Class:	IoT Systems	**Semester:**	Fall 2022
Project topic:	Final Project Auto-Window Open/Close		
Student:	Dylan Flores		

1. Project description

a. General idea of your IoT system

To preserve energy costs in AC generally around the area or even heat up the room when the sun is outside, the basic building block is to open a window automatically based on the set temperature or humidity inside and if the outside weather is too extreme then the window remains closed. However, if the conditions inside the room are more extreme, then the window will open and will allow the cooler breeze to dry out the stuffy ambience of the room and cool down the temperature so the room feels more comfortable. These settings can be changed however if desired and can be repurposed to open if it is too cold or too hot, or too humid or too dry.

b. System outcomes (profits to the end user)

The user will be able to save energy costs by spending less on AC and heater utilities by opening a window to let the fresh breeze in and will automatically close once a certain level of humidity is detected or a certain level of temperature is hit. The device also saves energy in the actuators by not being triggered unless the microcontroller tells it to. Each minute or so, a cluster of data from the sensors installed will showcase the live environmental feel from the room and store it onto the cloud for the user to see.

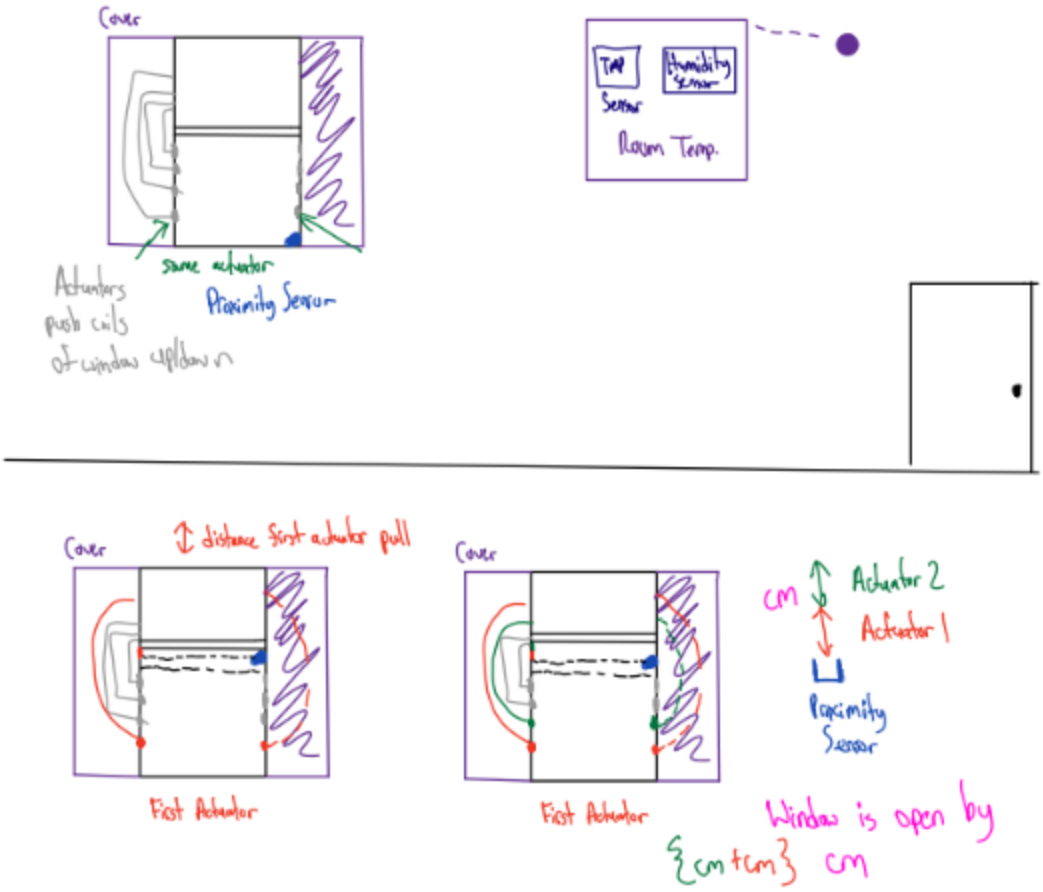
c. Technical description

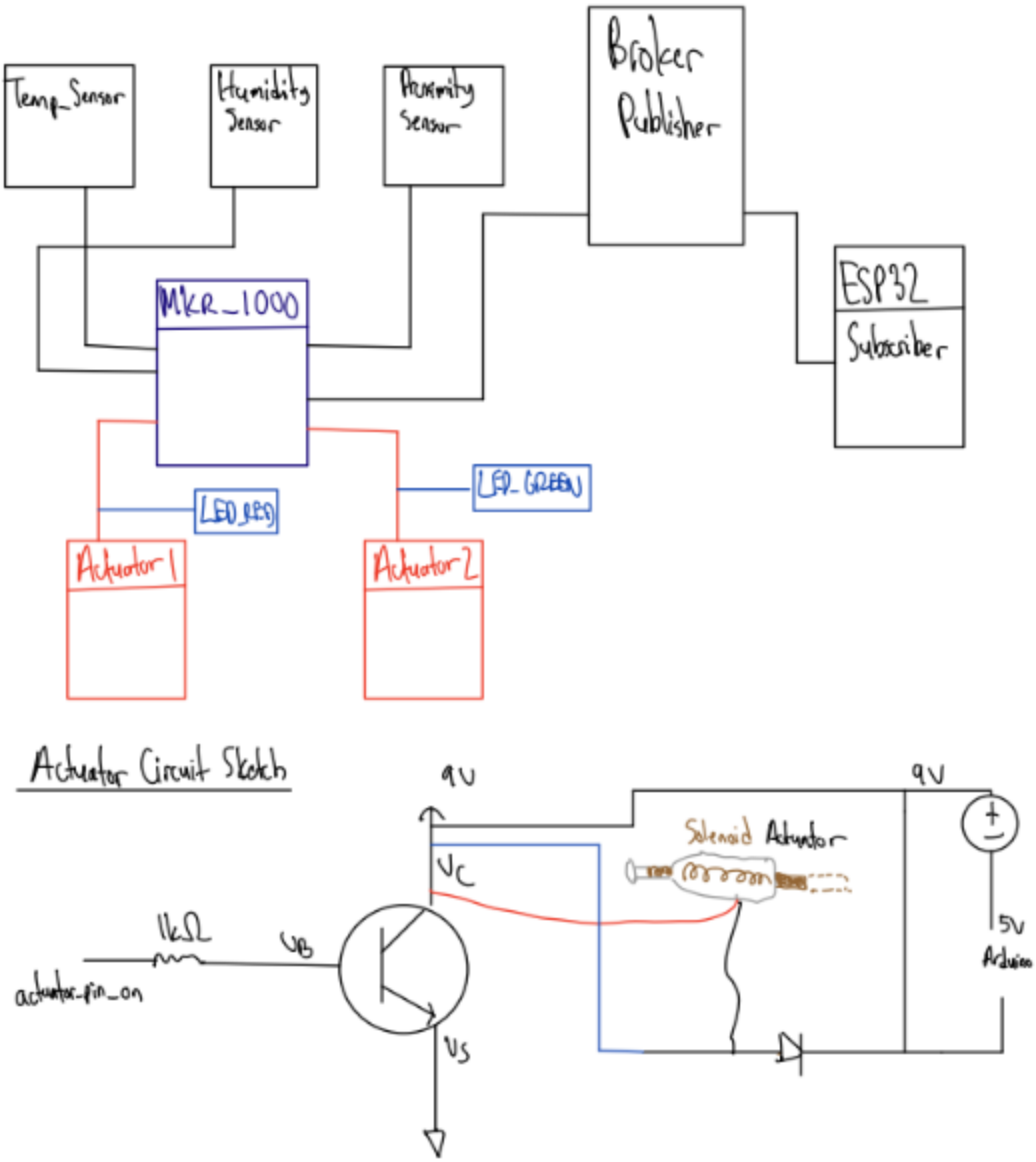
The transistors used for the actuators will isolate each actuator from the other and will be controlled sequentially whenever the previous actuator is open, the others are idle/asleep. When the window is opened, after about 10 minutes or so, a grab of data will be input and averaged and will decide whether to open the next actuator which opens the window further, keep the window at the same state, or close it altogether if the conditions are stable. The ESP32 will then input information and output the client's message to the user every second or so from the broker publisher.

d. Diagrams

Overview of Connecting Window w/ Multiple Actuators

IoT Systems – Final project





Both actuators tie to their own separate transistor because of the base voltage connected to the Arduino pins and when this pin activates, the voltage then spikes the circuit for the solenoid which is an inductor that runs the current to draw a magnetic field in to push the solenoid lever outwards. The diode regulates this high spike voltage and to protect the arduino and components from spiking or short circuiting damaging it or exploding it.

e. List of sensors

Vendor	Model	Comment
GearBox	TMP36	Temperature Sensor
MEAS	HPP801A031	Humidity Sensor

PARALLAX	SEN136B5B	Proximity Ping Distance Sensor

f. List of nodes

Vendor	Model	Comment
Espressif	ESP32	WiFi chip used as a subscriber to the client from the publisher.
Arduino	MKR1000	Device used to set sensors, set as publisher to MQTT client, connected to Blynk, connected to ThingSpeak.

g. List of other hardware components

Vendor	Model	Comment
NTE	TIP120	Transistor for Solenoid
Onsemi	1N4001	Diode used to regulate voltage spike in circuit from frying
Adafruit	Push-Pull-412	Actuator used to push/pull when a voltage spikes in

h. Selected cloud service

ThingSpeak was the cloud service used along with Blynk and MQTT that connected to the broker.

2. Project operation

a. Describe how your project works

Activating the operation allows a set of time when the window starts to open but if it is initially closed and the temperature is too hot or if the room humidity is not comfortable, the first actuator will activate pushing the window partially open, and the second actuator will activate sequentially if the room is still not uncomfortable after a set of time has passed by. However, if the room temperature stabilizes or goes below a few degrees from the readings before activating, the window closes but if it is still unstable (uncomfortable), the second actuator activates opening the window a little further. The proximity data sensor will determine which actuator to activate from the first to the last actuator if need be and the further the window pulls away activates the next actuator if it needs to be. As each actuator activates over time, a continuous check on the room temperature and humidity will reconfine the system to either continue opening the window through the use of the actuators until the window is fully open and leave it open until the temperature regulates or to start partially closing each segment of the window at a certain time or if the temperature falls below the original temperature of the room that started it initially.

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

i. specify publisher(s)
The publisher is the laptop connected to the MKR1000.

ii. specify subscriber(s)
The subscriber is the ESP32 WiFi chip.

iii. specify the location and type of MQTT broker
MQTT Mosquitto is the type of MQTT broker and it is located in the computer.

iv. specify the hierarchy / list of MQTT channels used

The list of MQTT channels used is the use of three topics that print out separate concurrent data.

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

The outputs of the sensors are what are passing through the cloud mechanism and the sensors are the current/past readings of the room recorded inside. The data processing is a message to the user that can be accessed through the mobile smartphone app, from the channels on ThingSpeak's website, or by connecting to the client itself.

3. Screenshots / pictures

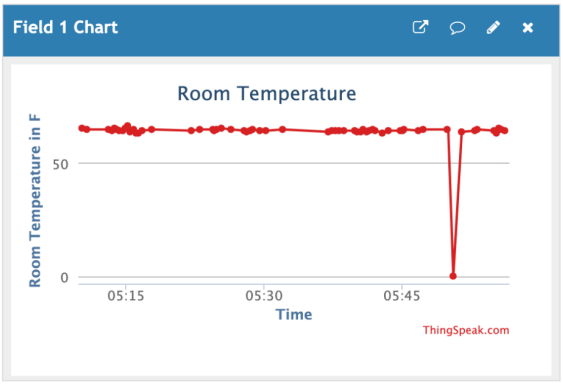
a. Publisher(s) operation (data being sent to MQTT broker)

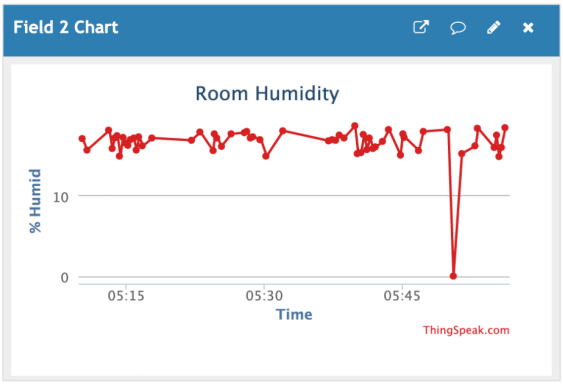
```
Distance 169.03 cm

Sending message to topic:
13
Temperature: 69.98 Degrees Farenheit
Humidity: 18.03 %
Distance 168.57 cm

Sending message to topic:
14
Temperature: 70.52 Degrees Farenheit
Humidity: 19.31 %
Distance 168.60 cm

Sending message to topic:
15
Temperature: 71.06 Degrees Farenheit
Humidity: 19.31 %
Distance 168.88 cm

Sending message to topic:
```

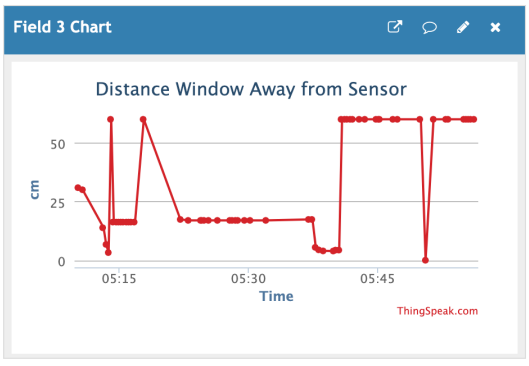
b. Subscriber(s) operation (data being received from MQTT broker)

```
Distance window is open by: 169.03 cm
**********************************************************************************************


**********************************************************************************************
Temperature in Deg. Farenheit: 69.98 F

Humidity is: 18.03%

Distance window is open by: 168.57 cm
**********************************************************************************************


**********************************************************************************************
Temperature in Deg. Farenheit: 70.52 F

Humidity is: 19.31%

Distance window is open by: 168.60 cm
**********************************************************************************************


**********************************************************************************************
Temperature in Deg. Farenheit: 71.06 F

Humidity is: 19.31%

Distance window is open by: 168.88 cm
**********************************************************************************************
```

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





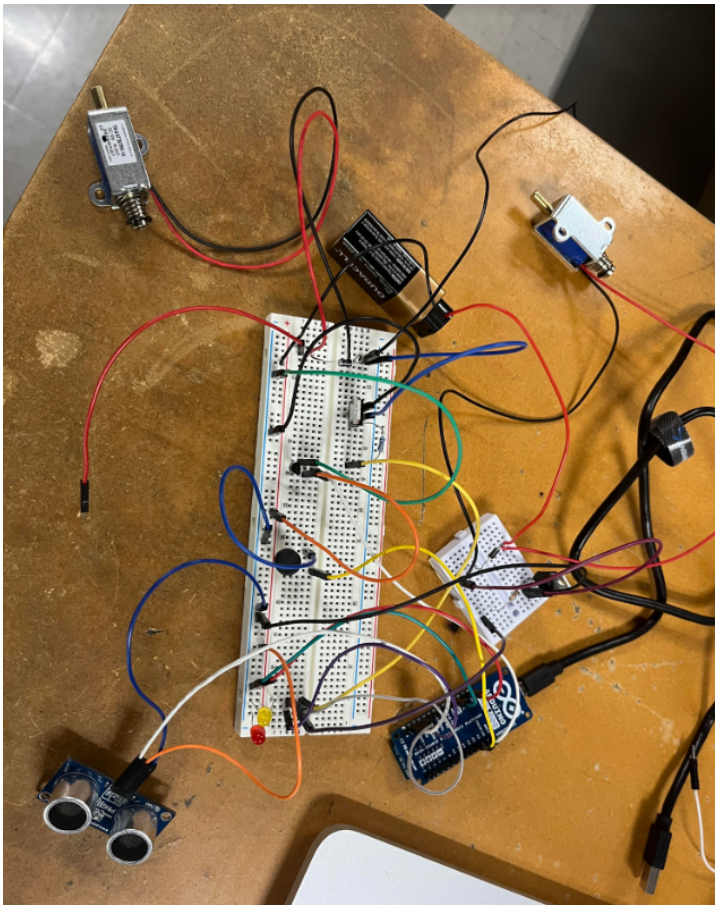


Blynk:

Link of Demo and Blynk Demo:
https://www.youtube.com/watch?v=VRRXK5Fq5HQ&list=PLMeCRZ1TwRlM4UZ8s6HjvLxbyrBXfphzO&index=1

Link to MQTT Client working:
https://www.youtube.com/watch?v=mfcSqa9qCVc&list=PLMeCRZ1TwRlM4UZ8s6HjvLxbyrBXfphzO&index=2

d. Picture(s) of wired circuit



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

The technical challenges I faced were wiring the actuators and it would not at first because the voltage supply was too low so I wired a 9V battery and the solenoids started to work perfectly. Keeping the wires inside the breadboard was another challenge because of how small some of the wires were but the main issue was getting the ESP32 to connect properly and I had to just press the boot button every time I uploaded, fixing the issue. Also, I had an issue with using ThingSpeak and MQTT client which paused the MQTT client after printing its topic message one iteration and the issue seems to be that ThingSpeak is not aggregated enough.

5. Code listings for each node and cloud codes

MKR1000 Code:

```
/* Dylan Flores
CpE 417
2001549196
Final Project */
#define BLYNK_TEMPLATE_ID "TMPLoVVf9Ycy"
#define BLYNK_DEVICE_NAME "Final"
#define BLYNK_AUTH_TOKEN "s4qBIA7ZM32InWbJhfnkWLKc78M4X2bW"
#define BLYNK_PRINT SerialUSB
#include <SPI.h>
#include <WiFi101.h>
#include <ArduinoMqttClient.h>
#include <BlynkSimpleWiFiShield101.h>
#include "ThingSpeak.h"
#define tempPin A1
#define humidityPin A2
#define sensorPin 7
#define solenoid 8
#define solenoid2 9
//const char ssid[] = "Dylan's iPhone";
const char ssid[] = "ECE_Labs";
const char pass[] = "348#ECEWiFi";
const char auth[] = BLYNK_AUTH_TOKEN;
int status = WL_IDLE_STATUS;     // the Wifi radio's status
unsigned long myChannelNumber = 1981479;
const char* myWriteAPIKey = "2T2OAUK7F74HM377";

WiFiClient client;
MqttClient mqttClient(client);
const char broker[] = "test.mosquitto.org";
```

```
int port = 1883;
const char topic[] = "final_sensors";
const char topic1[] = "final_sensors";
const char topic2[] = "final_sensors";
int estimate_sec = 0;
static float distance, temp, humidity;
bool firstActuatorOpen = false;
bool secondActuatorOpen = false;

// Set up pins and initialize
void setup() {
 // put your setup code here, to run once:
 Serial.begin(9600); // BAUD rate 9600 serial display terminal
 SerialUSB.begin(115200);
 Serial.println("Attempting to connect to WPA network...");
 status = WiFi.begin(ssid, pass);
 if(status != WL_CONNECTED) {
   Serial.println("Couldn't get a wifi connection");
   while(true);
 }
 else {
   Serial.print("Connected to network: ");
   Serial.println(ssid);
 }
 Serial.println("You're connected to the network\n");
//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("Connected to the MQTT broker!\n");
 pinMode(tempPin, INPUT);
 pinMode(humidityPin, INPUT);
 pinMode(solenoid, OUTPUT);
 pinMode(solenoid2, OUTPUT);
 Serial.println("Connecting to the Blynk Application...");
 Blynk.begin(auth, ssid, pass); // start blynk
 Serial.println("Connected to the Blynk App! \n Connecting to ThingSpeak Channel..");
```

```
 delay(1000);
 ThingSpeak.begin(client);
 delay(1000);
 Serial.println("Connected to ThingSpeak Channel!\n");
}
// Loop forever
void loop() {
 mqttClient.poll();
 sensor_temp();
 sensor_humidity();
 sensor();
 actuator();
 print_info_monitor();
 delay(1000);
 mqTTClient();
// thingSpeak();
 Blynk.run();
 Blynk.virtualWrite(V0, temp);
 Blynk.virtualWrite(V1, humidity);
 Blynk.virtualWrite(V2, distance);
 }

void thingSpeak(){
 ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
 ThingSpeak.setField(1, temp);
 ThingSpeak.setField(2, humidity);
 ThingSpeak.setField(3, distance);
 Serial.println("Temperature, humidity, & distance window opened datas sent to the
cloud\n");
}
void mqTTClient(){
 Serial.println("Sending message to topic: ");
 mqttClient.beginMessage(topic);

mqttClient.println("**********************************************************************
*****************");
 mqttClient.print("Temperature in Deg. Farenheit: ");
 mqttClient.print(temp);
 mqttClient.println(" F");
 mqttClient.endMessage();

 mqttClient.beginMessage(topic1);
```

```
 mqttClient.print("Humidity is: ");
 mqttClient.print(humidity);
 mqttClient.println("% ");
 mqttClient.endMessage();

 mqttClient.beginMessage(topic2);
 mqttClient.print("Distance window is open by: ");
 mqttClient.print(distance);
 mqttClient.println(" cm ");

mqttClient.println("**********************************************************************
*****************");
 mqttClient.println();
 mqttClient.endMessage();
}
// Distance proximity sensor logic
void sensor(){
 pinMode(sensorPin, OUTPUT);
 digitalWrite(sensorPin, LOW);
 delay(1);
 digitalWrite(sensorPin, HIGH);
 delay(1);
 digitalWrite(sensorPin, LOW);
 pinMode(sensorPin, INPUT);
 distance = pulseIn(sensorPin, HIGH);
 distance = distance / 29.0 / 2.0;
 if(distance > 600){
   distance = 600;
 }
}
// Temperature sensor pin logic
void sensor_temp(){
 float heat = analogRead(tempPin);
 float v = heat * (3300/1024);
 float realT = (v - 500) / 10.0;
 realT = ( (realT * 9.0) / 5.0) + 32.0;
 if(firstActuatorOpen == true || secondActuatorOpen == true){
   realT = realT - 60;
 }
 temp = realT;
}
// Humidity sensor pin logic
```

```
void sensor_humidity(){
 humidity = analogRead(humidityPin);
 humidity = (humidity / 1023.0 ) * 5; // humidity ADC voltage converted
 humidity = (5.0-humidity)*10.0/humidity; // humidity converted to match percentage
}
void print_info_monitor(){
 Serial.print("Temperature: ");
 Serial.print(temp);
 Serial.print(" Degrees Farenheit \n");
 Serial.print("Humidity: ");
 Serial.print(humidity);
 Serial.print(" % \n");
 Serial.print("Distance ");
 Serial.print(distance);
 Serial.println(" cm \n");
}
void actuator(){
 bool notComfortable = false;
 Serial.println(estimate_sec);
 if(estimate_sec >= 30) {
   estimate_sec = 0; // reset seconds clock
   digitalWrite(solenoid, LOW); // first actuator reset
   digitalWrite(solenoid2, LOW); // second actuator reset
   firstActuatorOpen = false;
   secondActuatorOpen = false;
 }
 // Determine if room is comfortable at normal temperatures
 if(temp > 60 || humidity > 18) {
   notComfortable = true;
 }
 else if(temp < 58 && humidity < 18) {
   digitalWrite(solenoid2, LOW);
 }
 else{
   notComfortable = false;
 }
 if(notComfortable == true){
   if(distance < 15 && estimate_sec < 10) {
     digitalWrite(solenoid, HIGH);
     firstActuatorOpen = true;
     delay(100);
   }
```

```
   else if(distance > 15 && distance < 200 && estimate_sec > 10) {
     digitalWrite(solenoid2, HIGH);
     secondActuatorOpen = true;
     delay(100);
   }
 }
 else{
   if(secondActuatorOpen == true && firstActuatorOpen == true){
      secondActuatorOpen = false; // close part of window
   }
   else if(firstActuatorOpen == true && secondActuatorOpen == false){
     firstActuatorOpen = false; // window fully closed
   }
 }
 delay(1000);
 estimate_sec++;
}
```

ESP32 Code:

```
#include <SPI.h>
#include <WiFi.h>
#include "ThingSpeak.h"
#include <ArduinoMqttClient.h>

const char ssid[] = "ECE_Labs";
const char pass[] = "348#ECEWiFi";
WiFiClient client;
MqttClient mqttClient(client);
const char broker[] = "test.mosquitto.org";
int port = 1883;
int status = WL_IDLE_STATUS;     // the Wifi radio's status
unsigned long myChannelNumber = 1981479;
const char* myReadAPIKey = "DG05W3RZOL9HRY6K";
float distance, temp, humidity;
const char topic[] = "final_sensors";

void setup() {
 Serial.begin(9600); // BAUD rate 9600 serial display terminal
 Serial.println("Attempting to connect to WPA network...");
 WiFi.mode(WIFI_STA); //Optional
 WiFi.begin(ssid, pass);
 Serial.println("\nConnecting");
```

```
 while(WiFi.status() != WL_CONNECTED){
  Serial.print(".");
   delay(100);
 }
 Serial.println("You're connected to the network\n");
 Serial.print("Attempting to connect to the MQTT broker on ESP32: ");
 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("Connected to the MQTT broker on ESP32!\n");
 mqttClient.onMessage(onMqttMessage);
 mqttClient.subscribe(topic);
}

void loop() {
 Serial.println("Reading from Client!...");
 delay(1000);
 mqttClient.poll();
 mqttClient.onMessage(onMqttMessage);

}

void onMqttMessage(int messageSize){
 Serial.print(mqttClient.messageTopic());
 while(mqttClient.available()){
   Serial.print((char)mqttClient.read());
 }
 Serial.println();
}
```