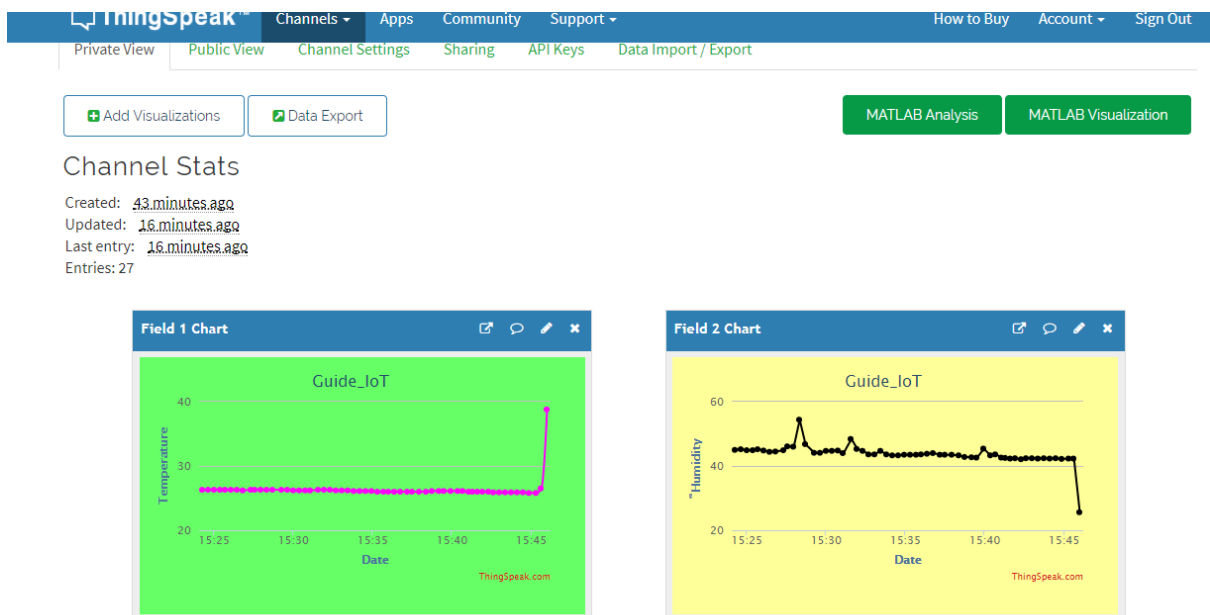


Lab 8

นาย ศุภากร บวรเศรษฐพงศ์ IT-4RC รหัส 5706021632162



Code

```
#include <LiquidCrystal_I2C.h>

#include "DHT.h"
#include <ESP8266WiFi.h>

#define PUMP_RLY 4 // output drive relay for pump GPIO4 (D2)
#define DHTPIN 2 // what pin we're connected to GPIO2 (D4)
#define DHTTYPE DHT22 // DHT 22

#define DEBUG

#define DEBUG_PRINTER Serial

#ifdef DEBUG

#define DEBUG_PRINT(...) { DEBUG_PRINTER.print(__VA_ARGS__); }
```

```

#define DEBUG_PRINTLN(...) { DEBUG_PRINTER.println(__VA_ARGS__); }

#else

#define DEBUG_PRINT(...) {}

#define DEBUG_PRINTLN(...) {}

#endif

const char* ssid    = "itfitm";
const char* password = "";

DHT *dht;

void connectWifi();
void reconnectWifiIfLinkDown();
void initDht(DHT **dht, uint8_t pin, uint8_t dht_type);
void readDht(DHT *dht, float *temp, float *humid);
void uploadThingsSpeak(float t, float h);

void setup() {
    Serial.begin(115200);
    delay(10);
    pinMode(PUMP_RLY, OUTPUT); // Initialize the PUMP_RLY(4) pin as an output
    digitalWrite(PUMP_RLY, HIGH); // Make sure relay is normal off

    connectWifi();

    initDht(&dht, DHTPIN, DHTTYPE);
}

void loop() {
    static float t_dht;
    static float h_dht;

```

```

readDht(dht, &t_dht, &h_dht);
if(t_dht > 40) // condition for make relay on
{
    digitalWrite(PUMP_RLY, HIGH); //If condition true do this!
} else
{
    digitalWrite(PUMP_RLY, LOW);
}
uploadThingsSpeak(t_dht, h_dht);

// Wait a few seconds between measurements.
delay(10 * 1000);
reconnectWifiIfLinkDown();
}

```

```

void reconnectWifiIfLinkDown() {
    if (WiFi.status() != WL_CONNECTED) {
        DEBUG_PRINTLN("WIFI DISCONNECTED");
        connectWifi();
    }
}

```

```

void connectWifi() {
    DEBUG_PRINTLN();
    DEBUG_PRINTLN();
    DEBUG_PRINT("Connecting to ");
    DEBUG_PRINTLN(ssid);

    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {

```

```

    delay(500);

    DEBUG_PRINT(".");
}

DEBUG_PRINTLN("");
DEBUG_PRINTLN("WiFi connected");
DEBUG_PRINTLN("IP address: ");
DEBUG_PRINTLN(WiFi.localIP());
}

void initDht(DHT **dht, uint8_t pin, uint8_t dht_type) {
    // Connect pin 1 (on the left) of the sensor to +5V
    // NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1
    // to 3.3V instead of 5V!
    // Connect pin 2 of the sensor to whatever your DHTPIN is
    // Connect pin 4 (on the right) of the sensor to GROUND
    // Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor

    // Initialize DHT sensor for normal 16mhz Arduino
    // NOTE: For working with a faster chip, like an Arduino Due or Teensy, you
    // might need to increase the threshold for cycle counts considered a 1 or 0.
    // You can do this by passing a 3rd parameter for this threshold. It's a bit
    // of fiddling to find the right value, but in general the faster the CPU the
    // higher the value. The default for a 16mhz AVR is a value of 6. For an
    // Arduino Due that runs at 84mhz a value of 30 works.
    // Example to initialize DHT sensor for Arduino Due:
    //DHT dht(DHTPIN, DHTTYPE, 30);

    *dht = new DHT(pin, dht_type, 30);
    (*dht)->begin();
    DEBUG_PRINTLN(F("DHTxx test!")) ;
}

```

```
}
```

```
void uploadThingsSpeak(float t, float h) {  
    static const char* host = "api.thingspeak.com"; ///////////  
    static const char* apiKey = "DMKEBXBZDUI3M5JV"; //////////  
  
    // Use WiFiClient class to create TCP connections  
    WiFiClient client;  
    const int httpPort = 80;  
    if (!client.connect(host, httpPort)) {  
        DEBUG_PRINTLN("connection failed");  
        return;  
    }  
  
    // We now create a URI for the request  
    String url = "/update/";  
    // url += streamId;  
    url += "?key=";  
    url += apiKey;  
    url += "&field1=";  
    url += t;  
    url += "&field2=";  
    url += h;  
  
    DEBUG_PRINT("Requesting URL: ");  
    DEBUG_PRINTLN(url);  
  
    // This will send the request to the server  
    client.print(String("GET ") + url + " HTTP/1.1\r\n" +  
        "Host: " + host + "\r\n" +  
        "Connection: close\r\n\r\n");
```

```
}
```

```
void readDht(DHT *dht, float *temp, float *humid) {
```

```
    if (dht == NULL) {
```

```
        DEBUG_PRINTLN(F("[dht11] is not initialised. please call initDht() first."));
```

```
        return;
```

```
    }
```

```
    // Reading temperature or humidity takes about 250 milliseconds!
```

```
    // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
```

```
    float h = dht->readHumidity();
```

```
    // Read temperature as Celsius
```

```
    float t = dht->readTemperature();
```

```
    // Read temperature as Fahrenheit
```

```
    float f = dht->readTemperature(true);
```

```
    // Check if any reads failed and exit early (to try again).
```

```
    if (isnan(h) || isnan(t) || isnan(f)) {
```

```
        DEBUG_PRINTLN("Failed to read from DHT sensor!");
```

```
        return;
```

```
    }
```

```
    // Compute heat index
```

```
    // Must send in temp in Fahrenheit!
```

```
    float hi = dht->computeHeatIndex(f, h);
```

```
    DEBUG_PRINT("Humidity: ");
```

```
    DEBUG_PRINT(h);
```

```
    DEBUG_PRINT("%\t");  
    DEBUG_PRINT("Temperature: ");  
    DEBUG_PRINT(t);  
    DEBUG_PRINT(" *C ");  
    DEBUG_PRINT(f);  
    DEBUG_PRINT(" *F\t");  
    DEBUG_PRINT("Heat index: ");  
    DEBUG_PRINT(hi);  
    DEBUG_PRINTLN(" *F");  
  
    *temp = t;  
    *humid = h;  
}
```