Class:	IoT Systems	**Semester:**	Fall 2024
Project topic:	Ambiance Monitoring and Music Recommendation System		
Student:	*Abraham Garcia*		

1. Project description

a. General idea of your IoT system

The Ambiance Monitoring and Music Recommendation System is an IoT system made to enhance users mood and vibes by tailoring music recommendations to the local weather. It uses a temperature sensor (TMP36) and a humidity sensor (HPP801A031) to determine local weather conditions, each sensor is connected to a dedicated node that communicates data to the cloud using MQTT network protocol. The system will use ThingSpeak cloud services for data storage, analysis, and processing to trigger a Spotify API applet to recommend or play a Spotify playlist to match the mood. ThingSpeak cloud services provide a dashboard to monitor real time data trend visuals.

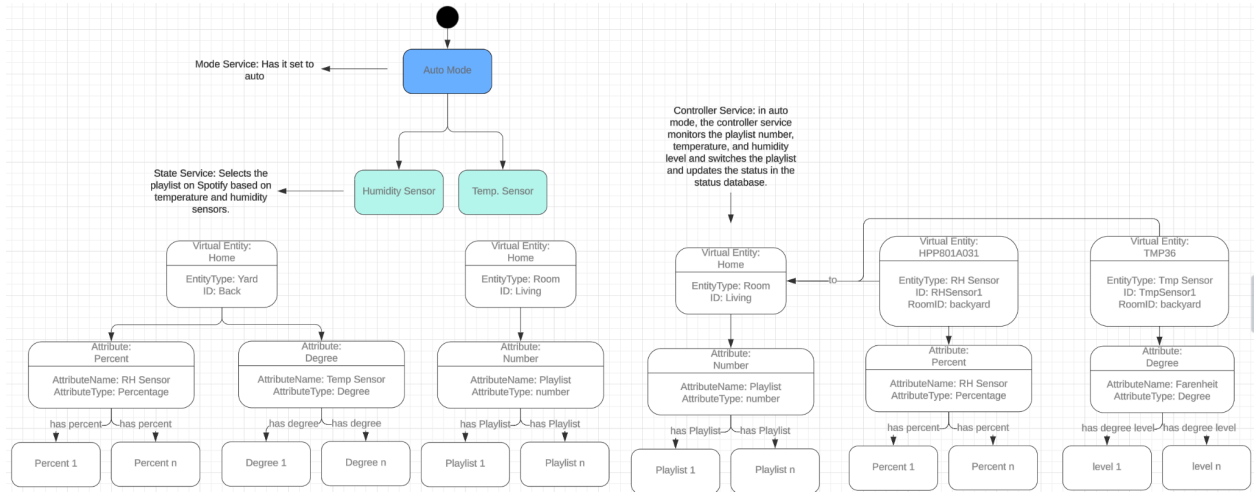
b. System outcomes (profits to the end user)

When certain weather conditions are met the system plays/recommends a playlist on Spotify, matching the mood set by the local weather.

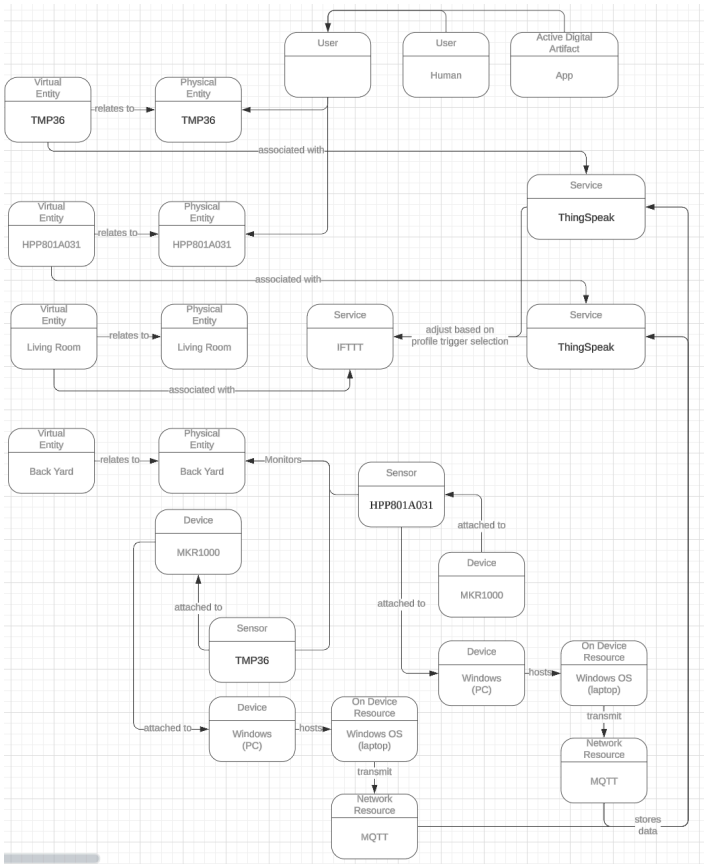
c. Technical description

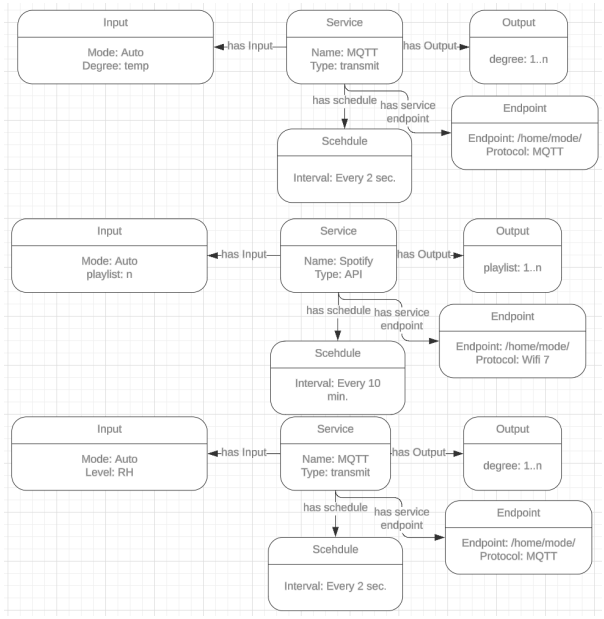
The IoT system will use a temperature sensor (TMP36) and a humidity sensor (HPP801A031) each connected to an Arduino MKR1000. On the MKR1000 the reading pin for humidity is connected to pin A2 and temperature is connected to A1. The humidity sensor (HPP801A031) is two pins with one connected to ground and the other connected to a 1MOhm resistor and pin A2 of the MKR1000 board, with the other side of the resistor connected to ground. The temperature sensor (TMP36) is connected from left to right: VCC, A1,and ground. Because the boards have the same name and the names conflict when I try to run them both on the same computer each MKR1000 node will be connected to a different computer on the same network.

d. Diagrams

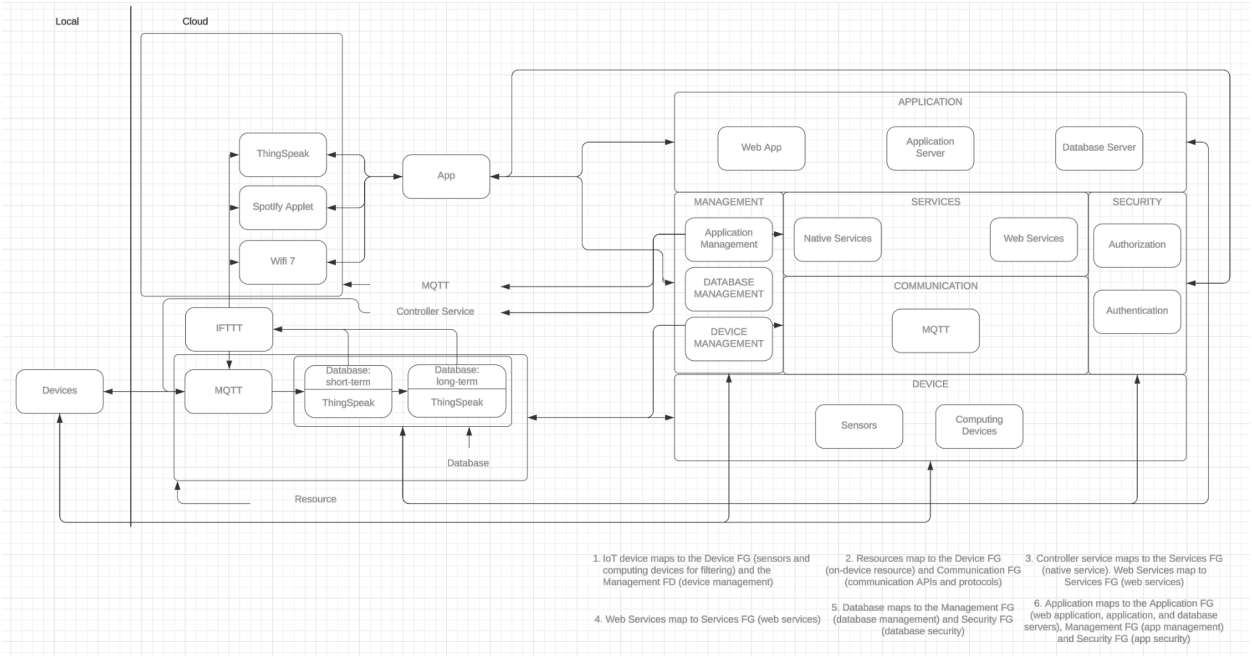


IoT Systems – Final project





IoT Systems – Final project



e. List of sensors

Vendor	Model	Comment
Adafruit	TMP36	https://www.adafruit.com/product/165
DigiKey	HPP801A031	https://www.digikey.com/en/products/detail/te-connectivity-measurement-specialties/HPP801A031/697731

f. List of nodes

Vendor	Model	Comment
Arduino	MKR1000	https://store-usa.arduino.cc/products/arduino-mkr1000-wifi?srsltid=AfmBOoo0s64_zQp0nLHiS4ST0EVpzSIQGZuaVH1xNOFseiecax4nTBiO
Arduino	MKR1000	https://store-usa.arduino.cc/products/arduino-mkr1000-wifi?srsltid=AfmBOoo0s64_zQp0nLHiS4ST0EVpzSIQGZuaVH1xNOFseiecax4nTBiO

g. List of other hardware components

Vendor	Model	Comment
	31 breadboard piece x2	
	Resistors 1M	
Sparkfun	10 M-M jumper cables	https://www.sparkfun.com/products/12796
Sparkfun	Jumper wire kit	https://www.sparkfun.com/products/124
Sparkfun	x2 USB cable Micro-B	https://www.sparkfun.com/products/13244

h. Selected cloud service

ThingSpeak and IFTTT

2. Project operation

a. Describe how your project works

The Ambiance Monitoring and Music Recommendation System is an IoT system uses a temperature sensor (TMP36) and a humidity sensor (HPP801A031) to determine local weather conditions, each sensor is connected to a dedicated node that communicates data to the cloud using MQTT network protocol. The system will use ThingSpeak cloud services for data storage, analysis, and processing to trigger a Spotify API applet to recommend or play a Spotify playlist to match the mood.

The IoT system will use a temperature sensor (TMP36) and a humidity sensor (HPP801A031) each connected to an Arduino MKR1000. On the MKR1000 the reading pin for humidity is connected to pin A2 and temperature is connected to A1. The humidity sensor (HPP801A031) is two pins with one connected to ground and the other connected to a 1MOhm resistor and pin A2 of the MKR1000 board, with the other side of the resistor connected to ground. The temperature sensor (TMP36) is connected from left to right: VCC, A1,and ground. Because the boards have the same name and the names conflict when I try to run them both on the same computer each MKR1000 node will be connected to a different computer on the same network.

b. Describe what's the data flow in your project

The data gets posted to two different topics in the same channel depending on if it's a humidity or temperature reading. They then get posted to mqtt channel with temperature going first then humidity. The subscriber is on a different device on the same network and reads the data by subscribing to the channel topics. At the send time the data is being sent to the subscriber the data is also being sent to ThingSpeak to trigger a webhook.

i. specify publisher(s)
   temp
   humid
ii. specify subscriber(s)
   tmp
   hum
iii. specify the location and type of MQTT broker
https://test.mosquitto.org/, port: 1883 (i'm not giving my home ip address)
iv. specify the hierarchy / list of MQTT channels used
   There is only one channel sending both the humid and temp topic values. It sends first the temperature value then the humidity value.

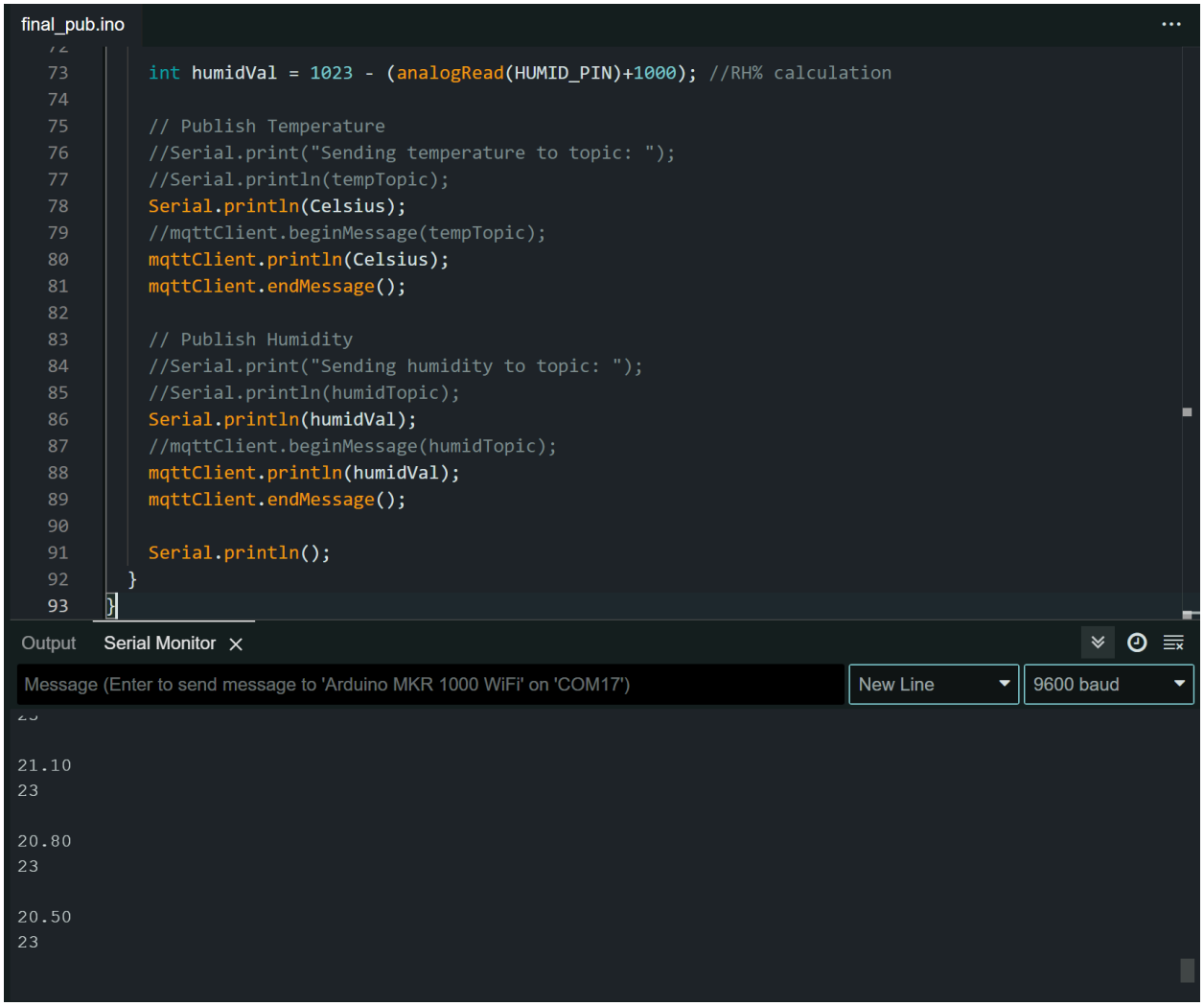
c. Describe cloud operation, including what's the data processing mechanism

ThingSpeak receives data at the same time the MQTT broker receives it from the sensor nodes. The data received from the node is displayed on the MATLAB visualization dashboard. Using MATLAB Analysis to program a calculation of the

weather conditions and send a trigger using ThingHTTP to IFTTT to trigger the Spotify applet to recommend a playlist.

3. Screenshots / pictures

a. Publisher(s) operation (data being sent to MQTT broker)



b. Subscriber(s) operation (data being received from MQTT broker)

temp:

IoT Systems – Final project

```
'Client null sending CONNECT
Client null received CONNACK (0)
Client null sending SUBSCRIBE (Mid: 1, Topic: temp, QoS: 0, Options: 0x00)
Client null received SUBACK
Subscribed (mid: 1): 0
Client null received PUBLISH (d0, q0, r0, m0, 'temp', ... (7 bytes))
24.70

Client null received PUBLISH (d0, q0, r0, m0, 'temp', ... (7 bytes))
24.40

Client null received PUBLISH (d0, q0, r0, m0, 'temp', ... (7 bytes))
23.80

Client null received PUBLISH (d0, q0, r0, m0, 'temp', ... (7 bytes))
24.10

Client null received PUBLISH (d0, q0, r0, m0, 'temp', ... (7 bytes))
24.40
```

humid:

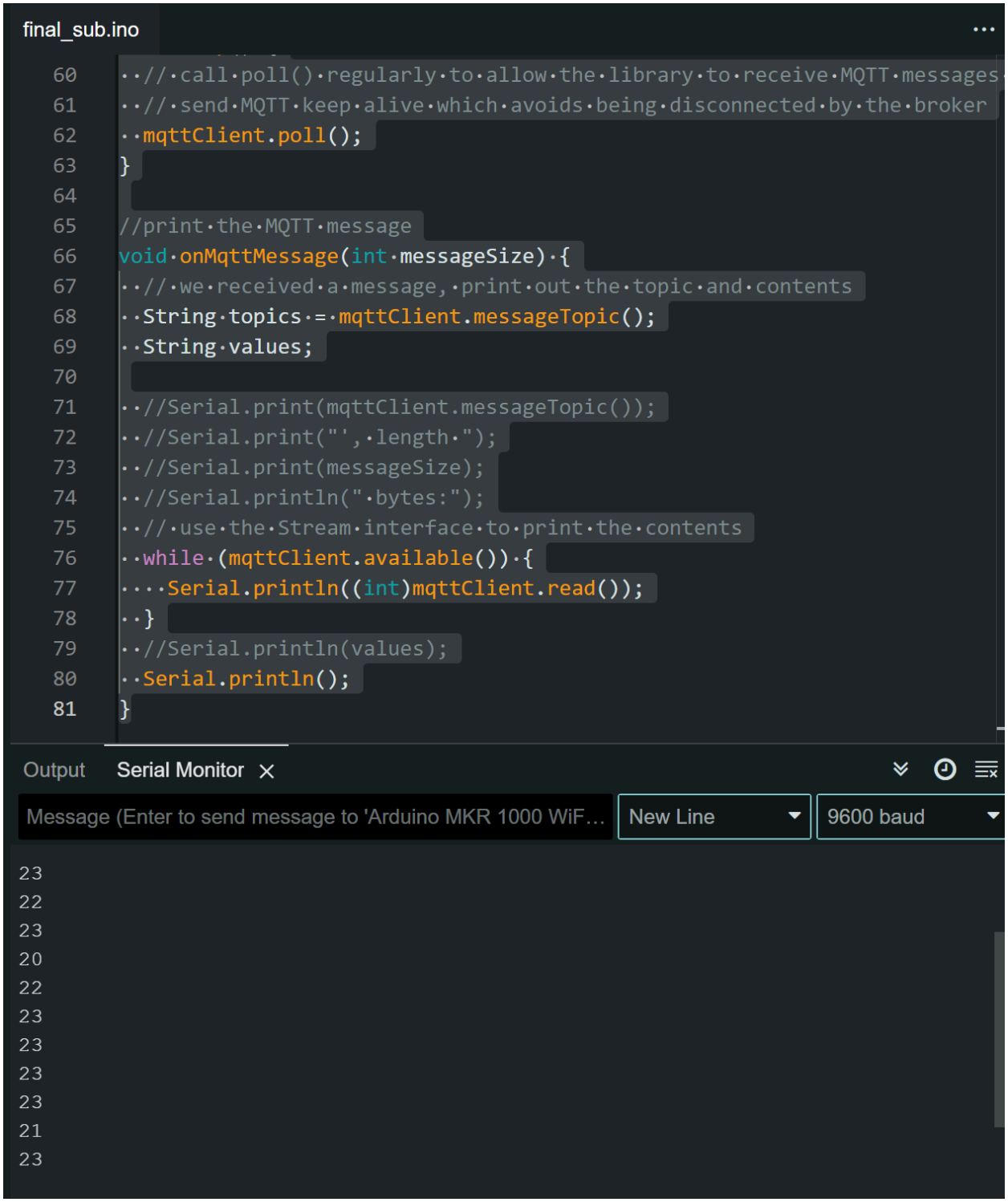
```
'Client null sending CONNECT
Client null received CONNACK (0)
Client null sending SUBSCRIBE (Mid: 1, Topic: humid, QoS: 0, Options: 0x00)
Client null received SUBACK
Subscribed (mid: 1): 0
Client null received PUBLISH (d0, q0, r0, m0, 'humid', ... (4 bytes))
22

Client null received PUBLISH (d0, q0, r0, m0, 'humid', ... (4 bytes))
22

Client null received PUBLISH (d0, q0, r0, m0, 'humid', ... (4 bytes))
21

Client null received PUBLISH (d0, q0, r0, m0, 'humid', ... (4 bytes))
22

Client null received PUBLISH (d0, q0, r0, m0, 'humid', ... (4 bytes))
20
```

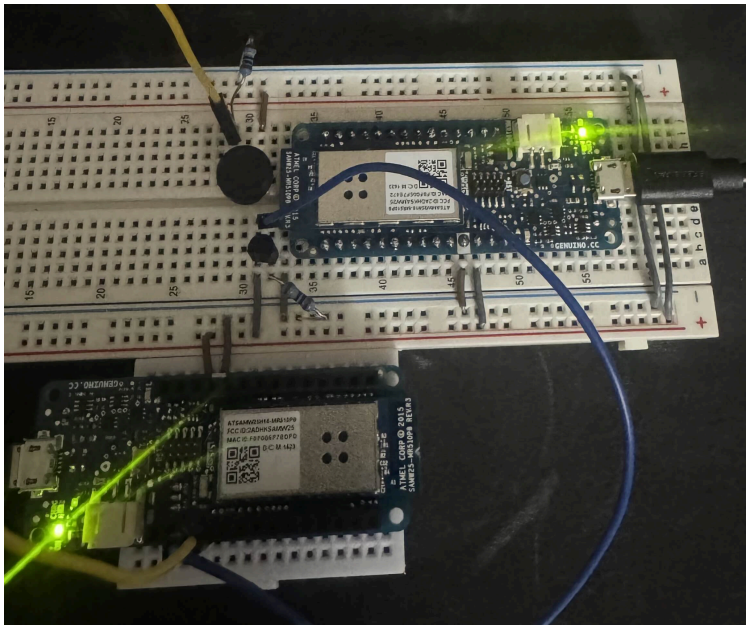


c. Log of the channels activity (if broker permits)

```
temp 23
temp 23
temp 23
temp 22
temp 23
temp 23
temp 23
temp 23
temp 23
temp 23
temp 22
temp 23
temp 23
temp 23
```

```
humid 23
humid 21
humid 20
humid 23
humid 23
humid 22
humid 23
humid 23
humid 23
humid 23
```

d. Picture(s) of wired circuit



4. Describe technical challenges you had and how you solved them

My first challenge was to get the subscriber ide to read the mqtt topic values because for some strange reason the terminal on the subscriber computer could read the data from the publisher computer but not in the arduino ide. I fixed it by changing the baud rate somehow.

5. Code listings for each node and cloud codes

final_pub:

```
#include <ArduinoMqttClient.h>
#include <WiFi101.h>

#define tempPin A1 //pin to read tmp
#define HUMID_PIN A2 //pin to read humid

char ssid[] = "SSID"; //network SSID, not sharing my network
char pass[] = "pass"; //network password, nor my password from home

//////////////////////////////////////////////////////////////////
unsigned long RH_Channel_No = 2762875;
const char * RH_WriteAPIKey = "VVXJEUB0OUPA5PH2";
unsigned long TMP_Channel_No = 2737117;
const char * TMP_WriteAPIKey = "Q68IWIYMTHIAA9RR";

WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);

const char broker[] = "ip_address"; //not sharing personal ip address
int port = 1883; //MQTT broker port

//MQTT topics
const char tempTopic[] = "temp";
const char humidTopic[] = "humid";

const long interval = 2000; //interval to send msg
unsigned long previousMillis = 0;


void setup() {
  //Initialize serial and wait for port to open:
```

```
  Serial.begin(9600);

  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  // attempt to connect to Wifi network:
  Serial.print("Attempting to connect to WPA SSID: ");
  Serial.println(ssid);

  while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
    // failed, retry
    Serial.print(".");
    delay(5000);
  }

  Serial.println("You're connected to the network");
  Serial.println();

  //connection to the broker
  Serial.print("Attempting to connect to the MQTT broker: ");
  Serial.println(broker);

  //connection to the broker failed
  if (!mqttClient.connect(broker, port)) {
    Serial.print("MQTT connection failed! Error code = ");
    Serial.println(mqttClient.connectError());
    while (1);
  }

  Serial.println("You're connected to the MQTT broker!");
  Serial.println();
}

void loop() {
  // call poll() regularly to allow the library to send MQTT keep alive
which
  // avoids being disconnected by the broker
  mqttClient.poll();
  unsigned long currentMillis = millis();
```

```
  if (currentMillis - previousMillis >= interval) {
    // save the last time a message was sent
    previousMillis = currentMillis;
    //record sensor valuse
    float heat = analogRead(tempPin); //analog read of tmp
    float v = heat * (3300/1024); //convert volt to K
    float Celsius =(v - 500 ) / 10; //temp in C

    int humidVal = 1023 - (analogRead(HUMID_PIN)+1000); //RH% calculation

    // Publish Temperature
    //Serial.print("Sending temperature to topic: ");
    //Serial.println(tempTopic);
    Serial.println(Celsius);
    //mqttClient.beginMessage(tempTopic);
    mqttClient.println(Celsius);
    mqttClient.endMessage();

    // Publish Humidity
    //Serial.print("Sending humidity to topic: ");
    //Serial.println(humidTopic);
    Serial.println(humidVal);
    //mqttClient.beginMessage(humidTopic);
    mqttClient.println(humidVal);
    mqttClient.endMessage();

    //ThingSpeak.setField(2, Celsius);////////////////////////////
    //ThingSpeak.writeFields(TMP_Channel_No, TMP_WriteAPIKey);/////
    //ThingSpeak.setField(1, humid_val);//////////////////////////
    //ThingSpeak.writeFields(RH_Channel_No, RH_WriteAPIKey);////

  }
}
```

final_sub:

```
#include <ArduinoMqttClient.h>
#include <WiFi101.h>
//#include "secret_arduino.h" //contained the ssid and password of the
WiFi
```

```
//WiFi that you want to be connected
char ssid[] = "ssd"; // your network SSID
char pass[] = "pass"; // your network password

WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);

const char broker[] = "IP_Address"; //i'm not sharing my ip address
int port = 1883; //channel of the broker

//topics, you can change the name
const char topic[] = "tmp";
const char topic2[] = "hum";
//const char topic3[] = "real_unique_topic_3";

void setup() {
  //Initialize serial and wait for port to open:
  Serial.begin(57600);

  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  // attempt to connect to Wifi network:
  Serial.print("Attempting to connect to SSID: ");
  Serial.println(ssid);
  while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
    // failed, retry
    Serial.print(".");
    delay(5000);
  }

  Serial.println("You're connected to the network");
  Serial.println();
  //connection to the broker
  Serial.print("Attempting to connect to the MQTT broker: ");
  Serial.println(broker);

  //connection to the broker failed
  if (!mqttClient.connect(broker, port)) {
```

```
    Serial.print("MQTT connection failed! Error code = ");
    Serial.println(mqttClient.connectError());
    while (1);
  }

  Serial.println("You're connected to the MQTT broker!");
  Serial.println();

  // subscribe to a topic
  mqttClient.subscribe(topic);
  mqttClient.subscribe(topic2);
  mqttClient.onMessage(onMqttMessage);
}

void loop() {
   // call poll() regularly to allow the library to receive MQTT messages
and
  // send MQTT keep alive which avoids being disconnected by the broker
  mqttClient.poll();
}

//print the MQTT message
void onMqttMessage(int messageSize) {
  // we received a message, print out the topic and contents
  String topics = mqttClient.messageTopic();
  String values;

  //Serial.print(mqttClient.messageTopic());
  //Serial.print("', length ");
  //Serial.print(messageSize);
  //Serial.println(" bytes:");
  // use the Stream interface to print the contents
  while (mqttClient.available()) {
    Serial.println((int)mqttClient.read());
  }
  //Serial.println(values);
  Serial.println();
}
```

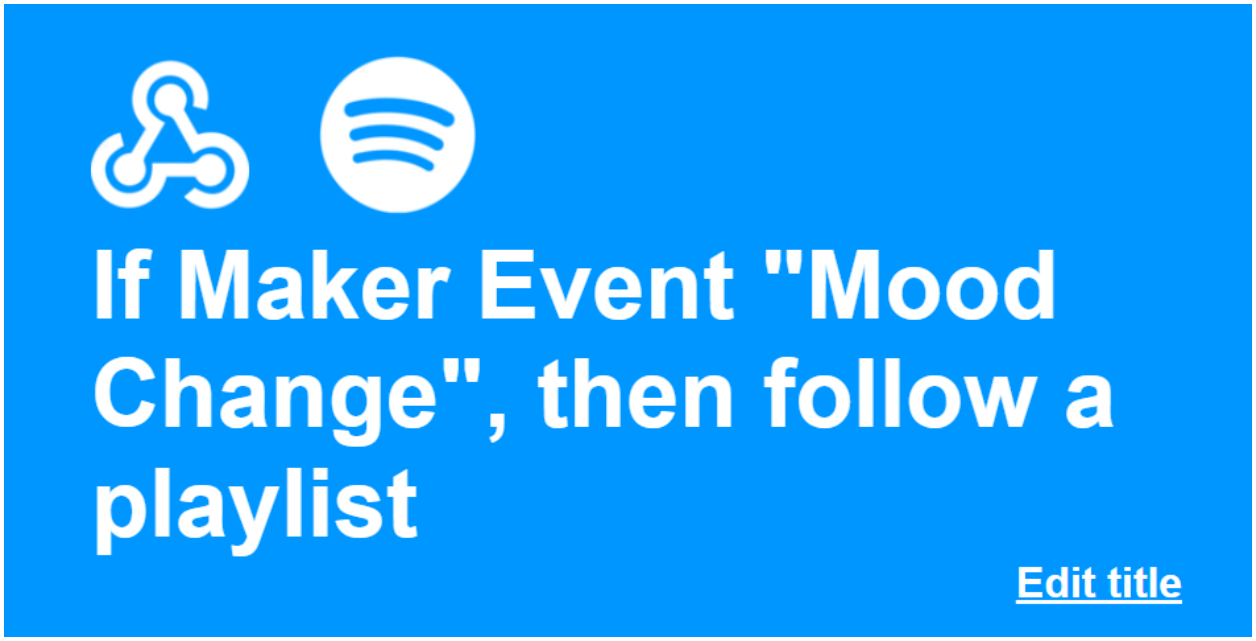
MATLAB ANALYSIS:

```
readAPIKey = 'WU6ABUMH50QRC20V'; % Replace with your ThingSpeak Read API
Key
channelID = 2762875; % Replace with your ThingSpeak Channel ID
threshold1 = 20; % Humidity Threshold value and
threshold2 = 20; % Temperature Threshold value to trigger Spotify
playlisy follow

% IFTTT Webhook URL
iftttWebhookURL =
'https://maker.ifttt.com/trigger/Mood_Change/with/key/cvRPM4rxwvQmnVOQlZ
smSVweF0uS32lldGbjvuo3xZ4';

% Read data from both fields
field1 = thingSpeakRead(channelID, 'Fields', 1, 'ReadKey', readAPIKey);
field2 = thingSpeakRead(channelID, 'Fields', 2, 'ReadKey', readAPIKey);

% Check if both thresholds are met
if isscalar(field1) && isscalar(field2) && field1 > threshold1 && field2
> threshold2
       webwrite(iftttWebhookURL, 'value1', field1, 'value2', field2); %
trigger the webhook
   disp('Webhook triggered.');
else
   disp('Thresholds not met.');
end
```



IoT Systems – Final project