VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

LOGIC DESIGN PROJECT (CO3091)

Final Projects

Building A Wireless Sensor System & Wireless Timer

System 1. Air Temperature And Humidity Sensor

Lecturer: Nguyen Cao Tri
Student performs: Nguyen Dinh Dat – 1811869

Ho Chi Minh City, 01/ 2022

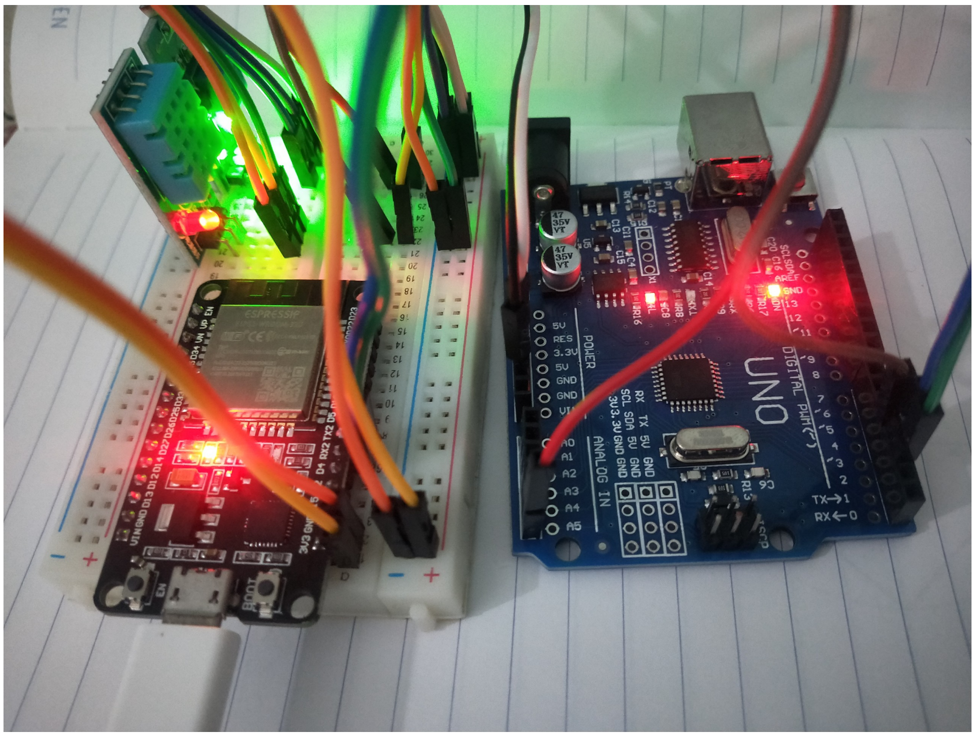
Content

1 Introduction

Building a system of wireless sensors for monitoring and supporting control of microclimate environments and devices in agricultural greenhouses or aquaculture environments,...

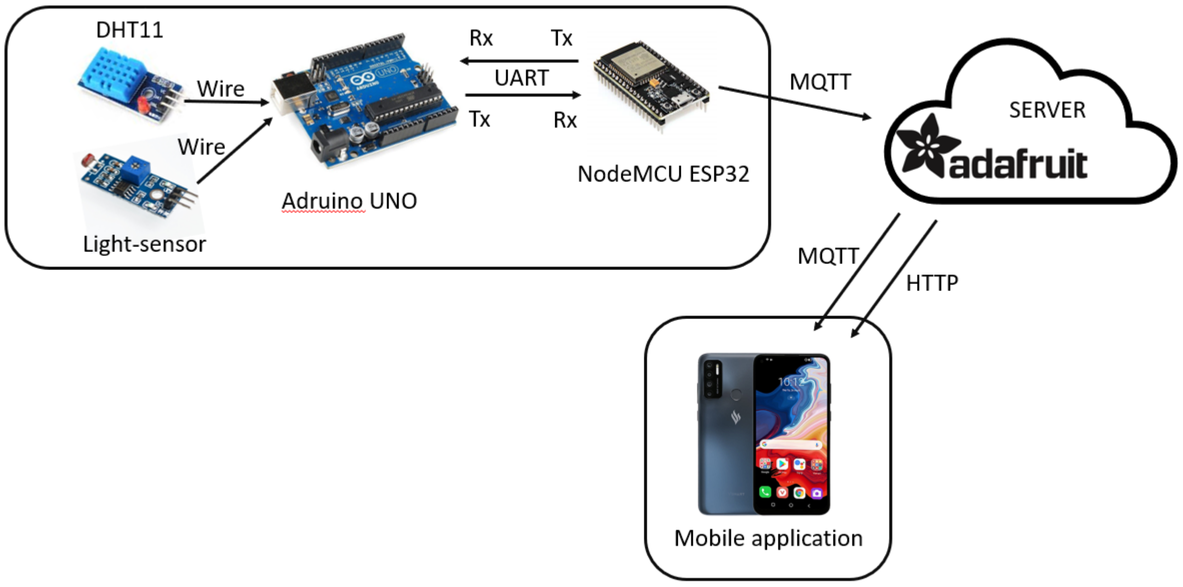
Each set of sensors will monitor one or several environmental parameters in real time and allow storing and statistical data on the server or providing central control stations to control related devices to ensure microclimate in greenhouses or aquaculture ponds. Allowing threshold alarms, .., allowing data transmission over wifi by central or bridged mechanism.

We use DHT11 sensor and module LDR light sensor. The data from these sensors will be collected by using Arduino Uno board. We use ESP32 to make a gateway and send it to the Adafruit Server.

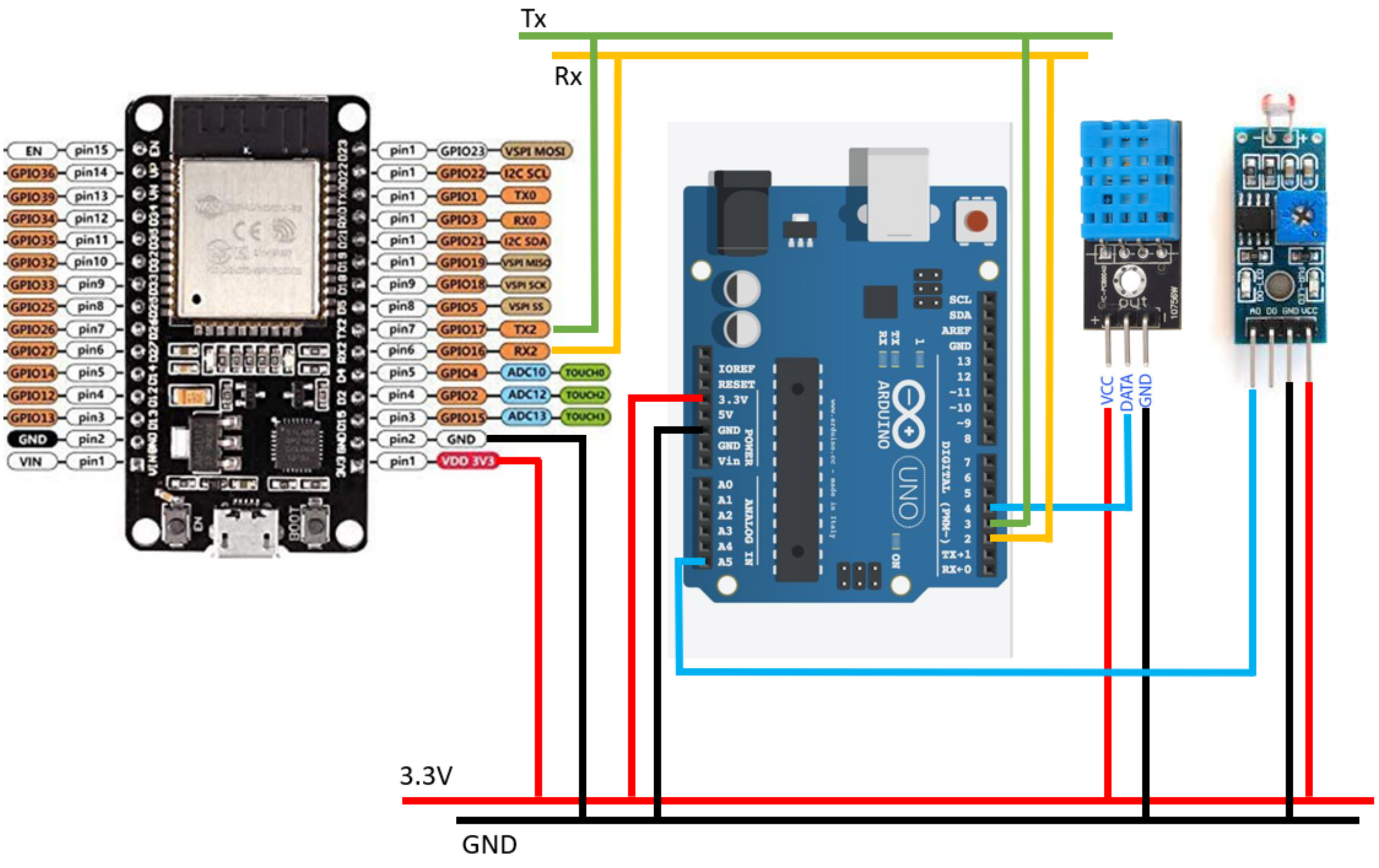


Hình 1: Hardware set up

The general description belows:



Hình 2: System description



Hình 3: Hardware description

2 Sensor implementation

Like we've mentioned at section 1, there are 3 type of sensors:

- **DHT11:** The main function of this sensor that it can take the value of temperature and humidity by using a capacitive humidity sensor and a thermistor.
- **LDR Light sensitive:** It using a light sensor and a simple ADC module to collect the digital value of light sensitive

After collecting data at the node sensor, we have to define a data framework to prepare for sending to gateway throughout UART communication. The frame data format of the sensor sending to the gateway is th JSON format. It totally has 5 feeds:

- **place:** This seed will contain the node sensor location. It consists of the name of ward and province.
- **node_id:** The node id will notify to user that which node sensor was sending the data to gateway and server. Because we just have one node sensor so that this value always be 1.
- **temperature:** This seed will contain the value of temperature which read from DHT11 sensor.
- **humidity:** his seed will contain the value of humidity which read from DHT11 sensor.
- **light:** his seed will contain the analog value of the light sensitive which read from module LDR light sensitive.

This is the example of 1 message which is sent from sensor node to gateway:

```
{"place": "Nghi Loc, Nghe An","node_id": 1, "temperature": 28.2,"humidity": 60,"light": 79}
```

The code implement station at the node sensor:

Listing 1: The C source code of Arduino Uno

```c
#include "DHT.h"
#include <SoftwareSerial.h>

SoftwareSerial sw(2,3);
int id = 99;
const int DHTPIN = 4;
const int DHTTYPE = DHT11;
int light = A5;

DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  dht.begin();
  sw.begin(9600);
}

void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float l = analogRead(light);


 sw.print("{");
 sw.print("\"place\": \"Nghi Loc, Nghe An\", ");
 sw.print("\"node_id\": 1, ");
 sw.print("\"temperature\": ");
 sw.print(t);
 sw.print(", ");
 sw.print("\"humidity\": ");
 sw.print(h);
 sw.print(", ");
 sw.print("\"light\": ");
 sw.print(l);
 sw.print("}");
 sw.println();

  delay(10000);
}
```

3 Gateway implementation

Like we've mentioned at section 1, we are using ESP32 as a gateway. Its features are UART received data, upload data to server and received data from server.

3.1 UART received data and data uploading process

At ARDUINO UNO board, we just only use command **Serial.print()** to send UART to ESP32. The data will be received on the gateway. The baudrate is **9600**. Whenever the **Serial.print()** in ARDUINO UNO board called, the information will be sent to gateway too. After receiving the data, these information will be sent with the rate is 10 seconds/cycle.

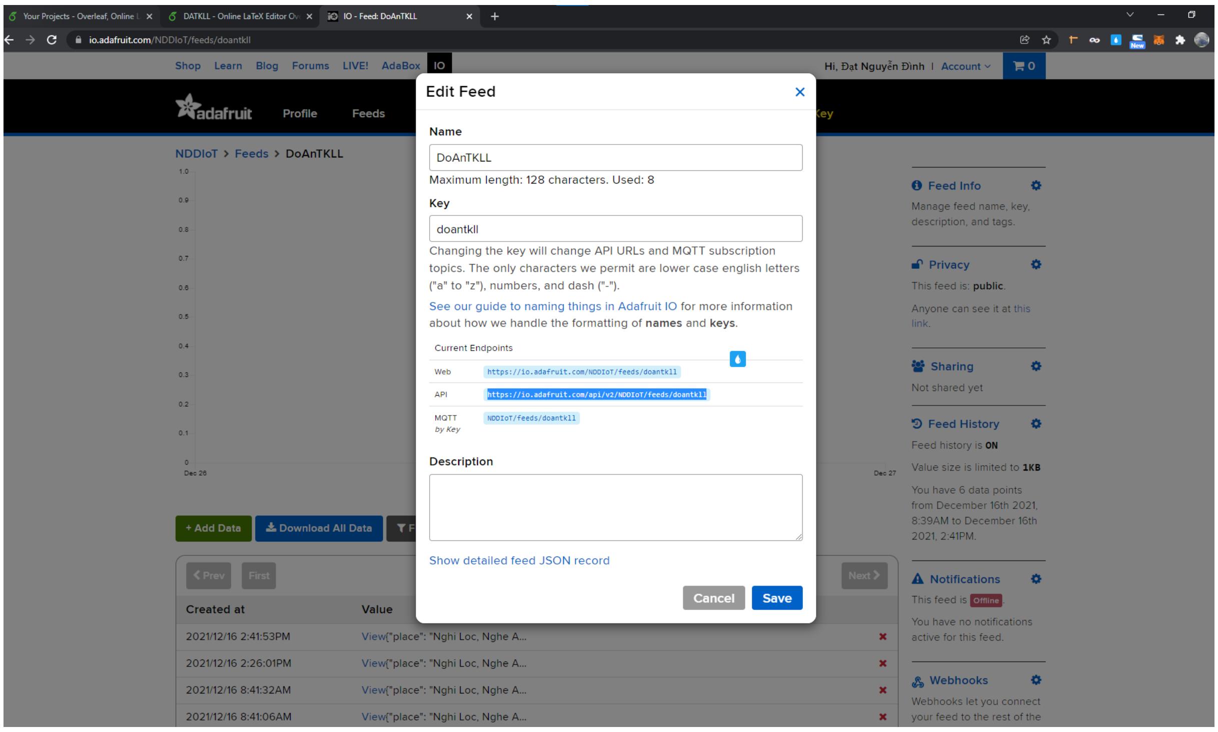
To send data to Adafruit Server, we using the MQTT protocol. Because the data which is sent from node sensor are already had the JSON format. So that, we just only take that data and send it to MQTT server. The code description will be showed below:

Listing 2: The Java source code to MQTT and send MQTT to server

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

#define RXD2 16
#define TXD2 17

#define WLAN_SSID "OPPO"         // name wifi
#define WLAN_PASS "12345670"     // password wifi

#define IO_SERVER     "io.adafruit.com"
#define IO_SERVERPORT 1883

#define IO_USERNAME   "NDDIoT"
#define IO_KEY        "aio_aGzS02OLLT4MqLhDWWaO6Ly4ZmfT"
WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, IO_SERVER, IO_SERVERPORT, ↩
    IO_USERNAME, IO_KEY);
Adafruit_MQTT_Publish SERIAL_PUBLISH = Adafruit_MQTT_Publish(&mqtt, ↩
    IO_USERNAME "/feeds/doantkll");

void MQTT_connect();

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);
  Serial.print(F("Connecting to "));
  Serial.println(WLAN_SSID);
  WiFi.begin(WLAN_SSID, WLAN_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("WiFi connected!");
  MQTT_connect();
  Serial.println("MQTT connected!");
}

void loop() {
  // put your main code here, to run repeatedly:
  MQTT_connect();
  if (Serial2.available() > 0) {
     char bfr[501];
     memset(bfr,0, 501);
     Serial2.readBytesUntil( '\n',bfr,500);
     Serial.println(bfr);
     MQTT_connect();
     SERIAL_PUBLISH.publish(bfr);
   }
}

void MQTT_connect() {

  while(!mqtt.connected()){
    Serial.print("Attempting MQTT connection...");
    if(mqtt.connect()){
      Serial.println("connected!");
    }else{
      Serial.println("try again in 5 seconds!");
      delay(5000);
    }
  }
}
```

3.2 Data received process

The goal that we want to build an Android App is have something can be "seen" for user to know the data exactly and conveniently. In this project, we want to get data information from Adafruit Server and show it in the Line Chart. In this case, we don't use MQTT protocol because it can't take the historic data. So that, we decide to use HTTP protocol by using OkHttp library. We just need an API HTTP URL of the feed which contains our data, the JSON format message will be generated and we just only take it down in the Android App.


Edit Feed

Name: DoAnTKLL

Maximum length: 128 characters. Used: 8

Key: doantkll

Changing the key will change API URLs and MQTT subscription topics. The only characters we permit are lower case english letters ("a" to "z"), numbers, and dash ("-").

See our guide to naming things in Adafruit IO for more information about how we handle the formatting of **names** and **keys**.

Current Endpoints

Web	https://io.adafruit.com/NDDIoT/feeds/doantkll
API	https://io.adafruit.com/api/v2/NDDIoT/feeds/doantkll
MQTT *by Key*	NDDIoT/feeds/doantkll

Description

Created at	Value
2021/12/16 2:41:53PM	View{"place": "Nghi Loc, Nghe A...
2021/12/16 2:26:01PM	View{"place": "Nghi Loc, Nghe A...
2021/12/16 8:41:32AM	View{"place": "Nghi Loc, Nghe A...
2021/12/16 8:41:06AM	View{"place": "Nghi Loc, Nghe A...

Feed Info: Manage feed name, key, description, and tags.

Privacy: This feed is: **public**. Anyone can see it at this link.

Sharing: Not shared yet

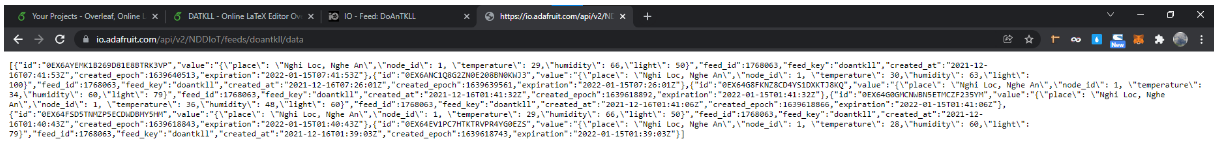
Feed History: Feed history is **ON**. Value size is limited to **1KB**. You have 6 data points from December 16th 2021, 8:39AM to December 16th 2021, 2:41PM.

Notifications: This feed is Offline. You have no notifications active for this feed.

Webhooks: Webhooks let you connect your feed to the rest of the

Hình 4: The HTTP API url at the feed information

Full feed data information: `https://io.adafruit.com/api/v2/NDDIoT/feeds/doantkll/data`


io.adafruit.com/api/v2/NDDIoT/feeds/doantkll/data

```
[{"id":"0EX6AYEMK1B269D81E8BTRK3VP","value":"{\"place\": \"Nghi Loc, Nghe An\",\"node_id\": 1, \"temperature\": 29,\"humidity\": 66,\"light\": 50}","feed_id":1768063,"feed_key":"doantkll","created_at":"2021-12-
16T07:41:53Z","created_epoch":1639640513,"expiration":"2022-01-15T07:41:53Z"},{"id":"0EX6ANC1Q8G2ZN0E208BN0KWJ3","value":"{\"place\": \"Nghi Loc, Nghe An\",\"node_id\": 1, \"temperature\": 30,\"humidity\": 63,\"light\":
100}","feed_id":1768063,"feed_key":"doantkll","created_at":"2021-12-16T07:26:01Z","created_epoch":1639639561,"expiration":"2022-01-15T07:26:01Z"},{"id":"0EX64G8FKNZ8CD4YS1DXKTJ8KQ","value":"{\"place\": \"Nghi Loc, Nghe An\",\"node_id\": 1, \"temperature\":
34,\"humidity\": 60,\"light\": 79}","feed_id":1768063,"feed_key":"doantkll","created_at":"2021-12-16T01:41:32Z","created_epoch":1639618892,"expiration":"2022-01-15T01:41:32Z"},{"id":"0EX64G0GMCNWBN5ETMCZF235YM","value":"{\"place\": \"Nghi Loc, Nghe
An\",\"node_id\": 1, \"temperature\": 36,\"humidity\": 48,\"light\": 60}","feed_id":1768063,"feed_key":"doantkll","created_at":"2021-12-16T01:41:06Z","created_epoch":1639618866,"expiration":"2022-01-15T01:41:06Z"},
{"id":"0EX64FSD5TNMZP5ECDWDBMY5HM","value":"{\"place\": \"Nghi Loc, Nghe An\",\"node_id\": 1, \"temperature\": 29,\"humidity\": 66,\"light\": 50}","feed_id":1768063,"feed_key":"doantkll","created_at":"2021-12-
16T01:40:43Z","created_epoch":1639618843,"expiration":"2022-01-15T01:40:43Z"},{"id":"0EX64EV1PC7HTKTRVPR4YG0EZS","value":"{\"place\": \"Nghi Loc, Nghe An\",\"node_id\": 1, \"temperature\": 28,\"humidity\": 60,\"light\":
79}","feed_id":1768063,"feed_key":"doantkll","created_at":"2021-12-16T01:39:03Z","created_epoch":1639618743,"expiration":"2022-01-15T01:39:03Z"}]
```

Hình 5: Full feed data information at JSON format

First, we initialize the OkHttpClient Object.

Listing 3: The Java source code to initialize OkHttp object

```java
OkHttpClient okHttpClient = new OkHttpClient.Builder()
            .connectTimeout(15, TimeUnit.SECONDS)
            .writeTimeout(15, TimeUnit.SECONDS)
            .readTimeout(15, TimeUnit.SECONDS)
            .retryOnConnectionFailure(true)
            .build();
```

Then, we make a request to server for taking the data.

Listing 4: The Java source code to send request to server and taking data from URL

```java
Request.Builder builder = new Request.Builder();
        builder.url("https://io.adafruit.com/api/v2/NDDIoT/feeds/↩
            doantkll/data");

        Request request = builder.build();
        try {
            Response response = okHttpClient.newCall(request).execute↩
                ();
            return response.body().string();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
```

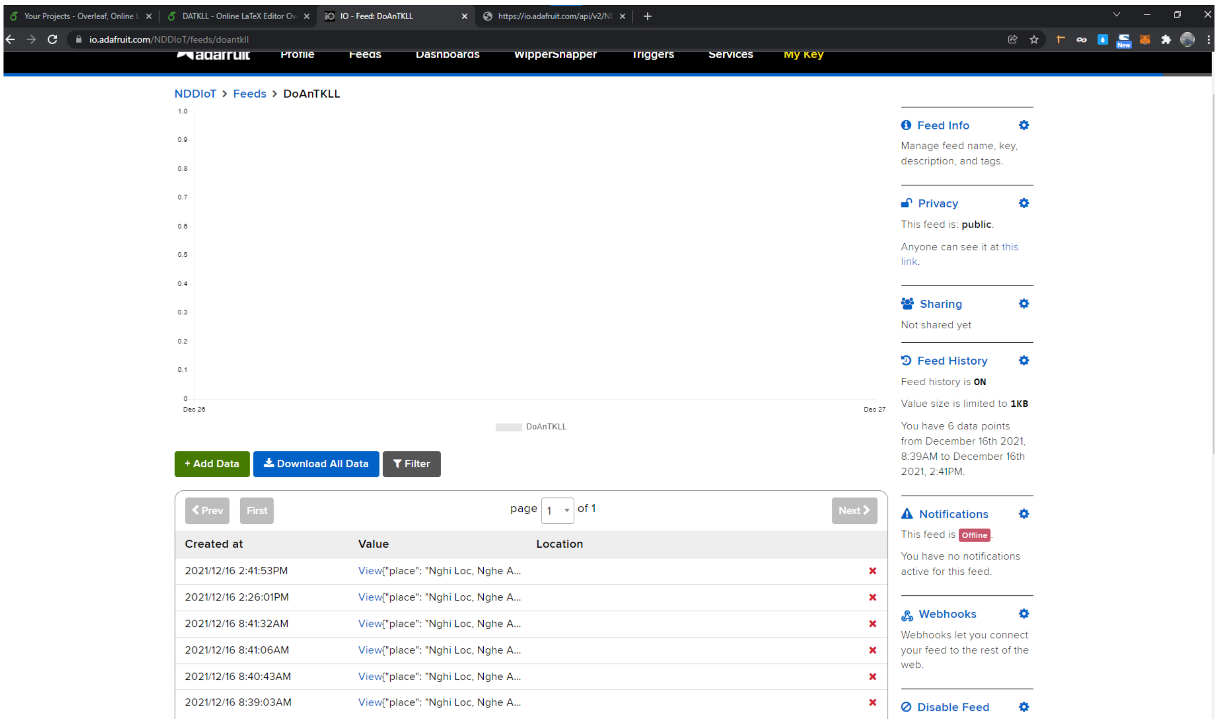
After receving data from Adafruit Server, we parse the JSON string to get the necessary information and draw it to the Line Chart by using MPAndroidChart library.

Listing 5: The Java source code to parse JSON string which received from server and graph it to Line Chart

```java
 @Override
    protected void onPostExecute(String s) {
        if (!s.equals("")) {

            listNgheAn = new ArrayList<NodeInfo>();

            try {

                JSONArray JA = new JSONArray(s);

                for (int i=JA.length()-1; i>=0; i--) {
                    JSONObject JO = JA.getJSONObject(i);

                    String val = JO.getString("value");
                    while (val.indexOf('/') > 0) {
                        val = val.substring(0, val.indexOf('/')) + val.substring(val.indexOf('/')+1);
                    }

                    //StatisticPage.textView.setText(val);

                    JSONObject _JO = new JSONObject(val);
                    NodeInfo obj = new NodeInfo();

                    String location = _JO.getString("place");
                    obj.ward = location.substring(0, location.indexOf(','));
                    obj.province = location.substring(location.indexOf(',')+1);
                    obj.node_id = _JO.getInt("node_id");
                    obj.temperature = (float) _JO.getDouble("temperature");
                    obj.humidity = (float) _JO.getDouble("humidity");
                    obj.light_sensitive = Math.round(_JO.getInt("light")); //*100.0/4095.0*100

                    if (obj.province.contains(" Nghe An")) {
                        listNgheAn.add(obj);
                    }

                }


                MainActivity.listNgheAn = listNgheAn;
                MainActivity.getIndex();

            } catch (JSONException e) {
                e.printStackTrace();
            }

        }

        super.onPostExecute(s);
    }
```

4 Server configuration

The server URL: `https://io.adafruit.com/NDDIoT/feeds/doantkll`.



page 1 of 1

Created at	Value	Location
2021/12/16 2:41:53PM	View{"place": "Nghi Loc, Nghe A...	
2021/12/16 2:26:01PM	View{"place": "Nghi Loc, Nghe A...	
2021/12/16 8:41:32AM	View{"place": "Nghi Loc, Nghe A...	
2021/12/16 8:41:06AM	View{"place": "Nghi Loc, Nghe A...	
2021/12/16 8:40:43AM	View{"place": "Nghi Loc, Nghe A...	
2021/12/16 8:39:03AM	View{"place": "Nghi Loc, Nghe A...	

Feed Info
Manage feed name, key, description, and tags.

Privacy
This feed is: **public**.
Anyone can see it at this link.

Sharing
Not shared yet

Feed History
Feed history is **ON**
Value size is limited to **1KB**
You have 6 data points from December 16th 2021, 8:39AM to December 16th 2021, 2:41PM.

Notifications
This feed is Offline.
You have no notifications active for this feed.

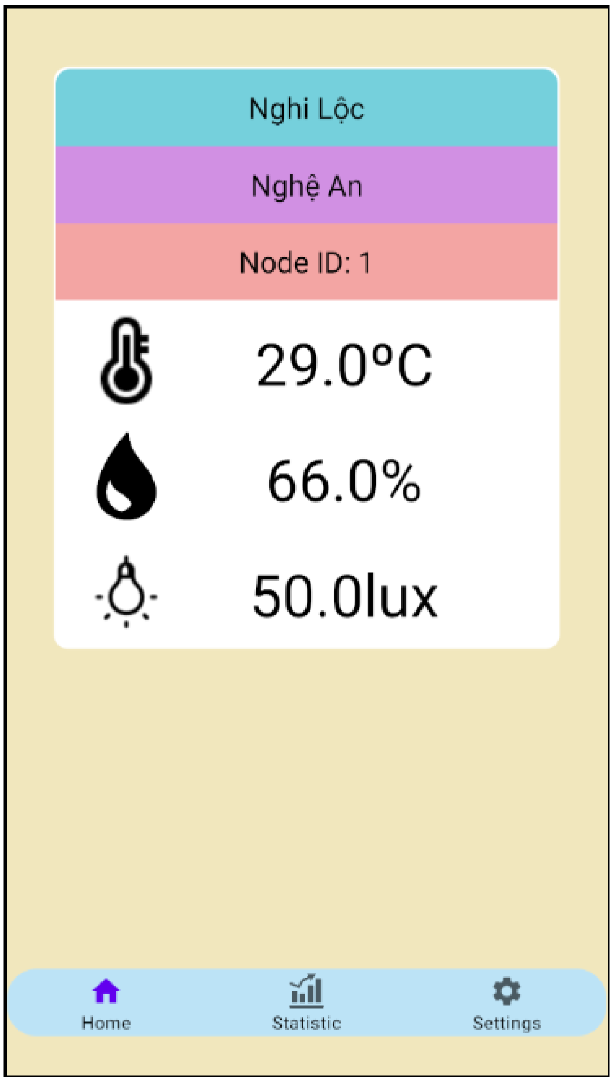
Webhooks
Webhooks let you connect your feed to the rest of the web.

Disable Feed

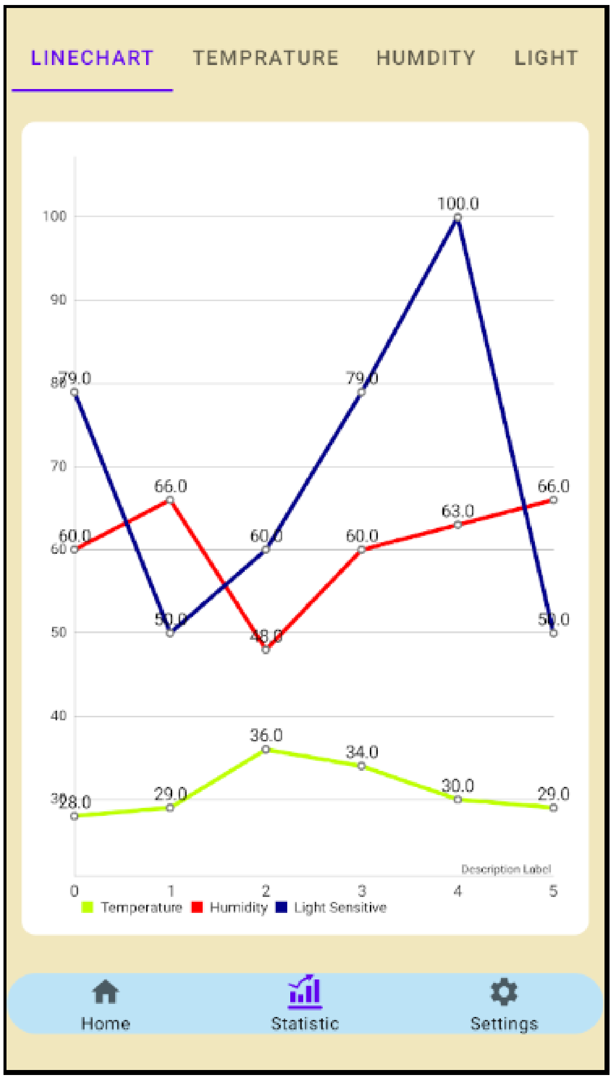
Hình 6: Interface of server Adafruit

5 Monitoring application

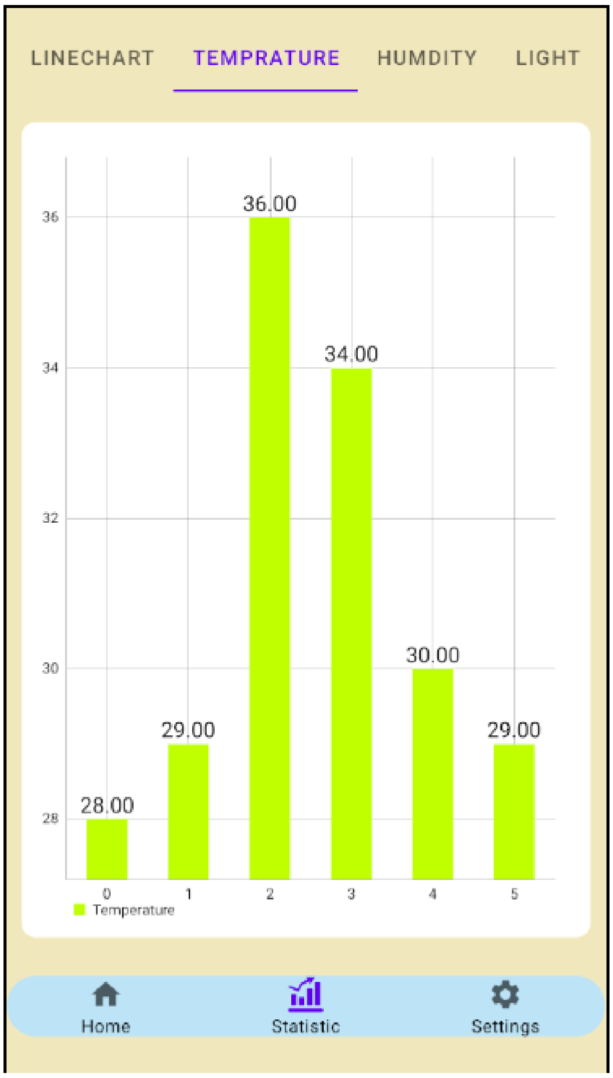
The phone app displays temperature, brightness, time and location data information at the sensor location, and the data is displayed on multiple graphs. When the application starts to open, the graph will display the data of the previous 5 fetches, and then continue to update the latest data to the graph.


Nghi Lộc
Nghệ An
Node ID: 1
29.0°C
66.0%
50.0lux

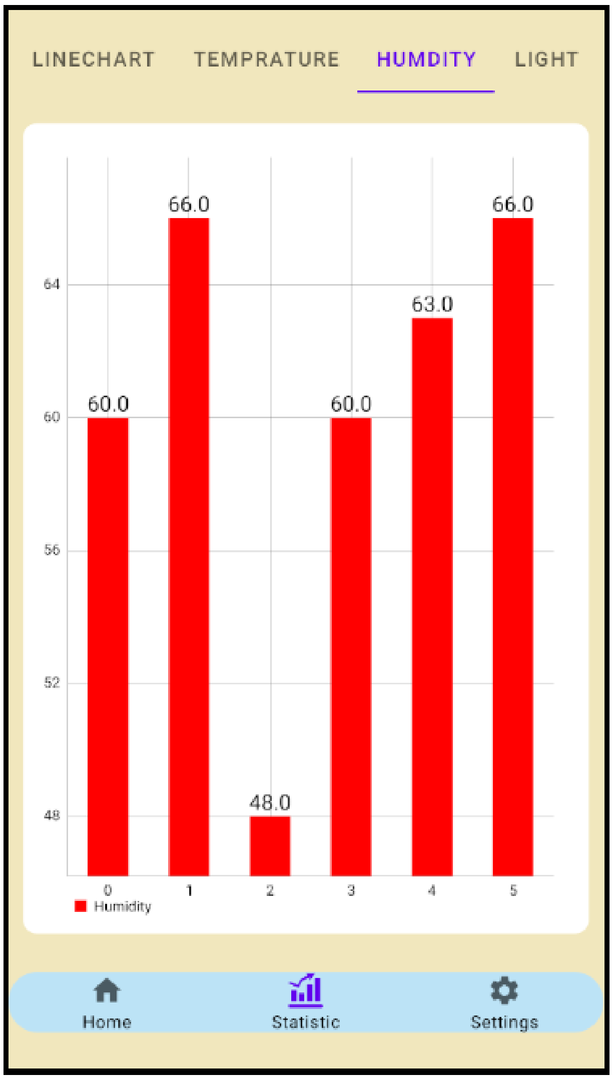
Hình 7: Interface of the mobile app HOME tag



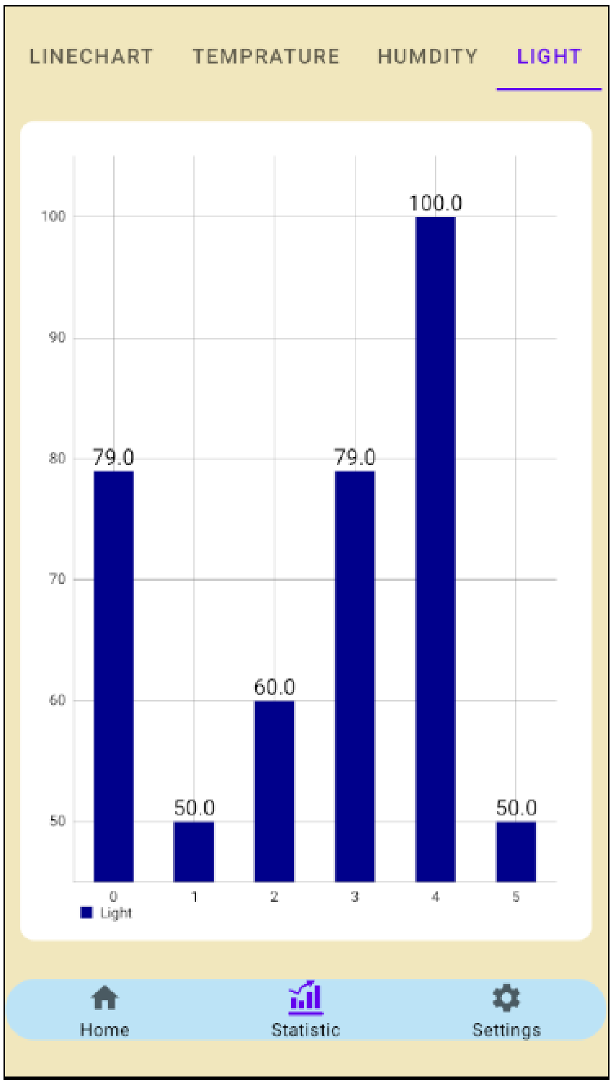
Hình 8: Interface of the mobile app Statistic tag



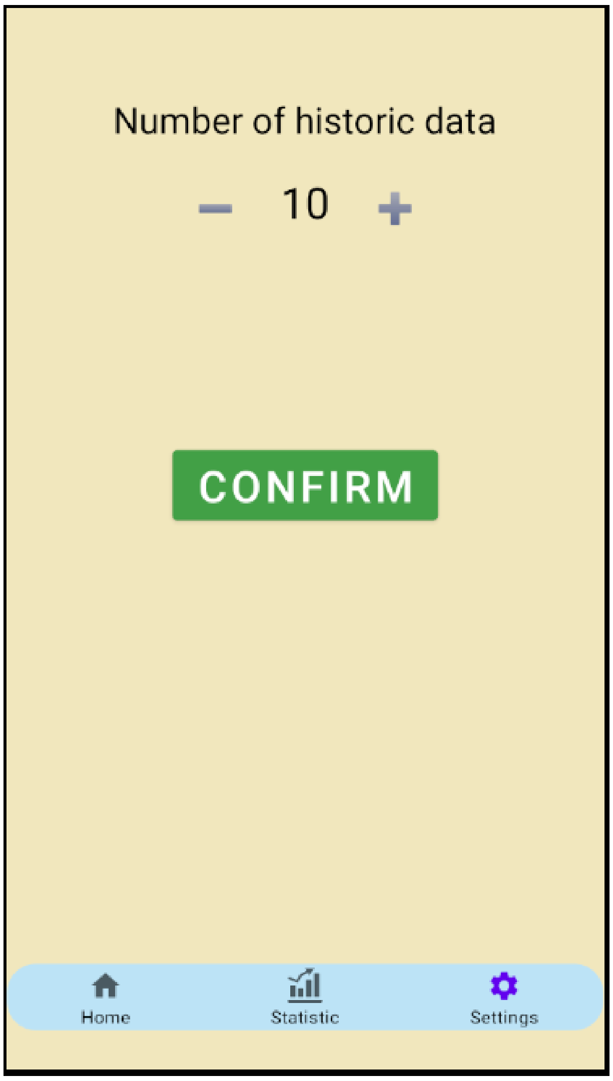
Hình 9: Interface of the mobile app Statistic tag



Hình 10: Interface of the mobile app Statistic tag



Hình 11: Interface of the mobile app Statistic tag



Hình 12: Interface of the mobile app SETTINGS tag

6 Conclusion

During the epidemic conditions, the team managed to complete the main requirements assigned and added some extra functions to the user's phone application such as being able to view temperature, humidity and light data on any chart. Help users have more ways to evaluate data.

The link Source Code for project: `https://drive.google.com/drive/folders/1TWW3J6LgNc-poZoL1d89R` `usp=sharing`.

The link Video Demo for project: `https://drive.google.com/file/d/1EhLr06QC02H0CEQettupLWs_ZW5rDEpX/view?usp=sharing`.

Tài liệu

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

[Web] Adafruit, https://io.adafruit.com/api/docs/

[Web] Github, https://github.com/PhilJay/MPAndroidChart

[Web] Codepath, https://guides.codepath.com/android/Using-OkHttp

[Accessories] DHT11, https://shopee.vn/product/148048328/4712723975?smtt=0.140313054-1637262474.9

[Accessories] LDR, https://shopee.vn/product/148048328/4806528974?smtt=0.140313054-1637262451.9

[Accessories] ARDUINO, https://shopee.vn/product/148048328/5321985783?smtt=0.140313054-1637262498.9

[Accessories] ESP32, https://shopee.vn/product/148048328/5109209715?smtt=0.140313054-1637262416.9