BỘ GIÁO DỤC & ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH

----------⸸Δ⸸----------

CƠ SỞ VÀ ỨNG DỤNG IOTS

BÀI TẬP CUỐI HỌC KỲ

ĐỀ TÀI: Hệ thống thu thập dữ liệu thông qua LoRa và tải dữ liệu lên NodeJS, Firebase, MQTT

Link video: https://youtu.be/DL2L-WOcjxs

Giảng viên hướng dẫn: Trương Quang Phúc

Sinh viên thực hiện:

Nguyễn Tấn An	**19151069**
Bùi Lê Anh	**19151099**
Nguyễn Quốc Khánh	**19151140**

Tp. Hồ Chí Minh, tháng 11 năm 2022

Thành viên	Phân công nhiệm vụ	Email
Nguyễn Tấn An	Thiết lập phần cứng và giao tiếp LoRa với ESP Master	19151069@student.hcmute.edu.vn
Bùi Lê Anh	Giao tiếp ESP Master với web server thông qua MQTT	19151099@student.hcmute.edu.vn
Nguyễn Quốc Khánh	Xây dựng web server NodeJS và giao diện người dùng. Giao tiếp ESP Master với web server thông qua Firebase	19151140@student.hcmute.edu.vn

MỤC LỤC

1. Giới thiệu

Nhờ vào Internet vạn vật (IoT - Internet of Things) mà chúng ta có thể tích hợp các hệ thống truyền thông khác nhau sử dụng để kết nối, thu thập dữ liệu và điều khiển nhiều thiết bị khác nhau chỉ với một chiếc điện thoại hoặc laptop được kết nối internet. IoT giúp chuyển các gói dữ liệu qua mạng được kết nối tiết kiệm thời gian và tiền bạc. Internet of Things hiện đang là một giải pháp hữu ích cho sản xuất, bán hàng, cuộc sống mang lại những hiệu quả vượt trội.

Dựa vào sự tiện lợi đó, hệ thống IoT thu thập dữ liệu, điều khiển thông qua LoRa và tải dữ liệu lên NodeJS, Firebase Realtime Database, MQTT được thực hiện để thu thập dữ liệu, giám sát, và điều khiển hệ thống với các cảm biến, vi điều khiển ESP8266 được kết nối với nhau thông qua mạng truyền thông LoRa và hệ thống lưu trữ dữ liệu Firebase Realtime Database. Hệ thống có khả năng thu thập và điều khiển linh hoạt, nhanh chóng ở khoảng cách xa.

Với công nghệ Lora (Long Range Radio), chúng ta có thể truyền dữ liệu với khoảng cách lên hàng km mà không cần sử dụng cáp vật lý để kết nối giữa các node, từ đó giảm chi phí và sự phức tạp từ việc lắp đặt. Do đó, LoRa có thể được áp dụng rộng rãi trong các ứng dụng thu thập dữ liệu như sensor network trong đó các sensor node có thể gửi giá trị đo đạc về trung tâm cách xa hàng km và có thể hoạt động với battery trong thời gian dài trước khi cần thay pin. Vì vậy Lora được chọn để giải quyết vấn đề khoảng cách khi truyền dữ liệu.

Cùng với đó, một hệ thống IoT không thể thiếu một server để có thể xử lý, lưu trữ dữ liệu. NodeJS khả năng tạo ra được các ứng dụng có tốc độ cực kỳ nhanh, xử lý được nhu cầu sử dụng của lượng khách truy cập ‘khổng lồ’ trong thời gian cực ngắn, có thể tương thích với nhiều thiết bị, chạy đa nền tảng, đồng thời đáp ứng được yêu cầu về thời gian thực realtime đối với hệ thống IoT. NodeJS là một lựa chọn hoàn hảo cho hệ thống khi có thể đáp ứng được yêu cầu về tốc độ xử lý, đa nền tảng và đáp ứng thời gian thực.

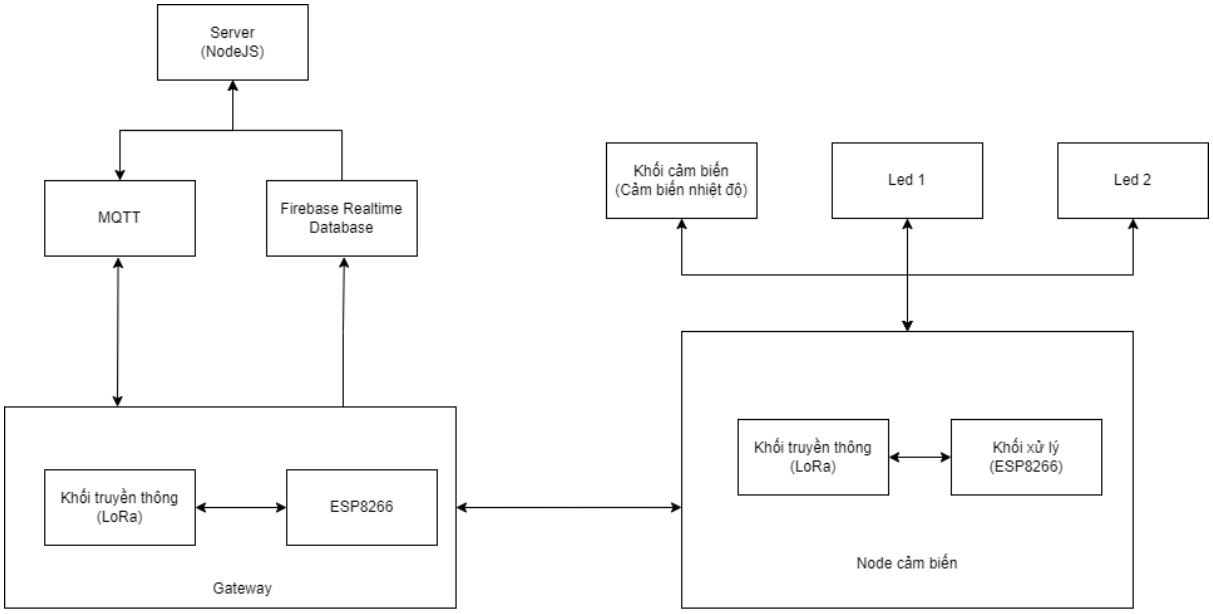
Để có thể lưu trữ dữ liệu một cách dễ dàng, tiện lợi, nhanh chóng hệ thống đã sử dụng Firebase Realtime Database. Realtime Database là một cơ sở dữ liệu NoQuery. Realtime Database được thiết kế cho việc thực hiện nhanh chóng. Điều này cho phép xây dựng trải nghiệm thời gian thực tuyệt vời. Điều đặc biệt là dịch vụ cho phép dùng thử miễn phí và sẽ chỉ phải trả phí nếu lượng truy cập trên ứng dụng vượt quá giới hạn mà firebase cung cấp và đối với quy mô của hệ thống thì bản dùng thử miễn phí hoàn toàn đáp ứng đủ dung lượng sử dụng.

MQTT (Message Queuing Telemetry Transport) là một giao thức truyền thông theo mô hình Publish/Subscribe. Giao thức này được sử dụng cho các thiết bị IoT với băng thông thấp, khả năng sử dụng được trong mạng lưới không ổn định và độ tin cậy cao phù hợp để sử dụng với vi điều khiển ESP8266. MQTT dựa trên một Broker. Hệ thống sử dụng Adafruit.io - một dịch vụ đám mây kết nối thông qua internet cho phép các thiết bị IoT dễ dàng gửi và nhận dữ liệu theo mô hình MQTT. [1]

Một hệ thống IoT không thể thiếu vi điều khiển để có thể giao tiếp, xử lý thông tin từ các cảm biến đến server. Hệ thống thu thập dữ liệu sử dụng ESP8266 vì module cung cấp khả năng xử lý và lưu trữ tích hợp mạnh mẽ, cho phép tích hợp dễ dàng với các cảm biến. ESP2866 là một hệ thống hoàn chỉnh hoặc khép kín trên mạch chip (SOC), module Wifi với ngăn xếp giao thức IP/TCP. ESP8266 được sản xuất bởi hãng Espressif Systems ở Thượng Hải, Trung Quốc do đó có giá thành khá rẻ và phù hợp với mô hình hệ thống. [2]

2. Thiết kế

2.1. Sơ đồ khối



Hình 1: Sơ đồ khối

Hệ thống bao gồm 3 phần chính là Node cảm biến, Gateway và Server.

Khối server: có chức năng giao tiếp với cơ sở dữ liệu (Firebase Realtime Database) để lưu trữ dữ liệu, hiển thị thông tin, dữ liệu cần thiết từ các cảm biến, gửi tín hiệu từ người dùng thông qua MQTT để điều khiển cơ cấu chấp hành từ xa. Đóng vai trò là trung tâm xử lý và giao diện người dùng.

Khối gateway: khối này có chức năng điều phối dữ liệu và thu thập dữ liệu của các node của hệ thống (ở hệ thống này sử dụng 1 node). Bộ xử lý master (ESP8266) của khối gateway sẽ gửi và nhận thông tin đến server NodeJS thông qua cơ sở dữ liệu Firebase Real Time Database và MQTT.

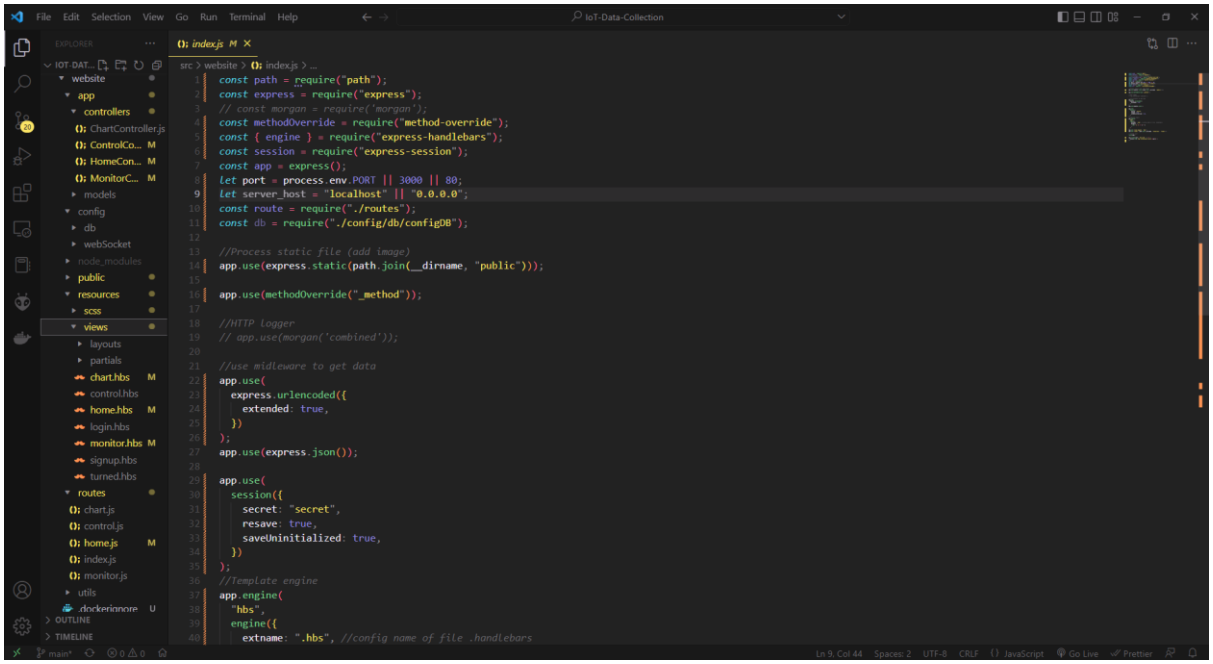
Khối node cảm biến: khối xử lý (ESP8266) sẽ đọc giá trị cảm biến DHT11, trạng thái các Led và sau đó gửi các giá trị đọc được sang khối master thông qua module truyền thông Lora E32.

Đối với luồng điều khiển: server NodeJS sẽ gửi tín hiệu điều khiển đến khối gateway thông qua server MQTT của adafruit.io. Khối gateway tiếp tục gửi tín hiệu điều khiển xuống node cảm biến thông qua hai module LoRa E32-TTL-100.

Và bộ xử lý ESP8266 của node sẽ thực hiện bật/tắt đèn dựa vào tín hiệu nhận được.

2.2. Khối server NodeJS

NodeJS là một nền tảng được xây dựng trên "V8 Javascript engine" được viết bằng c++ và Javascript. Nền tảng này được phát triển bởi Ryan Lienhart Dahl vào năm 2009. [3]



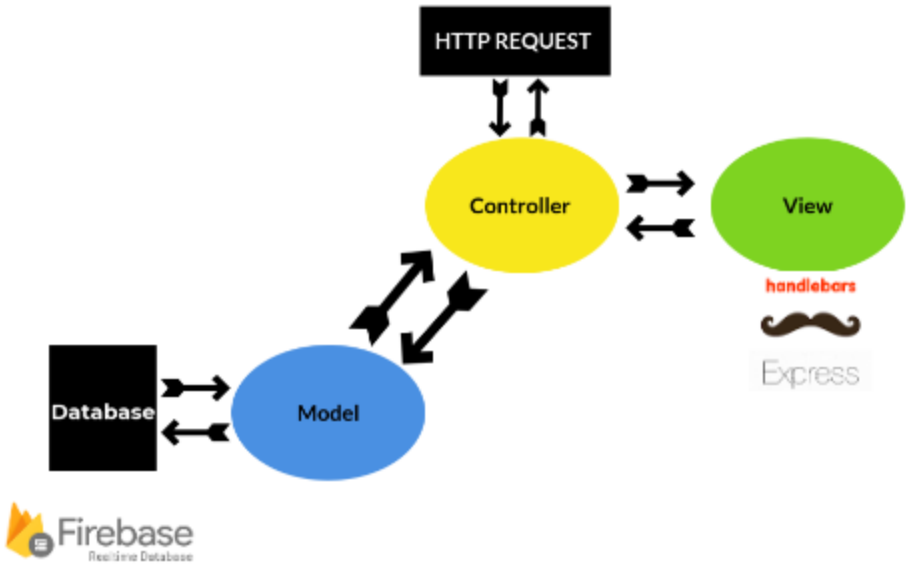
Hình 2: NodeJS Server

Web server sử dụng Expressjs - một framework được xây dựng trên nền tảng của Nodejs cung cấp các tính năng mạnh mẽ để phát triển web hoặc mobile. Expressjs hỗ trợ các method HTTP và midleware tạo ra API vô cùng mạnh mẽ và dễ sử dụng.

Server được thiết kế với mô hình MVC (Model – View – Controller) để có thể dễ dàng duy trì, mở rộng chức năng và sửa lỗi.

Mô hình MVC là một mẫu kiến trúc phần mềm để tạo lập giao diện người dùng trên máy tính. MVC chia thành ba phần được kết nối với nhau và mỗi thành phần đều có một nhiệm vụ riêng của nó và độc lập với các thành phần khác. Tên gọi 3 thành phần:

- Model (dữ liệu): Quản lí xử lí các dữ liệu. Là bộ phận có chức năng lưu trữ toàn bộ dữ liệu của ứng dụng. Một model là dữ liệu được sử dụng bởi chương trình. Đây có thể là cơ sở dữ liệu, hoặc file XML bình thường hay một đối tượng đơn giản. Ở hệ thống này, model là Firebase Realtime Database để lưu trữ dữ liệu như nhiệt độ, thông tin người dùng,…
- View (giao diện): Nới hiển thị dữ liệu cho người dùng. Chẳng hạn như hiển thị nhiệt độ, biểu đồ,... Nó bao gồm bất cứ thứ gì mà người dùng có thể nhìn thấy được. Trong hệ thống, thành phần view sử dụng express-handlebars (một thư viện javascrip rất mạnh mẽ, có thể binding data vào một templete để hiển thị ra website) làm giao diện người dùng.
- Controller (bộ điều khiển): Điều khiển sự tương tác của hai thành phần Model và View. Là bộ phận có nhiệm vụ xử lý các yêu cầu người dùng đưa đến thông qua View, nhận input và thực hiện các update tương ứng.

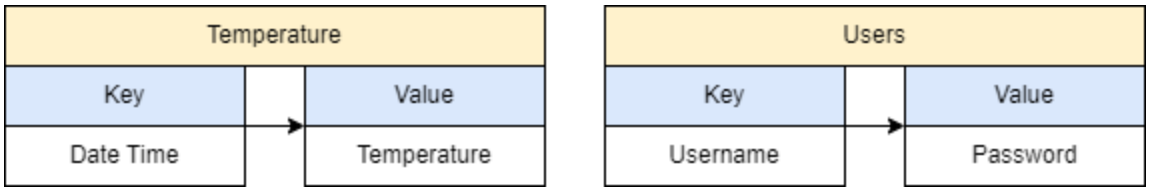


Hình 3: Mô hình MVC

Server có một số chức năng chính như: Login (đăng nhập), Signup (đăng ký), Monitor (giám sát), Chart (vẽ biểu đồ), Control (điều khiển).

2.2.1. Thiết kế cơ sở dữ liệu

Vì cơ sở dữ liệu hệ thống này dùng là Firebase Realtime Database – một cơ sở dữ liệu NoSQL nên cở sở dữ liệu của hệ thống sử dụng dạng Key-Value pair storage (lưu trữ theo từng cặp Key-Value) như hình bên dưới:

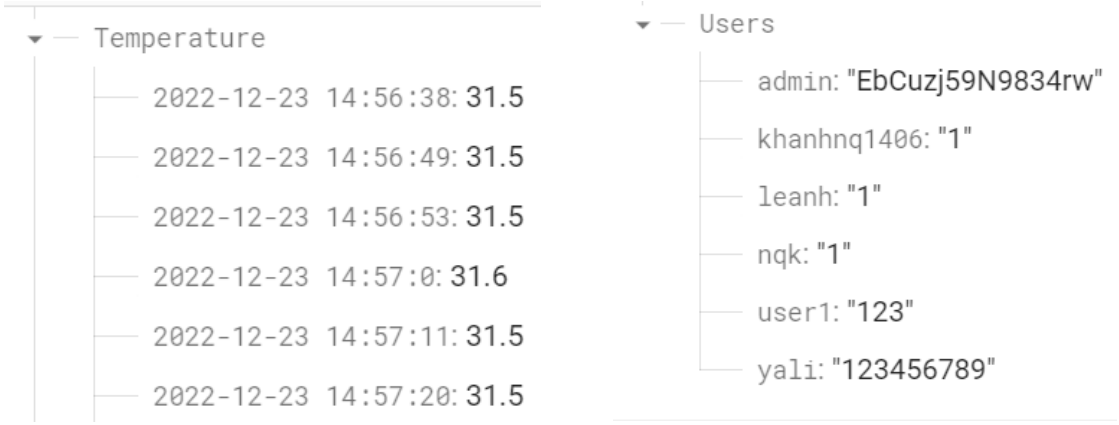


Hình 4: Cơ sở dữ liệu

Cơ sở dữ liệu bao gồm 2 bảng chính là Temperature và Users.

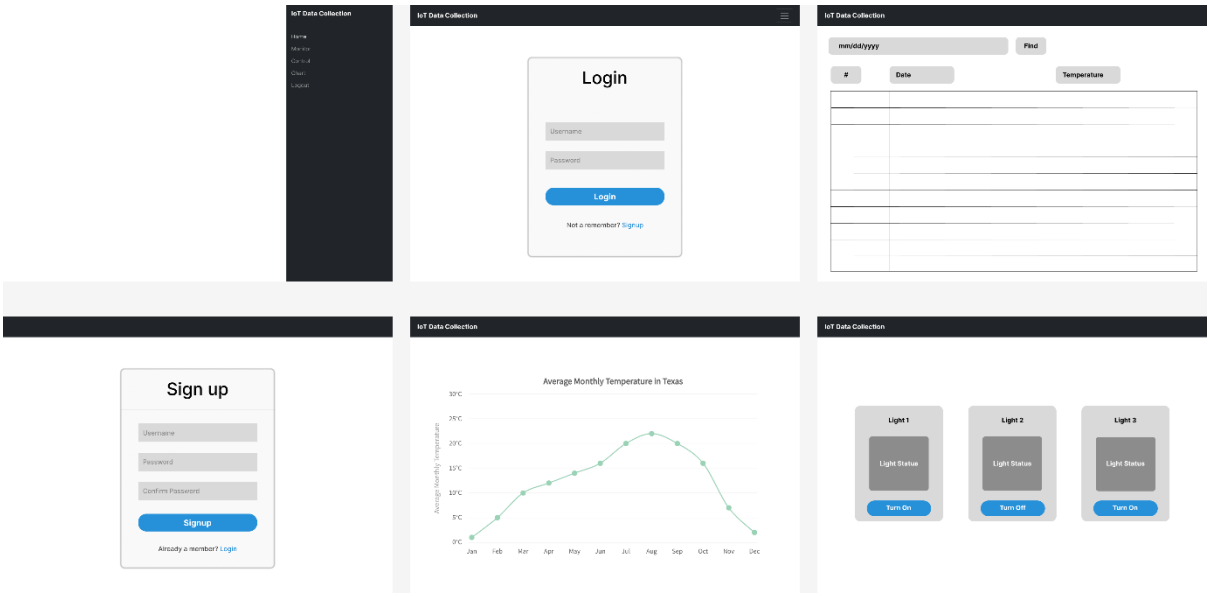
Bảng lưu trữ Temperature có chức năng lưu trữ nhiệt độ đo được tại một thời gian nhất định với key là thời gian thu thập và value là nhiệt độ.

Bảng lưu trữ Users có chức năng lưu thông tin đăng nhập của người dùng với key là username và value là password.



Hình 5: Bảng lưu trữ nhiệt độ và thông tin đăng nhập

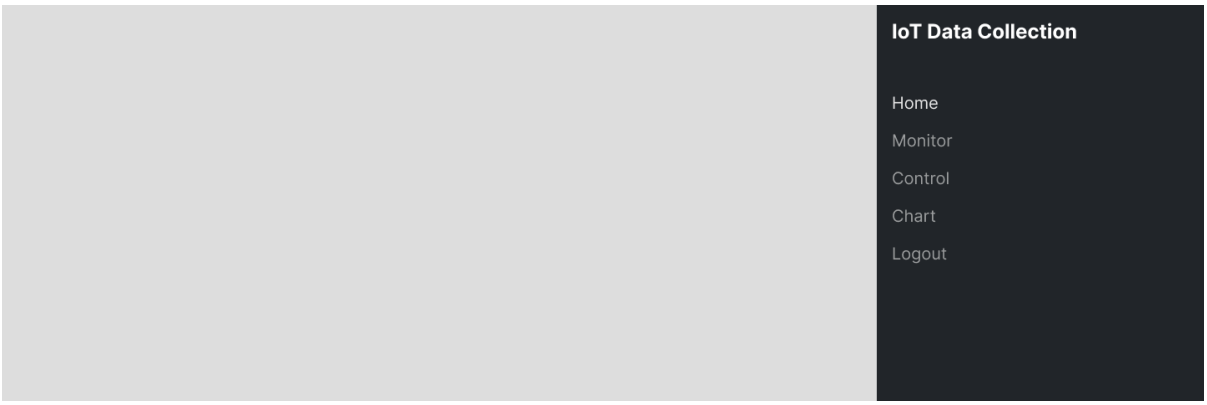
2.2.2. Thiết kế giao diện



Hình 6: Giao diện website

Website sẽ có 1 thanh navbar và 5 chức năng chính. Giao diện được thiết kế tổng quan trên Figma và được thực hiện lập trình bằng thư viện express-handlebars.

a) Thanh điều hướng



Hình 7: Thiết kế giao diện thanh điều hướng

Thanh điều hướng để dẫn đến 5 chức năng chính là Home, Monitor, Control, Chart và Logout. Khi người dùng yêu cầu mở thanh điều hướng thì thanh điều hướng sẽ xuất hiện ở bên phải màn hình.

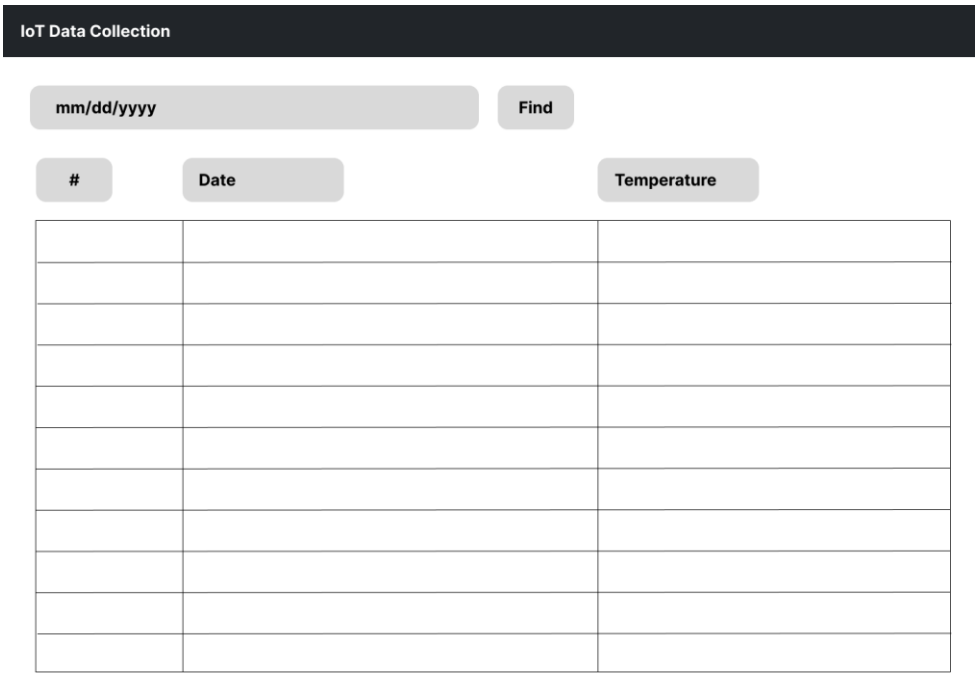
b) Giao diện đăng nhập/đăng ký

Hình: Thiết kế giao diện đăng nhập, đăng ký

Màn hình đăng nhập và đăng ký khá tương tự nhau. Cả 2 đều có ô input để nhập vào username và password, nút nhấn để xác nhận. Có thể điều hướng đến trang đăng ký từ đăng nhập và ngược lại.

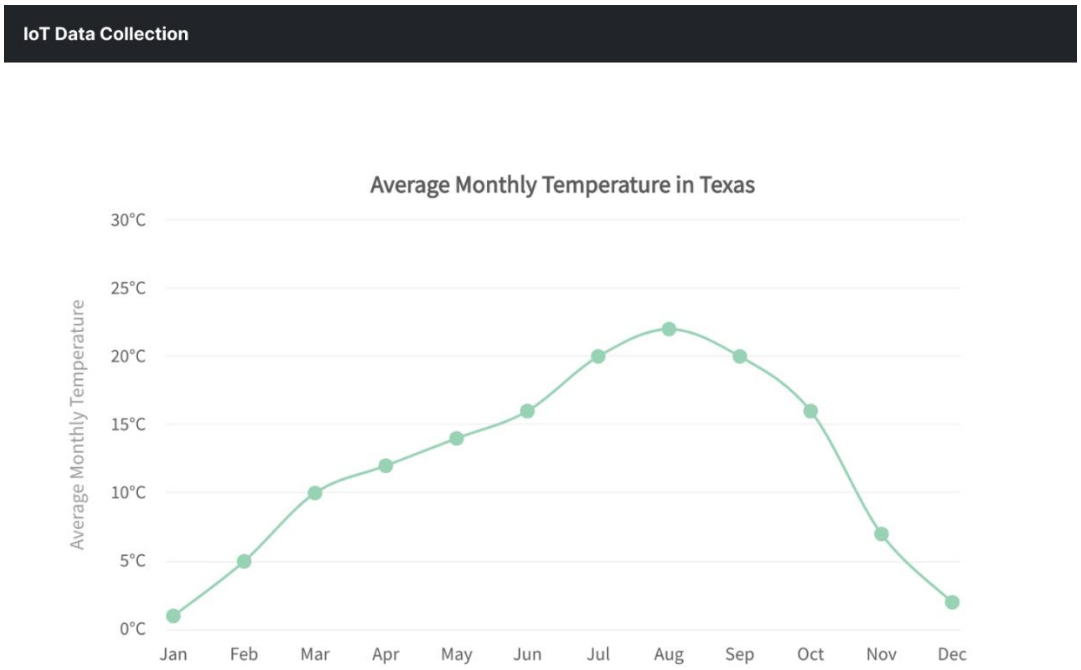
c) Giao diện Monitor

Giao diện Monitor được thiết kế để hiển thị nhiệt độ thu thập được và thời gian thu thập. Ngoài ra, người dùng cũng có thể tìm kiếm dữ liệu thông qua ngày tháng.



Hình 8: Thiết kế giao diện monitor

d) Giao diện biểu đồ

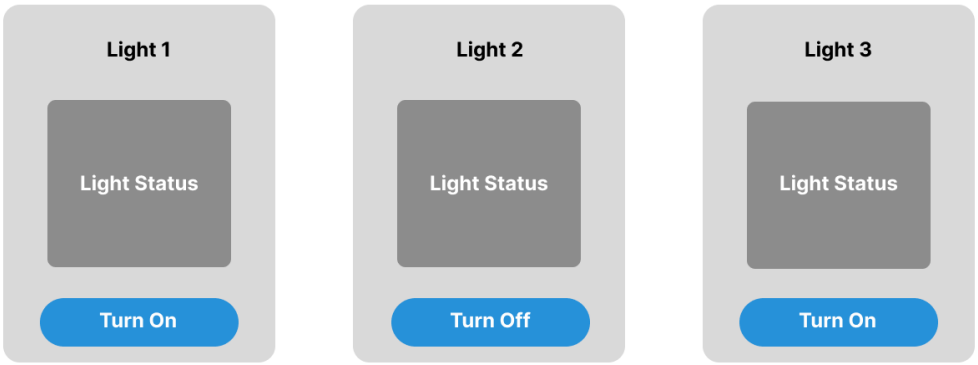


Hình: Thiết kế giao diện biểu đồ đường

Giao diện biểu đồ được thiết kế để thể hiện thông tin nhiệt độ một cách trực quan hơn. Giao diện được thiết kế đơn giản với trục tung là nhiệt độ và trục hoành là thời gian. Ngoài ra người dùng còn có thể lựa chọn số lượng điểm trên biểu đồ (tăng giảm lượng thời gian).

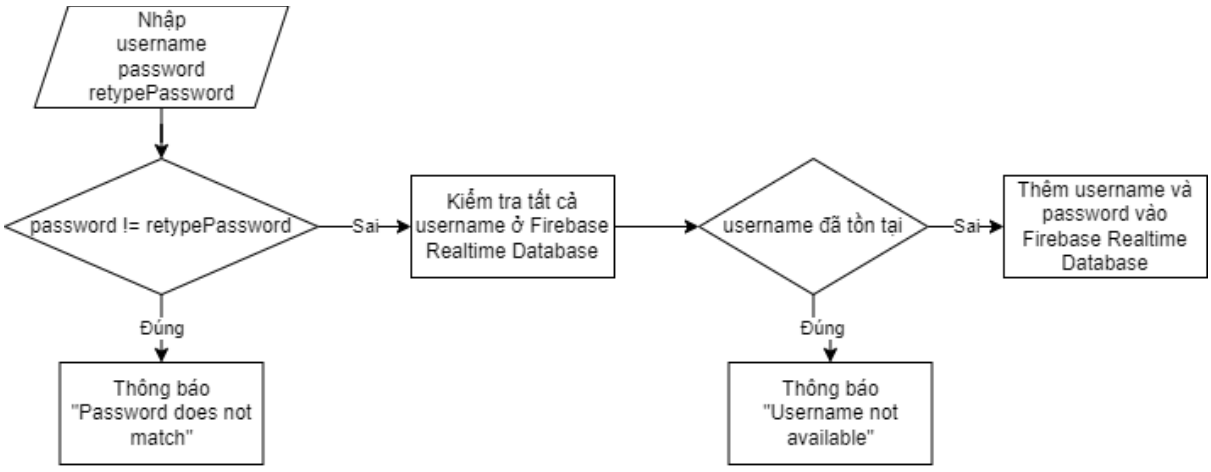
e) Giao diện điều khiển

Giao diện điều khiển được thiết kế phân vùng theo từng ô mỗi ô có 3 phần tử bao gồm tên đèn, trạng thái của đèn và nút nhấn để bật tắt đèn. Trạng thái đèn sẽ được hiển thị bằng hình ảnh trực quan thể hiện trạng thái đèn đang tắt hoặc mở để người dùng có thể dễ dàng hiểu rõ và điều khiển đèn chính xác.



Hình 9: Thiết kế giao diện điều khiển

2.2.3. Chức năng signup (đăng ký)



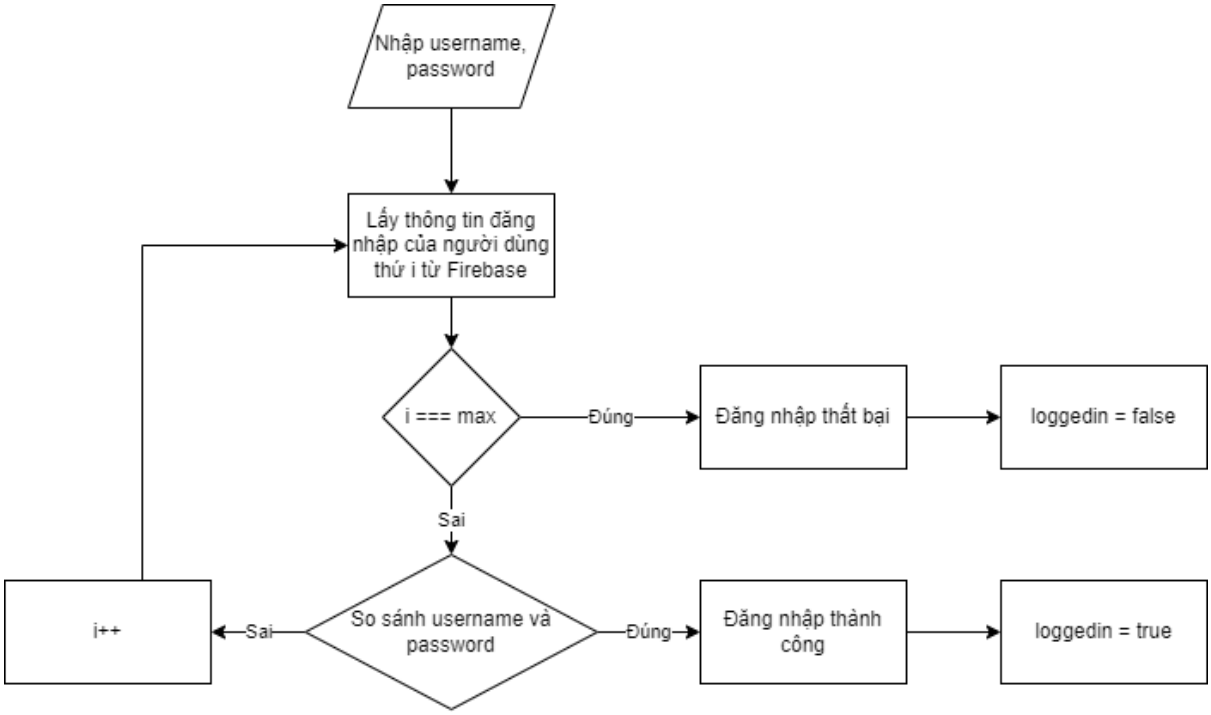
Hình 10: Lưu đồ giải thuật chức năng đăng ký

Đầu tiên người dùng sẽ nhập username, password và retypePassword với điều kiện là password và retypePassword phải trùng khớp với nhau. Sau đó, controller tại NodeJS sẽ kiểm tra tất cả các element được truy xuất từ bảng Users ở Firebase

Realtime Database. Nếu username người dùng nhập vào trùng với username ở cơ sở dữ liệu thì đăng ký không thành công. Nếu username người dùng nhập vào không trùng với bất kỳ username nào ở cơ sở dữ liệu thì thông tin username, password sẽ được tiến hành lưu vào cơ sở dữ liệu Firebase Realtime Database.

2.2.4. Chức năng login (đăng nhập)

Chức năng đăng nhập yêu cầu người dùng sử dụng username và password đã được đăng ký để đăng nhập và sử dụng các chức năng của trang web. Vì khi sử dụng bất kỳ chức năng nào thì website cũng sẽ có một biến "loggedin" để kiểm tra xem có đăng nhập hay chưa. Nếu người dùng chưa đăng nhập sẽ được trả về trang đăng nhập, nếu người dùng đã đăng nhập có thể tiếp tục sử dụng các chức năng của trang web.



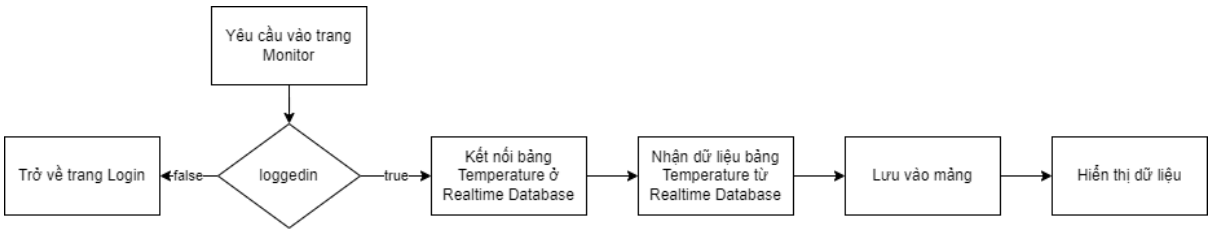
Hình 11: Lưu đồ giải thuật của chức năng đăng nhập

Người dùng sẽ nhập username và password, sau đó controller ở NodeJS sẽ kết nối với Firebase Realtime Database để lấy thông tin đăng nhập ở bảng Users. Firebase Realtime Database trả về một file json có chứa các element trong bảng Users. Kiểm tra username/password của mỗi element với username/password của

người dùng nhập vào. Nếu không trùng khớp, tăng element lên 1 để kiểm tra với element tiếp theo của bảng Users. Đến khi kết thúc bảng Users vẫn không trùng khớp thì đăng nhập thất bại. Nếu có element trùng khớp cả username và password thì đăng nhập thành công và đưa giá trị loggedin về true để có thể sử dụng các chức năng của website.

2.2.5. Chức năng monitor (giám sát)

Chức năng monitor giúp người dùng có thể giám sát được nhiệt độ thu thập dưới dạng bảng. Người dùng có thể sắp xếp dữ liệu theo nhiệt độ, thời gian tăng/giảm dần hoặc tìm kiếm thông tin nhiệt độ ở một ngày cụ thể.



Hình 12: Lưu đồ giải thuật chức năng monitor (giám sát)

Khi người dùng đã đăng nhập yêu cầu vào trang monitor, controller kết nối và nhận dữ liệu bảng Temperature từ Firebase Realtime Database sau đó lưu vào một mảng để thuận lợi cho việc sắp xếp theo nhiệt độ, thời gian tăng giảm dần theo yêu cầu của người dùng.

2.2.6. Chức năng chart (hiển thị thông tin dưới dạng biểu đồ)

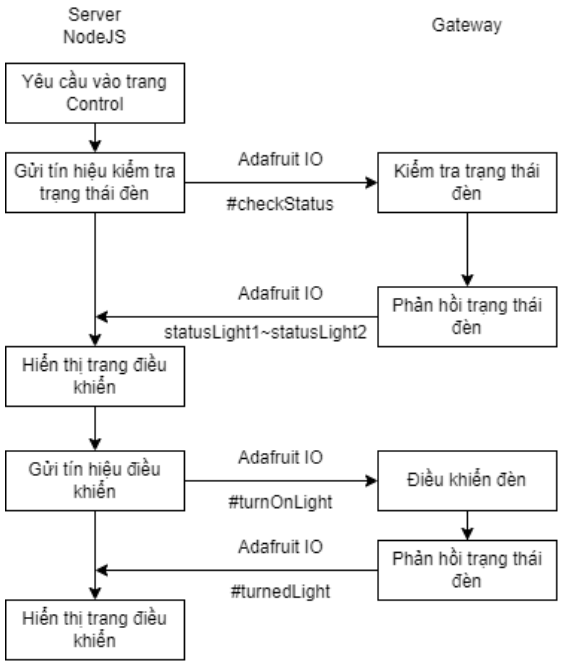
Chức năng chart được tạo dựa vào thư viện ChartJS. ChartJS là một framework của JavaScript dựa trên HTML5 để tạo ra các biểu đồ và đồ thị động, có tính tương tác cao và hỗ trợ responsive nhờ đó mà có thể tạo ra biểu độ cập nhật thời gian realtime.

Biểu đồ sẽ liên tục giao tiếp với Firebase Realtime Database để cập nhật dữ liệu. Khi Firebase Realtime Database có dữ liệu nhiệt độ mới được cập nhật thì chức năng chart sẽ ngay lập tức cập nhật và hiển thị lên biểu đồ.

2.2.7. Chức năng control (điều khiển)

Chức năng này được thực hiện để bật tắt đèn ở các node dựa vào giao thức MQTT và service gửi nhận dữ liệu của Adafruit IO. Các gói dữ liệu giao tiếp giữa web server và gateway sẽ được truyền nhận qua Adafruit IO.

Khi có yêu cầu vào trang điều khiển, controller sẽ gửi một tín hiệu để kiểm trang đèn (#checkStatus). Sau khi nhận được tín hiệu từ MQTT, gateway (ESP8266) sẽ kiểm tra trạng thái đèn từ các Node sau đó phản hồi trạng thái các đèn. Trạng thái các đèn sẽ được phản hồi dưới dạng một chuỗi được ngăn cách với nhau bằng ký tự đặc biệt (seperator) (statusLed1~statusLed2~statusLedn). Sau đó, khi server nhận được chuỗi này sẽ thực hiện việc tách các chuỗi và hiển thị trạng thái các đèn lên trang điều khiển.



Hình 13: Lưu đồ giải thuật chức năng điều khiển

Khi có yêu cầu điều khiển đèn từ người dùng, trang chủ sẽ gửi tín hiệu bật/tắt đèn (#turnOnLight hoặc #turnOffLight) dựa vào trạng thái hiện tại của đèn đến gateway thông qua MQTT. Nếu đèn đang bật thì sẽ gửi #turnOffLight để tắt và ngược lại nếu đèn đang tắt thì sẽ gửi #turnOnLight để bật. Sau khi gateway nhận được tín hiệu sẽ yêu cầu các Node thực hiện việc bật/tắt đèn sau đó phản hồi lại

đã bật/tắt đèn (#turnedOnLight hoặc #turnedOffLight) dựa vào yêu cầu của máy chủ cùng với trạng thái các đèn để máy chủ hiển thị.

2.3. Khối gateway và khối node cảm biến

2.3.1. Firebase Realtime Database

Realtime Database là một cơ sở dữ liệu NoSQL có chức năng lưu và đồng bộ dữ liệu trên mây dưới dạng JSON.

Kết nối Firebase với ESP:

- Bước 1: Khai báo biến thông tin cho Wifi và Firebase.
- Bước 2: Kết nối wifi cho ESP8266. (Hình 3)
- Bước 3: Khởi tạo thông tin xác thực của Firebase. (Hình 4)
- Bước 4: Gửi thông tin lên Firebase với hàm setFloat, setInt, setString,… và nhận thông tin từ firebase với hàm getFloat, getInt, getString,… (Hình 5)

```
WiFi.begin (WIFI_SSID, WIFI_PASSWORD);
Serial.print("Dang ket noi");
while (WiFi.status() != WL_CONNECTED) {
  Serial.print(".");
  delay(500);
}

Serial.println ("");
Serial.println ("Da ket noi WiFi!");
Serial.println(WiFi.localIP());
```

Hình 14: Kết nối wifi

```
Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
```

Hình 15: Khởi tạo thông tin firebase

```
void FireBase()
{
  srand (time(NULL));
  char ref[100] = "Temperature/";
  Firebase.setFloat ( fbdo,"Led 4", StatusLed_4);
  Firebase.setFloat ( fbdo,"Led 3", StatusLed_3);
  Firebase.setFloat ( fbdo,"Led 2", StatusLed_2);
  Firebase.setFloat ( fbdo,"Led 1", StatusLed_1);
  strcat(ref, dateTime.c_str());
  Firebase.setFloat( fbdo,ref, random(0,230));
```

Hình 16: Gửi nhận thông tin firebase

2.3.2. MQTT và Adfruit IO

Kết nối Adfruit IO với ESP8266:

- Bước 1: Khai báo biến thông tin cho Wifi và hai biến publishing, subscribing cho Adfruit IO.
- Bước 2: Kết nối wifi cho ESP8266.
- Bước 3: Thiết lập MQTT subscription
- Bước 4: Kết nối MQTT thông qua hàm MQTT_connect();
- Bước 5: Nhận dữ liệu từ MQTT thông qua hàm readSubscription, gửi dữ liệu đến MQTT thông qua hàm publish.

```
// Setup a feed called 'photocell' for publishing.
// Notice MQTT paths for AIO follow the form: <username>/feeds/<feedname>
Adafruit_MQTT_Publish pub = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/mqtt-server");

// Setup a feed called 'onoff' for subscribing to changes.
Adafruit_MQTT_Subscribe sub = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/mqtt-server");
```

Hình 17: Khai báo biến publishing, subscribing cho Adfruit IO

```
// Setup MQTT subscription for onoff feed.
mqtt.subscribe(&sub);
```

Hình 18: Thiết lập MQTT subscription

```
Adafruit_MQTT_Subscribe *subscription;
while ((subscription = mqtt.readSubscription(100))) {
  if (subscription == &sub) {
    Serial.print(F("Got: "));
    Serial.println((char *)sub.lastread);
    if (strcmp((char *)sub.lastread,"#turnOn") == 0) {
      Serial.println("Turned ON");
      pub.publish("#turnedOn");
      hasTurnedOnLight1 = true;
    }
```

Hình 19: nhận dữ liệu MQTT

2.3.3. LoRa

LoRa là viết tắt của long-range là một công nghệ điều chế RF cho mạng diện rộng công suất thấp (LPWAN) có khả năng truyền dữ liệu lên đến 5km ở khu vực đô thị và 10-15km ở khu vực nông thôn. Đặc điểm của công nghệ Lora là yêu cầu điện năng cực thấp.

Module Lora có 2 chế độ truyền nhận dữ liệu:

- Transparent transmission mode: chế độ này có thể gửi gói dữ liệu tới tất cả thiết bị có cùng địa chỉ và kênh được cấu hình.
- Fixed transmission: Chế độ này có thể gửi gói dữ liệu đến địa chỉ và kênh được xác định. Chế độ này có 2 dạng truyền nhận dữ liệu là:
    - Đến thiết bị có kênh, địa chỉ được chỉ định.
    - Đến các thiết bị có cùng kênh.

Module LoRa E32 có các chân sau:

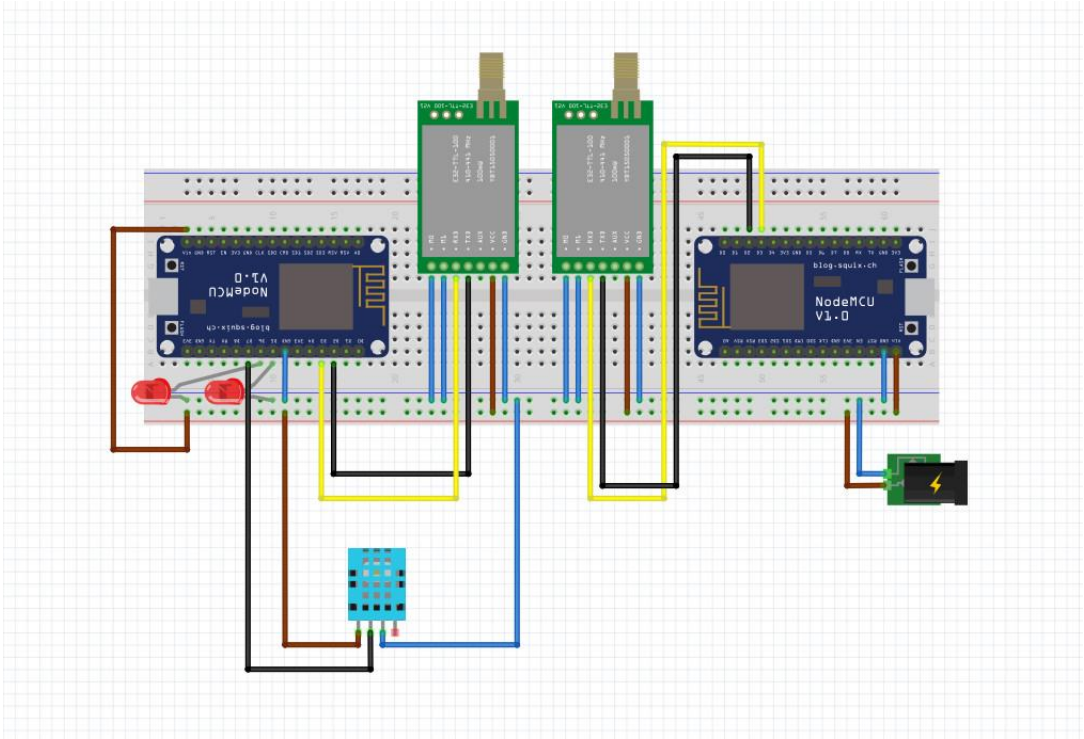
Pin No.	Pin item	Pin direction	Pin application
1	M0	Input (weak pull-up)	Làm việc với M1 & quyết định bốn chế độ hoạt động
2	M1	Input (weak pull-up)	Làm việc với M0 & quyết định bốn chế độ hoạt động
3	RXD	Input	Đầu vào TTL UART, kết nối với đầu ra TXD bên ngoài (MCU, PC).
4	TXD	Output	Đầu ra TTL UART, kết nối với RXD bên ngoài (MCU, PC) InputPin.
5	AUX	Output	Để chỉ ra trạng thái làm việc của mô - đun
6	VCC	Power supply 2.3V~5.5V DC	
7	GND	Ground	

Bảng 1: Các chân module LoRa E32

Mode	M1	M0	Explanation
Normal	0	0	UART và wireless đều mở
Wake-Up	0	1	Tương tự chế độ Normal nhưng có mã mở đầu được thêm vào dữ liệu truyền để đánh thức receiver.
Power-Saving	1	0	UART được vô hiệu hóa và không dây ở chế độ WOR (Wake On Radio) thiết bị sẽ bật khi có dữ liệu được nhận. Truyền bị vô hiệu hóa
Sleep	1	1	Truyền nhận bị vô hiệu hóa

Bảng 2: Các chế độ sử dụng module LoRa E32

Sơ đồ kết nối:



Hình 20: Sơ đồ kết nối

Node gateway: có chức năng nhận dữ liệu từ nodejs và điều khiển, giám sát node cảm biến thông qua module truyền thông Lora.

Node sensor: có chức năng thu thập dữ liệu từ cảm biến và điều khiển các trạng thái của 2 led đơn và truyền dữ liệu sang moudule truyền thông lora.

Module lora được cấu hình:

+ Địa chỉ và kênh được cấu hình:

- Gateway: địa chỉ 3 kênh 8.
- Node: địa chỉ 1 kênh 4.

+ Hoạt động ở chế độ Uart để giao tiếp với bộ vi xử lý(Esp8266) .

+ Air data rate（bps）：9. 6k

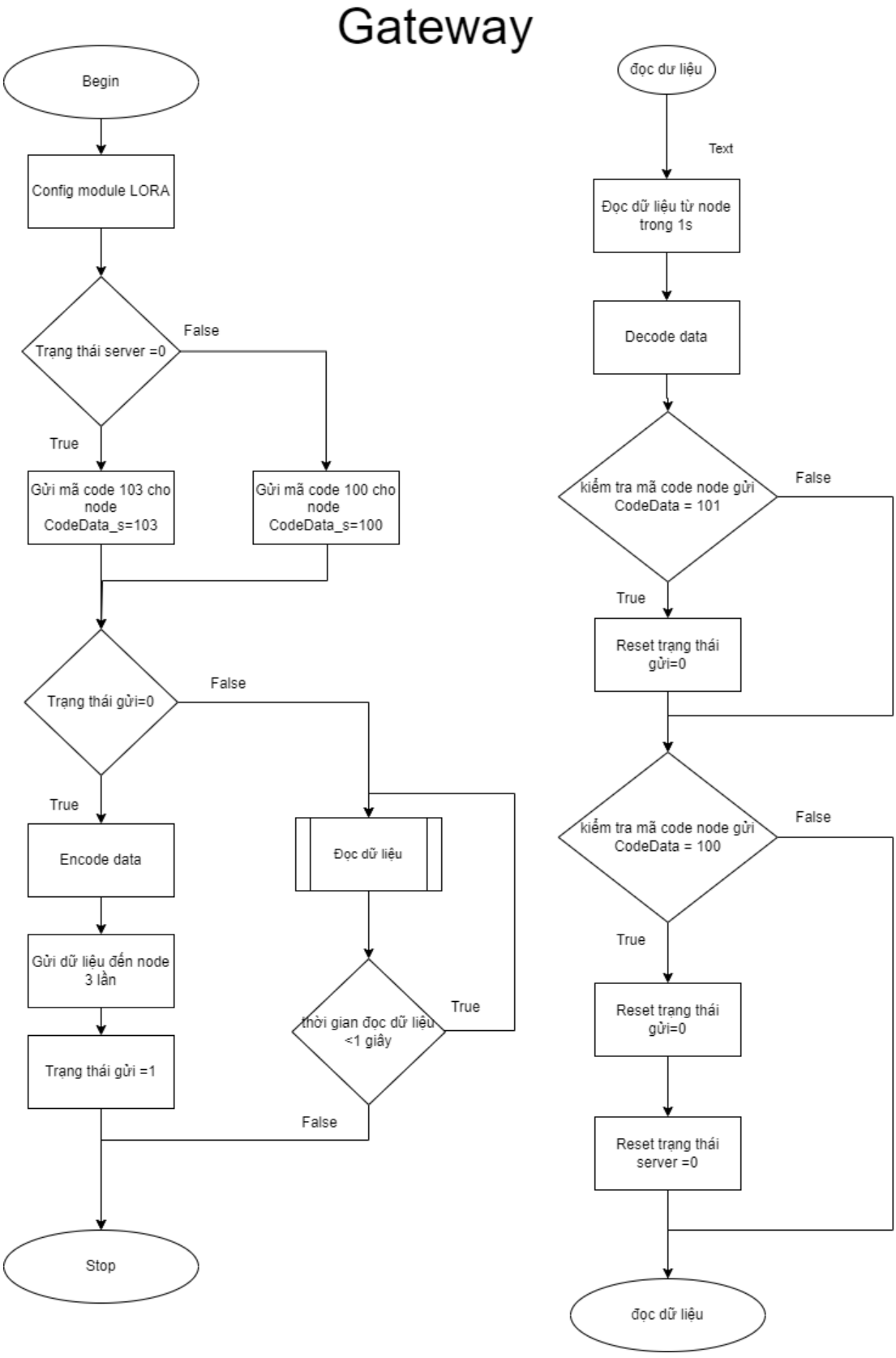
+ Transmission power: 20dbm

+ Mode hoạt động: FixedTranmission

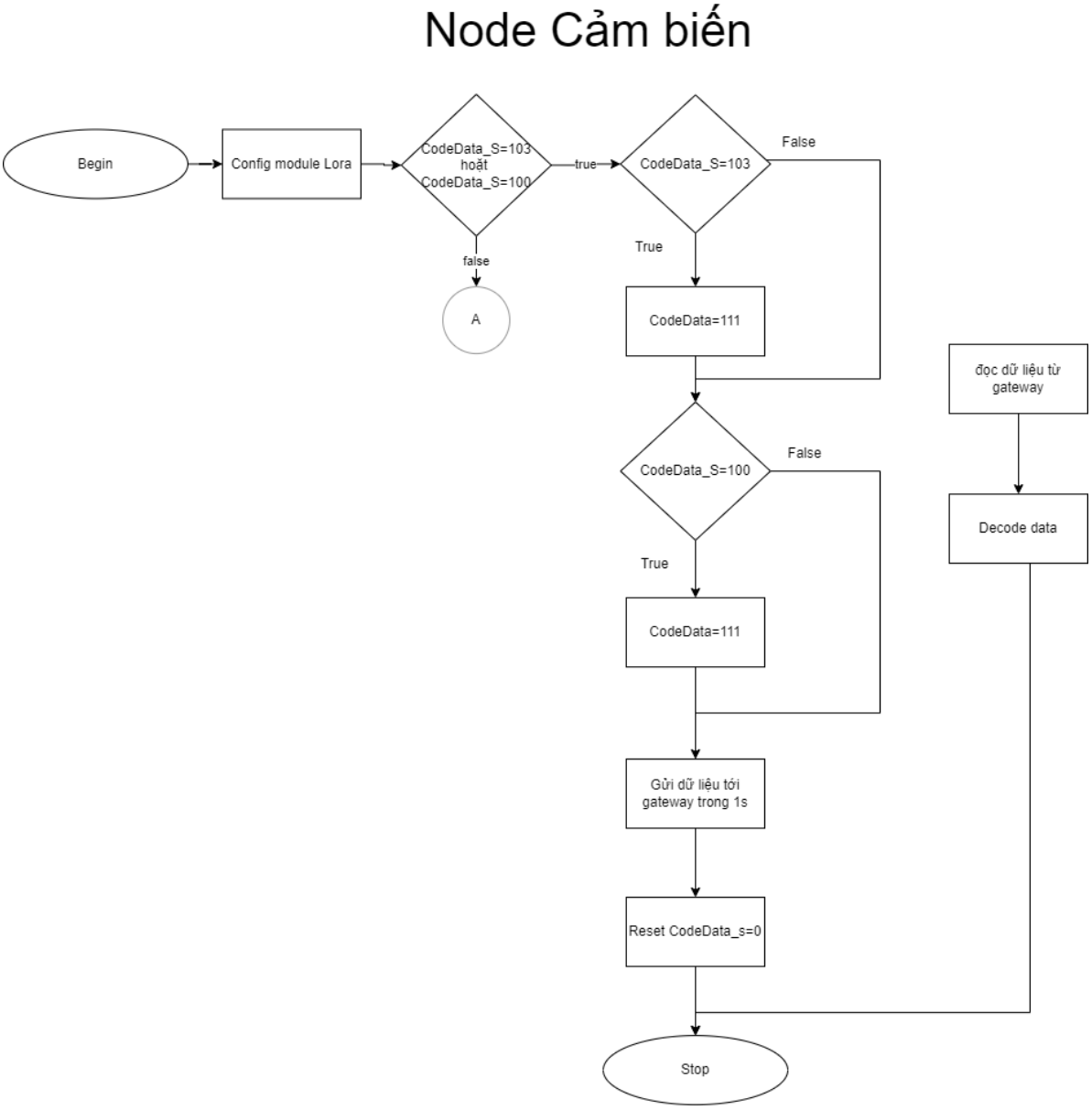
```
configuration.ADDL = 3;
configuration.ADDH = 0;
configuration.CHAN = 8;
configuration.SPED.uartBaudRate = UART_BPS_9600;
configuration.SPED.uartParity = MODE_00_8N1; // Parity bit
configuration.SPED.airDataRate = AIR_DATA_RATE_100_96 ;// Air data rate
configuration.OPTION.transmissionPower = POWER_20;
configuration.OPTION.fec = FEC_0_OFF;
configuration.OPTION.fixedTransmission = FT_FIXED_TRANSMISSION;
configuration.OPTION.wirelessWakeupTime = WAKE_UP_750;
configuration.OPTION.ioDriveMode = IO_D_MODE_PUSH_PULLS_PULL_UPS;
e32ttl.setConfiguration(configuration, WRITE_CFG_PWR_DWN_SAVE);
```

Hình 21: Cấu hình gateway

Lưu đồ giải thuật:



Hình 22: Lưu đồ giải thuật khối gateway



Hình 23: Lưu đồ giải thuật khối node cảm biến

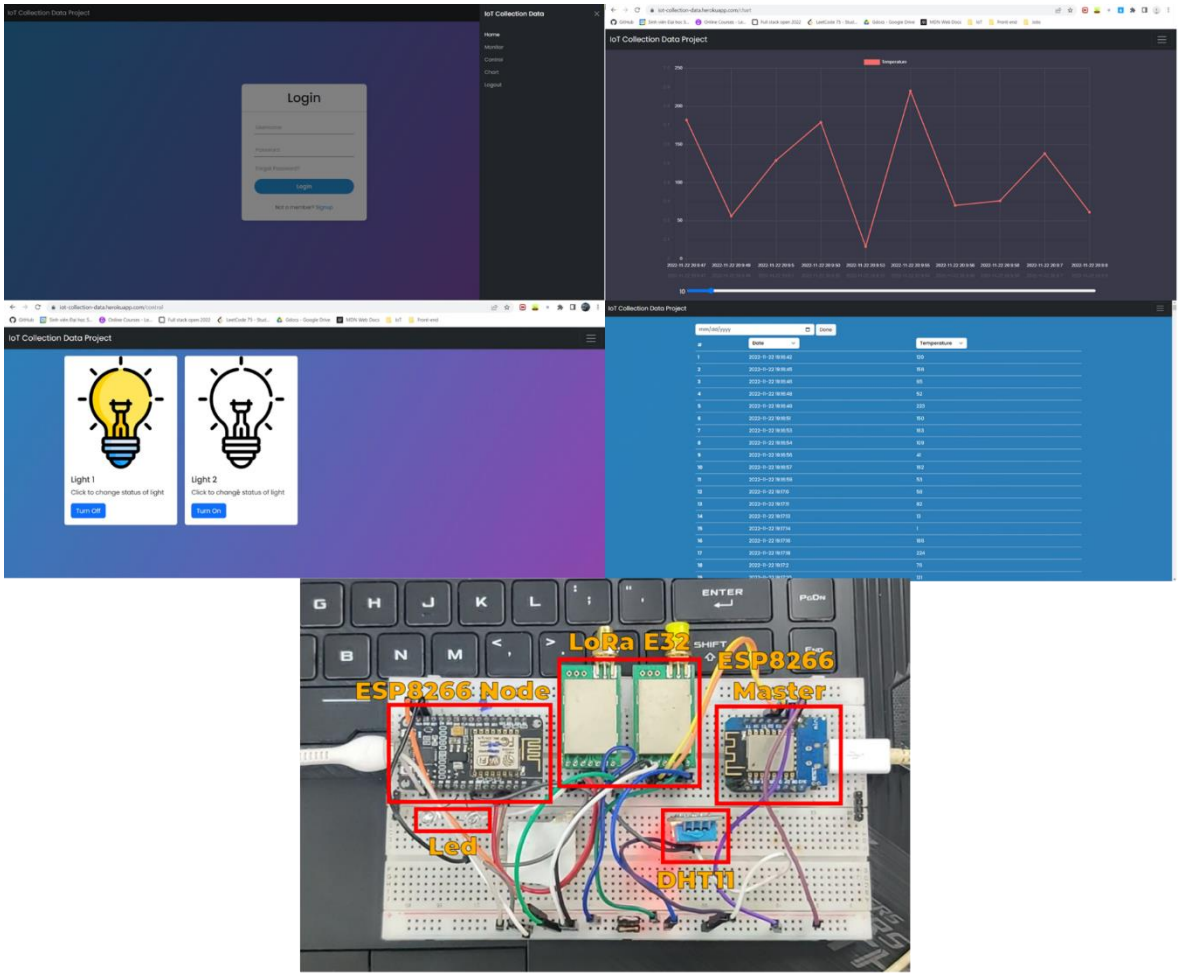
Nguyên lý hoạt động:

Gateway	Node
- Gửi lệnh đọc dữ liệu đến Node (CodeData_s=100) nếu không có tín hiệu đang nhận dữ liệu từ nodejs. Nếu đang có tín hiệu đang nhận dữ liệu từ nodejs thì Gateway sẽ gửi	- Luôn đọc dữ liệu từ Gateway và kiểm tra có lệnh gửi dữ liệu hay ghi dữ liệu xuống node không. - Khi nhận được lệnh đọc dữ liệu từ Gateway thì tiến hành giải mã dữ liệu.

code 103.Trạng thái gửi dữ liệu sẽ được gửi trong 1s - Sau khi kết thúc lệnh gửi dữ liệu thì tiến hành nhận dữ liệu từ Node. Nếu trong 1s không nhận được dữ liệu từ node sẽ tiến hành gửi dữ liệu cho node.	- Nếu nhận được lệnh 100 (CodeData_s=100) tức lệnh đọc dữ liệu từ Gateway. - Node sẽ tiến hành gửi dữ liệu cảm biến và trạng thái led cho Gateway - Nếu nhận được lênh 103 (CodeData_s = 103) tức là lệnh ghi dữ liệu từ Gateway và tiến hành giải mã dữ liệu và lưu dữ liệu đó. Sau đó gửi lại dữ liệu hiện tại cho Gateway kèm lệnh 111. - Ở trạng thái ghi dữ liệu node sẽ ghi dữ liệu trong 1s. Nếu quá 1s thì tiến hành đọc dữ liệu lại từ Gateway.

3. Kết quả

3.1. Tổng quan



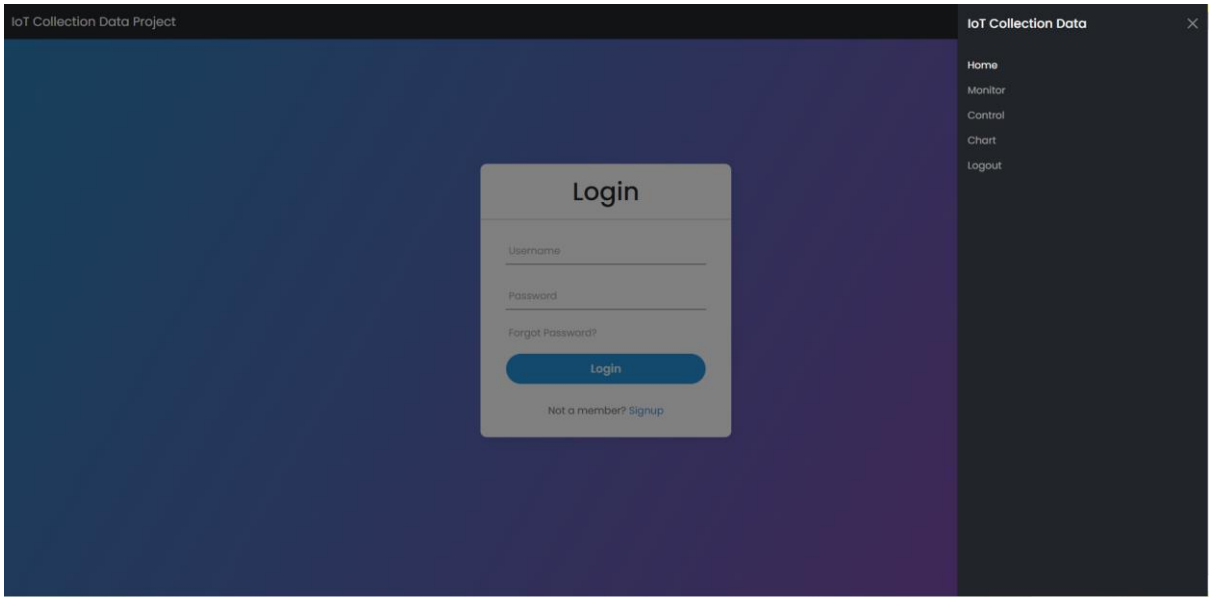
Hình 24: Kết quả phần cứng và phần mềm

Hệ thống hoàn thiện bao gồm một bộ phần cứng bao gồm 2 ESP8266, 2 Module LoRa E32, 2 Led, 1 cảm biến nhiệt độ DHT11 được kết nối với nhau thông qua dây bus và cắm trên test board. Cùng với đó là một web server với 4 trang giao diện chính là Đăng nhập, Biểu đồ, Giám sát và Điều khiển để lưu trữ và hiển thị thông tin cũng như điều khiển hệ thống

3.2. Kết quả phần mềm

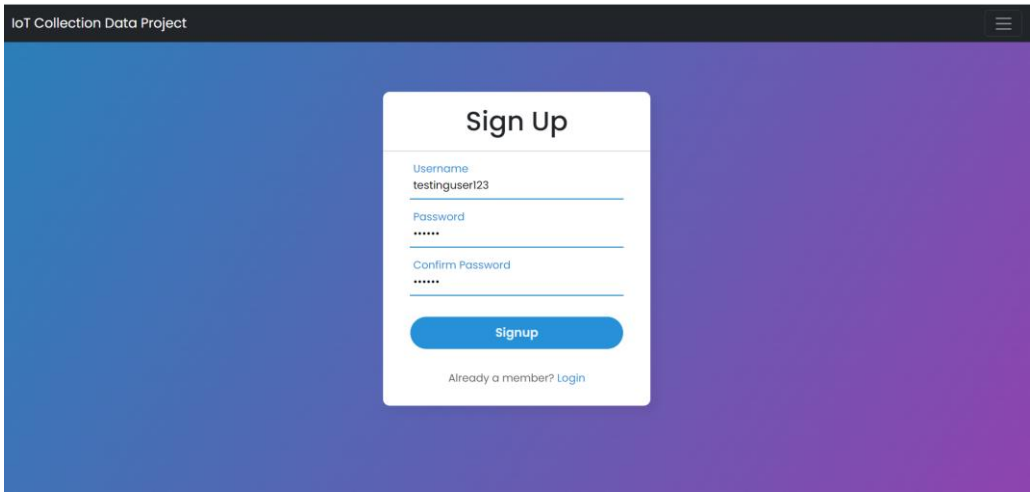
3.2.1. Giao diện chính của website

Website có 4 tính năng chính: Đăng nhập/Đăng ký/Đăng xuất, Giám sát, Điều khiển, Biểu đồ.



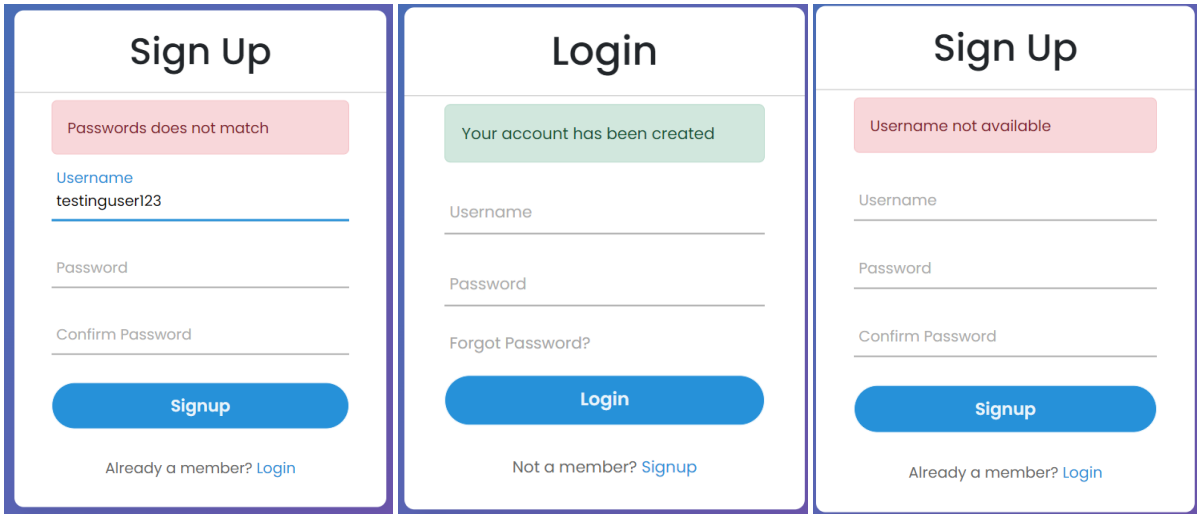
Hình 25: Giao diện chính của website

3.2.2. Chức năng đăng ký



Hình 26: Giao diện chức năng đăng ký

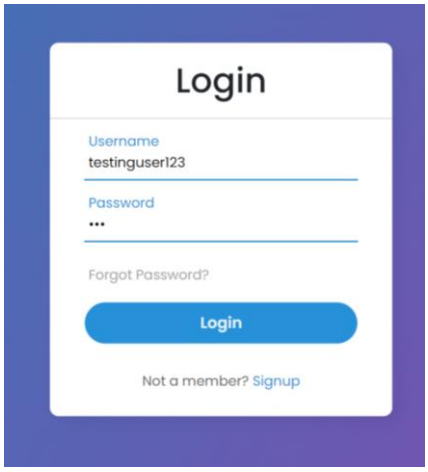
Khi đăng ký, người dùng phải nhập username, password và confirm password. Sau đó nhấn Signup. Nếu đăng ký thành công, hệ thống sẽ thêm dữ liệu và Firebase Realtime Database và người dùng sẽ được thông báo đăng ký thành công. Nếu đăng ký không thành công do username đã tồn tại, người dùng sẽ được thông báo lỗi và đăng ký lại. Nếu password và confirm password không trùng khớp người dùng cũng sẽ được thông báo lỗi và đăng ký lại. Hình bên dưới lần lượt là các trường hợp lỗi do không trùng khớp password – đăng ký thành công – lỗi do username đã tồn tại.



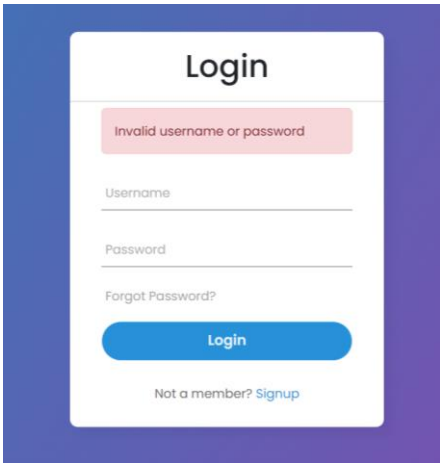
Hình 27: Giao diện đăng ký không trùng khớp – thành công – trùng username

3.2.3. Chức năng đăng nhập

Khi đăng nhập, người dùng sẽ phải nhập username và password. Nếu đăng nhập thành công, người dùng sẽ được chuyển hướng đến trang chủ. Nếu sai tên đăng nhập hoặc mật khẩu người dùng sẽ được thông báo và đăng nhập lại.



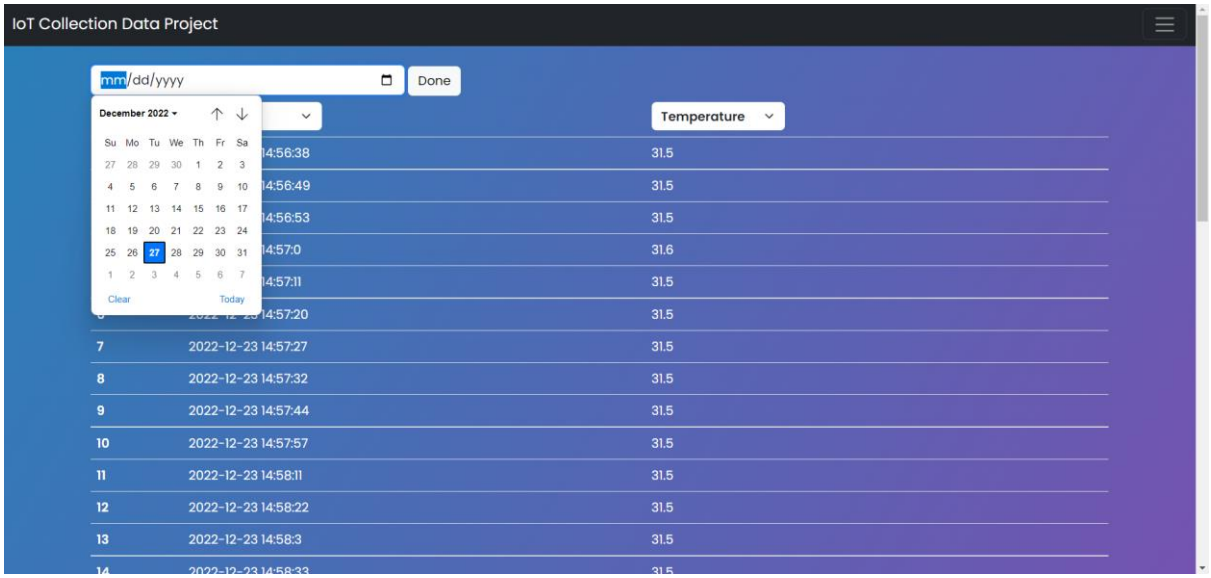
Hình: Giao diện đăng nhập



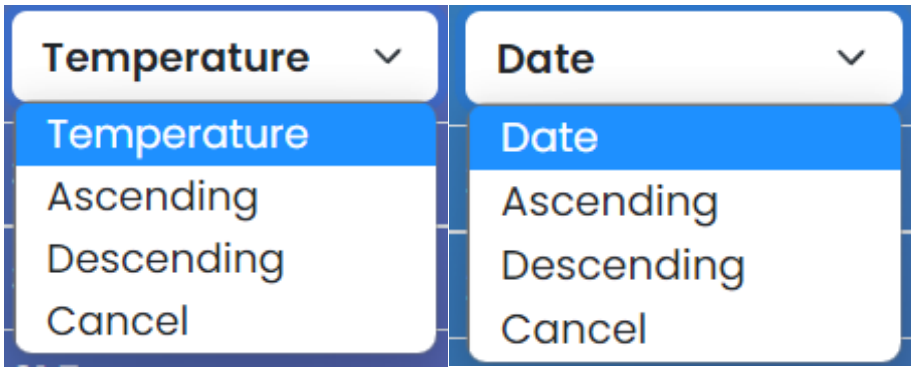
Hình: Giao diện đăng nhập thất bại

3.2.4. Chức năng monitor

Ở chức năng monitor, người dùng có thể quan sát nhiệt độ theo dạng bảng. Dữ liệu sẽ được lấy từ Firebase Realtime Database và hiển thị. Người dùng có thể lựa chọn một ngày xác định để xem giá trị nhiệt độ của ngày đó. Ngoài ra, còn có thể lựa chọn sắp xếp tăng giảm dần theo thời gian hoặc nhiệt độ.



Hình 28: Giao diện chức năng monitor



Hình 29: Tăng giảm dần theo nhiệt độ hoặc thời gian

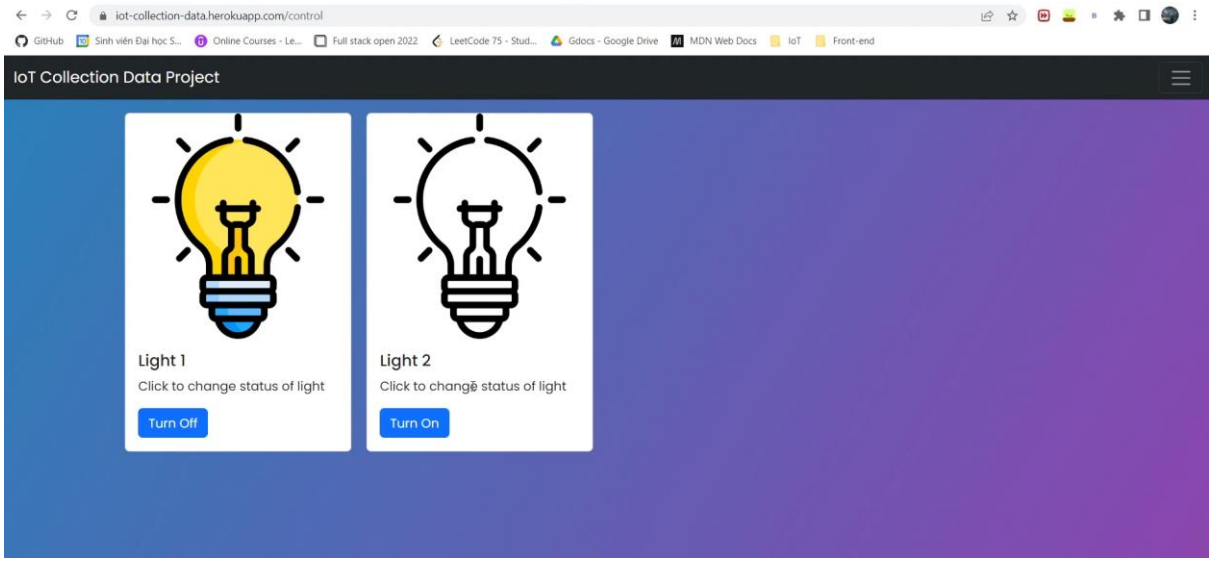
3.2.5. Chức năng chart

Hình 30: Chức năng chart

Ở trang này, người dùng có thể xem thông tin nhiệt độ dưới dạng biểu đồ đường. Biểu đồ này sẽ cập nhật thời gian thực từ Firebase Realtime Database khi có nhiệt độ mới cập nhật. Với trục tung là nhiệt độ và trục hoành là thời gian cùng thanh kéo bên dưới để lựa chọn số lượng điểm hiển thị mong muốn.

3.2.6. Chức năng control (điều khiển)

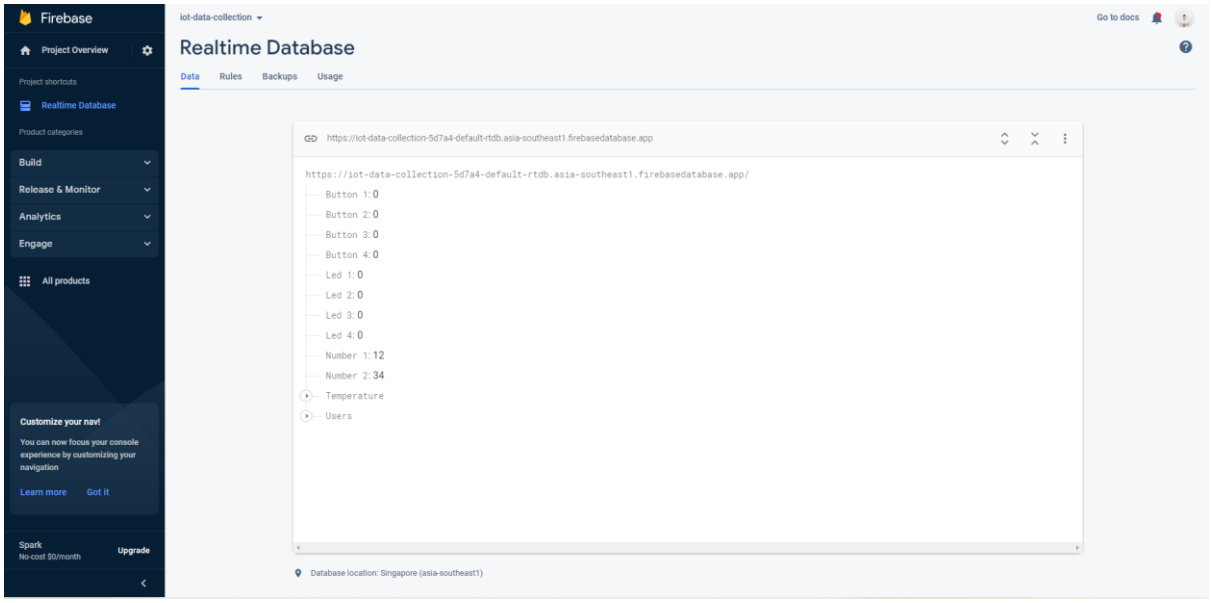
Người dùng có thể khiển bật tắt đèn từ web thông qua Adafruit IO MQTT đến hệ thống với các message đã được thiết lập sẵn. Trên website sẽ hiển thị số lượng đèn, tên đèn, trạng thái đèn thông qua hình ảnh (hình ảnh bên dưới có Light 1 đang bật và Light 2 đang tắt) và nút nhấn để gửi tín hiệu bật tắt đèn. Đèn được bật tắt với độ trễ khoảng 1-2s do server và gateway phải giao tiếp với nhau thông qua việc nhận gửi message.



Hình 31: Chức năng bật/tắt đèn trên website

3.2.7. Cơ sở dữ liệu Firebase Realtime Database

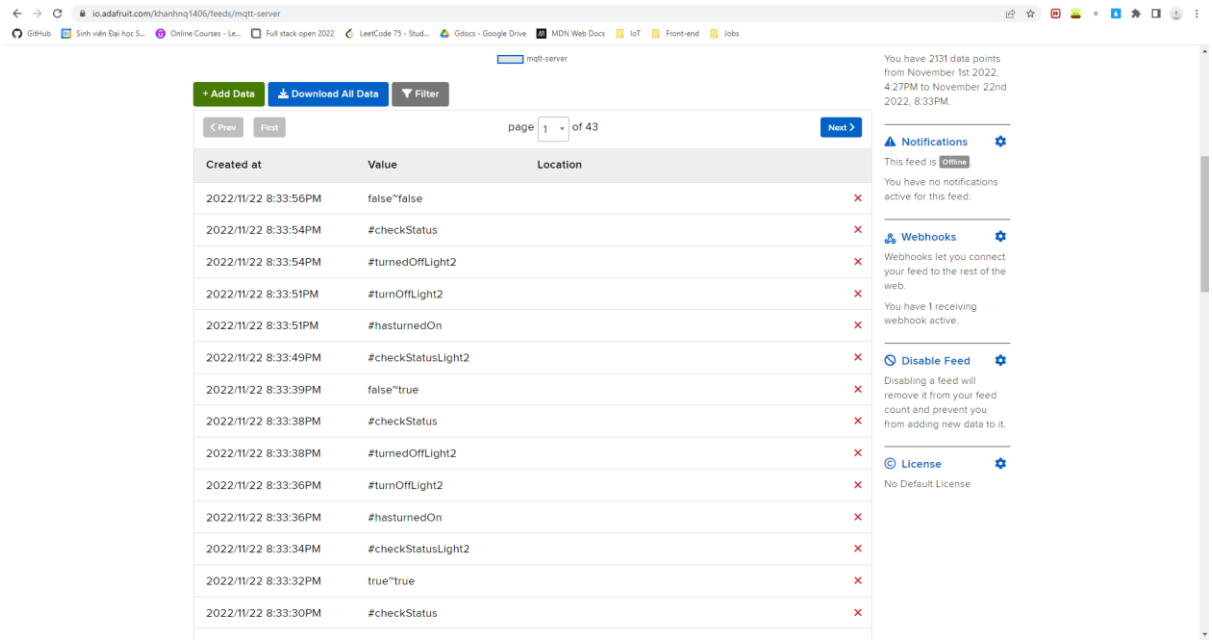
Firebase Realtime Database được sử dụng để lưu trữ trạng thái nút nhấn (Button), đèn (Led), nhiệt độ (Temperature) và thông tin người dùng (Users).



Hình 32: Bảng dữ liệu Realtime Database

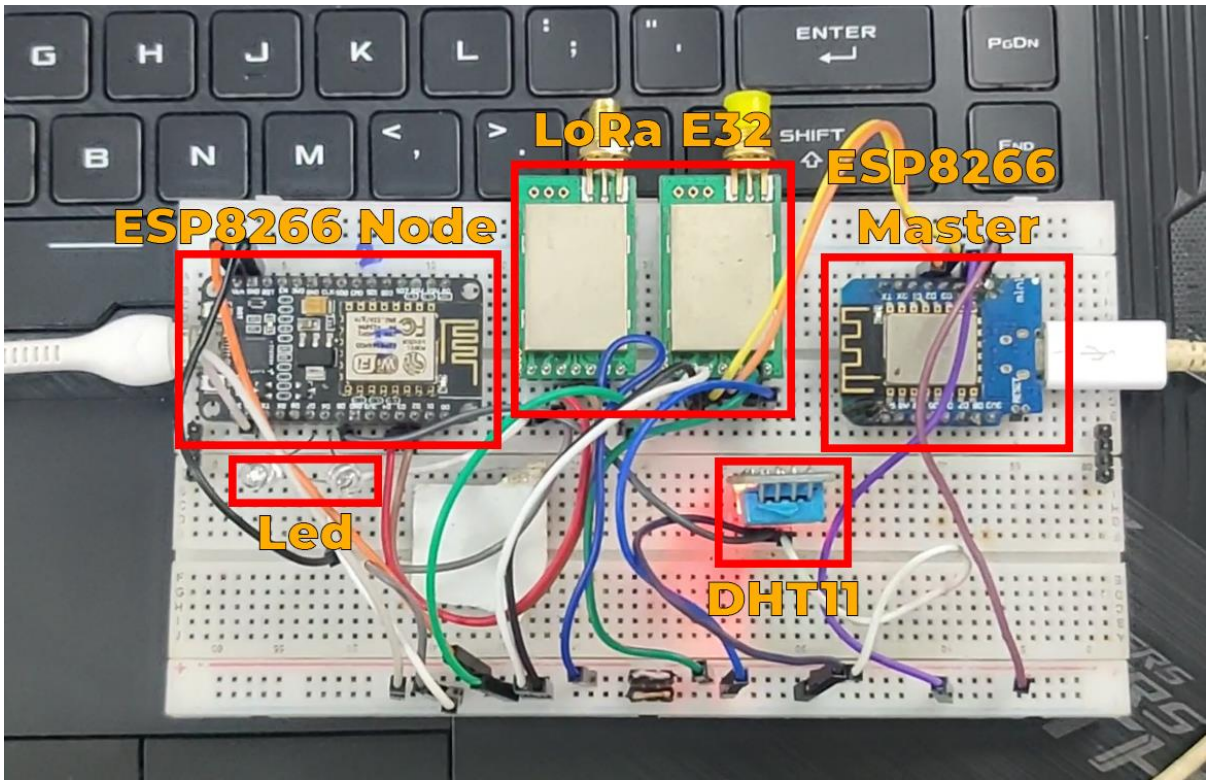
3.2.8. Adafruit MQTT

Trang web adafruit io dùng để giao tiếp, truyền tín hiệu giữa web server với esp8266 với các message như #checkStatus, #turnOnLight, #turnOffLight,…



Hình 33: Bảng seed MQTT

3.3. Kết quả phần cứng



Hình 34: Hệ thống thực tế

Mạch thực tế hệ thống bao gồm 2 ESP8266, 2 Module Lora E32, 1 cảm biến nhiệt độ DHT11, 2 bóng đèn led:

- ESP8266 Master được kết nối wifi và module LoRa để thực hiện chức năng gateway giữa web server và Node cảm biến.
- ESP8266 Node được kết nối với 2 bóng đèn led để điều khiển bật tắt khi có yêu cầu, kết nối với cảm biến DHT11 để thu thập nhiệt độ hiện tại và kết nối với module LoRa E32 để truyền/nhận dữ liệu đến gateway.

4. Kết luận và hướng phát triển

4.1. Kết luận

- Hệ thống truyền nhận tín hiệu nhanh chóng, ổn định với khoảng cách xa nhờ vào truyền thông LoRa.
- Hệ thống vẫn còn nhiều hạn chế chưa được tối ưu như: thời gian truyền nhận dữ liệu giữa 2 module lora và mức độ ưu tiên của dữ liệu của Gateway.
- Hệ thống gửi dữ liệu lên server nhanh chóng và ổn định thông qua Firebase Realtime Database.
- Giao tiếp MQTT giữa hệ thống và server có độ trễ và đáp ứng không tốt.

4.2. Hướng phát triển

- Tối ưu web server, cơ sở dữ liệu để có thể mở rộng khi có lưu lượng truy cập cao.
- Tăng tốc độ truyền nhận và tính chính xác giữa các node thông qua LoRa.
- Tối ưu tính chính xác và nhanh chống khi gửi nhận dữ liệu giữa web và các node thông qua MQTT.
- Tăng số lượng cảm biến, cơ cấu chấp hành và các node để mở rộng quy mô.
- Thêm chuẩn truyền thông modbus cho Node để có thể giao tiếp được các thiết bị trong thực tế hơn.
- Thiết kế pcb để hệ thống hoạt động ổn định.

5. Tài liệu tham khảo

[1] T. Thư, "MQTT là gì? Vai trò của MQTT trong IoT," 9 November 2020. [Online]. Available: https://viblo.asia/p/mqtt-la-gi-vai-tro-cua-mqtt-trong-iot-V3m5WL3bKO7.

[2] Wikipedia, "ESP8266," [Online]. Available: https://vi.wikipedia.org/wiki/ESP8266.

[3] P. Patel, "What exactly is Node.js?," 18 April 2018. [Online]. Available: https://www.freecodecamp.org/news/what-exactly-is-node-js-ae36e97449f5.