HO CHI MINH CITY, UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEER

Application Based Internet of Things Report

Student 1: Trần Long Vĩ - 1712938
Student 2: Huỳnh Phúc Khánh - 1810226

HO CHI MINH CITY, 06/ 2021

Content

1 Introduction

This project is an application base on the Internet of Things about collecting data from the sensor node (Arduino) included: Temperature and Light value and sending this data to Adafruit Server through a gateway. It also have an Android mobile phone to get and show this data. The diagram below is the architecture of this project:

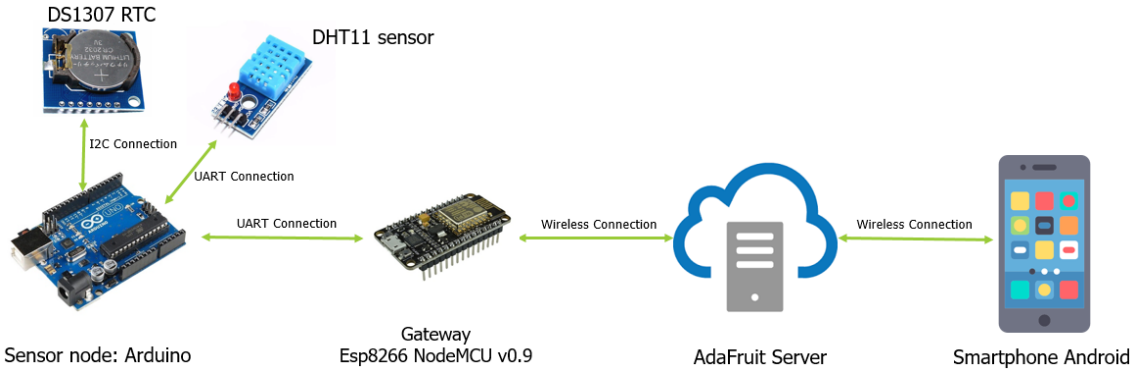

Figure 1: *Architecture diagram*

- Sensor node is used to get the value of the temperature and humidity from the environment by using DHT11 module and send to the gateway by UART Serial connection.
- Gateway: ESP8266 NodeMCU is used to receive the data from the sensor node (Arduino) and send to Adafruit Server through MQTT Protocol.
- AdaFruit.io is a free Server used to store the data that received from the gateway and send to the Smartphone based on MQTT Protocol.
- Smartphone Android: is used to show the data the received from the AdaFruit Server.

2 Sensor implementation

In this project, DHT11 module is used to get temperature and humidity from natural environment. Besides, a MCU used to a Sensor Node is Arduino UNO-R3. This Node's mission is receiving data from DHT11 module and real time module DS1307 RTC, packing data and sending to gateway via UART connection. Arduino always sends data packet to gateway Node each of 4 second.
Data format to send data from Sensor Node to gateway is defined by JSON type like below:

Listing 1: Json Data Format

```
1        {
2            "temperature" : /*float type*/,
3            "humidity" : /*float type*/,
4            "date" : [
5                yyyy,
6                mm,
```

```
            dd
        ],
        "time" : [
            hh,
            mm,
            ss
        ]
    }
```

- "temperature" and "humidity" paths are data got from DHT11 module, which type of these values is float.
- "date" and "time" paths are real time got from DS1307 RTC module, this is the time to get data from natural environment. Server.

The Source code of Arduino - sensor node is divided to 4 block: Data getting block, Real time block, Json package block and UART data sending block.

2.1 Data getting Block

The C source code of Data getting block is used to getting data from DHT11 via port 8, is provided below:

Listing 2: Getting data from DHT11 sensor

```
.............
#include <DHT.h>
...........

//************************ DHT Sensor ↩
    ***********************************
#define DHTPIN      8
#define DHTTYPE     DHT11
DHT dht(DHTPIN, DHTTYPE);
.........

void setup() {
    dht.begin();
    .........
}
void loop() {
    // Read data from DHT11
    float temp = dht.readTemperature();
    float humi = dht.readHumidity();
    ...........
}
```

2.2 Real time Block

The C source code of Real time block is used to getting real time from DS1307 RTC via I2C connection (Port A4-A5), is provided below:

Listing 3: Getting real time from DS1307 RTC

```
#include <RTClib.h>
.............
#include <time.h>
...........

//************************ RTC Sensor ↩
    ***********************************
RTC_DS1307 RTC;
.........

void setup() {
    ...........

    RTC.begin();
    delay(500);
    // time synchronization with computer
    RTC.adjust(DateTime(__DATE__, __TIME__));
    ...........
}
void loop() {
    ...........

    // Get current time from DS1307 RTC
    DateTime now = RTC.now();
    ...........
}
```

2.3 Json package Block

In this project, I used *ArduinoJson.h* library version 6 to pack data format. The C source code of Json package block is used to packing data to Json format defined by using ArduinoJson.h lib, is provided below:

Listing 4: Packing data to Json Type

```
.............
#include <ArduinoJson.h>
...........

void loop() {
    ...........

    //********** Json Data Packages by ArduinoJson.h lib ↩
        ****************
    StaticJsonDocument<256> jsonBuffer;
```

```
        JsonObject root = jsonBuffer.to<JsonObject>();
        // Add temperature-humidity path
        root["temperature"] = temp;
        root["humidity"] = humi;

        // Add date and time paths
        JsonArray _date = root.createNestedArray("date");
        _date.add(now.year());
        _date.add(now.month());
        _date.add(now.day());
        JsonArray _time = root.createNestedArray("time");
        _time.add(now.hour());
        _time.add(now.minute());
        _time.add(now.second());

        // Send Json data to Serial communication
        serializeJson(root, Serial);
        ...........
}
```

2.4 UART data sending Block

The C source code of UART data sending block is used to sending data package to gateway via UART serial communication (Port 10-11), is provided below:

Listing 5: Sending data package to gateway

```
...........
#include <SoftwareSerial.h>
...........

//************************ Rx - Tx ↩
    ***********************************
#define RXpin       10
#define TXpin       11
SoftwareSerial espSerial(RXpin,TXpin);
.........

void setup() {
    ...........

    espSerial.begin(115200);
    ...........
}
void loop() {
    ...........

    // Send Json data to ESP8266
    serializeJson(root, espSerial);
    ...........
}
```

3 Gateway implementation

Gateway node is implemented in NodeMCU v0.9 MCU. This Node's mission is receiving data from Sensor Node (Arduino), then packing and sending to Adafruit Server.

3.1 UART received data

When the Sensor Node sent data, gateway Node received data and try to decode. Gateway Node receives data in port D1 (corresponds to RX) and sends control request via D2 (corresponds to RX).
The C source code of UART data received is used to receiving data package from Sensor Node, is provided below:

Listing 6: Data received from Sensor Node

```
1            .............
2            #include <SoftwareSerial.h>
3            ...........
4
5            //************************* RX-TX ↩
                 *********************************************
6            #define RXpin               D1
7            #define TXpin               D2
8            SoftwareSerial unoSerial(RXpin, TXpin);
9            .........
10
11           void setup() {
12                ...........
13
14                unoSerial.begin(115200);
15                ...........
16           }
17           void loop() {
18                ...........
19
20                // Get Json Data Package from Arduino
21                DeserializationError error = deserializeJson(jsonBuffer↩
                      , unoSerial);
22                if (error){
23                     Serial.println("Fail receive data!");
24                     Serial.println(error.c_str());
25                     return;
26                }
27                ...........
28           }
```

deserializeJson() is a function of *AndroidJson.h* library, it used to get Json data package from serial communication.

3.2 Data uploading process

After the sensory data is received by the gateway, the data package will be uploaded to Adafruit Server by using MQTT protocol. Before data packet was sent, it is format to a string with Json format, is implemented below:

Listing 7: Json Data Format

```
{
    "temperature" : /*float type*/,
    "humidity" : /*float type*/,
    "date" : "yyyy-mm-ddThh:mm:ss"
}
```

Compared to the data format sent from the node, the packet sent to the server is changed the format of "date" to the standard date type of Json format. This will help clients is easier to decode data packet.
The C source code of UART data received is used to receiving data package from Sensor Node, is provided below:

Listing 8: MQTT sending data

```
.............
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

#include <ESP8266WiFi.h>
............

//************ Wifi via Access Point *********
#define WIFI_SSID       "Muoi Vinh"     //"6-No-2-Hope-3"
#define WIFI_PASSWORD   "0988258072"     //"01072000"

// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;

//************* Adafruit.io Setup **********
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "vitran"
#define AIO_KEY         "aio_kMjj660iSchQ5gVRBDF39N9COyEh"

//*************** Setup MQTT client class *****************
// Passing in the WiFi client and MQTT server and login details.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

//************* Feeds ***************
Adafruit_MQTT_Publish motion = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/temperature");
.........
```

```
28
29              void setup() {
30                  ...........
31
32                  connectWifi();
33                  ...........
34              }
35              void loop() {
36                  ...........
37
38                  if(!motion.publish(val.c_str()))
39                  Serial.println("Send data Failed!");
40                  else
41                  Serial.println("Data sent!");
42
43                  // Ping Adafruit.IO to keep the MQTT connection alive
44                  if(! mqtt.ping()) {
45                      mqtt.disconnect();
46                  }
47                  ...........
48              }
49              //*************** WiFi Connection *************
50              void connectWifi()
51              {
52                  // If already connected to WiFi, return to loop
53                  if (WiFi.status() == WL_CONNECTED){ return; }
54
55                  // Start a WiFi connection and enter SSID and Password
56                  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
57
58                  // While waiting on connection to start display "..."
59                  while (WiFi.status() != WL_CONNECTED) {
60                      delay(500);
61                      Serial.print(".");
62                  }
63                  Serial.println("Connected");
64
65                  // Run Procedure to connect to Adafruit IO MQTT
66                  MQTT_connect();
67              }
68
69              //************* MQTT Connect - Adafruit IO ***********
70              void MQTT_connect()
71              {
72                  int8_t ret;
73
74                  // Stop if already connected to Adafruit
75                  if (mqtt.connected()) { return; }
76                  Serial.println("Connecting to MQTT... ");
77
78                  mqtt.disconnect();
79
80                  // Connect to Adafruit, will return 0 if connected
```

```
81                  while ((ret = mqtt.connect()) != 0) {
82                      Serial.println(mqtt.connectErrorString(ret));
83                      Serial.println("Retrying MQTT...");
84                      mqtt.disconnect();
85                      delay(5000);                                    ↩
                            // wait 5 seconds
86                  }
87                  Serial.println("MQTT Connected!");
88              }
```

Data sent to MQTT server is character type, so we should change data format to string and use *c_str()* to change to character type.

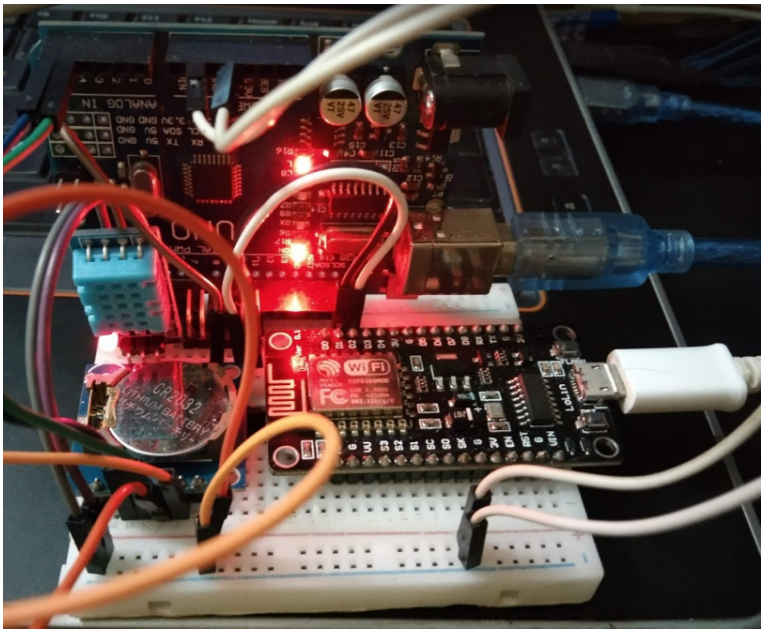

Figure 2: *Sensor Node and Gateway Node*

4 Server configuration

In this Project, as a description above, we use Adafruit.io server to serve data sent from Sensor Nodes.
Adafruit.io is a system that makes data useful. Our focus is on ease of use, and allowing simple data connections with little programming required. It includes client libraries that wrap our REST and MQTT APIs. So, it is very suitable for this Project.
Besides, we can use other web server like Thingspeak,...
A public server URL of this project is "https://io.adafruit.com/vitran/feeds/temperature".

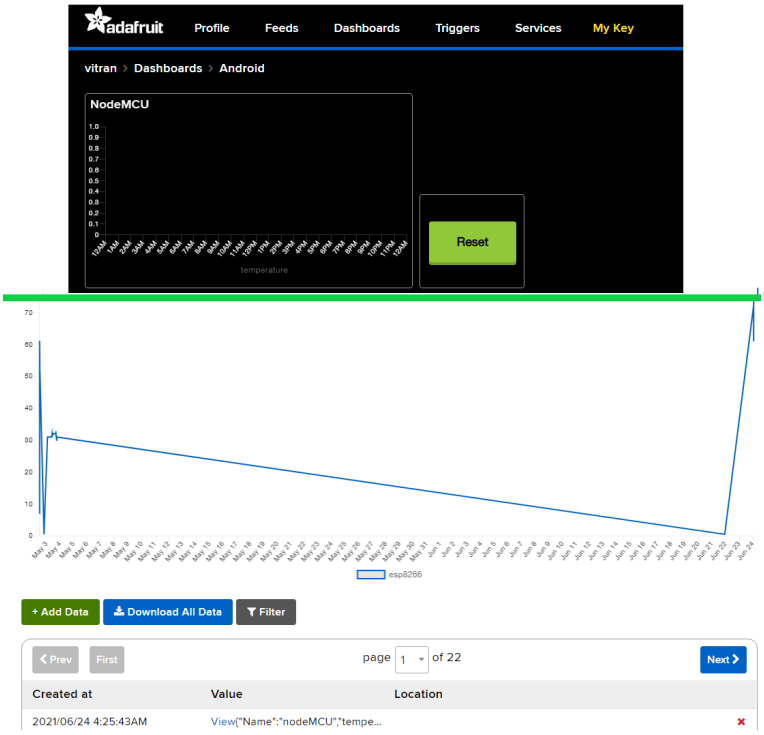

Figure 3: *Adafruit MQTT Server UI*

5 Monitoring application

After all, we build an Android app to get data from Adafruit server and present in the app UI as graphs and lastest updated data.
This app includes temperature and humidity graph and parameter path.
In parameter path, we can see name of Node sent data, the last time data was subscribed from server and and parts of the data after being decoded.

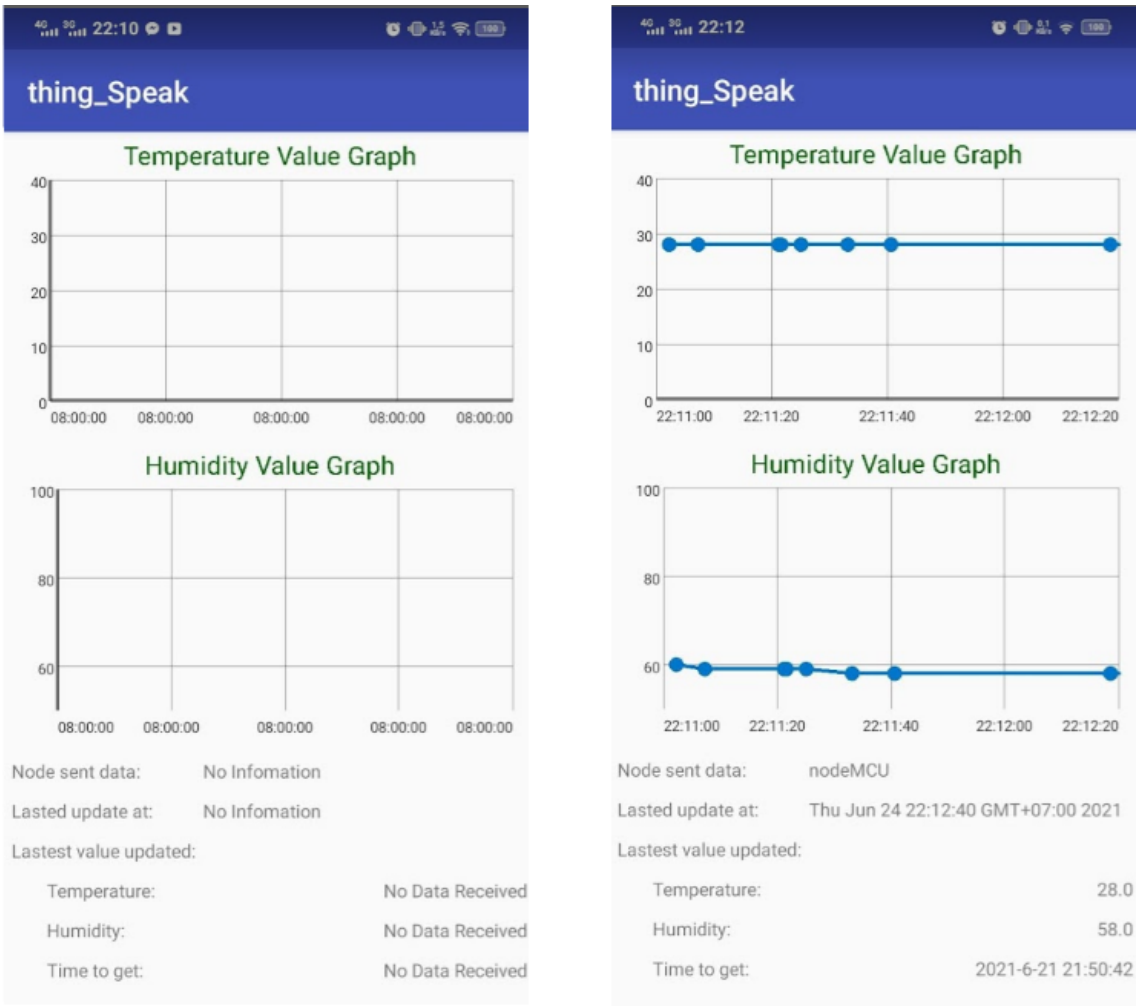

Figure 4: *Android app UI*

6 Conclusion

In conclusion, this project has realized serial communication between devices and wireless communication between devices and MQTT server.
In the future, this project can be extended with multiple Nodes and the android app can send commands to control the specific peripherals of each node.
You can watch demo video at
https://drive.google.com/file/d/1gWIa9NhSV-E2T5hbH-LKRj745DvZ4ytM/view?usp=sharing

Reference

[Web] Arduino, https://www.arduino.cc/

[Web] ArduinoJson v6, https://arduinojson.org/v6/doc/

[Web] Android Graph, https://www.geeksforgeeks.org/line-graph-view-in-android-with-example/