**Xin chào các bạn!**

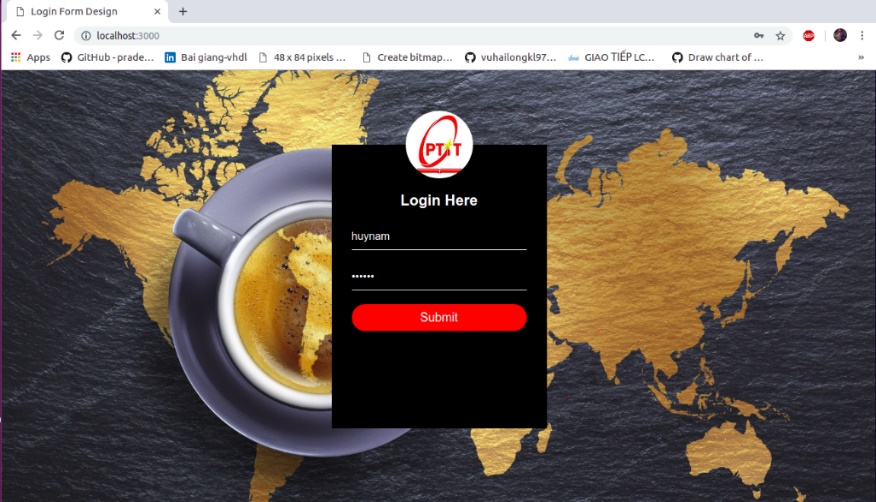
**Bài viết này sẽ hướng dẫn các bạn thiết kế giao diện web điều khiển các thiết bị đơn giản (led, speaker…) hiển thị giá trị đọc được từ các cảm biến và vẽ biểu đồ biểu diễn dữ liệu đó.**

# Mục tiêu:

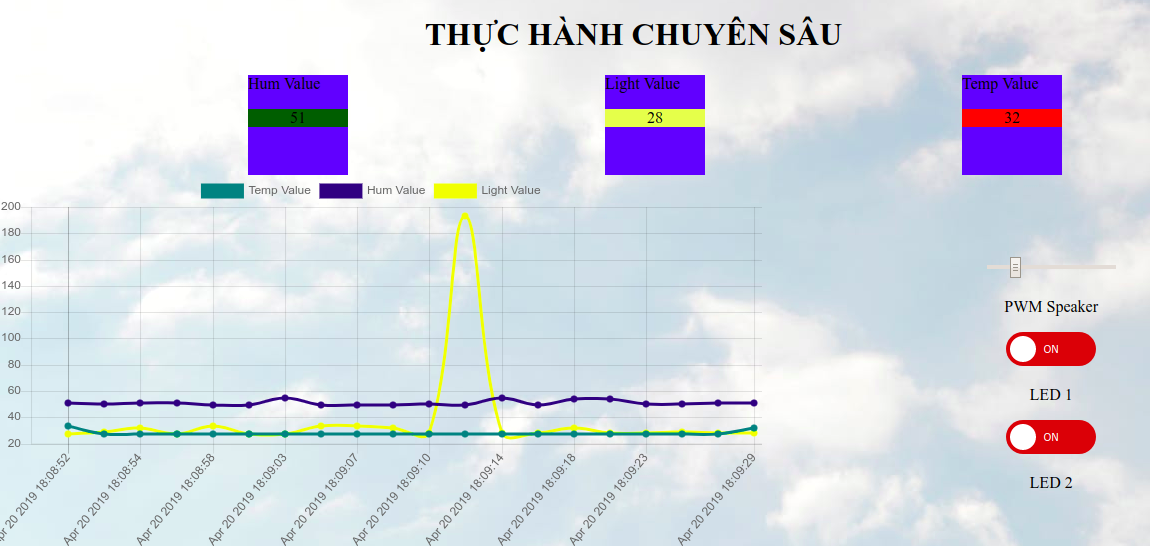
* Dùng ESP8266 để đọc các cảm biến như DHT11, Sensor Light
* Thiết kế giao diện web đơn giản để điều khiển và biểu diễn data của cảm biến
* Sử dụng Nodejs đưa dữ liệu vào database và đưa lên webserver

**Giao diện chính sau khi hoàn thành:**

**Login:**



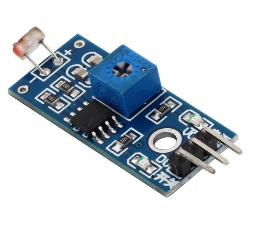
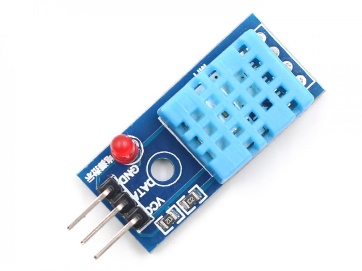
**Giao diện chính:**

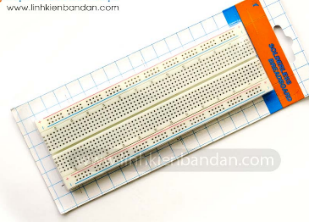


# Chuẩn bị linh kiện và thiết kế phần cứng:

## Yêu cầu phần cứng:

* 1 Node MCU ESP8266
* Dây Cáp Nạp Arduino (USB – Mini USB)
* 2 điện trở 330 Ohm
* 2 led đơn 5mm
* 10 jump cắm board test(đực-đực)
* 1 speaker
* 1 module DHT11
* 1 module sensor light( LDR)
* 1 board test trắng





# Nội dung:

**\* Chuẩn bị và install một số phần mềm cần sử dụng**

## 3.1 Download IDE Arduino

\* Trước hết chúng ta Download IDE Arduino

B1: Tải xuống các gói mới nhất, Linux 32 bit hoặc Linux 64 bit, từ liên kết chính thức bên dưới:

<https://www.arduino.cc/en/Main/Software>

B1: Open Terminal từ Ubuntu của bạn (phím tắt Ctrl + Alt + T)

$ cd ~/Downloads

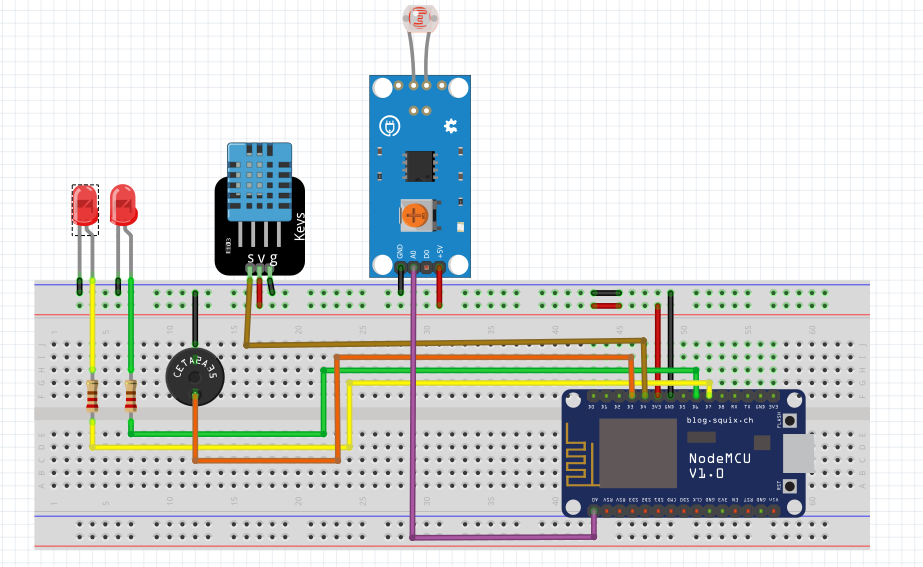
B2: Sau khi vào thư mục Download gõ tiếp câu lệnh để cài đặt

$ tar –xvf arduino-1.6.6-\*.tar.xz

## 3.2 Sơ đồ kết nối các chân của các cảm biến với NodeMCU ESP8266

DEVICE ESP8266

|  |  |  |
| --- | --- | --- |
| DHT11 | Data | D4 |
| Sensor Light | Analog | A0 |
| Speaker | Data | D3 |
| LED\_1 | Chân Led\_1 | D6 |
| LED\_2 | Chân Led\_2 | D7 |

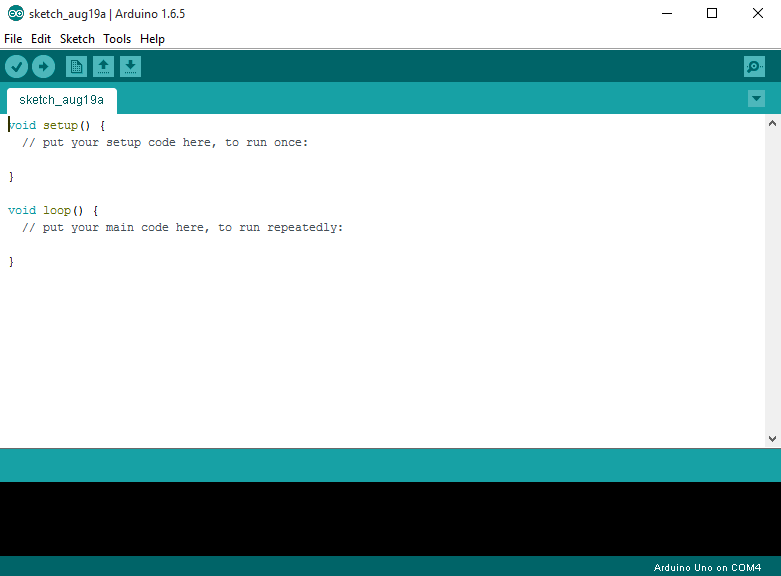


Sơ đồ kết nối được thiết kế trên phần mềm fritzing.

## 3.3 Đọc dữ liệu cảm biến và gửi data bằng giao thức MQTT

\* Sau khi đã cài xong IDE Arduino các bạn cần tải thư viện của ESP8266 và DHT11 để phục vụ cho nhu cầu của mình.

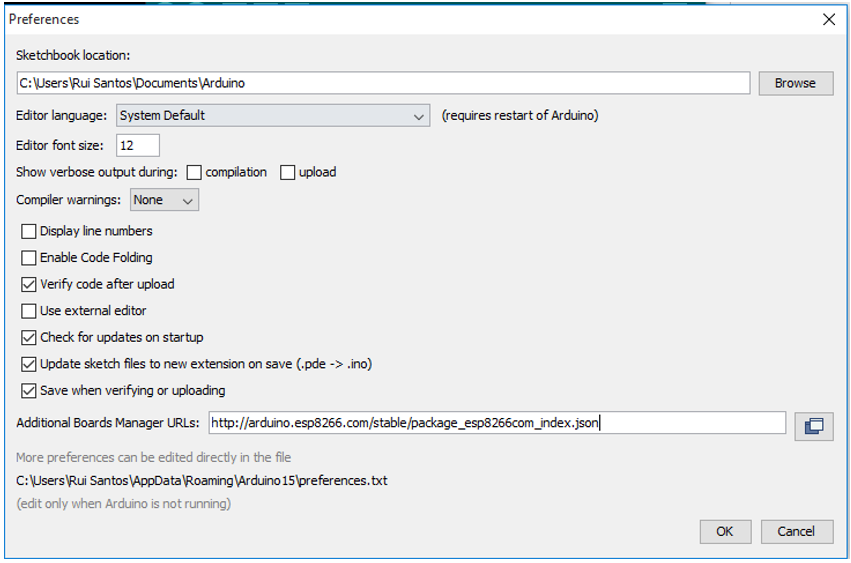
\* Dưới đây là giao diện sau khi chúng ta đã cài xong.



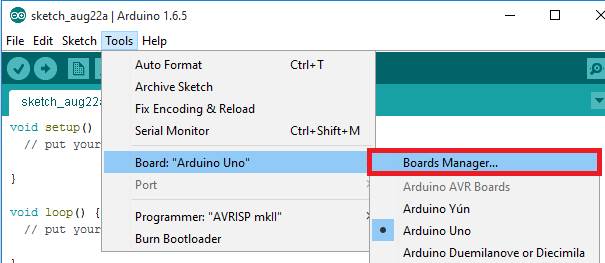
Để tiến hành cài đặt thư viện và chức năng nạp code cho IDE các bạn làm như sau:

Vào **File→ Preferences**, vào textbox Additional Board Manager URLs thêm đường link sau vào: <http://arduino.esp8266.com/stable/package_esp8266com_index.json>

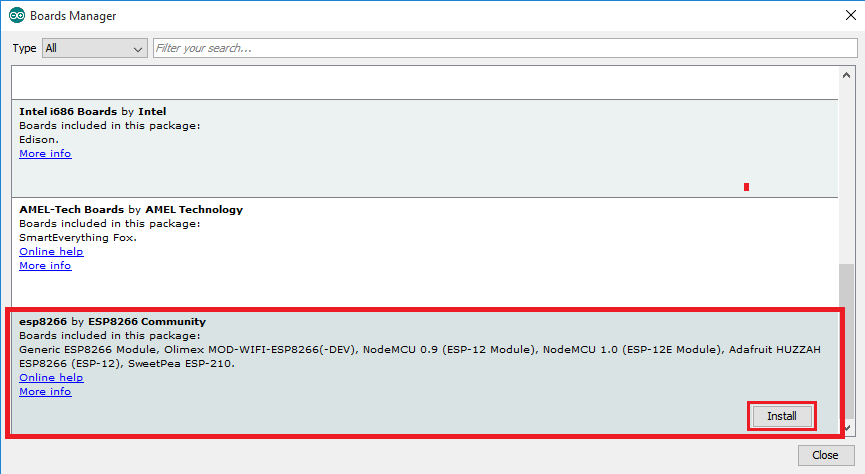
Click OK để chấp nhận.



Tiếp theo vào **Tool→Board→Boards Manager**



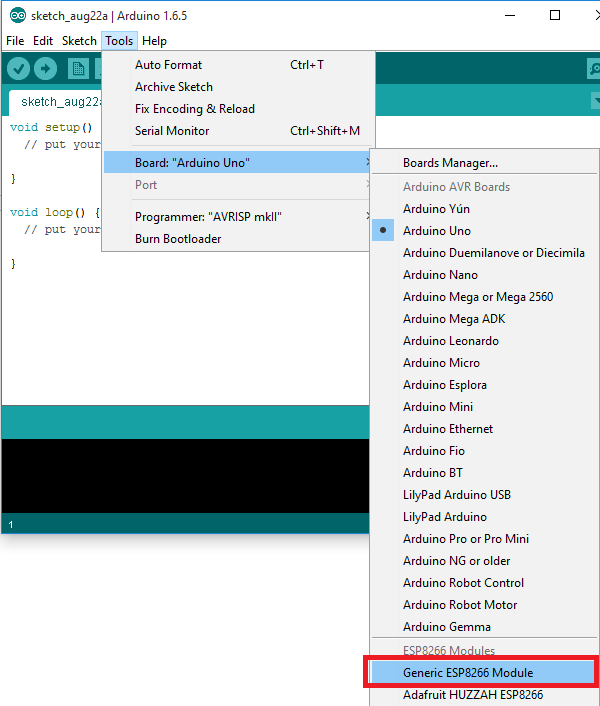
Đợi một lát để chương trình tìm kiếm. Ta kéo xuống và click vào **ESP8266 by ESP8266 Community**, click vào **Install**. Chờ phần mềm tự động download và cài đặt.



Chọn Board để lập trình cho ESP8266:

Kết nối mudule USB-to-UART vào máy tính. Vào **Tool→Board→Generic ESP8266 Modul**e, chọn cổng COM tương ứng với module USB-to-UART.

Board của chúng ta sẽ là Nodemcu 1.0 (esp-12e module) Kéo xuống và click để chọn board như vậy là chúng ta đã hoàn tất quá trình setup cho Esp8266

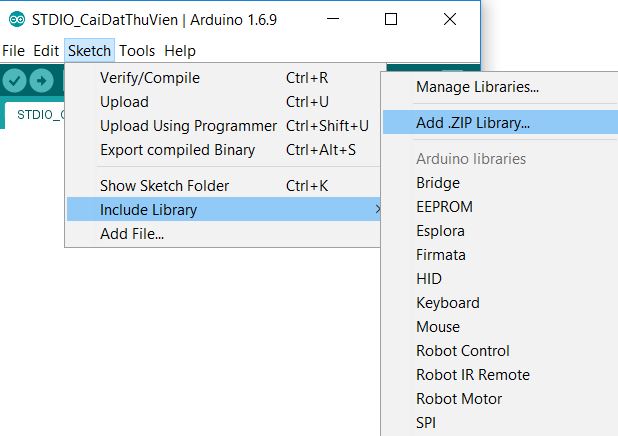


* Sau khi đã cài xong ESP8266 ta sẽ download và add thư viện DHT11 vào

Link: <https://www.arduinolibraries.info/libraries/dht-sensor-library>

Add thư viện sử dụng Add .Zip library

Sau khi tải thư viện về với file .zip, vào Sketch/ Include Library/Add sau đó chọn file Library DHT11 vừa tải về nhấn Ok là hoàn tất.



Hoàn tất quá trình cài đặt và add thư viện xong ta tiến hành lập trình.

**B1: Thiết lập cấu hình wifi và kết nối cảm biến với esp8266**

|  |
| --- |
| #include "DHT.h" // Khai báo thư viện DHT11  #include <ESP8266WiFi.h> // Sử dụng thư viện ESP8266  #include <PubSubClient.h>  #define DHTPIN 2 // PinOut của DHT11 tương ứng với D4 ESP8266  #define DHTTYPE DHT11  DHT dht(DHTPIN, DHTTYPE);  #define led1 D6 // Chân kết nối Led\_1 với D6 của ESP8266  #define led2 D7 // Chân kết nối Led\_2 với D7 của ESP8266  #define speaker 0 // Chân kết nối speaker với D3 của ESP8266  // Replace the next variables with your SSID/Password combination  const char\* ssid = "Happy Home"; // Name wifi kết nối  const char\* password = "dat251144"; // Password của wifi  const char\* mqtt\_server = "192.168.1.106"; // Địa chỉ ip mà laptop bạn kết nối đến wifi |

### B2: Tạo 1 hàm callback để điều khiển trong chương trình

|  |
| --- |
| void callback(char\* topic, byte\* message, unsigned int length) {  Serial.print("Message arrived on topic: ");  Serial.print(topic);  Serial.print(". Message: ");  String messageTemp;  for (int i = 0; i < length; i++) {  Serial.print((char)message[i]);  messageTemp += (char)message[i];  }  Serial.println();  if (String(topic) == "Topic 1") {  Serial.print("setup led1 ");  if (strcmp(messageTemp.c\_str(), "true") == 0)  digitalWrite(led1, 1);  else  digitalWrite(led1, 0);  Serial.println(messageTemp);  }  if (String(topic) == "Topic 2") {  Serial.print("setup led2 ");  if (strcmp(messageTemp.c\_str(),"true")==0)  digitalWrite(led2,1);  else  digitalWrite(led2,0);  Serial.println(messageTemp);  }  if (String(topic) == "Topic 3") {  Serial.print("setup led1 pwm ");  Serial.println(messageTemp.toInt());  analogWrite(speaker,messageTemp.toInt());  }  } |

**B3: Viết hàm reconnect() kiểm tra xem gói tin mà mình gửi đã thành công hay chưa**

|  |
| --- |
| void reconnect() {  // lặp cho đến khi được kết nối trở lại  while (!client.connected()) {  Serial.print("Attempting MQTT connection...");  // Attempt to connect  // publish gói tin "Connected!" đến topic ESP8266/connection/board  if (client.connect("ESP8266Client", "huynam", "huynam")) {  Serial.println("connected");  // Subscribe  // đăng kí nhận gói tin tại topic ESP8266/LED\_GPIO16/status  client.subscribe("Topic 1");  client.subscribe("Topic 2");  client.subscribe("Topic 3");  }  else {  // in ra màn hình trạng thái của client khi không kết nối được với MQTT broker  Serial.print("failed, rc=");  Serial.print(client.state());  Serial.println(" try again in 5 seconds");  // Wait 5 seconds before retrying  delay(1000);  }  }  } |

**B4: Viết hàm Setup()**

void setup() {

Serial.begin(115200);

// hàm thực hiện chức năng kết nối Wifi và in ra địa chỉ IP của ESP8266

Serial.println("DHTxx test!");

setup\_wifi();

// cài đặt server là broker.mqtt-dashboard.com và lắng nghe client ở port 1883

client.setServer(mqtt\_server, 1883);

// gọi hàm callback để thực hiện các chức năng publish/subcribe

client.setCallback(callback);

// Đặt các chân data ở chế độ Output

pinMode(led1, OUTPUT);

pinMode(led2, OUTPUT);

pinMode(sensor, INPUT);

dht.begin();

}

**B5: Viết hàm Loop() của chương trình**

|  |
| --- |
| void loop()  {  int value = analogRead(sensor); // gán giá trị đọc được từ analog của sensor light cho value  int humidity = dht.readHumidity(); // gán giá trị hum của dht11 cho humidity  int temperature = dht.readTemperature(); // gán giá trị temp của dht11 cho temperature  if (!client.connected()) {  reconnect();  }  client.loop();  char tempString[8];  sprintf(tempString, "%d", temperature);  Serial.print(" Temperature: ");  Serial.print(tempString);  client.publish("home/sensors/temperature", tempString); // Publish giá trị temp cho vào tempString  char humString[8];  sprintf(humString, "%d", humidity);  Serial.print(" Humidity: ");  Serial.println(humString);  client.publish("home/sensors/humidity", humString); // Publish giá trị hum cho vào humString  char Illumination[8];  sprintf(Illumination, "%d", value);  Serial.print(" Value Light: ");  Serial.println(Illumination);  client.publish("home/sensors/illumination", Illumination); // Publish giá trị value light cho vào Illumination  delay(2000);  } |

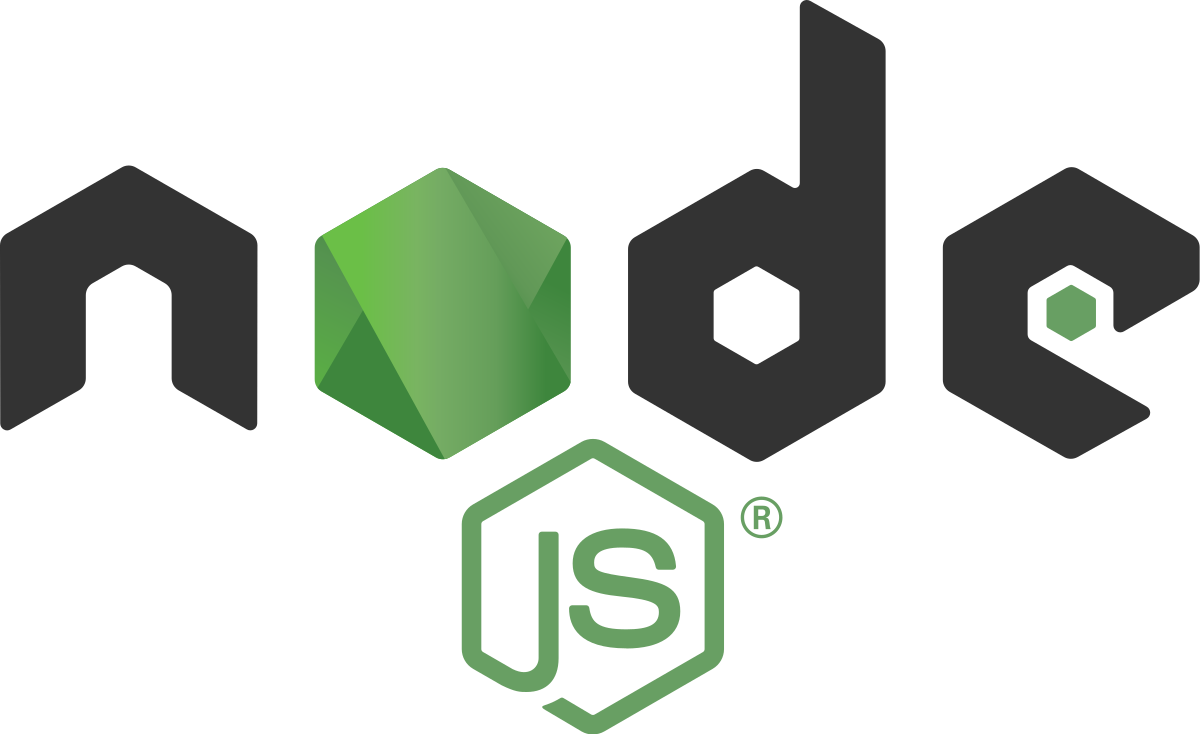
* **Như vậy chúng ta đã lập trình xong chương trình để đọc data của các sensor.**

**3.4 Cài đặt Nodejs, Mqtt, MySQL**

**3.4.1 Nodejs:**

NodeJS là một mã nguồn được xây dựng dựa trên nền tảng Javascript V8 Engine, nó được sử dụng để xây dựng các ứng dụng web như các trang video clip, các forum và đặc biệt là trang mạng xã hội phạm vi hẹp. NodeJS là một mã nguồn mở được sử dụng rộng bởi hàng ngàn lập trình viên trên toàn thế giới. NodeJS có thể chạy trên nhiều nền tảng hệ điều hành khác nhau từ WIndow cho tới Linux, OS X nên đó cũng là một lợi thế. NodeJS cung cấp các thư viện phong phú ở dạng Javascript Module khác nhau giúp đơn giản hóa việc lập trình và giảm thời gian ở mức thấp nhất.

Một lý do Node.js được ưa chuộng nữa là đa phần các lập trình viên viết Web, Mobile đều biết, và giờ đây, nhờ Node.js mà họ có thể triển khai các ứng dụng Server Side bằng Javascript, mà không cần dùng ngôn ngữ nào khác (như trước kia phải cần Java, PHP …​)



* **Các bước cài đặt Nodejs:**

B1: Bắt đầu bằng cách cập nhật danh sách các gói bằng cách gõ:

*$ sudo apt-get update*

**B2:** Cài đặt Node js bằng trình quản lý gói apt

*$ sudo apt-get install nodejs npm*

Kiểm tra lại phiên bản NodeJS sau khi cài đặt:

*$ node –v*

**3.4.2:Cơ Sở Dữ Liệu MySQL**

MySQL là hệ quản trị cơ sở dữ liệu tự do nguồn mở phổ biến nhất thế giới và được các nhà phát triển rất ưa chuộng trong quá trình phát triển ứng dụng.

MySQL được sử dụng cho việc bổ trợ NodeJs, PHP, Perl, và nhiều ngôn ngữ khác, làm nơi lưu trữ những thông tin trên các trang web viết bằng NodeJs, PHP hay Perl,…

**Cài đặt MySQL trên Ubuntu**

|  |
| --- |
| $ sudo apt-get update  $ sudo apt-get install mysql-server |

**Truy cập vào MySQL**

$ mysql –u root –p

Trong đó: root là user. Sau đó nhập mật khẩu để truy cập. (Mật khẩu được tạo khi cài MySQL)

**Tạo Database trong MySQL**

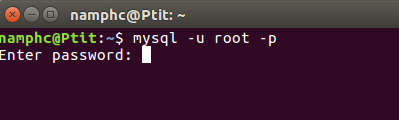
**>> CREATE DATABASE Tên\_cơ\_sở\_dữ\_liệu;**

**Xóa Database**

>> DROP DATABASE Tên\_csdl;

**Xem danh sách Database**

**>> SHOW DATABASES;**



**Chọn Database**

>> USE DATABASE Tên\_csdl

**Tạo bảng**

>> CREATE TABLE Tên\_bảng(Tên\_cột\_1 kiểu\_dữ liệu, …..);

https://scontent.fhan2-2.fna.fbcdn.net/v/t1.15752-9/57581554_2240457876208640_4020572761855361024_n.png?_nc_cat=111&_nc_ht=scontent.fhan2-2.fna&oh=5913bba175538c56cc993567fcbff69e&oe=5D432327

**Xóa bảng**

>> DROP TABLE Tên\_bảng;

**Xem cấu trúc bảng**

>> DESC Tên\_bảng;

**Xem dữ liệu trong bảng**

>> SELECT \* FROM Tên\_bảng;

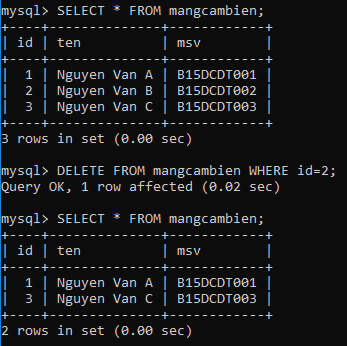
**Thêm dữ liệu vào bảng**

>> INSERT INTO Tên\_bảng(tên\_cột\_1, tên\_cột\_2) VALUES(dữ\_liệu\_cột\_1, dữ\_liệu\_cột\_2, ….);

https://scontent.fhan2-2.fna.fbcdn.net/v/t1.15752-9/57581554_2240457876208640_4020572761855361024_n.png?_nc_cat=111&_nc_ht=scontent.fhan2-2.fna&oh=5913bba175538c56cc993567fcbff69e&oe=5D432327

**Xóa dữ liệu trong bảng**

>> DELETE FROM Tên\_bảng WHERE mệnh\_đề;

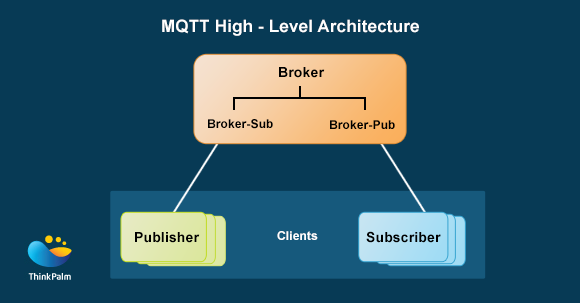


**3.4.3 MQTT Message Queue Telemetry Transport)**

1. Khái niệm

Đây là một giao thức truyền thông điệp (message) theo mô hình publish/subscribe (xuất bản – theo dõi), sử dụng băng thông thấp, độ tin cậy cao và có khả năng hoạt động trong điều kiện đường truyền không ổn định.

Kiến trúc mức cao (high-level) của MQTT gồm 2 phần chính là Broker và Clients.



Trong đó, broker được coi như trung tâm, nó là điểm giao của tất cả các kết nối đến từ client. Nhiệm vụ chính của broker là nhận mesage từ publisher, xếp các message theo hàng đợi rồi chuyển chúng tới một địa chỉ cụ thể. Nhiệm vụ phụ của broker là nó có thể đảm nhận thêm một vài tính năng liên quan tới quá trình truyền thông như: bảo mật message, lưu trữ message, logs,...

Client thì được chia thành 2 nhóm là publisher và subscriber. Client là các software components hoạt động tại edge device nên chúng được thiết kế để có thể hoạt động một cách linh hoạt (lightweight). Client chỉ làm ít nhất một trong 2 việc là publish các message lên một topic cụ thể hoặc subscribe một topic nào đó để nhận message từ topic này.

1. Cài đặt Mosquitto  
   Đầu tiên ta chạy lệnh sau :

*$ sudo apt-get update*

Cài đặt Mosquitto bằng cách sử dụng apt install:

*$ sudo apt install mosquitto mosquitto-clients*

Theo mặc định, Ubuntu sẽ khởi động dịch vụ Mosquitto sau khi cài đặt. Chạy thử các lệnh sau để kiểm tra kiểm tra hệ thống để đảm bảo mosquitto đã được chạy.

– Subscirbe topic:

*$ mosquitto\_sub -h localhost -t test*

– Publish topic:

*$ mosquitto\_pub -h localhost -t test -m "hello world"*

1. Bảo mật máy chủ MQTT

Việc bảo mật cho hệ thống là tối quan trọng trong bất kỳ trường hợp nào. Bảo mật còn quyết định sự sống còn của hệ thống.

Hãy cấu hình Mosquitto để sử dụng mật khẩu. Mosquitto bao gồm một tiện ích để tạo một tệp mật khẩu đặc biệt được gọi là mosquitto\_passwd. Lệnh này sẽ nhắc bạn nhập mật khẩu cho tên người dùng được chỉ định và đặt kết quả vào /etc/mosquitto/passwd.

B1: mở file /etc/mosquitto/conf.d/default.conf và điền vào đoạn code sau

*$ sudo nano /etc/mosquitto/conf.d/default.conf*

*allow\_anonymous false*

*password\_file /etc/mosquitto/passwd*

* allow\_anonymous false : sẽ vô hiệu hóa tất cả các kết nối không được xác thực
* password\_file :dòng này cho Mosquitto biết nơi tìm thông tin mật khẩu và người dùng.

Lưu và thoát tập tin.

B2: Tạo user, password:

*$ sudo mosquitto\_passwd -c /etc/mosquitto/passwd <user> <pasword>*

Khởi động lại Mosquitto và kiểm tra các thay đổi:

*$ sudo systemctl restart mosquitto*

Bây giờ nếu cố gắng xuất bản một tin nhắn mà không có mật khẩu ví dụ như:

*$ mosquitto\_pub -h localhost -t "test" -m "hello world"*

Hệ thống sẽ báo lỗi sau:

*Connection Refused: not authorised.*

*Error: The connection was refused.*

* Để có thể puslish và subscribe bạn cần phải có tên tài khoản và mật khẩu mà bạn đã khai báo cho máy chủ MQTT biết ở bước trên:

# Xây dựng trang web index.js và login.html, index.html

## Xây dựng index.js

**B1: Cấu hình cho index.js**

|  |
| --- |
| var mysql = require('mysql');  var mqtt = require('mqtt');  var express = require('express');  var session = require('express-session');  var bodyParser = require('body-parser');  var path = require('path');  var app = express();  app.set('view engine', 'html');  app.set('views', \_\_dirname )  app.engine('html', require('ejs').renderFile); |

Chúng ta cần khai báo các Topic để Subcribe dữ liệu, bời vì chúng ta gửi 3 data của cảm biến do vậy cần phải có 3 Topic khác nhau. Humidity và Temperature của cảm biến DHT11, value\_light của Sensor Light.

|  |
| --- |
| var count = 0;  var client = mqtt.connect("mqtt://localhost:1883", { clientId: "mqttjs01", username: "huynam", password: "huynam" });// username và password là tài khoản và mật khẩu của MQTT đã được tạo ở hướng dẫn trên  var topic1 = "Topic 1"; // Topic 1 giá trị gửi là giá trị temperture đọc được từ DHT11  var topic2 = "Topic 2"; // Topic 1 giá trị gửi là giá trị temperture đọc được từ DHT11  var topic3 = "Topic 3"; // Topic 1 giá trị gửi là giá trị temperture đọc được từ DHT11  var message = "test message"; |

Kết nối cổng server để truy cập:

|  |
| --- |
| var server = app.listen(3000, () => { //Start the server, listening on port 3000.  console.log("Conect to requests on port 3000...");  }) |

**B2: Tạo connect để kết nối với database nơi mà lưu trữ data của mình**.

|  |
| --- |
| var connection = mysql.createConnection({  host: 'localhost', // dùng loclhost kết nối server  user: 'root', // user của Mysql đã được hướng dẫn tạo ở trên  password: 'huynam', // pass được đặt là pass của Mysql  database: 'Name\_database' // Name\_database là 1 tên database được tạo trong Mysql  }); |

**B3: Tạo bảng trong database và kết nối tới bảng đó**

|  |
| --- |
| connection.connect(function (err) {  if (err)  throw err;  console.log("mysql connected");  var sql = "DROP TABLE IF EXISTS sensors"; // xóa table sensors nếu đã có trong Mysql  connection.query(sql, function (err, result) {  if (err)  throw err;  console.log("drop tables sensors ok");  });  sql = "CREATE TABLE sensors( id INT(10) PRIMARY KEY auto\_increment , Sensor\_ID varchar(10) not null, Date\_and\_Time datetime not null, Temperature int(3) not null,Humidity int(3) not null,Illumination int(3) not null)"  // Create 1 table sensors  connection.query(sql, function (err, result) {  if (err)  throw err;  console.log("create tables sensors ok");  });  }); |

**B4: Link đến file login.html và index.html để tạo giao diện đăng nhập và home để hiển thị**

|  |
| --- |
| app.get('/', function (request, response) {  response.sendFile(path.join(\_\_dirname + '/login.html')); // gọi đến hàm login  });  app.get('/home', (request, response) => {  if (request.session.loggedin) {  connection.query("SELECT \* FROM sensors", function (err, result, fields) {  if (err) throw err;  response.sendFile(path.join(\_\_dirname + '/index.html')); // gọi đến hàm index.html  });  } else {  response.send('Please login to view this page!!');  response.end();  }  }) |

Kiểm tra username và password trong database:

|  |
| --- |
| app.post('/auth', function (request, response) {  var username = request.body.username; //Khai báo username  var password = request.body.password; //Khai báo password  if (username && password) { // Kiểm tra username và pass  // Kết nối với tables accounts trong database và kiểm tra 2 trường username và password  connection.query('SELECT \* FROM accounts WHERE username = ? AND password = ?', [username, password], function (error, results, fields) {  if (results.length > 0) {  request.session.loggedin = true;  request.session.username = username;  response.redirect('/home');  // Nếu user và password đúng sẽ connect đến /home chính là giao diện chính của web hiển thị  }  } else {  response.send('No Connect Account!');  }  response.end();  });  } else {  response.send('Please enter Username and Password!');  response.end();  }  }); |

**B5: Tiếp theo sẽ là phần điều khiển thiết bị trong bài viết này mình sẽ điều khiển thiết bị là 2 led sáng bật, và speaker tăng giảm.**

|  |
| --- |
| socket.on('led\_status1', (data) => {  console.log("LED STATUS: led1 %d", data.led);  publish(topic1, data.led.toString(), options);  });  socket.on('led\_status2', (data) => {  console.log("LED STATUS: led2 %d", data.led);  publish(topic2, data.led.toString(), options);  });  socket.on('led\_pwm', (data) => {  console.log("PWM SPEAKER: led1 %d ", data.led1);  publish(topic3, data.led1.toString(), options);  }); |

**B6: Đẩy dữ liệu đọc được từ các biến vào database**

Kiểm tra các Topic của mình gửi có đúng với topic\_list[] không nếu đúng thì lần lượt ta lưu vào trong topic\_list[]. Trong đó topic\_list[0],topic\_list[1], topic\_list[2] lần lượt là Temperature, Humidity, Value light.

|  |
| --- |
| client.on('message', function (topic, message, packet) {  console.log("message is " + message);  console.log("topic is " + topic);  //message = JSON.parse(message);  if (topic == topic\_list[0]) {  cnt\_check++;  //Temp = message["Temperature"];  Temp = message;  }  else if (topic == topic\_list[1]) {  cnt\_check++;  //Hum = message["Humidity"];  Hum = message;  }  else if (topic == topic\_list[2]) {  cnt\_check++;  //Illumination = message["Illumination"];  Illumination = message;  } |

Do ta gửi 3 Topic chính vì thế chúng ta sẽ nhận đồng thời 3 Topic mới lưu vào database, chứ không phải nhận 1 cái rồi lưu luôn.

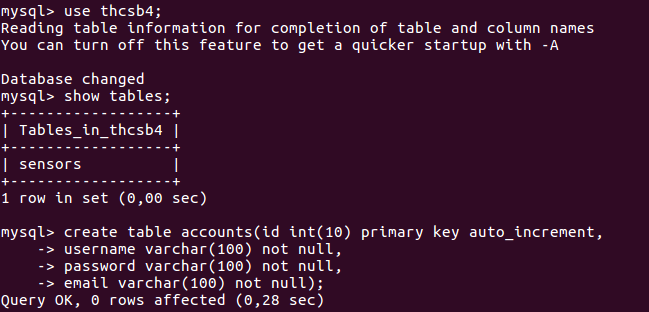
Vì thế ta sử dụng biến cnt\_check để kiểm tra. Mỗi khi nhận được 1 Topic biến sẽ tăng lên 1. Đến khi nào biến cnt\_check = 3 thì ta mới bắt đầu lưu dữ liệu.

|  |
| --- |
| if( cnt\_check == 3 ){  cnt\_check = 0;  console.log(Temp,Hum,Illumination);  console.log("ready to save");  var first\_name = "DHT-11";  var Date\_and\_Time = new Date().toMysqlFormat();  let query = "INSERT INTO `sensors` (Sensor\_ID,Date\_and\_Time,Temperature,Humidity,Illumination) VALUES ('" +  first\_name + "', '" + Date\_and\_Time + "', '" + Temp + "', '" + Hum + "', '"+ Illumination + "')";  connection.query(query, (err, result) => {  if (err) {  throw err;  }}}; |

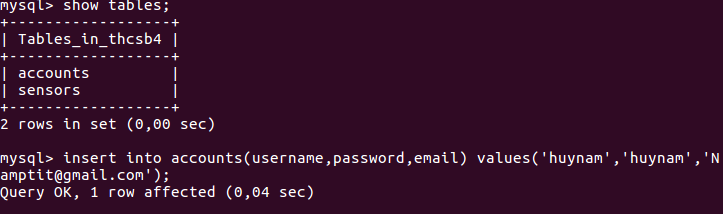
**4.2: Xây dựng trang web login.html**

Khi chúng ta dùng login user và password ta sẽ lấy trong cơ sở dữ liệu để kiểm tra. Do đó username và password sẽ tạo trong cơ sở dữ liệu.

B1: Tạo table trong database với tên là accounts trong đó có các trường là username, password.



B2: Sau khi đã tạo thành công ta sẽ insert giá trị vào trong table accounts.

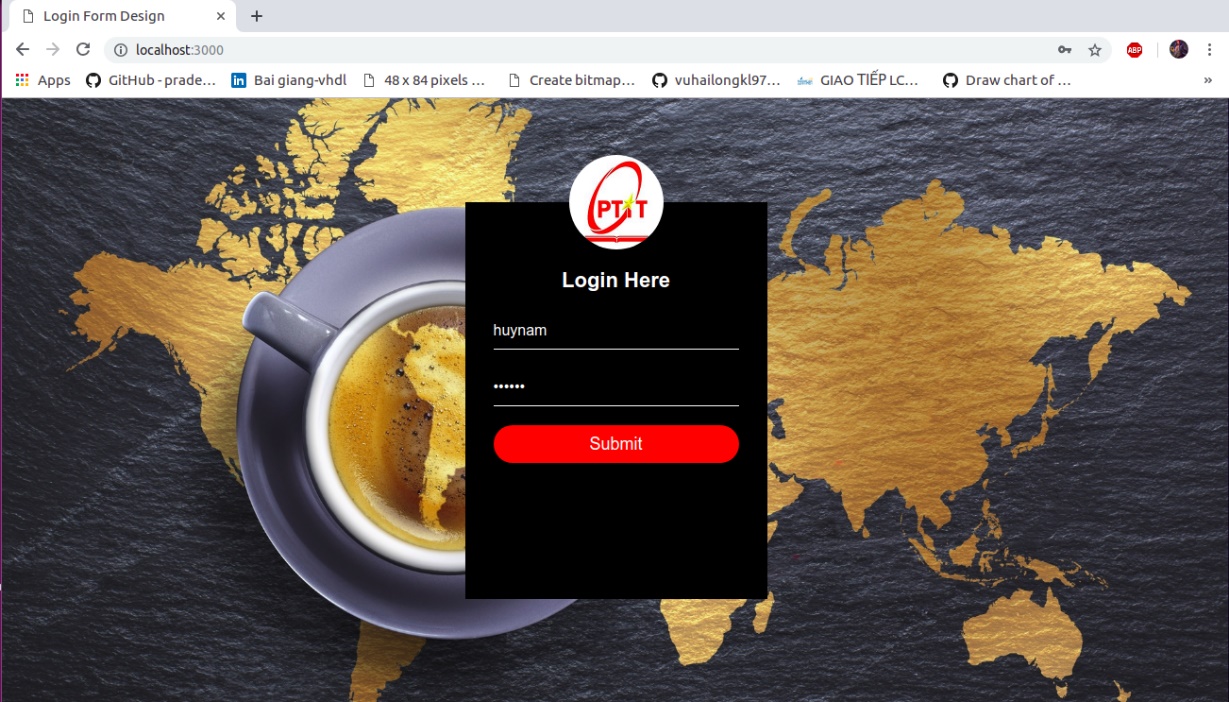


* Giá trị được insert vào table accounts chính là user và password cho tài khoản của login.html mà chúng ta đăng nhập

B3: Tạo file login.html

|  |
| --- |
| <body>  <div class="loginbox">  <img src="https://upload.wikimedia.org/wikipedia/commons/thumb/d/d5/Posts\_%26\_Telecoms\_Institute\_of\_Technology\_logo.jpg/200px-Posts\_%26\_Telecoms\_Institute\_of\_Technology\_logo.jpg" class="avatar">  <h1>Login Here</h1>  <form action="auth" method="POST">  <input type="text" name="username" placeholder="Username" required>  <input type="password" name="password" placeholder="Password" required>  <input type="submit">  </form>  </div>  </body> |

* Các bạn có thể dùng thêm CSS, Bootstrap để tạo giao diện đẹp hơn cho mình.



**4.3: Xây dựng /home là index.html**

Chúng ta sẽ đi khái quát về trang /home giao diện chính của mình có những gì.

* Chúng ta sẽ cần 1 div để Plotly giá trị data đọc được từ các biển.
* 3 box để hiển thị giá trị của Temp, Hum, Value light.
* 2 button ON/OFF điều khiển led
* Và cuối cùng 1 range để điều khiển tăng giảm âm của speaker.

**4.3.1 Đầu tiên ta sẽ tạo 3 box để hiển thị giá trị của Temp, Hum, Value light, tạo 1 class rectangle để hiển thị.**

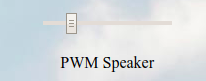
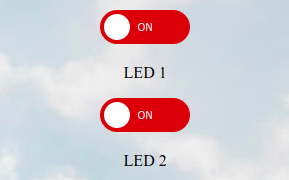
|  |
| --- |
| .rectangle {  height: 100px;  width: 100px;  background-color: blue;  float: left;  margin-left: 20%;  position:relative;  } |

Sau khi tạo xong class ta chỉ cần gọi trong body class tương ứng với id mà mình muốn hiển thị.

|  |
| --- |
| <div id = "hum" class = "rectangle">Hum Value  <center> <p id = 'humidity'>0 </p> </center>  </div>  <div id = "illu" class = "rectangle">Light Value  <center> <p id = 'Light'>0 </p> </center>  </div>  <div id = "temp" class = "rectangle">Temp Value  <center> <p id = 'temperature'>0 </p> </center>  </div> |

**4.3.2 Tạo 2 switch và thanh range cho speaker.**

|  |
| --- |
| <div class="slidecontainer">  // Nguyên lý để tăng giảm độ to nhỏ của còi là PWM giá trị của độ rộng xung.  <input type="range" min="0" max="255" value="50" id="myRange" onchange="updateSlider(this.value)">  <p>PWM Speaker</p>  // Sau khi tạo được button On/Off gọi hàm calc\_led1 là hàm mà ta sẽ dùng để check  Và kiểm tra trạng thái của led được bật hay tắt  <label class="switch">  <input id='led1' type="checkbox" onclick="calc\_led1()"><div class="slider round">  <!--ADDED HTML -->  <span class="on">OFF</span><span class="off">ON</span><!--END-->  </div>  </label>  <p>LED 1</p>  <label class="switch">  <input id='led2' type="checkbox" onclick="calc\_led2()"><div class="slider round">  <!--ADDED HTML -->  <span class="on">OFF</span><span class="off">ON</span><!--END-->  </div>  </label>  <p>LED 2</p>  </div> |



**4.3.3 Plot giá trị đọc được.**

**B1:Khai báo thư viện trong javarscrip:**

|  |
| --- |
| <link href="https://fonts.googleapis.com/css?family=Lato" rel="stylesheet">  <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.4.0/Chart.min.js"></script>  <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.0.4/socket.io.js"></script> |

**B2:Viết kiểu dữ liệu mà ta muốn biểu diễn trong bài viết này mình sẽ hướng dẫn các bạn biểu diễn bằng line.**

|  |
| --- |
| var chart = new Chart(ctx, {  // The type of chart we want to create  type: 'line',  // The data for our dataset  data: {  labels: [],  datasets: [{  label: "Temp Value",  borderColor: "teal",  data: [],  fill: false,  backgroundColor: 'teal',  },  {  label: "Hum Value",  borderColor: "navy",  data: [],  fill: false,  backgroundColor: 'navy',  },  {  label: "Light Value",  borderColor: "yellow",  data: [],  fill: false,  backgroundColor: 'yellow',  }  ]  },  // Configuration options go here  options: {}    }); |

Và cuối cùng đó là dữ liệu sẽ được biểu diễn như sau, và giá trị đọc được các bạn muốn thay đổi màu background color khi giá trị thay đổi. Mình đã thực hiện và kiểm tra trong hướng dẫn dưới các bạn cùng xem. Ý tưởng đơn giản là kiểm tra dữ liệu đọc được sẽ ‘>’hay ‘<’ hay ‘=’ thì sẽ gọi color cho background đó.

Ví dụ:

Khi giá trị humidity của mình nhỏ hơn 65 thì mình sẽ thay đổi màu của nó.

|  |
| --- |
| if(data.hum < 65){  document.getElementById('humidity').style.backgroundColor = '#035403';  } |

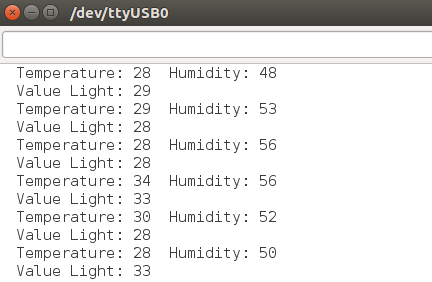
Tương tự với các giá trị Temp, value\_light:

|  |
| --- |
| var my\_data = [data.temp , data.hum, data.illu ];  console.log(my\_data);  var index = 0;  document.getElementById('temperature').innerHTML = data.temp; //update the date  if(data.temp < 30){  document.getElementById('temperature').style.backgroundColor = '#8e0101';  }  if(data.temp >= 30){  document.getElementById('temperature').style.backgroundColor = '#f70404';  }  document.getElementById('humidity').innerHTML = data.hum; //update the date  if(data.hum < 65){  document.getElementById('humidity').style.backgroundColor = '#035403';  }  if(data.hum > 65 && data.hum < 80){  document.getElementById('humidity').style.backgroundColor = '#04a504';  }  if(data.hum > 80){  document.getElementById('humidity').style.backgroundColor = '#04f704';  }  document.getElementById('Light').innerHTML = data.illu; //update the date  if(data.illu < 30){  document.getElementById('Light').style.backgroundColor = '#F4FA58';  }  if(data.illu > 30 && data.illu < 100){  document.getElementById('Light').style.backgroundColor = '#cece04';  }  if(data.illu > 150){  document.getElementById('Light').style.backgroundColor = '#969601';  } |

* Như vậy mình đã hướng dẫn các bạn tạo 1 webserver đơn giản để có thể điều khiển thiết bị cũng như biểu diễn data của các cảm biển thiển thị lên web theo thời gian thực.

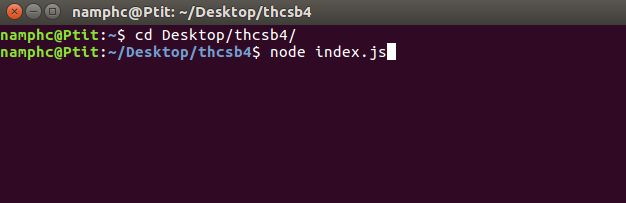
# Hướng dẫn cách chạy chương trình và kết quả:

B1: Mở file arduino và chạy chương trình sau khi build xong kiểm tra trên cổng Serial xem giá trị được in ra. Nếu giá trị được in ra như sau:

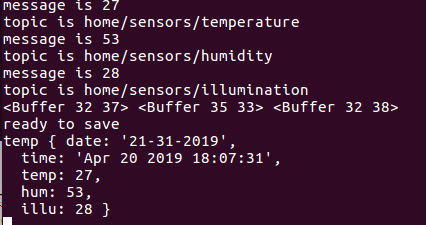


B2: Chạy server với file index.js

Mở cửa sổ Terminal hoặc Ctrl+Alt+T mở đến thư mục của mình có index.js chạy lệnh như sau:

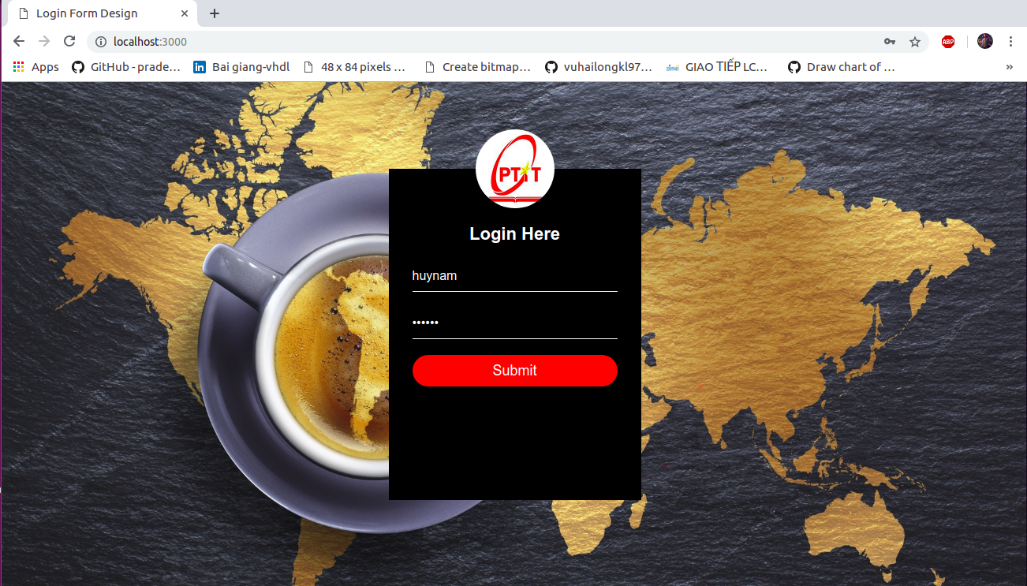


Nếu chương trình của bạn chạy Ok mà không báo lỗi thì kết quả như sau:

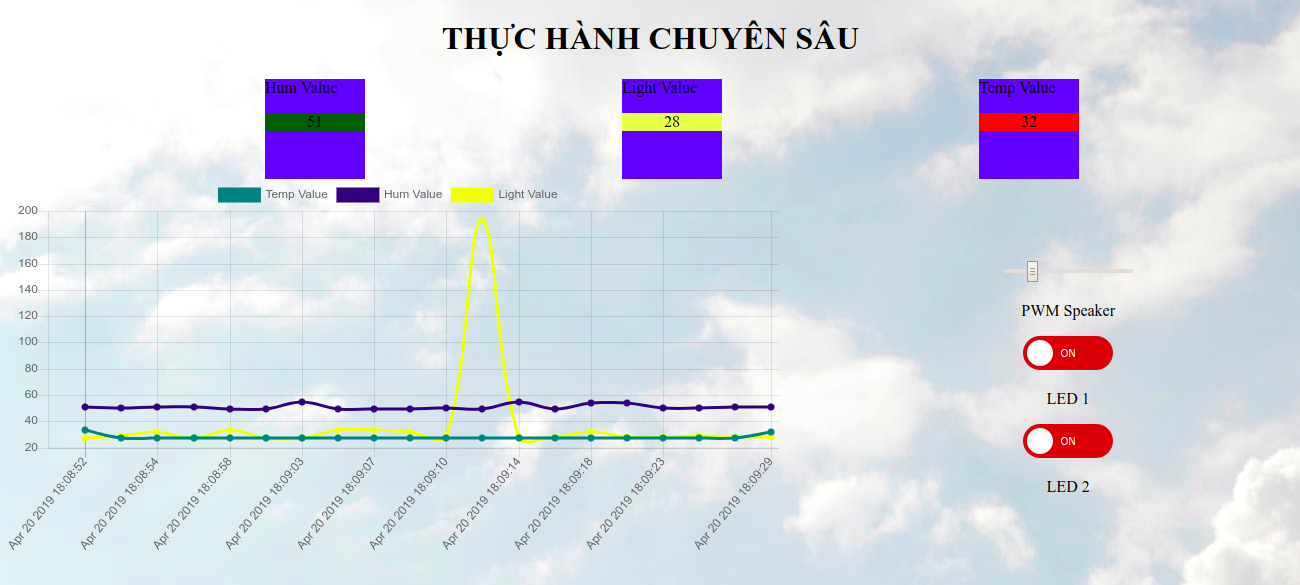


* Như vậy server đã hoạt động và dữ liệu đã được upload.

B3: Mở Google chrome và chạy localhost:3000 sau đó đăng nhập user và password cho login.



Nếu đăng nhập thành công giao diện sẽ chuyển đến /home giao diện chính của ta.



* Giao diện của mình làm còn đơn giản các bạn muốn làm tốt hơn và đẹp hơn có thể tham khảo phần giao diện trên google.
* Bài viết hướng dẫn còn thiếu sót và sơ sài mong thầy và các bạn xem xét và đóng góp ý kiến. Em/mình xin chân thành cảm ơn!
* Dưới đây là link download toàn bộ project các bạn có thể tham khảo

<https://drive.google.com/drive/folders/1QUYUe4W3dGpaPLNedV91QK6t1akO0_Oe?usp=sharing>

**6: Tài liệu tham khảo:**

<https://www.w3schools.com/>

<https://github.com/priyanshrastogi/temperature-plot>