.put(date, avgDate);
            }
        }
        List<AvgHumidity> list = new ArrayList<>();
        heatMapData.forEach((K,V)-> list.add(new
        AvgHumidity(K.toString(),Double.toString(heatMapData.get(K).calculateTempAvg()),
        Double.toString(heatMapData.get(K).calculateHumidityAvg()))));
        return list;
    } catch (Exception ex){
        System.out.println("logs" + ex.toString());
    }
    return null;
}

```

4.7 Client-Side programming code snippet

The frontend part of the project is done using Javascript. Google map API's are used to get the map and plot the location points of the device. The google map API provides a wide range of functions for producing heat maps[15]. Below code shows the javascript code which calls the API endpoint for fetching the data from DynamoDB. The data is then filtered to get the latest Sensor data based on time and device Id.

```

// the below code gets the JSON data from dynamoDB
var getJSON = function(url, callback) {
    var xhr = new XMLHttpRequest();
    xhr.open('GET', url, true);
    xhr.responseType = 'json';
    xhr.onload = function() {
        var status = xhr.status;
        if (status == 200) {
            callback(null, xhr.response);
        } else {
            callback(status);
        }
    };
    xhr.send();
};

// getJSON function(url, callback())
getJSON('http://localhost:8881/dynamoDB/alldevice', function(err, data) {

    if (err != null) {
        console.error(err);
    } else {
        getJsonData(data);
    }
});
}

// the getJsonData() splits each received record and stores it in variables
function getJsonData(jsonObject){
    a = getPoints();
    for(var i=0;i<2;i++){
        var temperature = 0;
        var humidity = 0;
        var time;
        var deviceData = [];
        var location ;
        if( jsonObject.length!=0 ){
            deviceData = jsonObject.filter(function(entry){
                return entry.deviceId == i+1
            });
            temperature = deviceData[deviceData.length-1].data.temp.toString();
            // the if condition decides the gradient color of the data points on the map
            if(temperature < "32"){
                heatMapData2[i] = {location: a[i]}
            }
            else{
                heatMapData[i] = {location: a[i]}
            }
            if(temperature < lowTemp){
                lowTemp = temperature;
                alertStringTemp = dict[i+1];
            }
            humidity = deviceData[deviceData.length-1].data.humidity.toString();
            if(humidity < lowHumidity){
                lowHumidity = humidity;
                alertStringHumidity = dict[i+1];
            }
            location = dict[i+1];
            var utcSeconds = deviceData[deviceData.length-1].eventTime;
            time = new Date(0); // The 0 there is the key, which sets the date to the epoch
            time.setUTCMinutes(utcSeconds); }

```

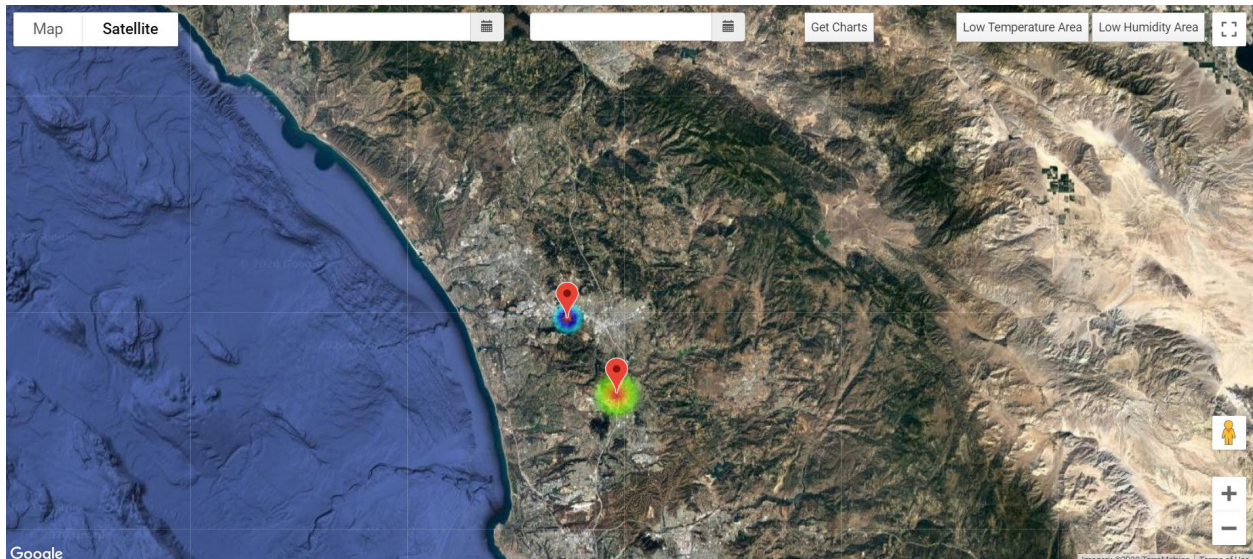


Figure 13. Screenshot of the Map with data points in satellite view

Figure 13 shows two data points with different colors. This is because of the gradient changes based on the temperature value at that location. In the top left corner, there is a button to toggle between the satellite and the map view as required.

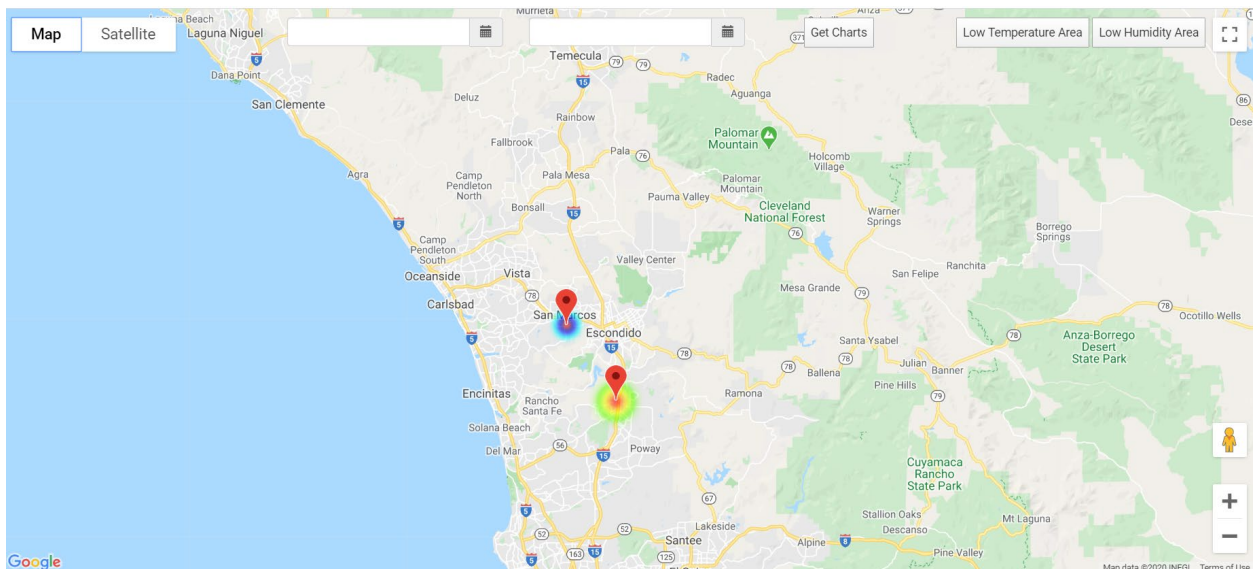


Figure 14. Screenshot of the Map with data points in map view

Figure 14 shows the Map view of the application. When clicked on a certain location on the graph it shows the temperature and humidity of that place at that time along with the location name.

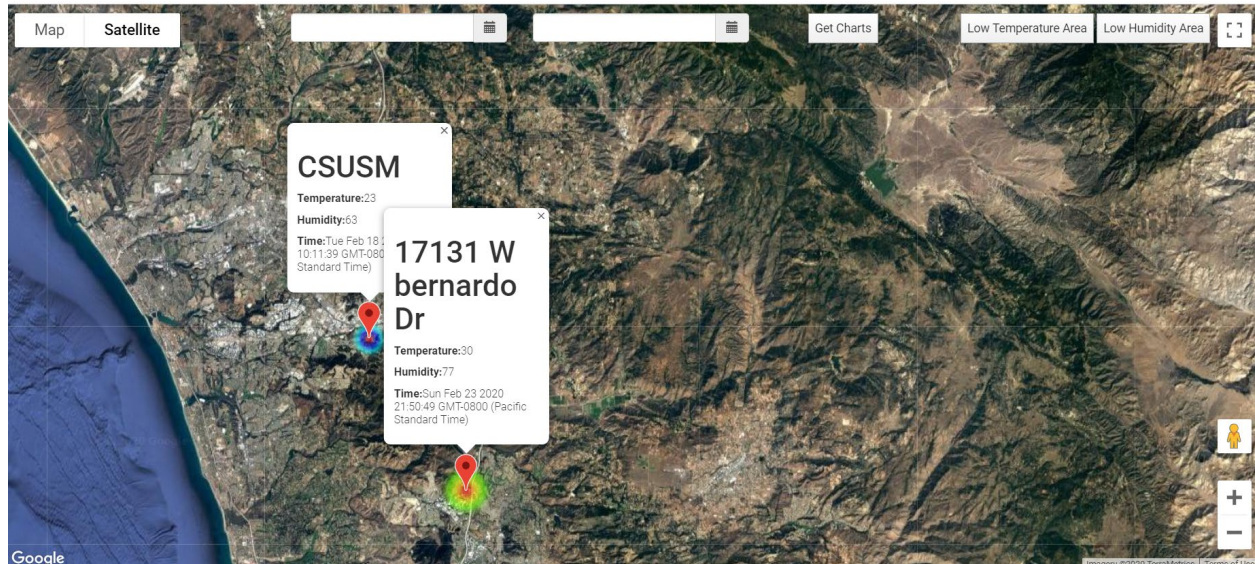


Figure 15. Screenshot of the Map with Data information on that location

Figure 15 shows the data information on both the data points (locations).

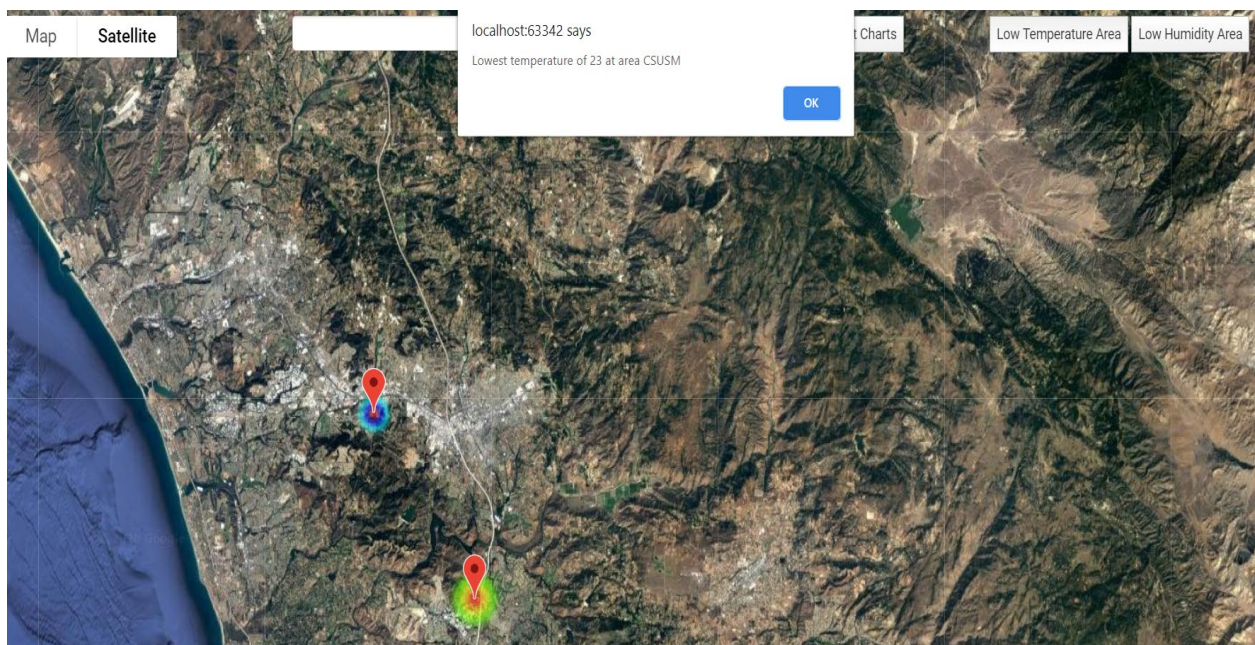


Figure 16. Screenshot of the map showing popup for lowest temperature value

The map also contains 2 buttons on the top right corner named “Low-Temperature Area” and “Low Humidity Area”. The purpose of these buttons is to give the location name with the lowest temperature/humidity, so the user does not have to click on all the data points to find the lowest sensor values. Figure 16 shows the same.

Temperature chart

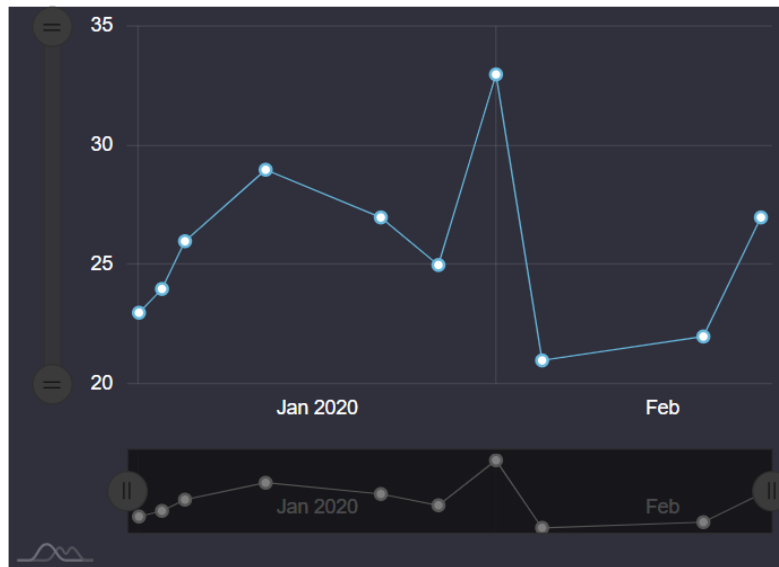


Figure 17. Temperature Chart

On the top center of the map, there is a button to get charts for the selected date range. Users need to select both the start and end date to get the charts. Figure 17 shows the temperature chart and figure 18 below shows Humidity chart. Also, when hovering over on the data points on the chart it will show the Temperature/Humidity values along with the date.

Humidity chart

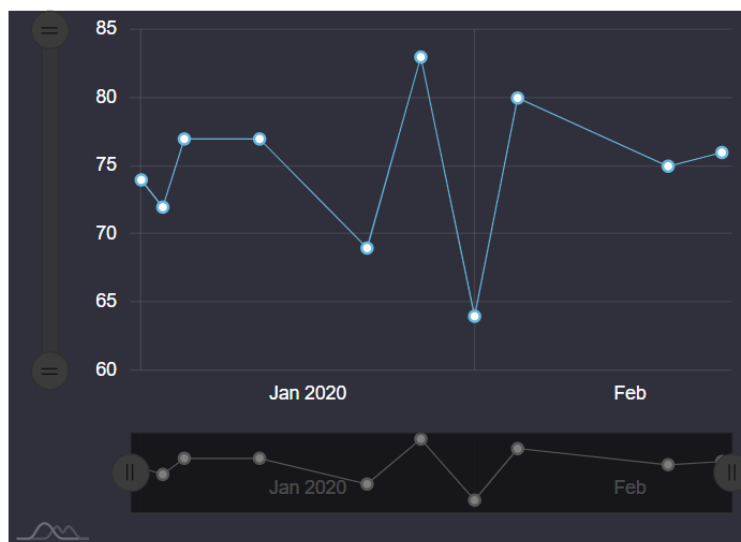


Figure 18. Humidity chart

If the temperature ever reaches a given threshold value, then the user receives an email notification stating that the temperature has reached above threshold. Figure 19 shows the screenshot of the email notification.

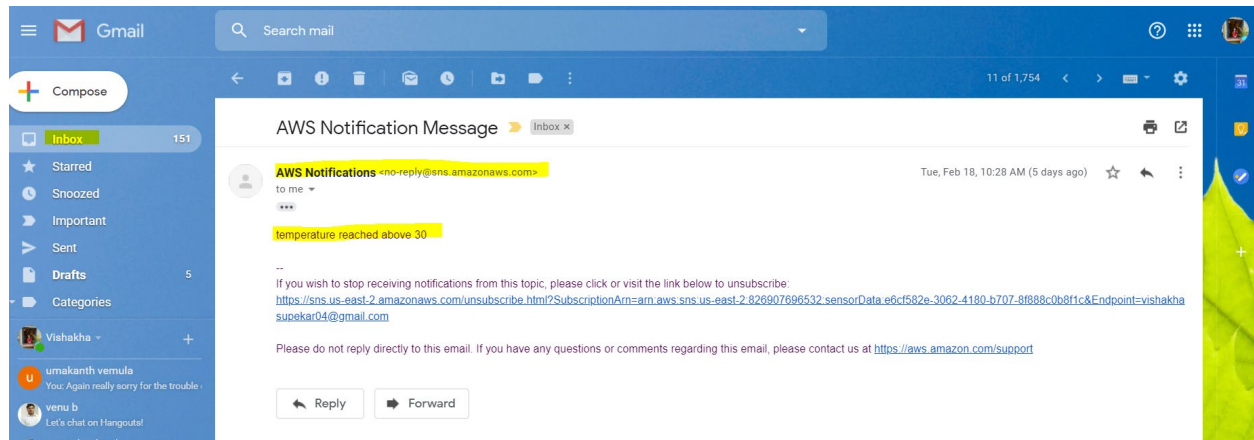


Figure 19. Email notification screenshot

5. Applications

The project offers features such as email/SMS notification. Heat map generation and persistence data storage of sensor values which can be manipulated in a number of ways. Based on the implementation of the project can be used as a prototype to build different projects/systems or can be used in tandem with existing projects to enhance its functionality. The following are some of the proposed applications of the project.

5.1 Regulating the Air Conditioner (AC) temperature of a room

The device (STM32L475 board) can be placed in different rooms around the college campus. These devices will send the sensor data of every room to the cloud. The stored data can be monitored and based on the time of the day the average temperature of the room can be calculated. These average temperature readings can then be auto configured with the AC unit to set the temperature as per the average temperature at that time [7]. Suppose, after monitoring the data after a few days it was found that at 12:30 pm, the temperature in room1 increases to 35 degrees. In this case, we need to set the AC temperature manually for cooler settings. This manual process can be auto set in the AC unit when a notification from the device is sent to it.

Also, the map will help to identify the positions of the devices in the room and would provide a heat map for reference.

5.2 Monitoring humidity in parks

The sensor devices can be placed at different parks and the parks' humidity or air pollution level can be monitored. With the help of the map, one can find the park with less humidity and temperature to enjoy a relaxed time. The web application also provides a feature to find out a place with the lowest humidity and temperature.

It has been observed that high humidity or temperature has adverse effects on the health of patients with respiratory diseases [18]. So, such people can use the application to find a park that has favorable environmental conditions for exercise, cycling or just for a lazy stroll.

5.3 Smart Farming

Nowadays, the concept of smart cities is not just related to automation in electronic gadgets. With the advancement in technology and the availability of the Internet worldwide, the concept of smart farming is gaining popularity [17]. The project can be used as a prototype in predictive farming as the gathered data is available on DynamoDB. Also, devices can be placed inside the greenhouse to monitor the temperature and humidity of different greenhouse. This will ensure that the temperature can be monitored separately for each crop type. Notifications can be sent to alert the sudden changes in the temperature levels to ensure healthy plant growth.

6. Conclusion and Future Enhancements

The project successfully achieved its objective of collecting the sensor data using STM32L475 microcontroller board, storing it on the cloud and visualizing the data on a map. The web application displays the Google heat map along with the sensor data information. The user of the application can find out the places with the lowest temperature and humidity and can also receive notifications about the same. Since the data is on the cloud, it can be further utilized by professionals for carrying out detailed studies related to climate, agriculture and health industry.

In future, the web application can be further enhanced by utilizing other map functionality like searching the locations through the search bar. Other device features such as barometer, magnetometer, etc could also be used and can also be stored on the cloud to widen the project application beyond temperature and humidity. The board can be used with other sensor devices for sensing pollutants in the room and taking necessary measures based on the data gathered.

Also, in future, the sensor data can be stored on the cloud using the low-energy Bluetooth feature of the board where a smartphone application can be developed to get the sensor data and using the mobile GPS track the devices on a map.

7.References

- [1] Margapuri, Venkat, and Mitchell Neilsen. "Prototype Implementation of Temperature Control System with CAN and FreeRTOS on STM32F407 Discovery Boards." *EPiC Series in Computing* 63 (2019): 130-139.
- [2] P. Brynda, Z. Kosov and J. Kopřiva, "Mobile sensor unit for online air quality monitoring," 2016 Smart Cities Symposium Prague (SCSP), Prague, 2016, pp. 1-4. doi: 10.1109/SCSP.2016.7501028
- [3] R. K. Kodali and S. Mandal, "IoT based weather station," 2016 *International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, Kumaracoil, 2016, pp. 680-683.doi: 10.1109/ICCICCT.2016.7988038
- [4] [ARM MBED "https://os.mbed.com/platforms/ST-Discovery-L475E-IOT01A/"](https://os.mbed.com/platforms/ST-Discovery-L475E-IOT01A/)
- [5] City of Prague – Prague City Hall (2014): Prague Environment 2012 Yearbook report on state of the environment 55, 14-19.
- [6] Z. Kosov, "Pollution Gathering: Real time Pollution Map," in Proc. International Conference on Advances in Software 2015 , Antalya, 2015; pp 29--34.
- [7] Thomas, Auswin George, Pedram Jahangiri, Di Wu, Chengrui Cai, Huan Zhao, Dionysios C. Aliprantis, and Leigh Tesfatsion. "Intelligent residential air-conditioning system with smart-grid functionality." *IEEE Transactions on Smart Grid* 3, no. 4 (2012): 2240-2251.
- [8] M. H. Asghar, A. Negi, and N. Mohammadzadeh, "Principle application and vision in internet of things (iot)," in International Conference on Computing, Communication Automation, May 2015, pp. 427–431.
- [9] Y. Zhou, Q. Zhou, Q. Kong, and W. Cai, "Wireless temperature amp; humidity monitor and control system," in 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), April 2012, pp. 2246–2250.
- [10] B. Deaky, N. B. Lupulescu and D. Ursutiu, "Extended educational use of the Microcontroller Student Learning Kit (MCU SLK)," 2011 *IEEE Global Engineering Education Conference (EDUCON)*, Amman, 2011, pp. 913-916.doi: 10.1109/EDUCON.2011.5773254

-
- [11] S. Nazeem Basha, Dr. S.A.K. Jilani², Mr.S. Arun³, "An Intelligent Door System using Raspberry Pi and Amazon Web Services IoT" March 2016 International Journal of Engineering Trends and Technology (IJETT) – Volume 33 Number 2
- [12] Monteiro PL, Zanin M, Ruiz EM, Pimentão J, Sousa PADC. Indoor Temperature Prediction in an IoT Scenario. *Sensors (Basel)*. 2018;18(11):3610. Published 2018 Oct 24. doi:10.3390/s18113610
- [13] A. Detti, G. Tropea, G. Rossi, J. A. Martinez, A. F. Skarmeta and H. Nakazato, "Virtual IoT Systems: Boosting IoT Innovation by Decoupling Things Providers and Applications Developers," *2019 Global IoT Summit (GloTS)*, Aarhus, Denmark, 2019, pp. 1-6.doi: 10.1109/GIOTS.2019.8766422
- [14] Kheriji, Amira Abbes, Faouzi Bouani, Mekki Ksouri, and Mohamed Ben Ahmed. "A microcontroller implementation of model predictive control." *World Academy of Science, Engineering and Technology* 5 (2011): 5-21.
- [15] Chen, Hsinchun, Xin Li, Michael Chau, Yi-Jen Ho, and Chunju Tseng. "Using open web APIs in teaching web mining." *IEEE Transactions on Education* 52, no. 4 (2009): 482-490.
- [16] Ryu, Minwoo, Jaeseok Yun, Ting Miao, Il-Yeup Ahn, Sung-Chan Choi, and Jaeho Kim. "Design and implementation of a connected farm for smart farming system." In *2015 IEEE SENSORS*, pp. 1-4. IEEE, 2015.
- [17] Kassim, Mohamed Rawidean Mohd, Ibrahim Mat, and Ahmad Nizar Harun. "Wireless Sensor Network in precision agriculture application." In *2014 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pp. 1-5. IEEE, 2014.
- [18] Strauss, RICHARD H., E. R. McFadden, R. H. Ingram, E. C. Deal, and J. J. Jaeger. "Influence of heat and humidity on the airway obstruction induced by exercise in asthma." *The Journal of clinical investigation* 61, no. 2 (1978): 433-440.
- [19] Kalid, Sultana, Ali Syed, Azeem Mohammad, and Malka N. Halgamuge. "Big-data NoSQL databases: A comparison and analysis of "Big-Table", "DynamoDB", and "Cassandra"." In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*(, pp. 89-93. IEEE, 2017.
- [20] Tärneberg, William, Vishal Chandrasekaran, and Marty Humphrey. "Experiences creating a framework for smart traffic control using aws iot." In *2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, pp. 63-69. IEEE, 2016.

-
- [21] [Global X By Mirae Asset "https://www.globalxetfs.com/leading-companies-in-the-development-of-the-internet-of-things/"](https://www.globalxetfs.com/leading-companies-in-the-development-of-the-internet-of-things/)
- [22] Fang, Biyi, Qiumin Xu, Taiwoo Park, and Mi Zhang. "AirSense: an intelligent home-based sensing system for indoor air quality analytics." In *Proceedings of the 2016 ACM International joint conference on pervasive and ubiquitous computing*, pp. 109-119. 2016.
- [23] Wei, Peter, Xiaoqi Chen, Rishikanth Chandrasekaran, Fengyi Song, and Xiaofan Jiang. "Adaptive and Personalized Energy Saving Suggestions for Occupants in Smart Buildings." In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, pp. 247-248. 2016.
- [24] [Google](https://www.google.com/search?q=STM32+IoT+Discovery+kit+with+Sensor+placement+and+Features&rlz=1C1CHBF_enUS867US867&sxsrf=ALeKk01EdE6M5IR-twuul8ljHTgRUVi2JA:1586485600046&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjbyYa159zoAhXRN3OKHexbCHcQ_AUoAXoECAwQAw&biw=1504&bih=642#imgsrc=sBrtiAj6Kyx_RM)
"https://www.google.com/search?q=STM32+IoT+Discovery+kit+with+Sensor+placement+and+Features&rlz=1C1CHBF_enUS867US867&sxsrf=ALeKk01EdE6M5IR-twuul8ljHTgRUVi2JA:1586485600046&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjbyYa159zoAhXRN3OKHexbCHcQ_AUoAXoECAwQAw&biw=1504&bih=642#imgsrc=sBrtiAj6Kyx_RM"
- [25] [Aeroqual "https://www.aeroqual.com"](https://www.aeroqual.com)
- [26] [Purple Air "https://www2.purpleair.com/pages/technology"](https://www2.purpleair.com/pages/technology)
- [27] [Purple Air "https://www.purpleair.com/map?opt=1/mAQI/a10/cC0#1/25/-30"](https://www.purpleair.com/map?opt=1/mAQI/a10/cC0#1/25/-30)
- [28] [ST.com"https://www.st.com/content/dam/technology-tour-2017/hands-on-training_stm32.pdf"](https://www.st.com/content/dam/technology-tour-2017/hands-on-training_stm32.pdf)
- [29] [Linuxgizmos.com"http://linuxgizmos.com/amazon-aims-an-aws-savvy-version-of-freertos-at-iot-gizmos/"](http://linuxgizmos.com)
- [30] [Primo.AI "http://primo.ai/index.php?title=AWS Internet of Things \(IoT\)"](http://primo.ai/index.php?title=AWS_Internet_of_Things_(IoT))

APPENDIX

Technical terminologies

MCU Ecosystem - It is a collection of compatible and interoperable hardware and software components that allows development and prototyping of innovative applications based on the STM32 32-bit Microcontroller.

AWS (Amazon Web Services) - It is a third-party cloud service provider. It provides cloud services such as SNS, data storage services, IoT services etc. Users pay only for the services that they use. As compared to other cloud service providers AWS is the cheapest and provides a wide range of services.

SNS (Simple Notification Service) - It is one of the cloud services provided by AWS. It is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and serverless applications. It is used in the project for sending email notification when the temperature reaches a certain threshold value.

STM (STMicroelectronics) - STMicroelectronics is an electronics and semiconductor manufacturer headquartered in Switzerland. It is a leading Integrated Device Manufacturer delivering solutions that are key to Smart Driving, Smart Industry, Smart Home & City.

MQTT (Message Queuing Telemetry Transport) - MQTT is a lightweight publish-subscribe network protocol for transferring messages between devices such as sensors. [Wikipedia](https://en.wikipedia.org/wiki/MQTT)
"<https://en.wikipedia.org/wiki/MQTT>"

JSON (JavaScript Object Notification) - Json is a human readable file format and data interchange format. It transfers information in the form of data objects which is Key-value pair format. It is lightweight and easy for humans to read and write.

API (Application Programming Interface) - It is an interface which determines the request response calls and its format (JSON/ XML) and how the system should use it.

Serial Communication - It is the telecommunication process of transmitting data one bit at a time in a sequential manner over the communication channel or computer bus. It is used for most computer networks.

FreeRTOS (Real Time Operating System) - It is an open source real time operating system for microcontrollers. It allows secure connection of low-power small devices to connect to AWS cloud services such as AWS IoT core. [AWS Amazon "https://aws.amazon.com/freertos/"](https://aws.amazon.com/freertos/)

AWS IoT Core - This is one of the cloud services provided by AWS for connecting devices such as sensors to cloud over Wi-Fi.

UART (Universal Asynchronous Receiver/Transmitter) - It is a computer hardware whose main purpose is to transmit and receive serial data. It is not a protocol.

Spring boot - It is a web application framework which follows model-view-controller design pattern. With Spring boot it is easy to develop scalable and restful web applications.

MVC - Model-View-Controller design pattern