

# Task 6: Internet of Things Laboratory

## Table of content

I. Introduction .....	2
II. Instructions .....	5
2.1. Hardware design .....	5
2.2. Programming for NodeMCU .....	6
2.3. Data live on ThingSpeak.....	9
2.4. Programming in MIT App Inventor 2.....	9

## I. Introduction

The internet of things (as IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people. It has the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. It is also an ecosystem consists of web-enabled smart devices that use embedded processors, sensors and communication hardware to collect, send and act on data they acquire from their environments. The IoT helps people work more effectively and enables companies to automate processes and reduce labor costs.

In this task, writing the guide for how to transfer data from an embedded system to the website via ThingSpeak and using the MIT App Inventor 2 for Android, it is an open source web application originally provided by Google and currently maintained by Massachusetts Institute of Technology (MIT). Temperature and humidity data are measured from DHT11 sensors and sent to the website using ESP8266 NodeMCU.

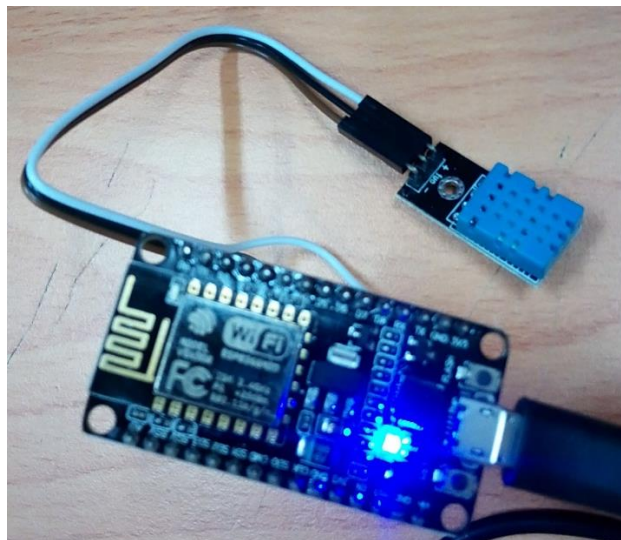


Figure 1: The hardware

After that users can access their accounts on ThingSpeak to view the data uploaded on real time.

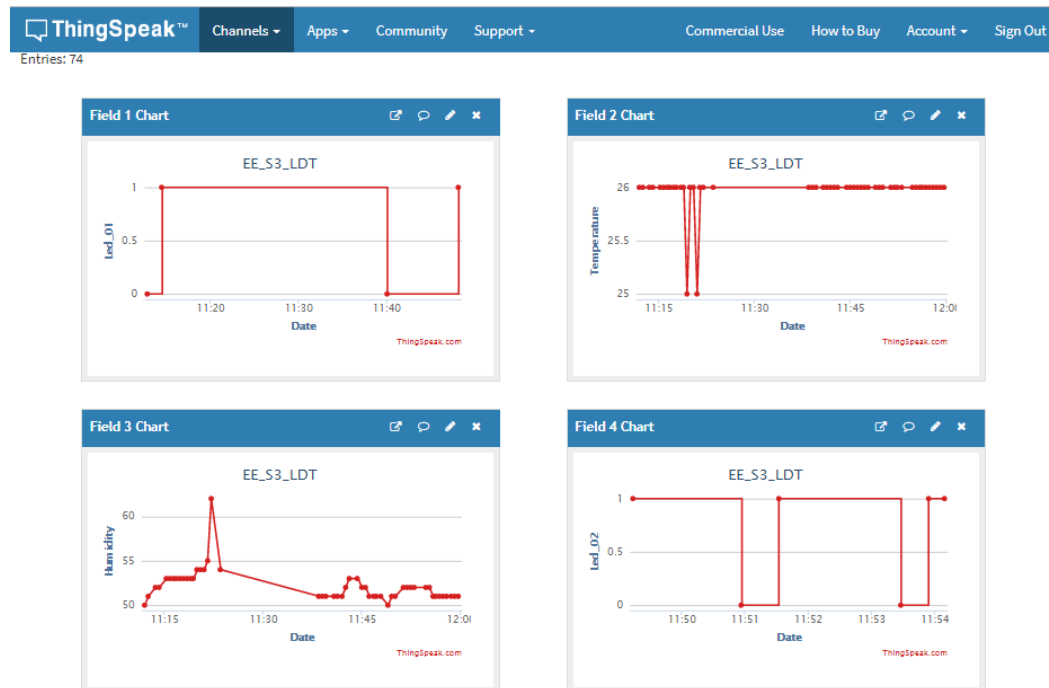


Figure 2: The data on real-time

The next step is to create an software applications for the Android operating system on smartphone to view data and control the system easily through a wifi connection. MIT App Inventor 2 is a platform that allows developers to create applications using the graphical interface with drag and drop of code blocks to create applications that can run on Android devices.

```

initialize global Data_2 to "https://api.thingspeak.com/channels/852480/field..."
initialize global Data_3 to "https://api.thingspeak.com/channels/852480/field..."
initialize global read_API_KEY to "VZP2XVZDZJSSJ72L"
initialize global Read_End to "&results=1"
initialize global Temp_Display to 0
initialize global Hum_Display to 0

```

Figure 3: Getting data information

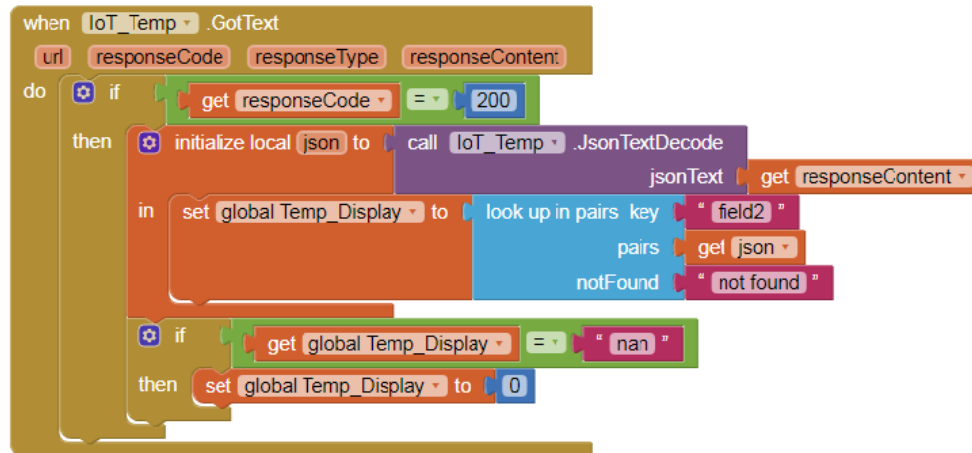


Figure 4: Sorting the commands

After successful initialization, downloading App Inventor software and entering the security code to run the successfully created project.

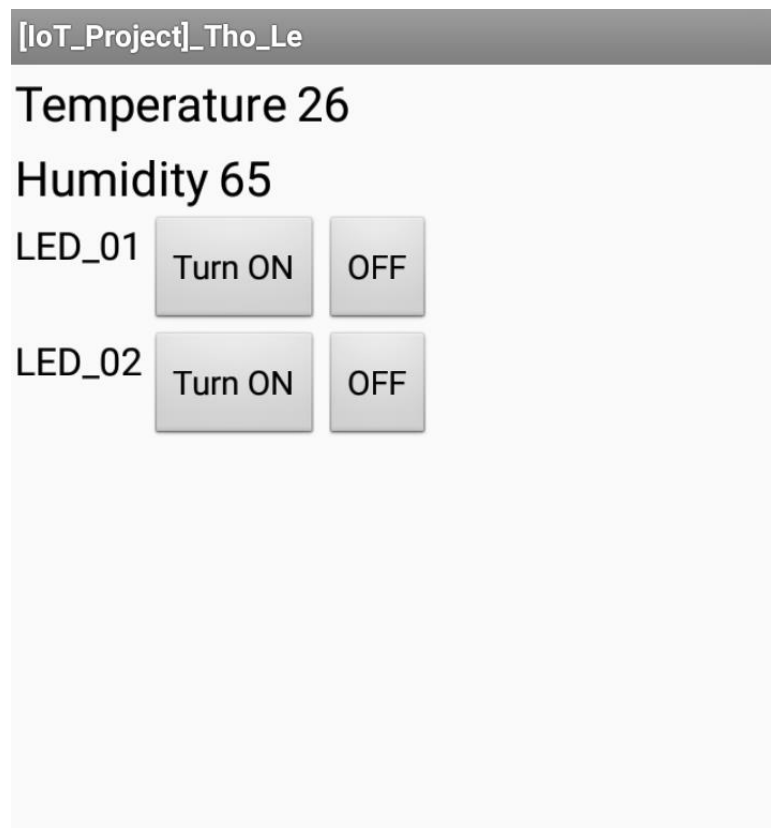


Figure 5: Application for the project

## II. Instructions

### 2.1. Hardware design

Connect components according to the following diagram

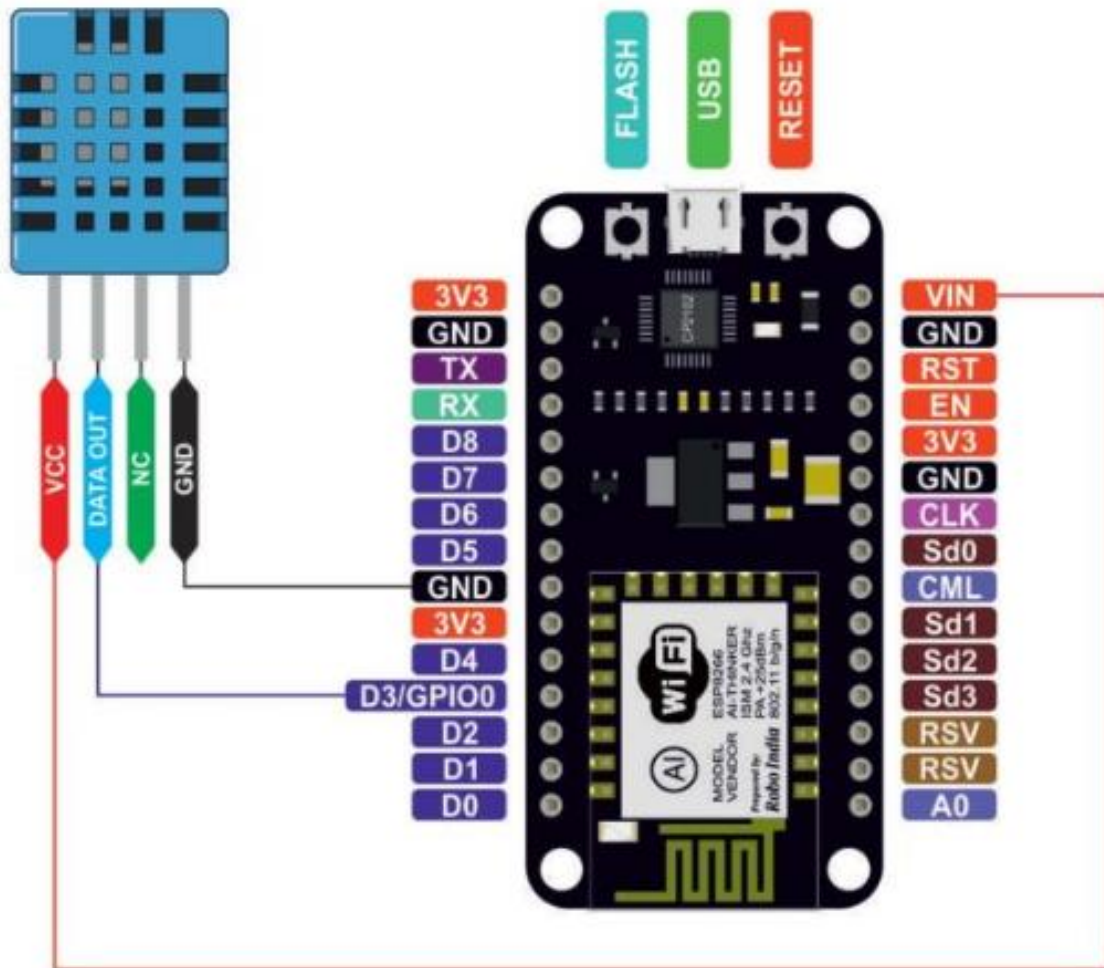



Figure 5. Circuit Diagram

The nature signal was convert to the digital signal by DHT11 sensor. Then, The NodeMCU will receive and transmit the data on ThingSpeak. Moreover, we can control the LED by API key and MIT App Inventor 2.

## 2.2. Programming for NodeMCU

- **Step 1:** To declare libraries ThingSpeak, DHT11 and ESP8266. Then, setting up the data that using in project as following figure:



```
IoT_ESP8266 $

#include <ThingSpeak.h>    //Thingspeak library
#include <ESP8266WiFi.h>    //ESP8266 library
#include "DHT.h"           //DHT sensor library

const int DHTPIN = D3;     //Read data from pin D3
const int DHTTYPE = DHT11; //Declare DHT11 or DHT11

DHT dht(DHTPIN, DHTTYPE);
float t;
float h;

//Setup network parameters
const char* ssid = "International University"; //ssid
// const char* password = " "; //password

//Setup Thingspeaks
char thingSpeakAddress[] = "api.thingspeak.com";
unsigned long channelID = 852480; //Your chanel ID
char* writeAPIKey = "E1MCGYU60JL72RLW"; // Your write key
char* readAPIKey = "VZP2XVZDZJSSJ72L"; //Your read key
//Setup field in Thingspeak
unsigned int Led_01= 1; // LED Field
unsigned int Temperature= 2; // Temperature Field
unsigned int Humidity= 3; // Humidity Field
unsigned int Led_02= 4; // LED Field
```

Figure 6. Declaring libraries

- **Step 2:** Configuring time to update data and setting up to turn on/off LEDs

```

/*Time update data*/
const unsigned long Posting_Interval = 30L * 1000L;
long lastUpdateTime = 0;
WiFiClient client;
const unsigned long Turn_off_Interval = 30L * 1000L;
long lastUpdateTime_van1 = 0;
/*-----
void setup()
{
    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(D5, OUTPUT);
    Serial.begin(9600);
    Serial.println("Start");
    connectWiFi();
    dht.begin();          // Start sensor
}

```

Figure 7. Configuring timer

- **Step 3:** Creating WiFi connection to transmit and receive data

```

// Set up Wifi Connection
int connectWiFi()
{
    Serial.begin(9600);
    delay(10);

    // Connect WiFi
    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.hostname("Name");
    WiFi.begin(ssid);
    // WiFi.begin(ssid, password);

    if (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
        Serial.println("Connecting to WiFi");
    }
}

```

Figure 8. The WiFi connection

- **Step 4:** To create an account on ThingSpeak
  - ❖ Going to <https://thingspeak.com/> and creating an account to login to your account.
  - ❖ Create a new channel and enter basic details of the channel.
  - ❖ Save information:
    - ✓ Channel ID
    - ✓ API keys to write and read  
(For example: E1MCGYU60JL72RLW, VZP2XVZDZJSSJ72L)
    - ✓ Some links are shown
- **Step 5:** The control algorithm is written as follows:

```

void loop()
{
  air_parameter();
  long rssi = WiFi.RSSI();
  if (millis() - lastUpdateTime >= Posting_Interval)
  {
    lastUpdateTime = millis();
    write2TSDData( channelID , Temperature , t , Humidity , h );
    Serial.print("-----WRITE DATA TO THINGSPEAK----");
    Serial.println();
  }

  if ( readTSDData( channelID, Led_01 ) == 1 )
  {
    digitalWrite(LED_BUILTIN,LOW);
  }
  else
  {
    digitalWrite(LED_BUILTIN,HIGH);
  }

  if ( readTSDData( channelID, Led_02 ) == 1 )
  {
    digitalWrite(D5,HIGH);
  }
  else
  {
    digitalWrite(D5,LOW);
  }
}

```

Figure 9. The main programming



## 2.3. Data live on ThingSpeak

Open the channel at ThingSpeak and the data will be shown as follow:

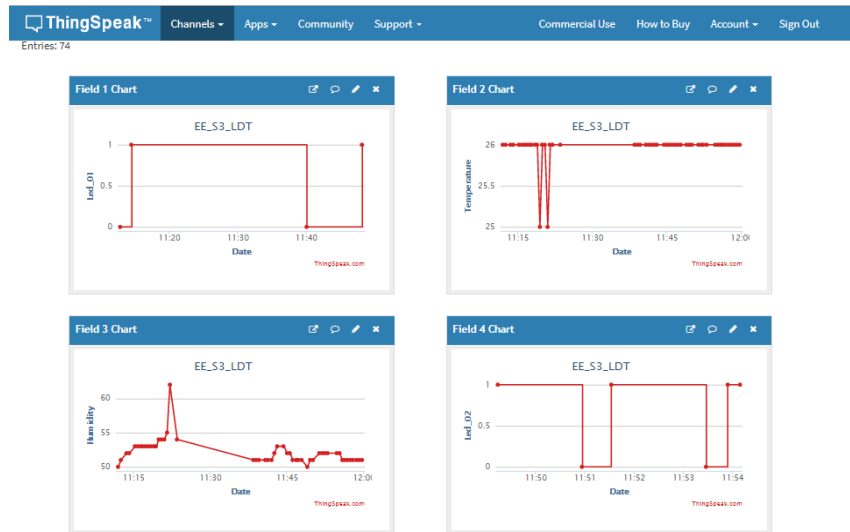


Figure 10. The result at ThingSpeak

## 2.4. Programming in MIT App Inventor 2

- At first, to design the user interface. The first Print Screen above show the main visible and non-visible elements:

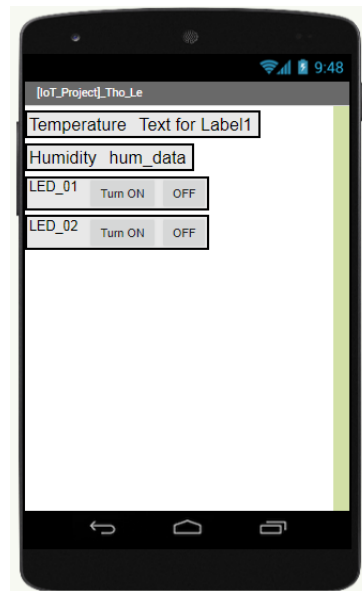


Figure 11. The user interface

- Secondly, to design the blocks:
  - ❖ The status variables must be declared as Global.

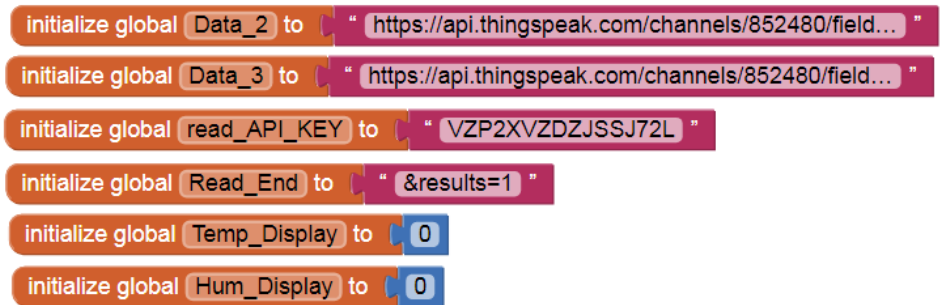


Figure 12. To Declare the link



Figure 13. To Declare controlling LEDs

- ❖ At every 2 seconds (defined by Clock1), the data was updated.

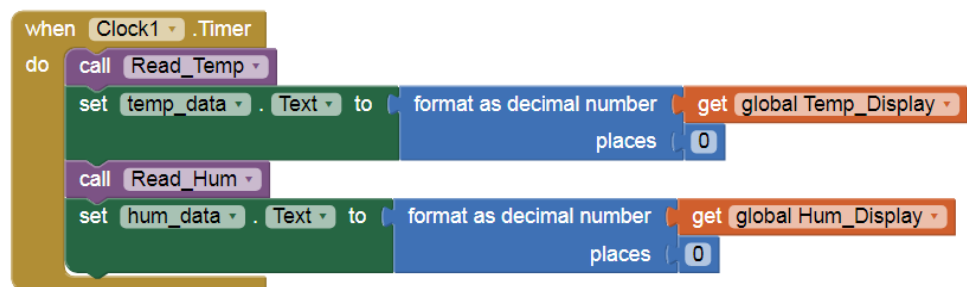


Figure 14. The Clock

- ❖ Functions to get data from ThingSpeak to application.

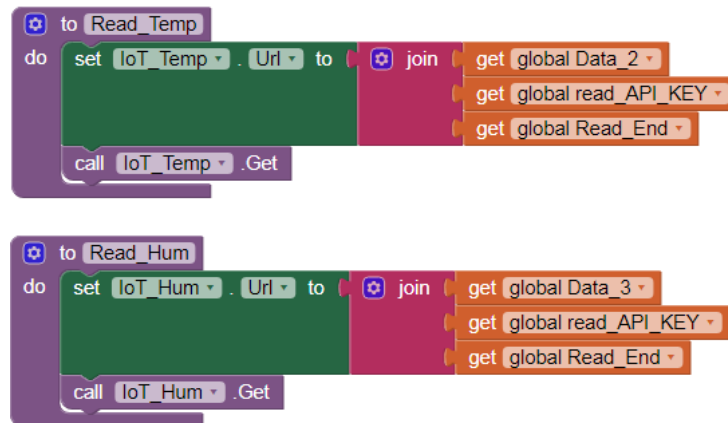


Figure 15. Getting the data on ThingSpeak

- ❖ The data type reading method.

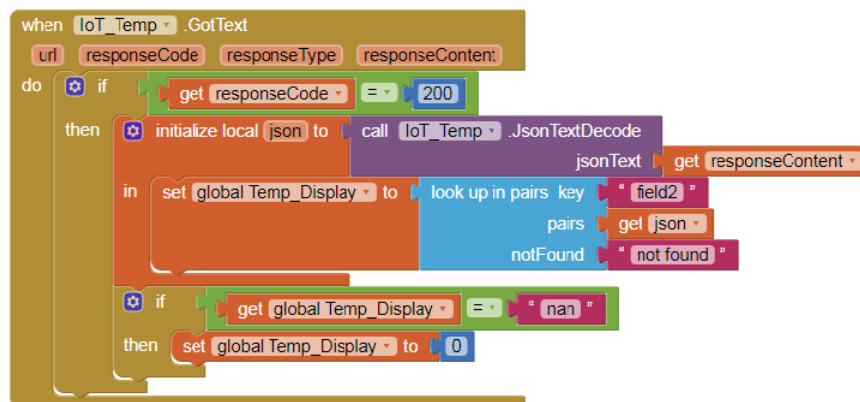


Figure 16. Reading temperature

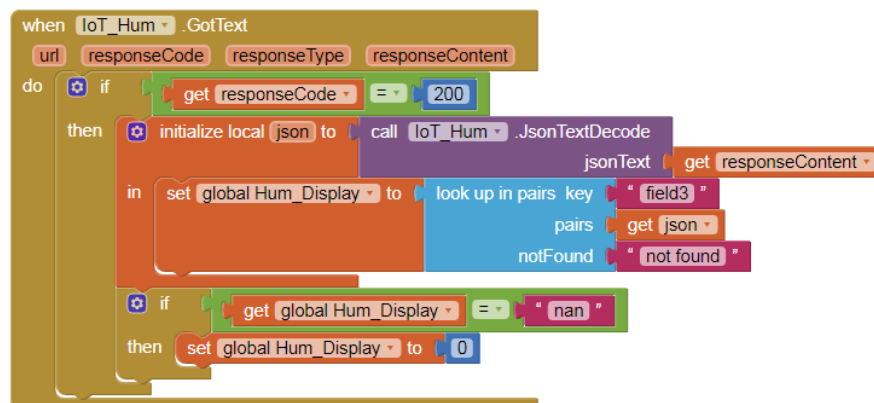


Figure 17. Reading humidity

❖ How to control turning LEDs on/off

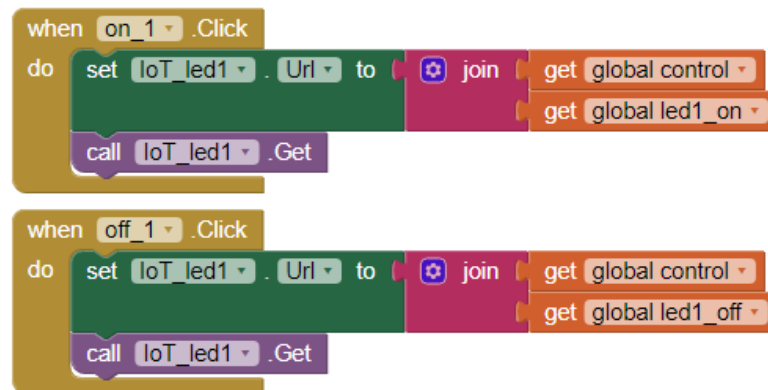


Figure 18. Turning on/off LED 1

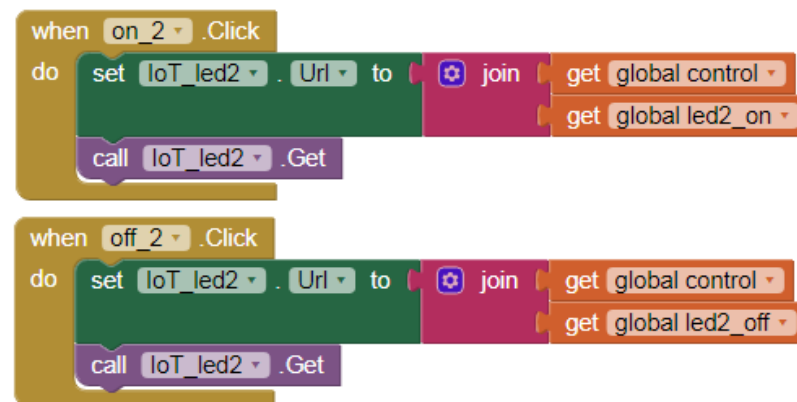


Figure 19. Turning on/off LED 2

❖ To run the project as following figure:

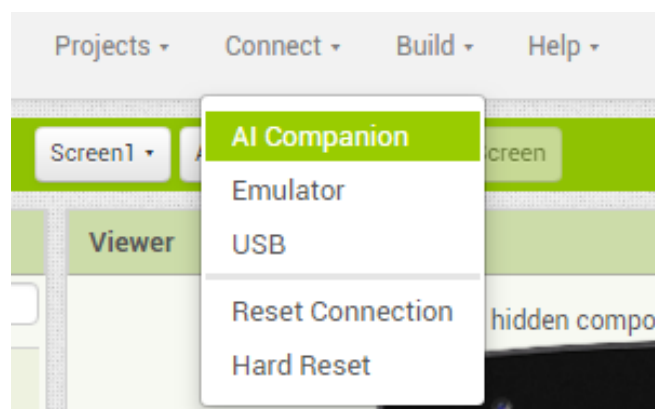


Figure 20. Running the code

- ❖ To open the MIT App Inventor 2 to scan QR or enter the 6-character code

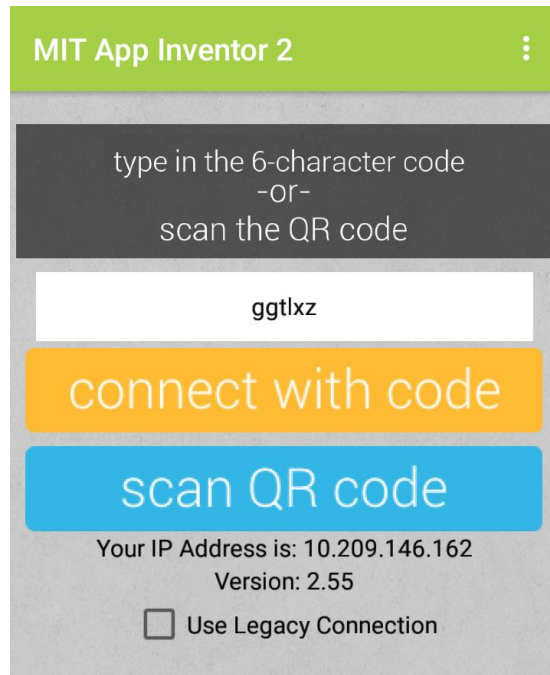


Figure 21. The MIT App Inventor 2

- Finally, here is the result

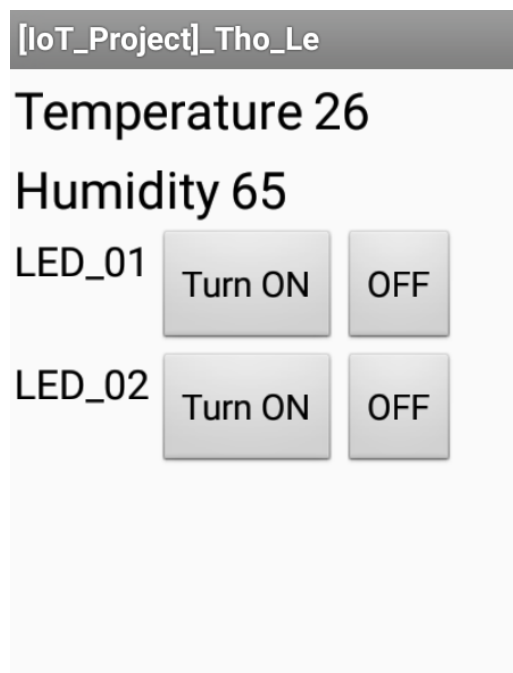


Figure 22. The result