CMPE 277 SMARTPHONE APPLICATION DEVELOPMENT

FINAL PROJECT REPORT

Environmental Monitoring System Using IoT and Cloud Service at Real-Time

TEAM SPARTANS

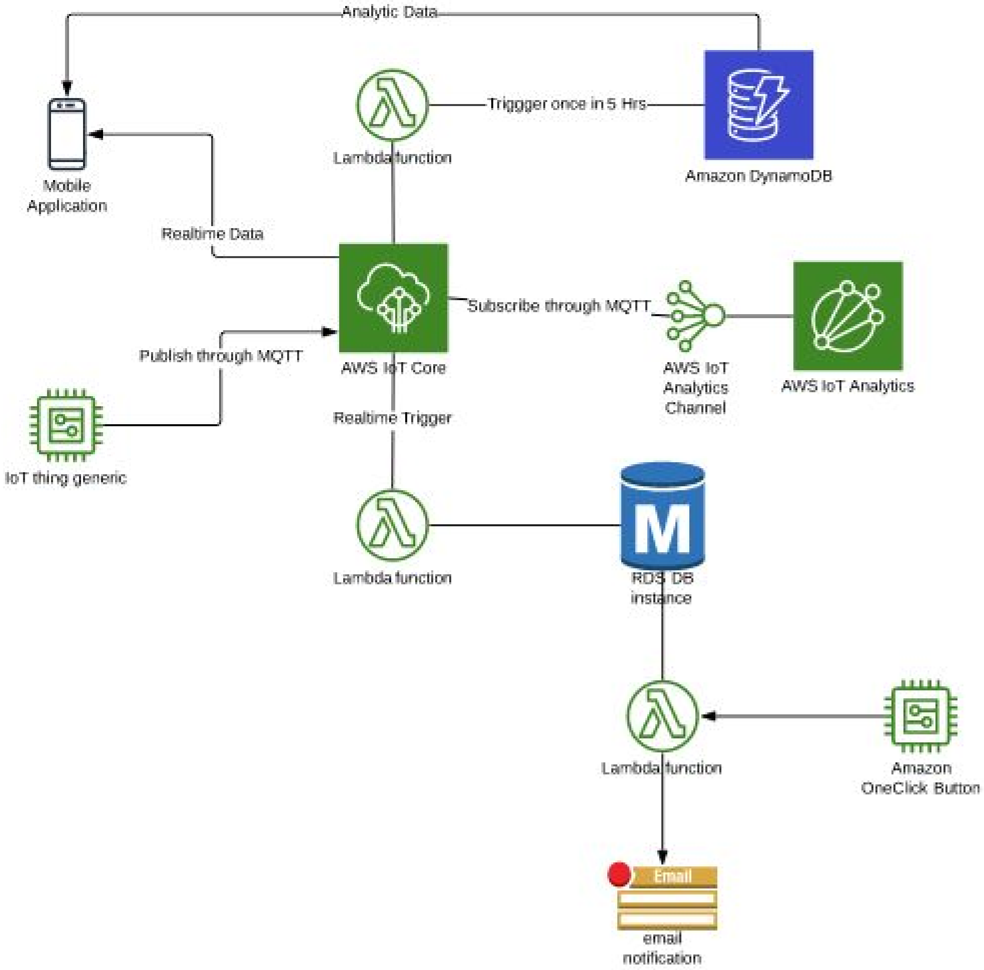
MUKESH RANJAN SAHAY
MUTHU KUMAR SUKUMARAN
SUDHA AMARNATH
THIRUMALAI NAMBI DOSS PALANI

ABSTRACT

Monitoring environmental activities has become very crucial over the days for controlling the electronic appliances around us. With the help of a new technology stack, we have designed and developed a mobile application using the Internet of Things (IoT) based on a real-time environmental monitoring system. Sensors are connected to the cloud and the device is subscribed to data published to the cloud.From our application, we can get the weather details like Temperature, Humidity, Heat Index and Air Quality. The application also possesses the capability to send a notification to the subscribed users when the temperature reaches the threshold limit. We are using Amazon One-Click button to send a notification to the user with the temperature, humidity and heat index details. server. This is a low-cost system which gives insight into the design and implementation of a complete IoT application involving all aspects from sensing and wireless transmission to cloud storage and data retrieval from the cloud via a mobile application. The results of the project show the real-time monitoring of temperature and humidity levels from any location in the world and its statistical analysis. This system can be extended to enable remote controlling of various appliances based on the sensed data.

We have used the NodeMCU (ESP-8266) because of its simplicity, robustness and lost cost. The system uses the WiFi module ESP8266 in order to upload sensor data from DHT-11 to the cloud data store We are using DHT11 sensor to capture the data.It guarantees a great consistency and outstanding long-term steadiness. DHT-11 sensor comprises of a resistive-type humidity measurement component and an NTC temperature measurement component. It is small sensor with fast response and high quality. It has a comparatively low cost and strong anti-interference ability, digital signal output, and precise calibration. It has a temperature range from 0 to 50°C and humidity range from 20 to 90%RH, and the signal transmission range of 20m. We have installed the DHT library and used it to get the required data.

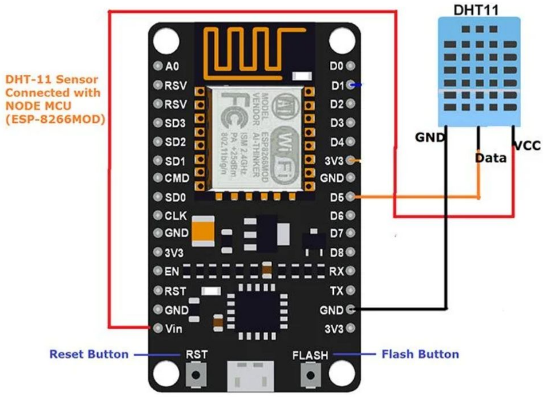
ARCHITECTURE



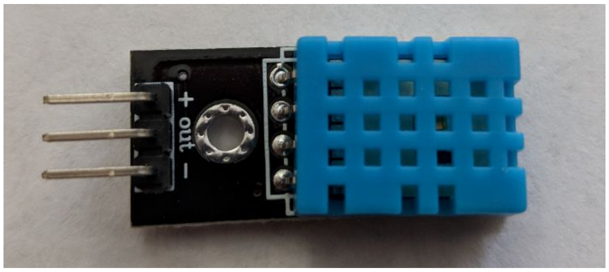
TECHNOLOGIES USED

Hardware

The below diagram depicts a picture of the NodeMCU microcontroller used in our system.



NodeMCU - ESP8266



DHT11 - Temperature and Humidity Sensor

Significance of using DHT11 Sensor

- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 20-80% humidity readings with 5% accuracy
- Good for 0-50°C temperature readings ±2°C accuracy
- No more than 1 Hz sampling rate (once every second)
- Body size 15.5mm x 12mm x 5.5mm
- 4 pins with 0.1" spacing

Software

Software used in the application

- Arduino
- Android Studio

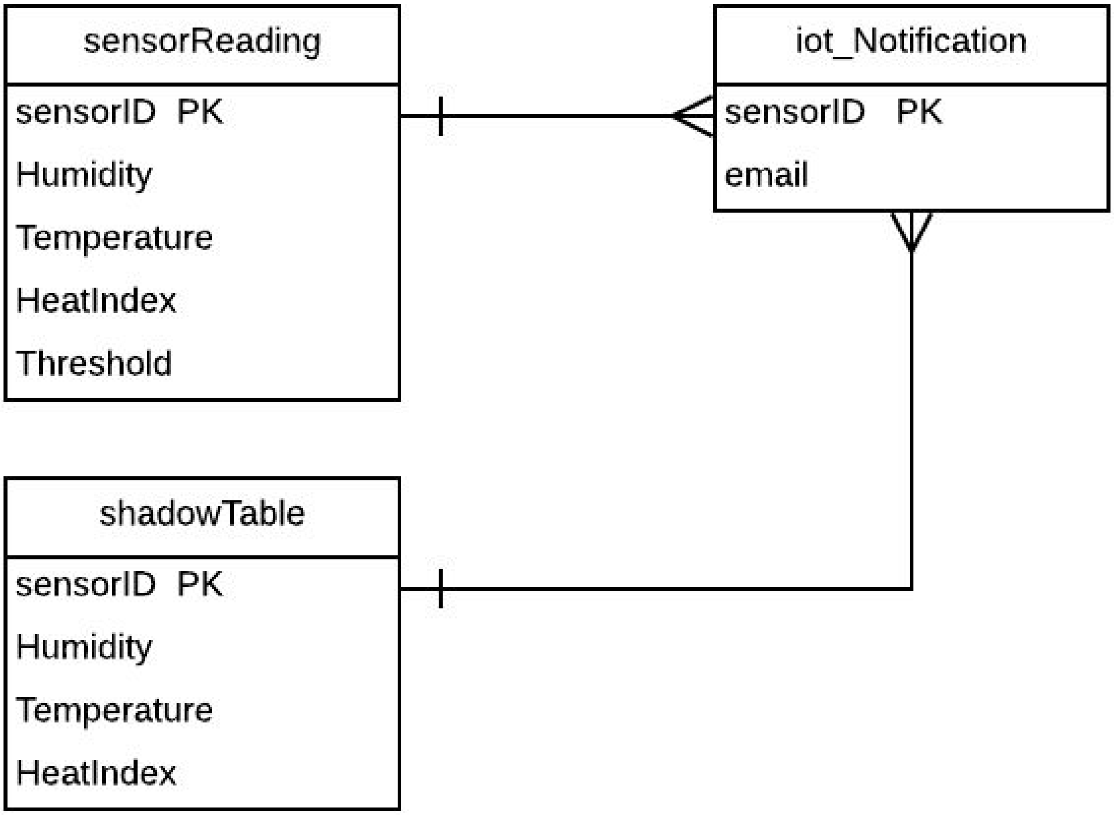
CLOUD SERVICES

Below are the cloud services used in our temperature monitoring services.

- AWS Mobile SDK
- AWS IoT Core
- AWS IoT Topic
- AWS Cognito
- AWS IoT 1-Click
- AWS SNS
- AWS SES
- AWS IoT Analytics
- AWS Lambda
- AWS DynamoDB
- AWS RDS
- AWS Mobile Hub

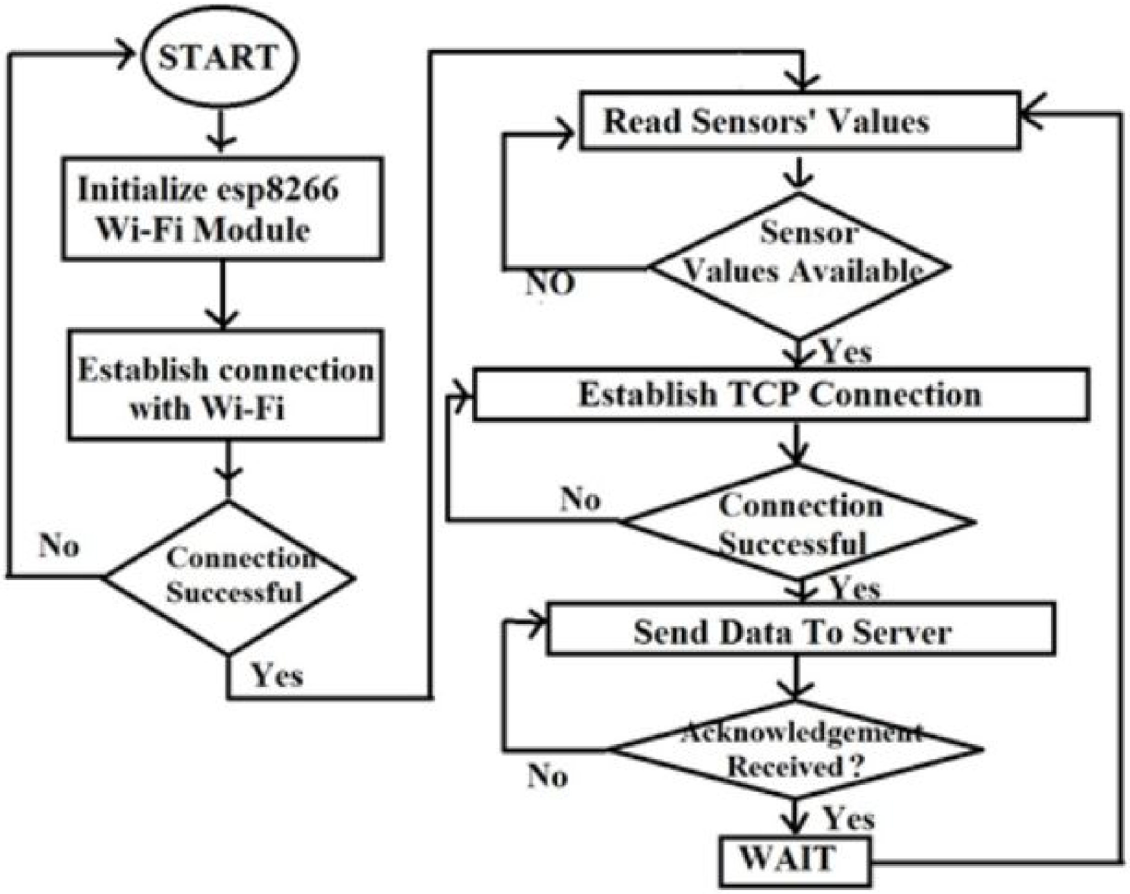
Data Model

Sensors are captured from the environment and sent to Sensor Reading table and Shadow Table. IoT notification will be initiated when the readings threshold value breaches the user defined value.



FLOW DIAGRAM

Below flow diagram depicts the collection of sensor data from DHT11 and send them to cloud through ESP8266 with MQTT protocol.



SETUP HARDWARE AND SOFTWARE MODULES

Steps to run the Arduino Code:

1. Download Arduino software version 1.8.9.
2. Addthis link http://arduino.esp8266.com/stable/package_esp8266com_index.json in file preferences "New Boards".
3. Select Board Manager: Search ESP8266 and install SDK version 2.4.1. This would install all the NODE MCU board softwares.
4. Select Tools Board à NodeMCU 1.0 ESP 12E Module and Baud rate 115200 and connected COM port.

Arduino Libraries installed:

1. aws-sdk-arduino **from** https://github.com/odelot/aws-sdk-arduino
2. arduinoWebSockets v.2.1.0 from https://github.com/Links2004/arduinoWebSockets/releases/tag/2.1.0
3. pubsubclient v2.6.0 from https://github.com/knolleary/pubsubclient/releases/tag/v2.6
4. Install DHT sensor library by Adafruit - 'DHT sensor'. v.1.3.4
5. Install Adafruit Unified Sensor v.1.0.3 by Adafruit

Running Arduino program:

1. Open and the Arduino code **WeatherMonitoringMetricsToAwsIotShadow.ino**
2. Update the AWS keys, IoT topic Connect the NodeMCU module to the COM port.
3. Compile and download the code to the NodeMCU flash

Setup AWS IOT Core for MQTT:

1. Setup IOT project thing and IoT topic.
2. Setup IOT shadow.
3. Setup AWS Cognito Identity

Setup Andriod App:

Following AWS classes are imported from SDK library aws-android-sdk-iot:2.13.+ in the build.gradle

1. def aws_version = "2.13.+"
2. implementation "com.amazonaws:aws-android-sdk-iot:$aws_version"
3. implementation ('com.amazonaws:aws-android-sdk-iot:$aws_version@aar') { transitive = true}
4. implementation "com.amazonaws:aws-android-sdk-mobile-client:$aws_version"

Import following classes for AWS Cognito, and MQTT functions from AWS libraries.

1. import com.amazonaws.auth.CognitoCachingCredentialsProvider;
2. import com.amazonaws.mobileconnectors.iot.AWSIotMqttClientStatusCallback;
3. import com.amazonaws.mobileconnectors.iot.AWSIotMqttManager;
4. import com.amazonaws.mobileconnectors.iot.AWSIotMqttNewMessageCallback;
5. import com.amazonaws.mobileconnectors.iot.AWSIotMqttQos;
6. import com.amazonaws.regions.Regions;
7. Configure AWS Region, AWS Cognito user identiy

Finally Build and Install project APK to the Android phone

REST INTERFACES AND APIs IN THE APP

```
// MQTT Client
AWSIotMqttManager mqttManager;
mqttManager = new AWSIotMqttManager(clientId,
CUSTOMER_SPECIFIC_ENDPOINT);
Intent analyticsIntent = new Intent(this,AnalyticsActivity.class);
Intent airqualityIntent = new Intent(this, AirQualityActivity.class);
public class WeatherMonitorAsyncTask extends AsyncTask<Void, Void, String>
  mqttManager.connect(credentialsProvider, new
AWSIotMqttClientStatusCallback()
  public String doInBackground(Void... params) {
          mqttManager.subscribeToTopic(topic, AWSIotMqttQos.QOS0,
                                                  new
AWSIotMqttNewMessageCallback()
          setCurrentSensorValues(message);
          return getCurrentSensorValues();
  }

  // AIR Quality messgage
public class AirQualityActivity extends AppCompatActivity {
public void airQuality(View view) {
        RequestQueue requestQueue;
        airURL =
"http://www.airnowapi.org/aq/forecast/zipCode/?format=application/json&zipCod
e="+zipcode+"&date=" + currentDate + "&distance=25&API_KEY=" + APIKEY;
        JsonArrayRequest jsonArrayRequest = new
JsonArrayRequest(Request.Method.GET, airURL, (String) null, new
Response.Listener<JSONArray>()
        airQualityGauge.setValue(Integer.parseInt(AQI));
}
```

```
// Analytics Activity
public class AnalyticsActivity extends AppCompatActivity {
Cartesian cartesian = AnyChart.line();
anyChartView.setChart(cartesian);
}
private class CustomDataEntry extends ValueDataEntry {}
```

```
// Gauge library
import pl.pawelkleczkowski.customgauge.CustomGauge;
```

```
// Android import classes used
import android.app.DownloadManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonArrayRequest;
import com.android.volley.toolbox.Volley;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.w3c.dom.Text;
import java.text.SimpleDateFormat;
import java.util.Date;
import pl.pawelkleczkowski.customgauge.CustomGauge;
import android.content.ComponentName;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import android.os.Handler;
import java.util.Calendar;
import java.text.SimpleDateFormat;
import java.util.Date;
import org.json.JSONArray;
import org.json.JSONObject;
import org.w3c.dom.Text;
import com.amazonaws.auth.CognitoCachingCredentialsProvider;
import com.amazonaws.mobileconnectors.iot.AWSIotMqttClientStatusCallback;
import com.amazonaws.mobileconnectors.iot.AWSIotMqttManager;
import com.amazonaws.mobileconnectors.iot.AWSIotMqttNewMessageCallback;
import com.amazonaws.mobileconnectors.iot.AWSIotMqttQos;
import com.amazonaws.regions.Regions;
import java.io.UnsupportedEncodingException;
import java.util.UUID;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import de.nitri.gauge.Gauge;
import eu.sergehelfrich.ersa.Dew;
import eu.sergehelfrich.ersa.Scale;
import eu.sergehelfrich.ersa.Temperature;
import pl.pawelkleczkowski.customgauge.CustomGauge;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import com.anychart.AnyChart;
import com.anychart.AnyChartView;
import com.anychart.chart.common.dataentry.DataEntry;
import com.anychart.chart.common.dataentry.ValueDataEntry;
import com.anychart.charts.Cartesian;
import com.anychart.core.cartesian.series.Line;
import com.anychart.data.Mapping;
import com.anychart.data.Set;
import com.anychart.enums.Anchor;
import com.anychart.enums.MarkerType;
import com.anychart.enums.TooltipPositionMode;
import com.anychart.graphics.vector.Stroke;
import java.util.ArrayList;
import java.util.List;
```

High-level Components on AWS IoT:

Following are high-level components of AWS IoT Platform –

1. Device gateway and MQTT Broker
2. Rules Engine
3. Registry (Things, Shadow and Shadow Service)
4. Security and Identity (Certificates and Policies)

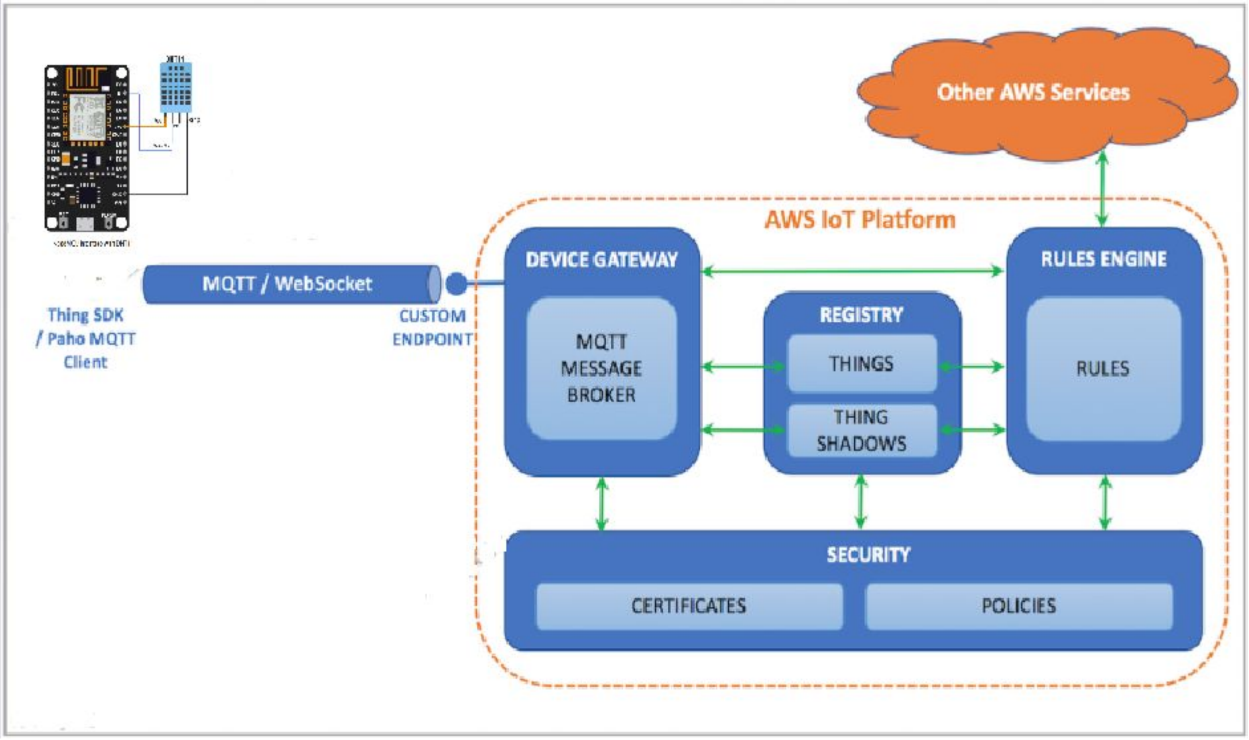
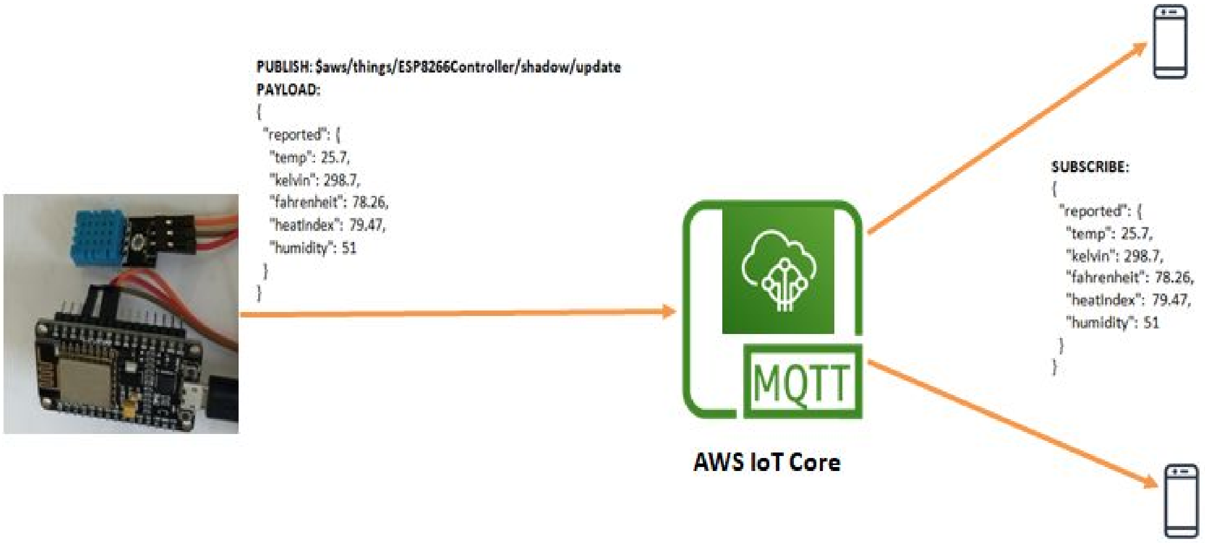


Fig : Hardware and Software Component Diagram

Below figure depicts the flow of sensor data from DHT11 to connected mobile devices.



```
PUBLISH: $aws/things/ESP8266Controller/shadow/update
PAYLOAD:
{
 "reported": {
  "temp": 25.7,
  "kelvin": 298.7,
  "fahrenheit": 78.26,
  "heatIndex": 79.47,
  "humidity": 51
 }
}
```

```
SUBSCRIBE:
{
 "reported": {
  "temp": 25.7,
  "kelvin": 298.7,
  "fahrenheit": 78.26,
  "heatIndex": 79.47,
  "humidity": 51
 }
}
```

TESTING

Following are the unit test cases we considered during the design and development of the project:

1. Validate if the sensor is correctly connected to the nodeMCU and to the WiFi.
2. Validate that the MCU is recorded as the publisher for the MQTT topic in AWS IoT Core.
3. Check the incoming data to the AWS IoT Core and make sure that it is getting updated as per the condition where the sensor is located.
4. Make sure that the mobile app is registered as the subscriber for the MQTT topic in the AWS IoT Core.
5. Validate the data received by the mobile app and check with the IoT Core data.
6. Check the analytics graph generated by the mobile app and validate it with the data in the database.
7. Validate the Air Quality Index displayed on graph as compared on the Open Weather API.

8. Check the email notification received after clicking the AWS IoT 1-Click button.
9. Validate the information received in the email as compared to the current sensor data.
10. Validate the data stored in the AWS RDS and DynamoDB and confirm its accuracy.

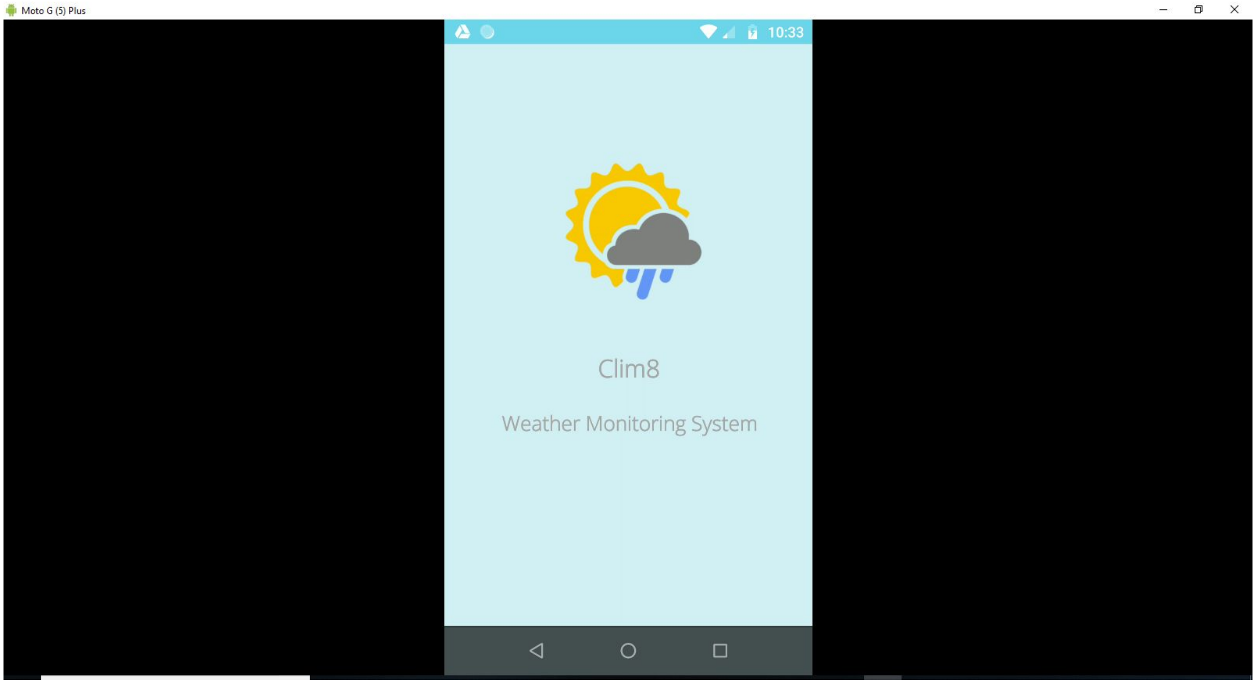
INDIVIDUAL CONTRIBUTION

- **MUKESH RANJAN SAHAY**
  - Worked on the basic setup of DynamoDB to pass data for Analytics
  - Worked on the Integrating AWS Mobile SDK to the Android Application
  - Worked on the AWS IoT 1-Click button
- **THIRUMALAI NAMBI DOSS PALANI**
  - Worked on the configuration of NodeMCU Sensor
  - Worked on sending data from the sensor to AWS IoT Core
  - Created the Lambda function to update values from AWS IoT Core to AWS RDS
- **SUDHA AMARNATH**
  - Worked on the configuration of DHT11 Sensor
  - Worked on sending data from IoT Core to IoT Analytics
  - Worked on sending data from the sensor to AWS IoT Core
- **MUTHU KUMAR SUKUMARAN**
  - Worked on the Mobile UI

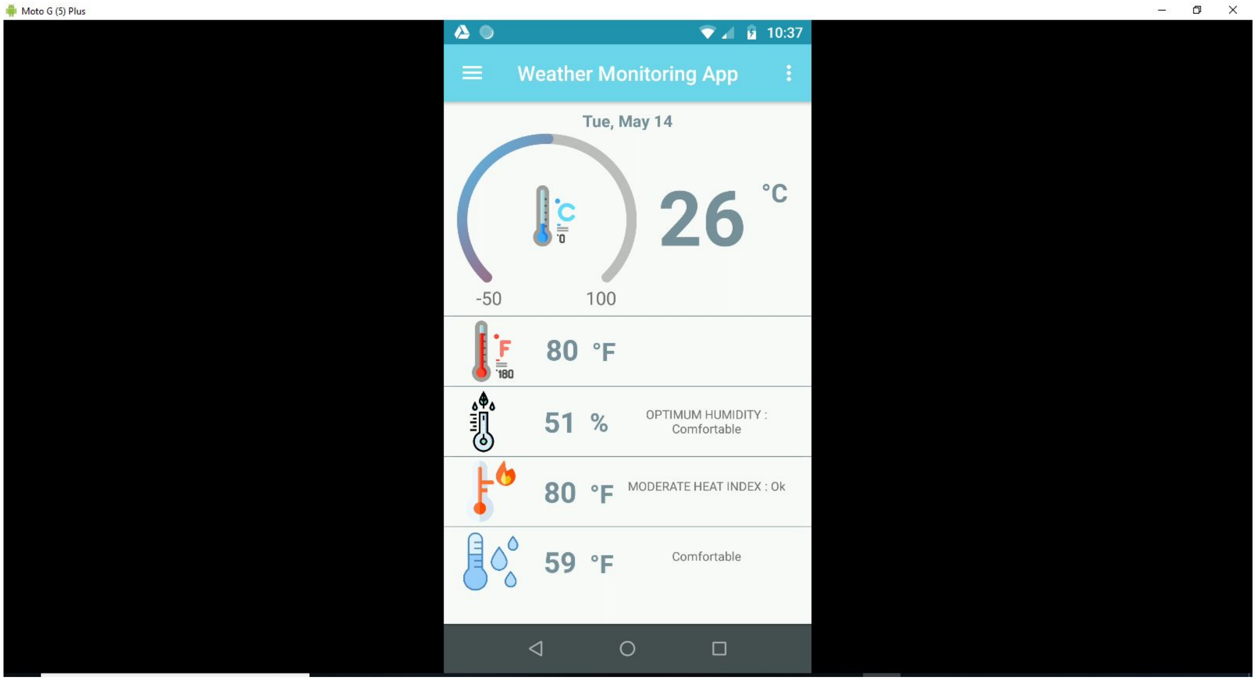
- Handling data in the Mobile Application
- Worked on AWS Mobile Hub to pass data from DynamoDB to the Application

SCREENSHOTS

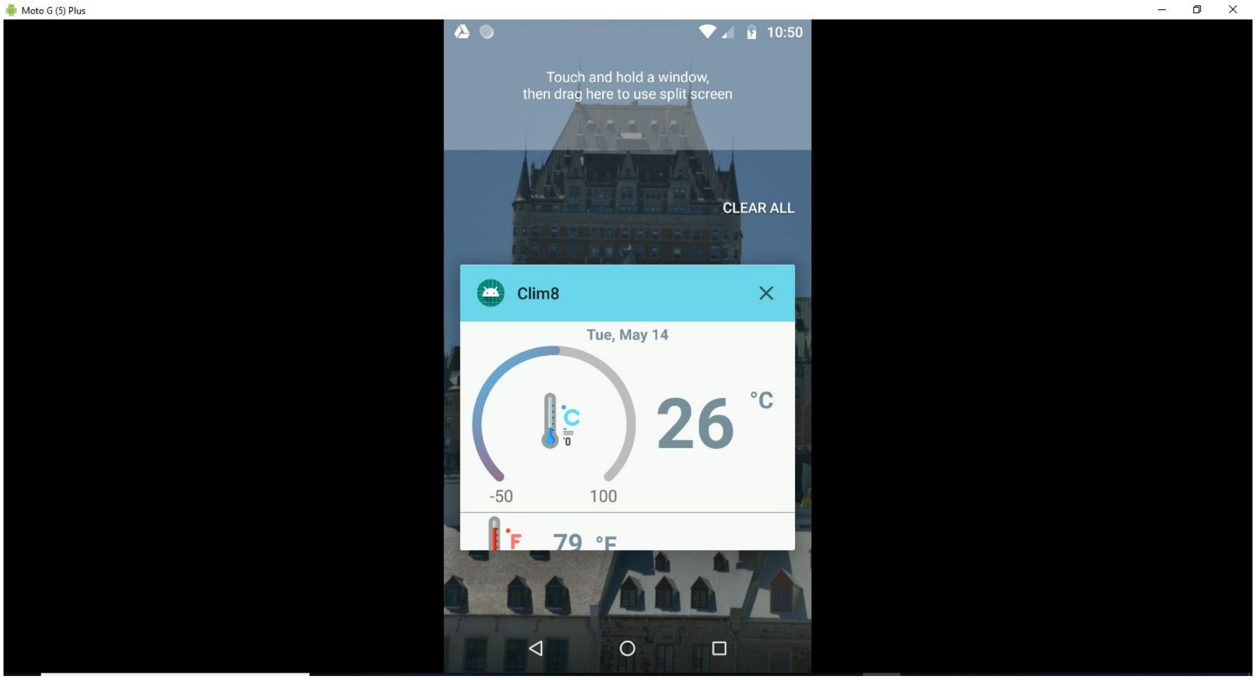
1. Launcher Activity of the Application



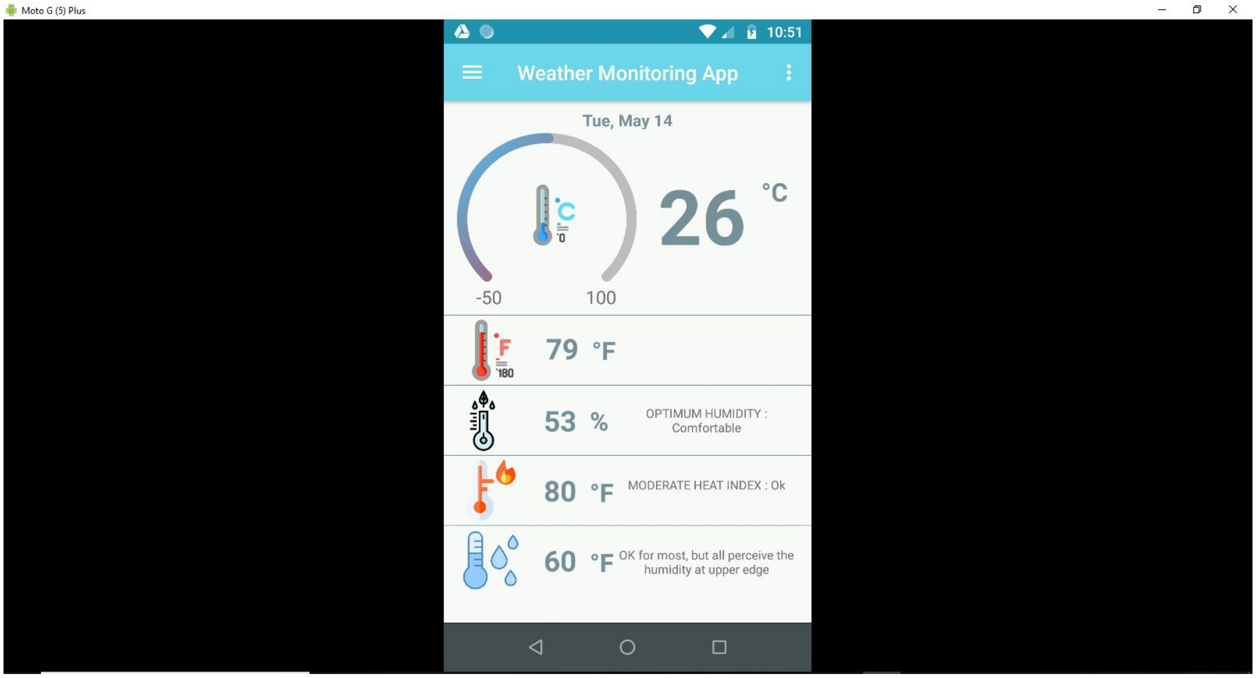
2. Sensor Values Displayed on the Screen.
   a. Temperature In C
   b. Fahrenheit in F
   c. Humidity in %
   d. Heat Index in C
   e. Dew point in F
   f. Health related messages displayed depending on the parameters



3. Application Paused in the Background



4. Restore of the current values when the Main Activity is resumed



Moto G (5) Plus

10:51

Weather Monitoring App

Tue, May 14

26 °C

-50 100

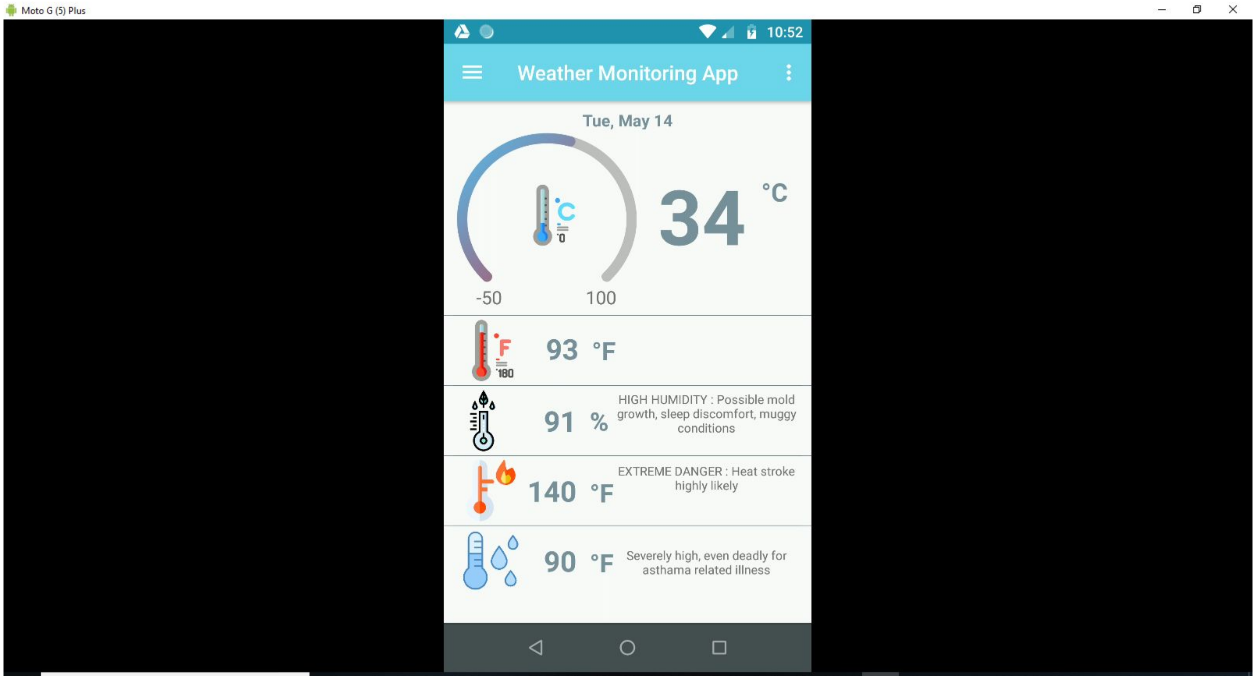
79 °F

53 % OPTIMUM HUMIDITY : Comfortable

80 °F MODERATE HEAT INDEX : Ok

60 °F OK for most, but all perceive the humidity at upper edge

5. Manually stimulating DHT sensor value changes by testing the sensor by covering it in hand for a few minutes. Values from sensors are changed appropriate messages are displayed on the App screen



Moto G (5) Plus

10:52

Weather Monitoring App

Tue, May 14

34 °C

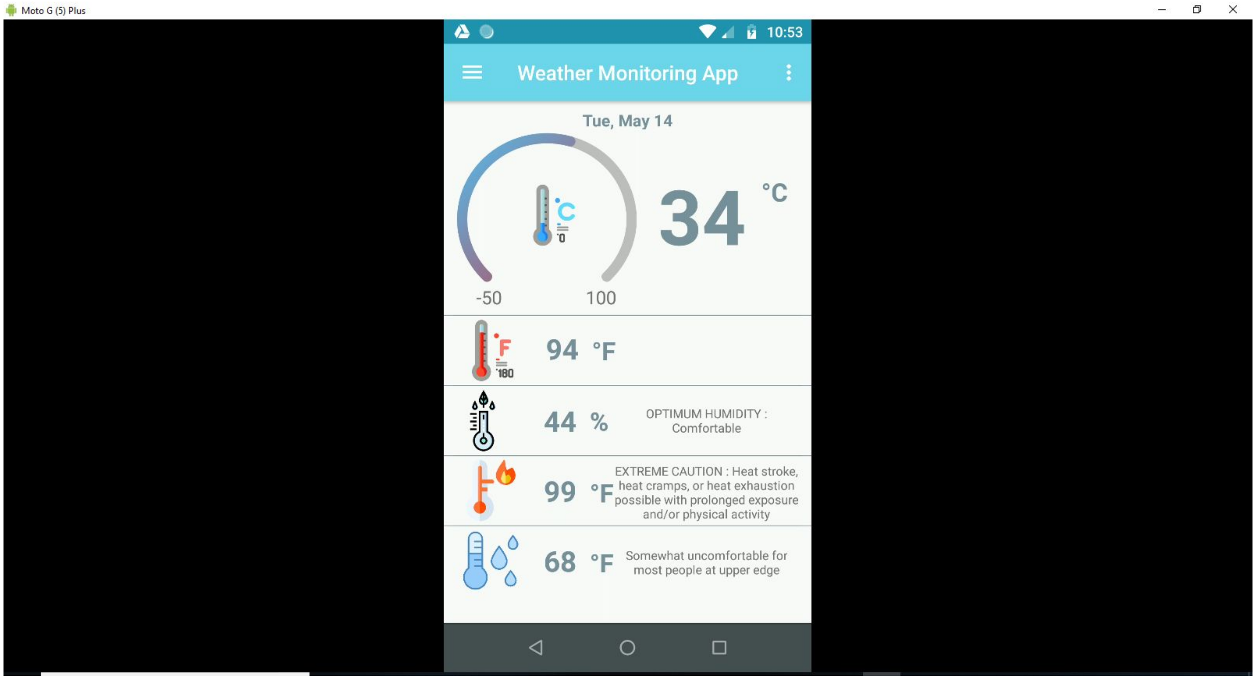
-50 100

93 °F

91 % HIGH HUMIDITY : Possible mold growth, sleep discomfort, muggy conditions

140 °F EXTREME DANGER : Heat stroke highly likely

90 °F Severely high, even deadly for asthama related illness



Weather Monitoring App

Tue, May 14

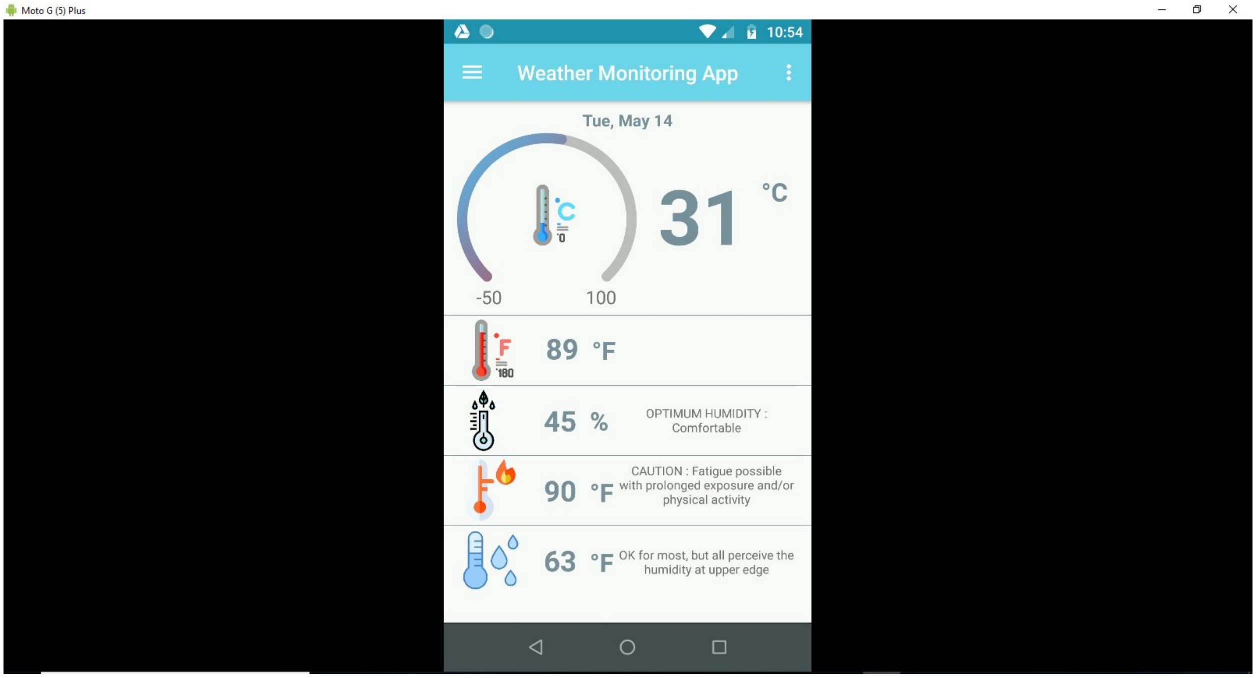
34 °C

94 °F

44 % OPTIMUM HUMIDITY : Comfortable

99 °F EXTREME CAUTION : Heat stroke, heat cramps, or heat exhaustion possible with prolonged exposure and/or physical activity

68 °F Somewhat uncomfortable for most people at upper edge



Weather Monitoring App

Tue, May 14

31 °C

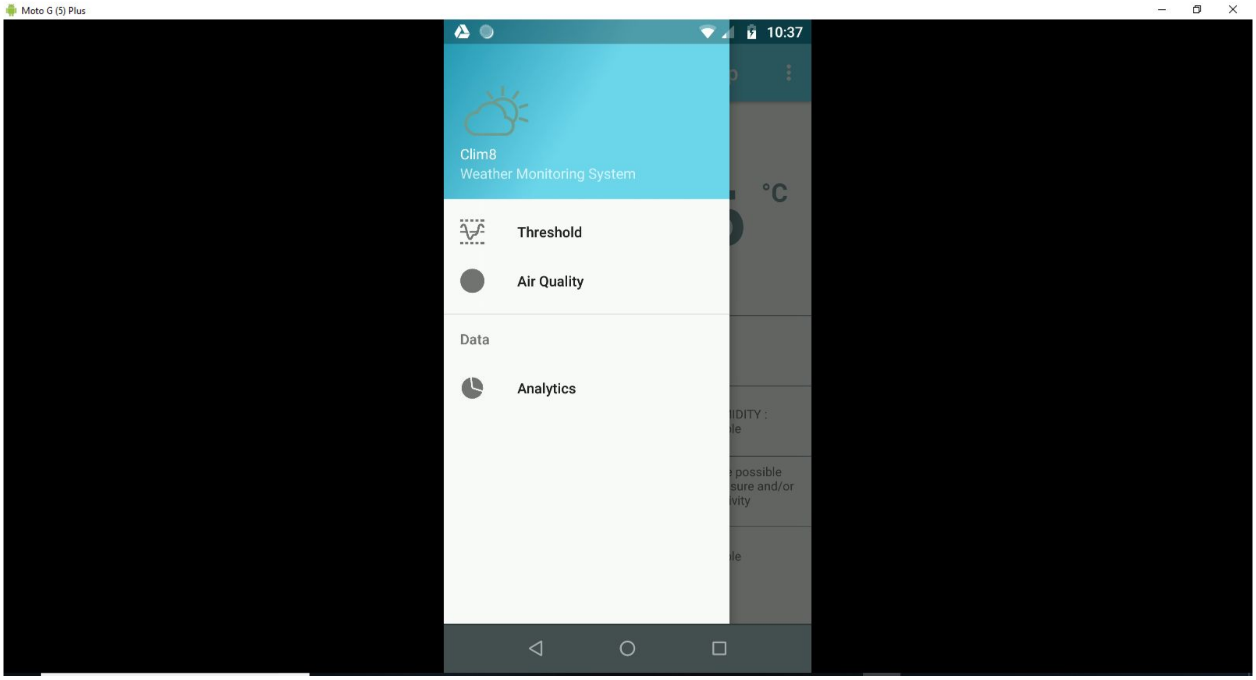
89 °F

45 % OPTIMUM HUMIDITY : Comfortable

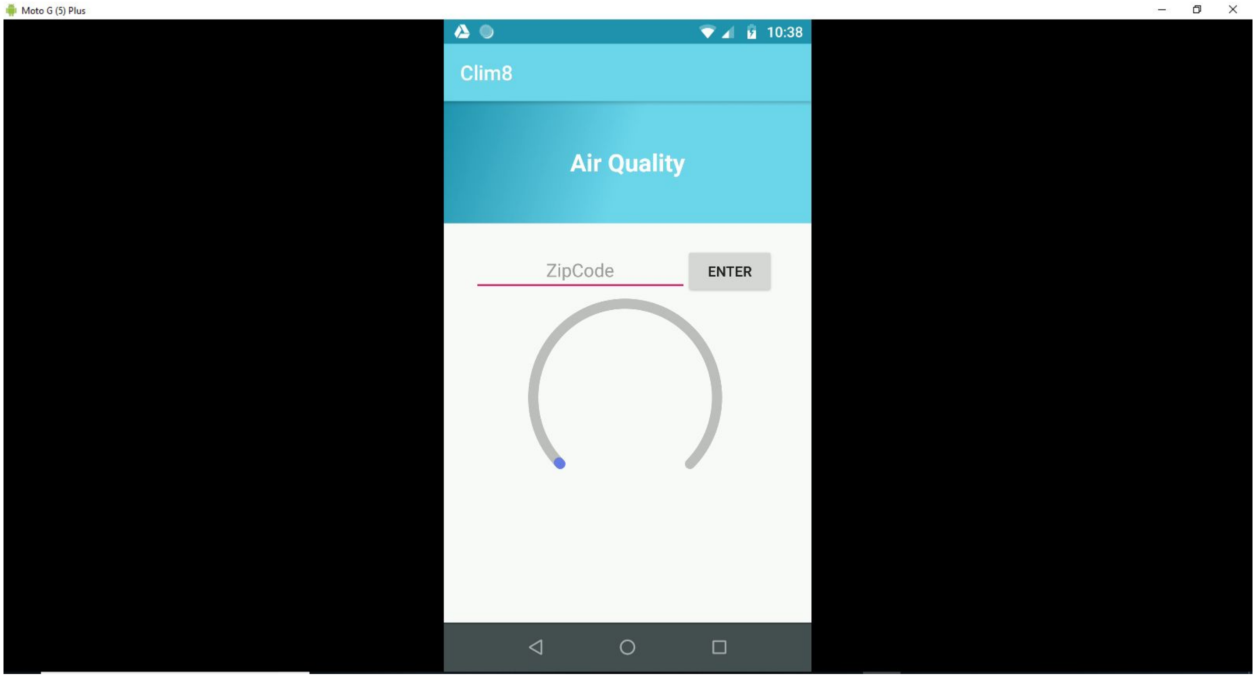
90 °F CAUTION : Fatigue possible with prolonged exposure and/or physical activity

63 °F OK for most, but all perceive the humidity at upper edge

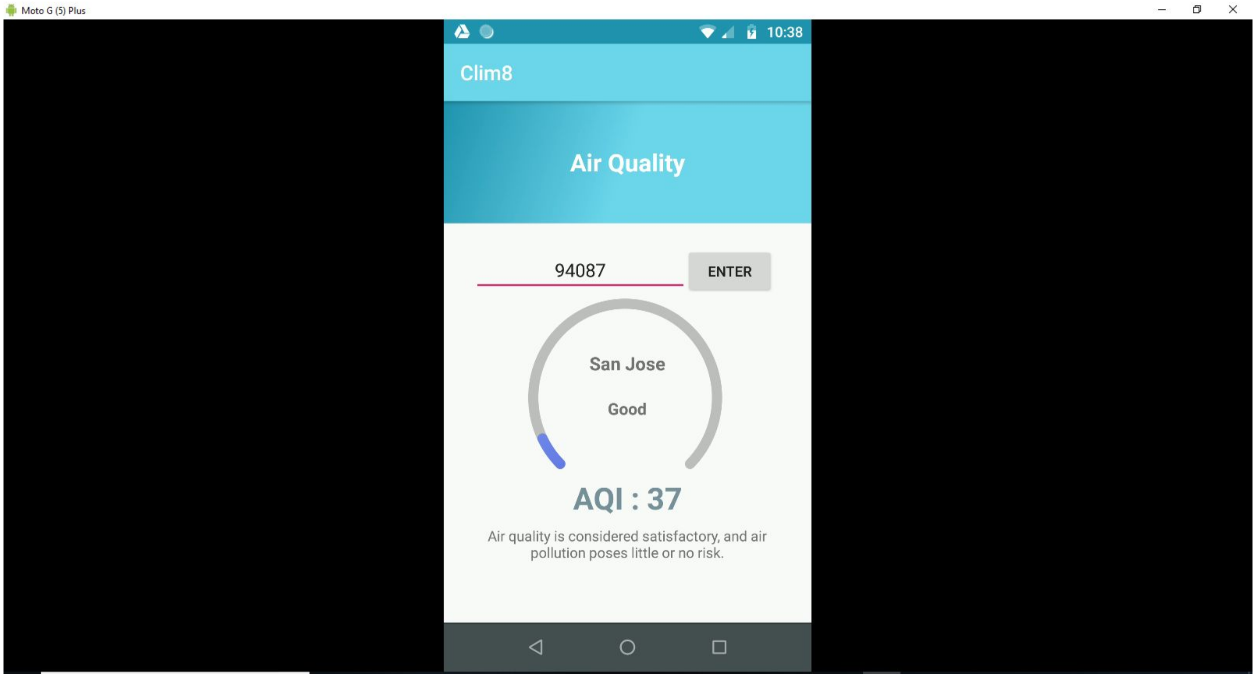
6. Navigation Pane options



7. Air Quality activity display



7. Entering Zipcode for San Jose, CA for air quality and AQI is displayed



94087

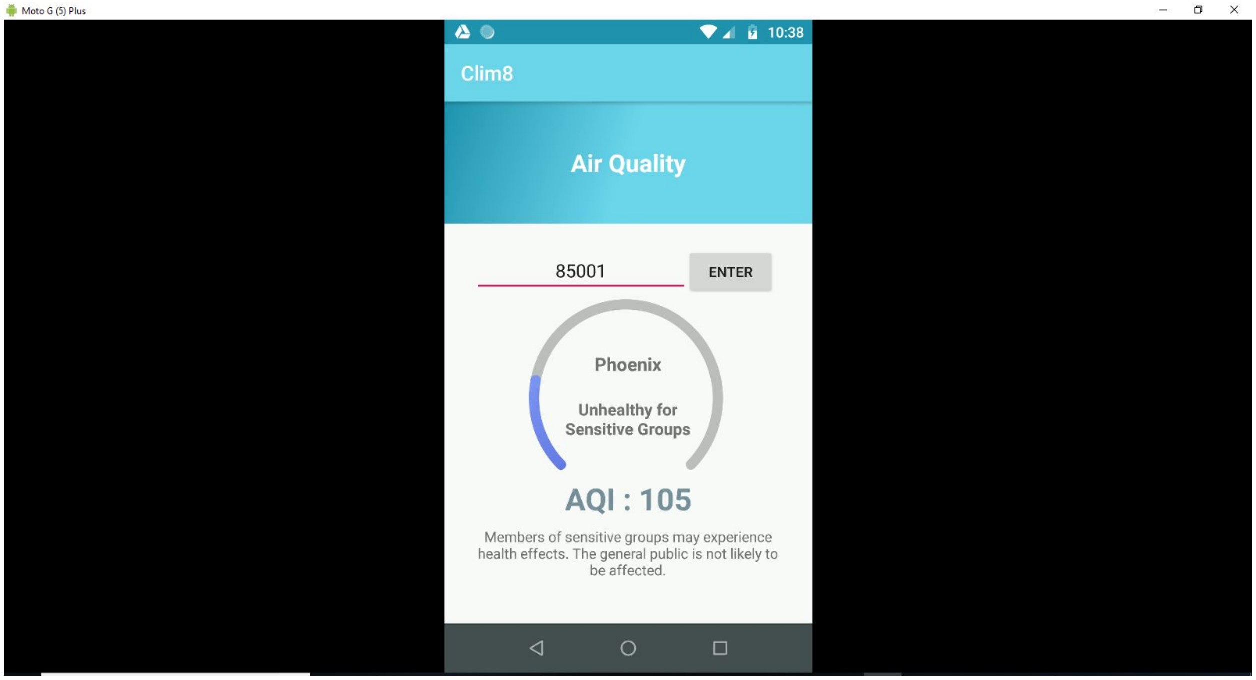
San Jose

Good

AQI : 37

Air quality is considered satisfactory, and air pollution poses little or no risk.

8. Entering Zipcode for Phoenix, AZ for air quality and AQI is displayed - showing more degraded air quality



85001

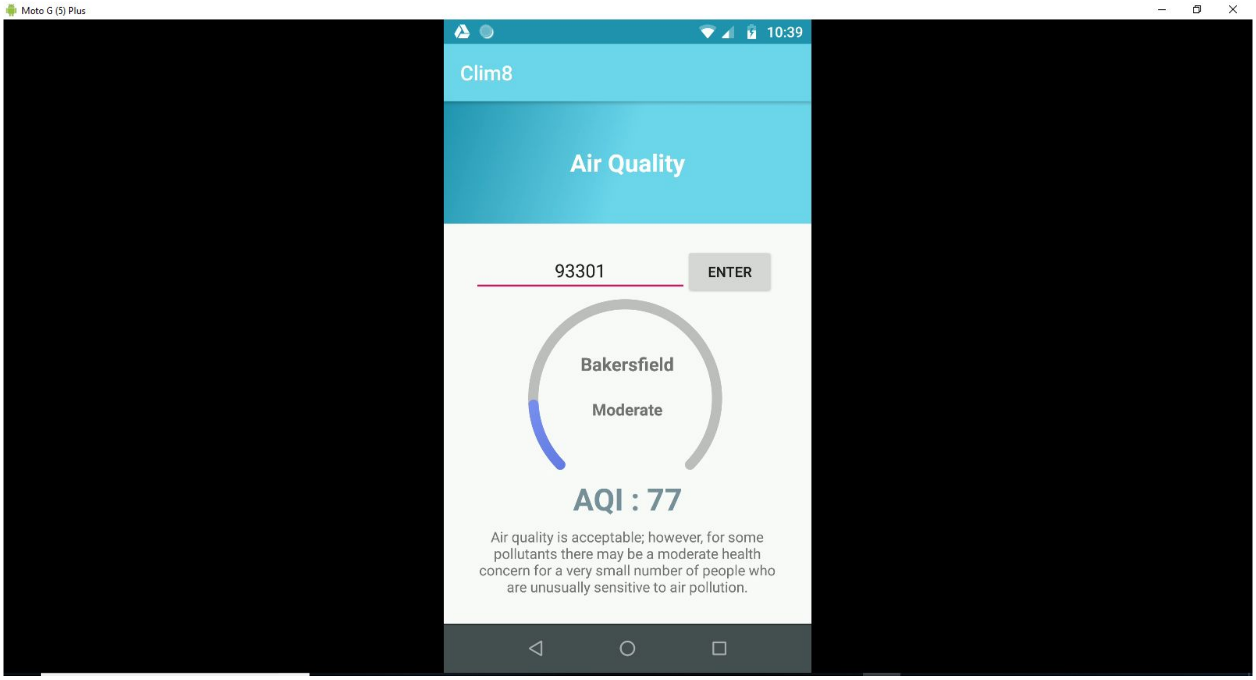
Phoenix

Unhealthy for Sensitive Groups

AQI : 105

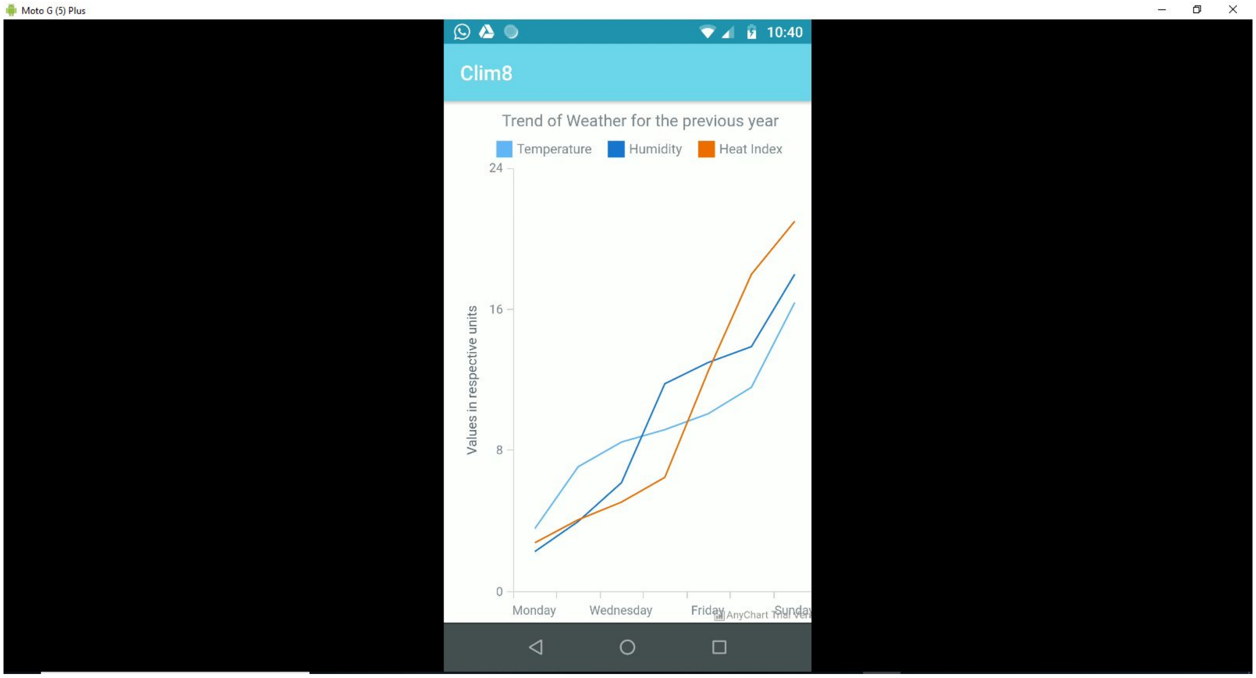
Members of sensitive groups may experience health effects. The general public is not likely to be affected.

9. Moderate AQI in Backerfield, CA.

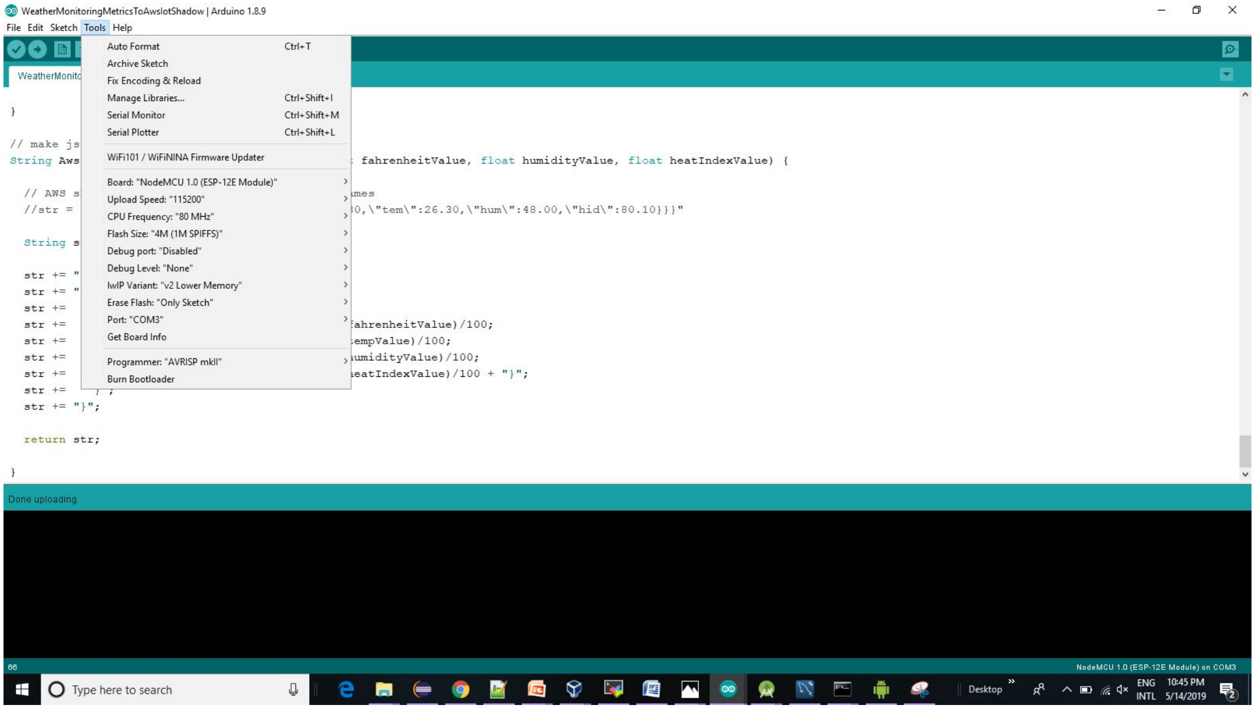


Air quality is acceptable; however, for some pollutants there may be a moderate health concern for a very small number of people who are unusually sensitive to air pollution.

10. Analytics Graph.



11. Arduino Software showing NodeMCU board



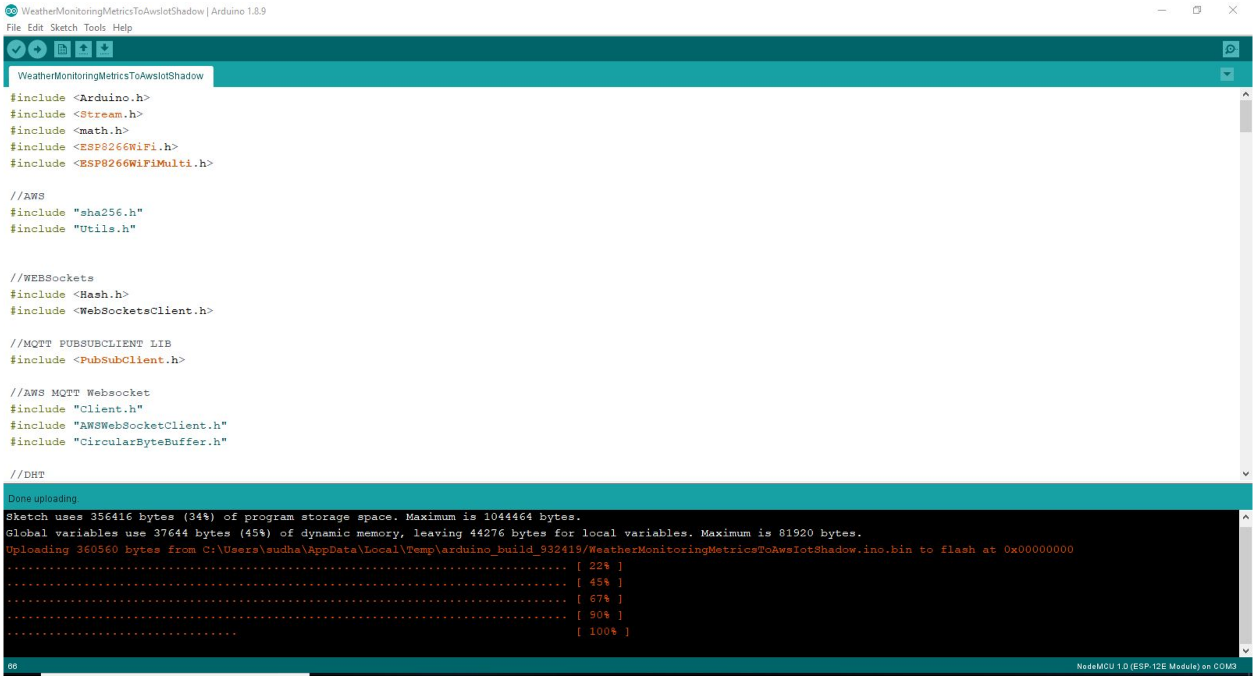
WeatherMonitoringMetricsToAwsIotShadow | Arduino 1.8.9

Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "NodeMCU 1.0 (ESP-12E Module)"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Flash Size: "4M (1M SPIFFS)"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
Erase Flash: "Only Sketch"
Port: "COM3"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader

```
}

// make js
String Aws                          fahrenheitValue, float humidityValue, float heatIndexValue) {

  // AWS s                          mes
  //str =                           0,\"tem\":26.30,\"hum\":48.00,\"hid\":80.10}}}"

  String s

  str += "
  str += "
  str +=
  str +=                            ahrenheitValue)/100;
  str +=                            empValue)/100;
  str +=                            umidityValue)/100;
  str +=                            eatIndexValue)/100 + "}";
  str +=       } ;
  str += "}";

  return str;

}
```

Done uploading.

12. Downloading Arduino code to the nodeMCU device



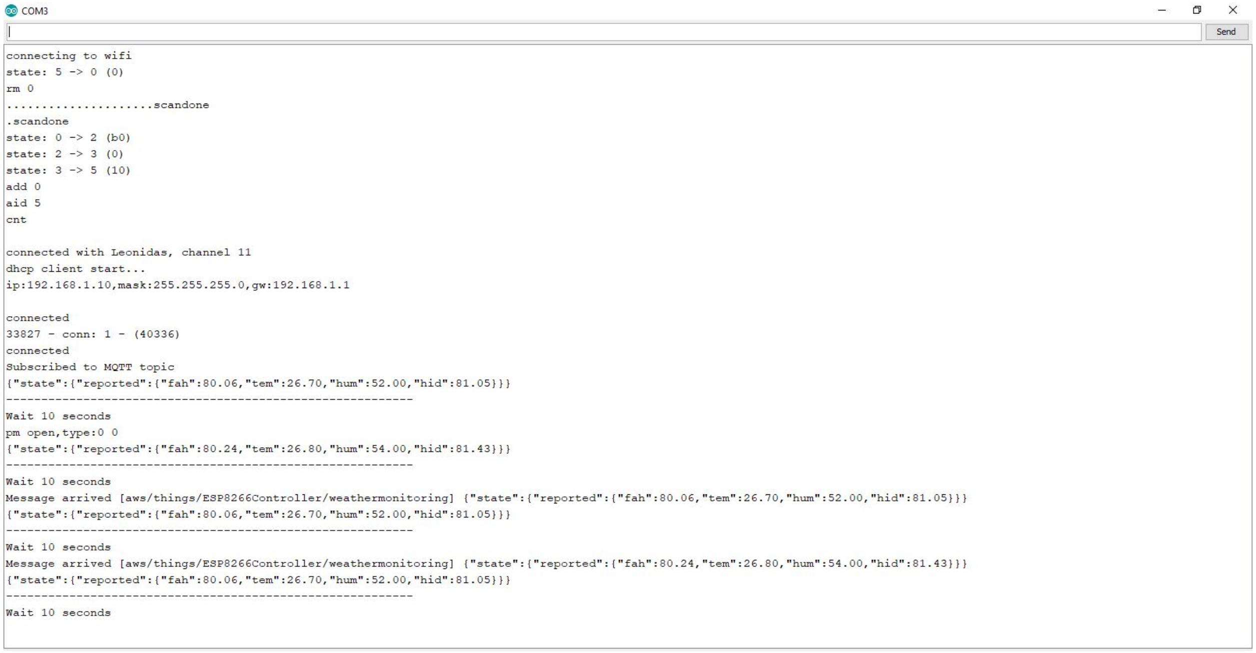
WeatherMonitoringMetricsToAwsIotShadow | Arduino 1.8.9

```
#include <Arduino.h>
#include <Stream.h>
#include <math.h>
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>

//AWS
#include "sha256.h"
#include "Utils.h"


//WEBSockets
#include <Hash.h>
#include <WebSocketsClient.h>

//MQTT PUBSUBCLIENT LIB
#include <PubSubClient.h>

//AWS MQTT Websocket
#include "Client.h"
#include "AWSWebSocketClient.h"
#include "CircularByteBuffer.h"

//DHT
```

Done uploading.

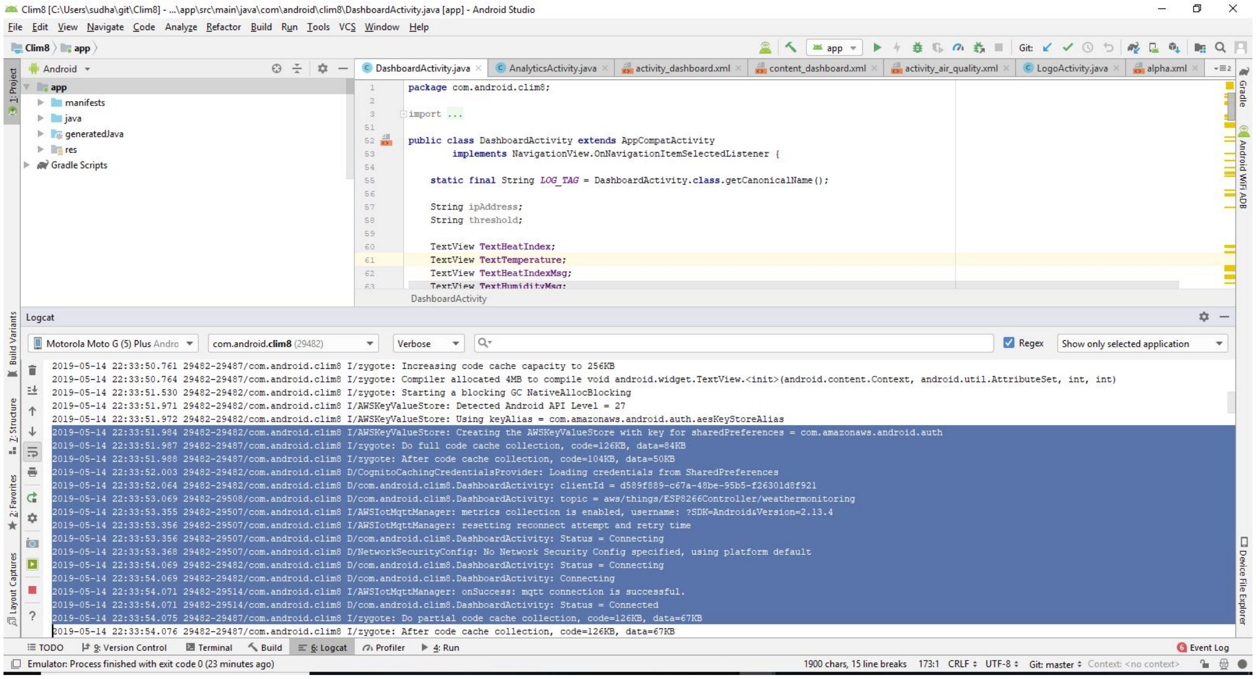
```
Sketch uses 356416 bytes (34%) of program storage space. Maximum is 1044464 bytes.
Global variables use 37644 bytes (45%) of dynamic memory, leaving 44276 bytes for local variables. Maximum is 81920 bytes.
Uploading 360560 bytes from C:\Users\sudha\AppData\Local\Temp\arduino_build_932419/WeatherMonitoringMetricsToAwsIotShadow.ino.bin to flash at 0x00000000
................................................................................ [ 22% ]
................................................................................ [ 45% ]
................................................................................ [ 67% ]
................................................................................ [ 90% ]
................................                                                 [ 100% ]
```

13. MQTT connection from NodeMCU to AWS IOT thing and sensor values are sent to AWS IOT shadow table.



```
connecting to wifi
state: 5 -> 0 (0)
rm 0
....................scandone
.scandone
state: 0 -> 2 (b0)
state: 2 -> 3 (0)
state: 3 -> 5 (10)
add 0
aid 5
cnt

connected with Leonidas, channel 11
dhcp client start...
ip:192.168.1.10,mask:255.255.255.0,gw:192.168.1.1

connected
33827 - conn: 1 - (40336)
connected
Subscribed to MQTT topic
{"state":{"reported":{"fah":80.06,"tem":26.70,"hum":52.00,"hid":81.05}}}
-----------------------------------------------------------
Wait 10 seconds
pm open,type:0 0
{"state":{"reported":{"fah":80.24,"tem":26.80,"hum":54.00,"hid":81.43}}}
-----------------------------------------------------------
Wait 10 seconds
Message arrived [aws/things/ESP8266Controller/weathermonitoring] {"state":{"reported":{"fah":80.06,"tem":26.70,"hum":52.00,"hid":81.05}}}
{"state":{"reported":{"fah":80.06,"tem":26.70,"hum":52.00,"hid":81.05}}}
-----------------------------------------------------------
Wait 10 seconds
Message arrived [aws/things/ESP8266Controller/weathermonitoring] {"state":{"reported":{"fah":80.24,"tem":26.80,"hum":54.00,"hid":81.43}}}
{"state":{"reported":{"fah":80.06,"tem":26.70,"hum":52.00,"hid":81.05}}}
-----------------------------------------------------------
Wait 10 seconds
```

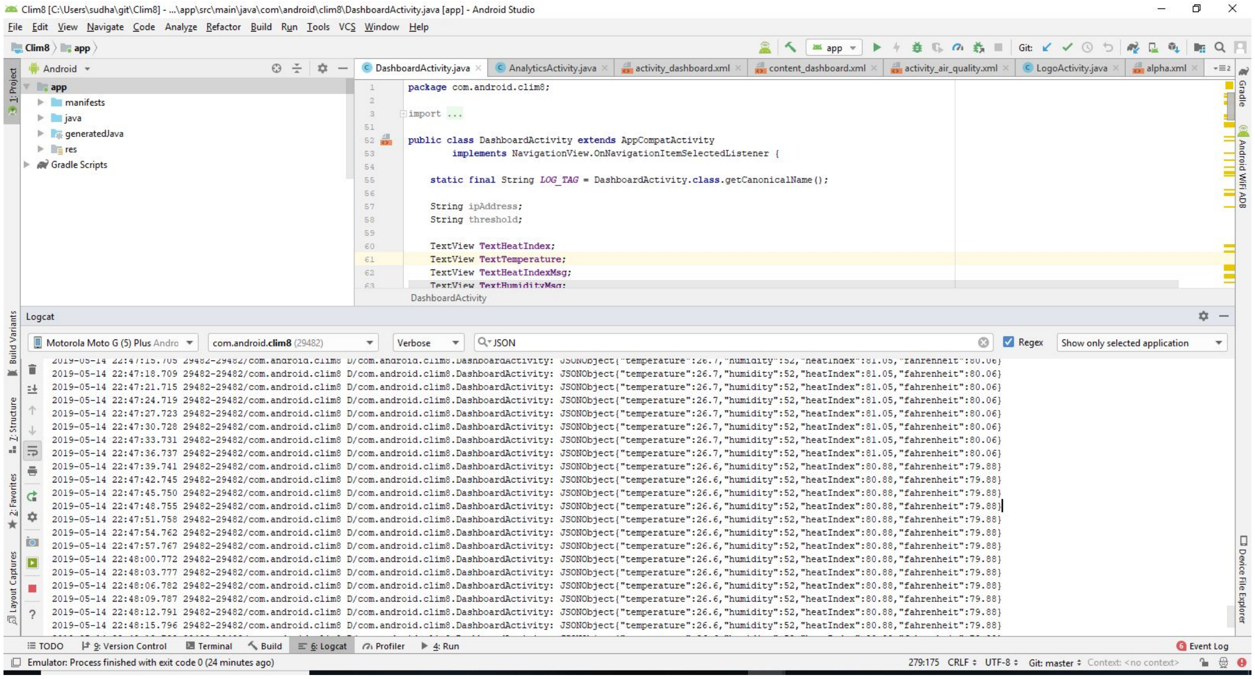
14. Android App upon starting is connected to AWS IOT and Subscribed to the same AWS IOT topic



```
package com.android.clim8;

import ...

public class DashboardActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {

    static final String LOG_TAG = DashboardActivity.class.getCanonicalName();

    String ipAddress;
    String threshold;

    TextView TextHeatIndex;
    TextView TextTemperature;
    TextView TextHeatIndexMsg;
    TextView TextHumidityMsg;
```

```
2019-05-14 22:33:50.761 29482-29487/com.android.clim8 I/zygote: Increasing code cache capacity to 256KB
2019-05-14 22:33:50.764 29482-29487/com.android.clim8 I/zygote: Compiler allocated 4MB to compile void android.widget.TextView.<init>(android.content.Context, android.util.AttributeSet, int, int)
2019-05-14 22:33:51.530 29482-29482/com.android.clim8 I/zygote: Starting a blocking GC NativeAllocBlocking
2019-05-14 22:33:51.971 29482-29482/com.android.clim8 I/AWSKeyValueStore: Detected Android API Level = 27
2019-05-14 22:33:51.972 29482-29482/com.android.clim8 I/AWSKeyValueStore: Using keyAlias = com.amazonaws.android.auth.aesKeyStoreAlias
2019-05-14 22:33:51.984 29482-29482/com.android.clim8 I/AWSKeyValueStore: Creating the AWSKeyValueStore with key for sharedPreferences = com.amazonaws.android.auth
2019-05-14 22:33:51.987 29482-29487/com.android.clim8 I/zygote: Do full code cache collection, code=126KB, data=84KB
2019-05-14 22:33:51.988 29482-29487/com.android.clim8 I/zygote: After code cache collection, code=104KB, data=50KB
2019-05-14 22:33:52.003 29482-29482/com.android.clim8 D/CognitoCachingCredentialsProvider: Loading credentials from SharedPreferences
2019-05-14 22:33:52.064 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: clientId = d589f889-c67a-48be-95b5-f26301d8f921
2019-05-14 22:33:53.069 29482-29508/com.android.clim8 D/com.android.clim8.DashboardActivity: topic = aws/things/ESP8266Controller/weathermonitoring
2019-05-14 22:33:53.355 29482-29507/com.android.clim8 I/AWSIotMqttManager: metrics collection is enabled, username: ?SDK=Android&Version=2.13.4
2019-05-14 22:33:53.356 29482-29507/com.android.clim8 I/AWSIotMqttManager: resetting reconnect attempt and retry time
2019-05-14 22:33:53.356 29482-29507/com.android.clim8 D/com.android.clim8.DashboardActivity: Status = Connecting
2019-05-14 22:33:53.368 29482-29507/com.android.clim8 D/NetworkSecurityConfig: No Network Security Config specified, using platform default
2019-05-14 22:33:54.069 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: Status = Connecting
2019-05-14 22:33:54.069 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: Connecting
2019-05-14 22:33:54.071 29482-29514/com.android.clim8 I/AWSIotMqttManager: onSuccess: mqtt connection is successful.
2019-05-14 22:33:54.071 29482-29514/com.android.clim8 D/com.android.clim8.DashboardActivity: Status = Connected
2019-05-14 22:33:54.075 29482-29487/com.android.clim8 I/zygote: Do partial code cache collection, code=126KB, data=67KB
2019-05-14 22:33:54.076 29482-29487/com.android.clim8 I/zygote: After code cache collection, code=126KB, data=67KB
```

15. MQTT messages are received from AWS IOT in Json Format at regular intervals


Clim8 [C:\Users\sudha\git\Clim8] - ...\app\src\main\java\com\android\clim8\DashboardActivity.java [app] - Android Studio

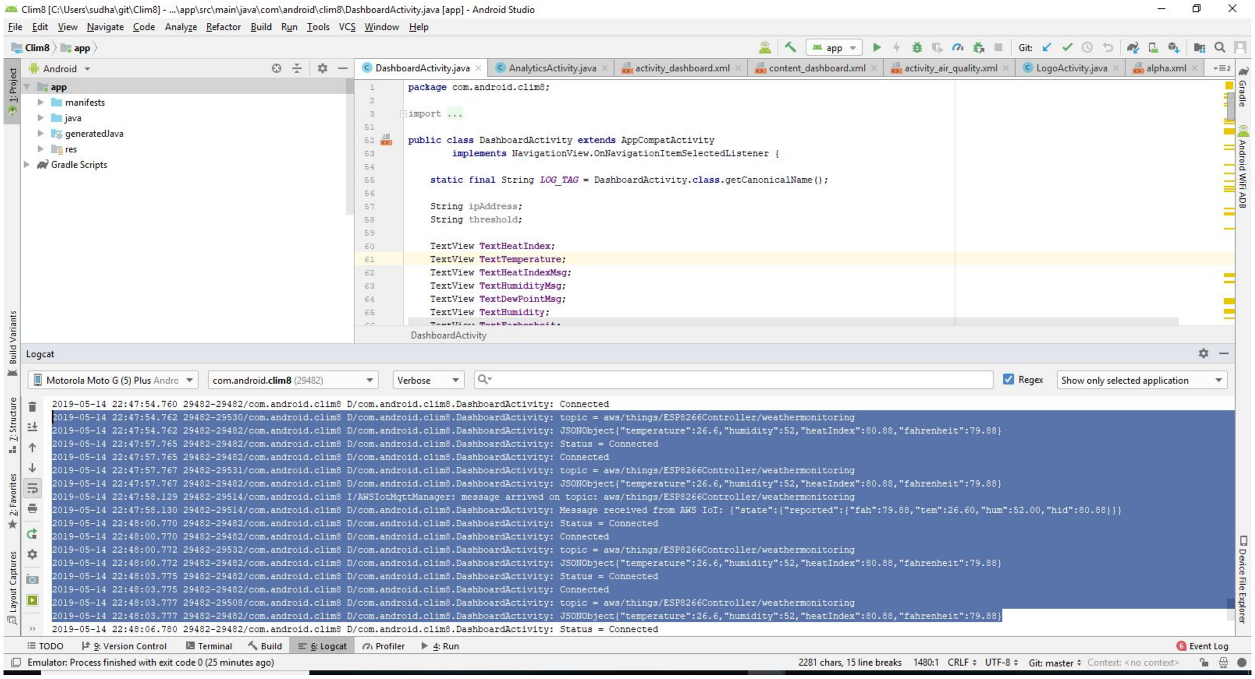
```
package com.android.clim8;

import ...

public class DashboardActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {

    static final String LOG_TAG = DashboardActivity.class.getCanonicalName();

    String ipAddress;
    String threshold;

    TextView TextHeatIndex;
    TextView TextTemperature;
    TextView TextHeatIndexMsg;
    TextView TextHumidityMsg;
```

Logcat: Motorola Moto G (5) Plus Andro | com.android.clim8 (29482) | Verbose | JSON | ☑ Regex | Show only selected application

```
2019-05-14 22:47:15.705 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.7,"humidity":52,"heatIndex":81.05,"fahrenheit":80.06}
2019-05-14 22:47:18.709 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.7,"humidity":52,"heatIndex":81.05,"fahrenheit":80.06}
2019-05-14 22:47:21.715 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.7,"humidity":52,"heatIndex":81.05,"fahrenheit":80.06}
2019-05-14 22:47:24.719 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.7,"humidity":52,"heatIndex":81.05,"fahrenheit":80.06}
2019-05-14 22:47:27.723 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.7,"humidity":52,"heatIndex":81.05,"fahrenheit":80.06}
2019-05-14 22:47:30.728 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.7,"humidity":52,"heatIndex":81.05,"fahrenheit":80.06}
2019-05-14 22:47:33.731 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.7,"humidity":52,"heatIndex":81.05,"fahrenheit":80.06}
2019-05-14 22:47:36.737 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.7,"humidity":52,"heatIndex":81.05,"fahrenheit":80.06}
2019-05-14 22:47:39.741 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
2019-05-14 22:47:42.745 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
2019-05-14 22:47:45.750 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
2019-05-14 22:47:48.755 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
2019-05-14 22:47:51.758 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
2019-05-14 22:47:54.762 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
2019-05-14 22:47:57.767 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
2019-05-14 22:48:00.772 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
2019-05-14 22:48:03.777 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
2019-05-14 22:48:06.782 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
2019-05-14 22:48:09.787 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
2019-05-14 22:48:12.791 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
2019-05-14 22:48:15.796 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
```

Emulator: Process finished with exit code 0 (24 minutes ago)

16. Verbose output displayed overwriting existing values


Clim8 [C:\Users\sudha\git\Clim8] - ...\app\src\main\java\com\android\clim8\DashboardActivity.java [app] - Android Studio

```
package com.android.clim8;

import ...

public class DashboardActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {

    static final String LOG_TAG = DashboardActivity.class.getCanonicalName();

    String ipAddress;
    String threshold;

    TextView TextHeatIndex;
    TextView TextTemperature;
    TextView TextHeatIndexMsg;
    TextView TextHumidityMsg;
    TextView TextDewPointMsg;
    TextView TextHumidity;
```

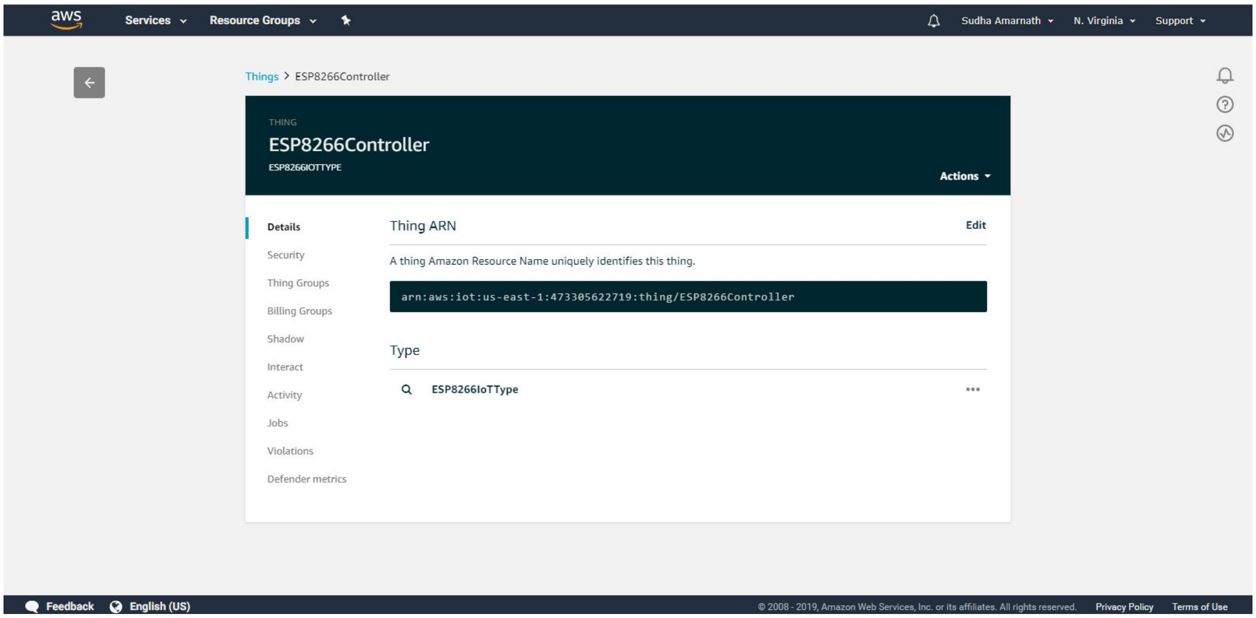
Logcat: Motorola Moto G (5) Plus Andro | com.android.clim8 (29482) | Verbose | ☑ Regex | Show only selected application

```
2019-05-14 22:47:54.760 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: Connected
2019-05-14 22:47:54.762 29482-29530/com.android.clim8 D/com.android.clim8.DashboardActivity: topic = aws/things/ESP8266Controller/weathermonitoring
2019-05-14 22:47:54.762 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
2019-05-14 22:47:57.765 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: Status = Connected
2019-05-14 22:47:57.765 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: Connected
2019-05-14 22:47:57.767 29482-29531/com.android.clim8 D/com.android.clim8.DashboardActivity: topic = aws/things/ESP8266Controller/weathermonitoring
2019-05-14 22:47:57.767 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
2019-05-14 22:47:58.129 29482-29514/com.android.clim8 I/AWSIotMqttManager: message arrived on topic: aws/things/ESP8266Controller/weathermonitoring
2019-05-14 22:47:58.130 29482-29514/com.android.clim8 D/com.android.clim8.DashboardActivity: Message received from AWS IoT: {"state":{"reported":{"fah":79.88,"tem":26.60,"hum":52.00,"hid":80.88}}}
2019-05-14 22:48:00.770 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: Status = Connected
2019-05-14 22:48:00.770 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: Connected
2019-05-14 22:48:00.772 29482-29532/com.android.clim8 D/com.android.clim8.DashboardActivity: topic = aws/things/ESP8266Controller/weathermonitoring
2019-05-14 22:48:00.772 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
2019-05-14 22:48:03.775 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: Status = Connected
2019-05-14 22:48:03.775 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: Connected
2019-05-14 22:48:03.777 29482-29508/com.android.clim8 D/com.android.clim8.DashboardActivity: topic = aws/things/ESP8266Controller/weathermonitoring
2019-05-14 22:48:03.777 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: JSONObject{"temperature":26.6,"humidity":52,"heatIndex":80.88,"fahrenheit":79.88}
2019-05-14 22:48:06.780 29482-29482/com.android.clim8 D/com.android.clim8.DashboardActivity: Status = Connected
```

Emulator: Process finished with exit code 0 (25 minutes ago)

2281 chars, 15 line breaks 1480:1 CRLF UTF-8 Git: master

17. IOT - Thing Configured in AWS with topic as shown in this output



THING

ESP8266Controller

ESP8266IOTTYPE

Thing ARN

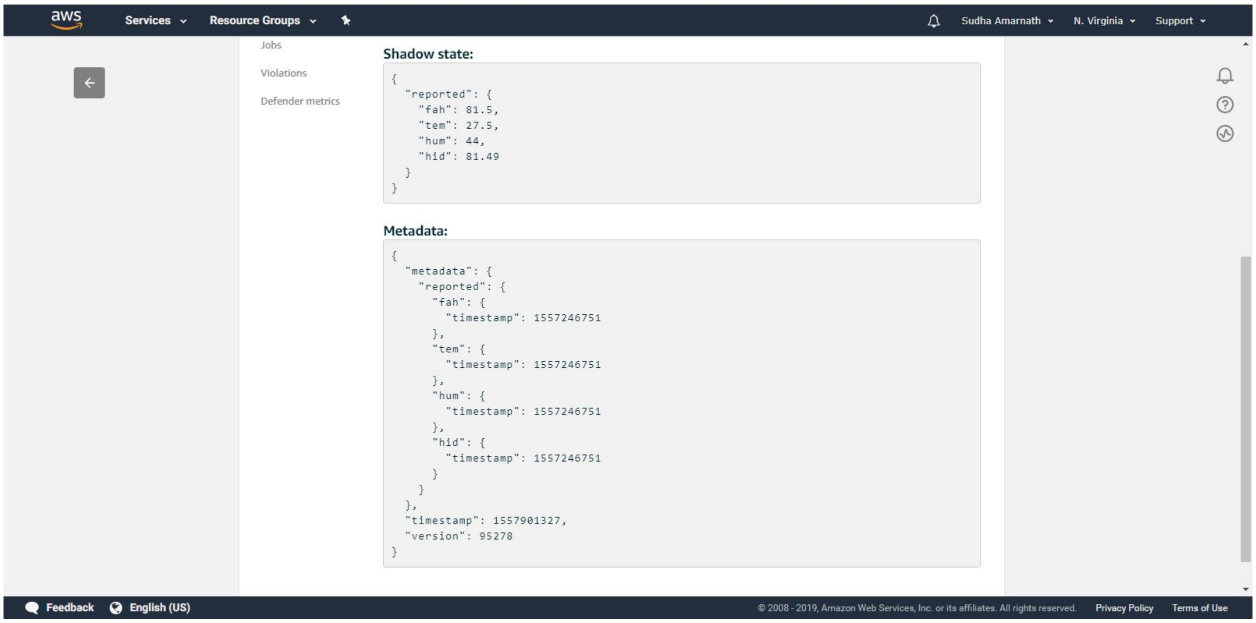
A thing Amazon Resource Name uniquely identifies this thing.

```
arn:aws:iot:us-east-1:473305622719:thing/ESP8266Controller
```

Type

ESP8266IoTType

18. AWS Shadow table



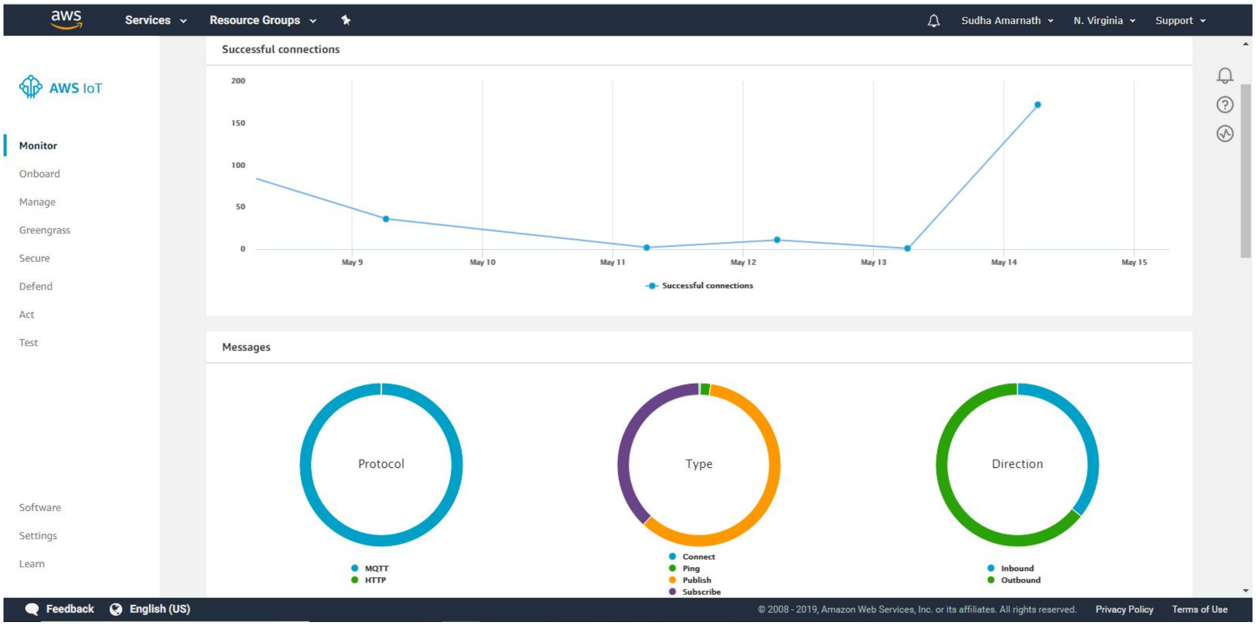
Shadow state:

```
{
  "reported": {
    "fah": 81.5,
    "tem": 27.5,
    "hum": 44,
    "hid": 81.49
  }
}
```

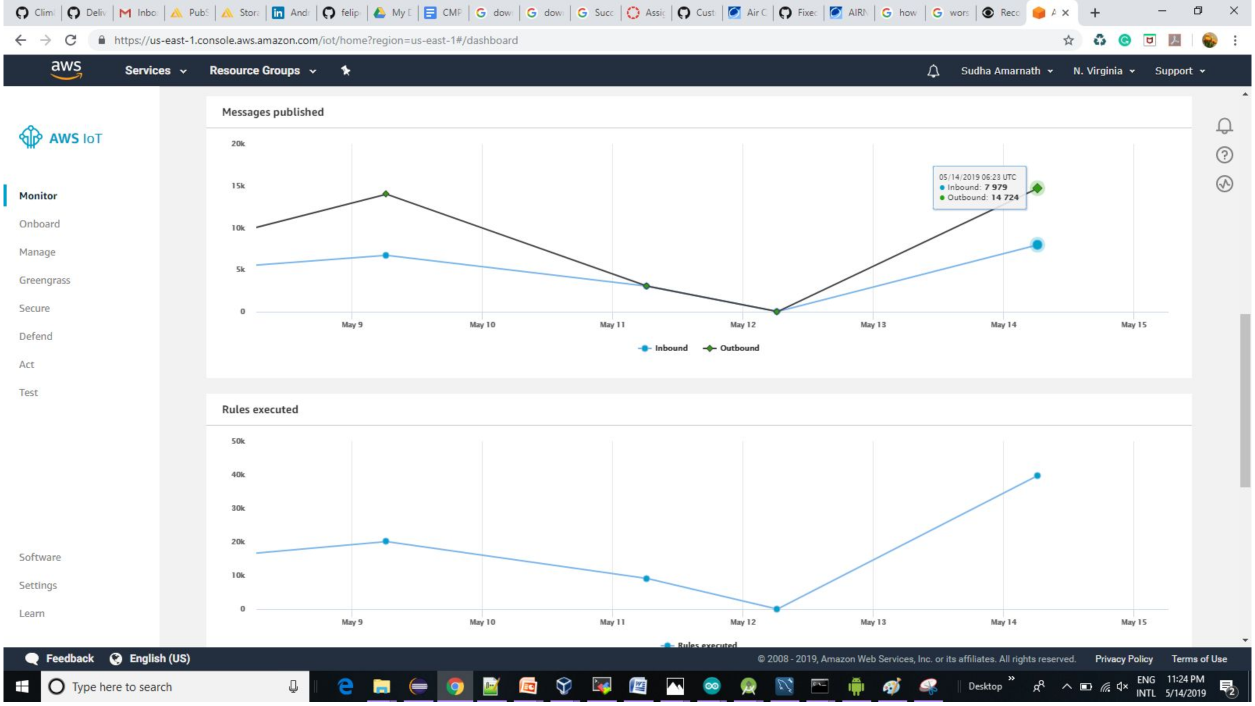
Metadata:

```
{
  "metadata": {
    "reported": {
      "fah": {
        "timestamp": 1557246751
      },
      "tem": {
        "timestamp": 1557246751
      },
      "hum": {
        "timestamp": 1557246751
      },
      "hid": {
        "timestamp": 1557246751
      }
    }
  },
  "timestamp": 1557901327,
  "version": 95278
}
```

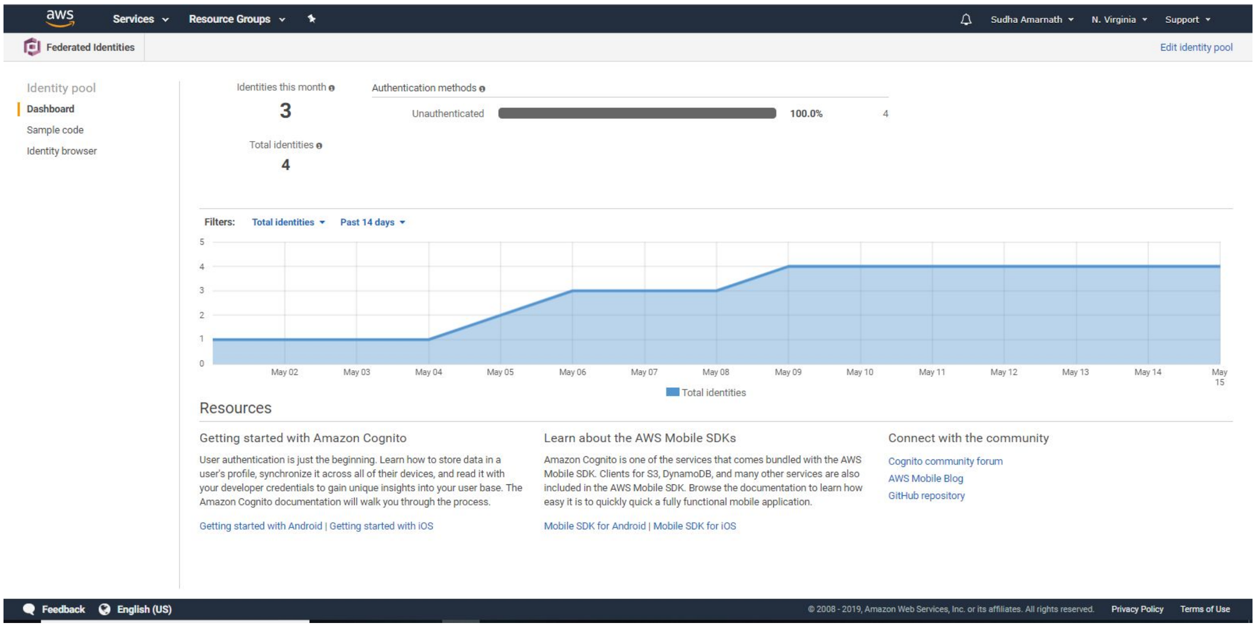
19. MQTT connections establishment from the devices. Usage is shown in AWS



20. The number of MQTT messages received and advertised to the IOT devices subscribed to the IOT topic



20. Federated Identities for the IOT Thing in AWS Cognito. This is used to access AWS IOT Topic from the Andriod Apps



Identities this month
3

Total identities
4

Authentication methods

Unauthenticated 100.0% 4

Resources

Getting started with Amazon Cognito

User authentication is just the beginning. Learn how to store data in a user's profile, synchronize it across all of their devices, and read it with your developer credentials to gain unique insights into your user base. The Amazon Cognito documentation will walk you through the process.

Getting started with Android | Getting started with iOS

Learn about the AWS Mobile SDKs

Amazon Cognito is one of the services that comes bundled with the AWS Mobile SDK. Clients for S3, DynamoDB, and many other services are also included in the AWS Mobile SDK. Browse the documentation to learn how easy it is to quickly quick a fully functional mobile application.

Mobile SDK for Android | Mobile SDK for iOS

Connect with the community

Cognito community forum
AWS Mobile Blog
GitHub repository

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

GITHUB SOURCE CODE

https://github.com/muthu-05/Clim8

REFERENCES

https://learn.adafruit.com/dht/overview