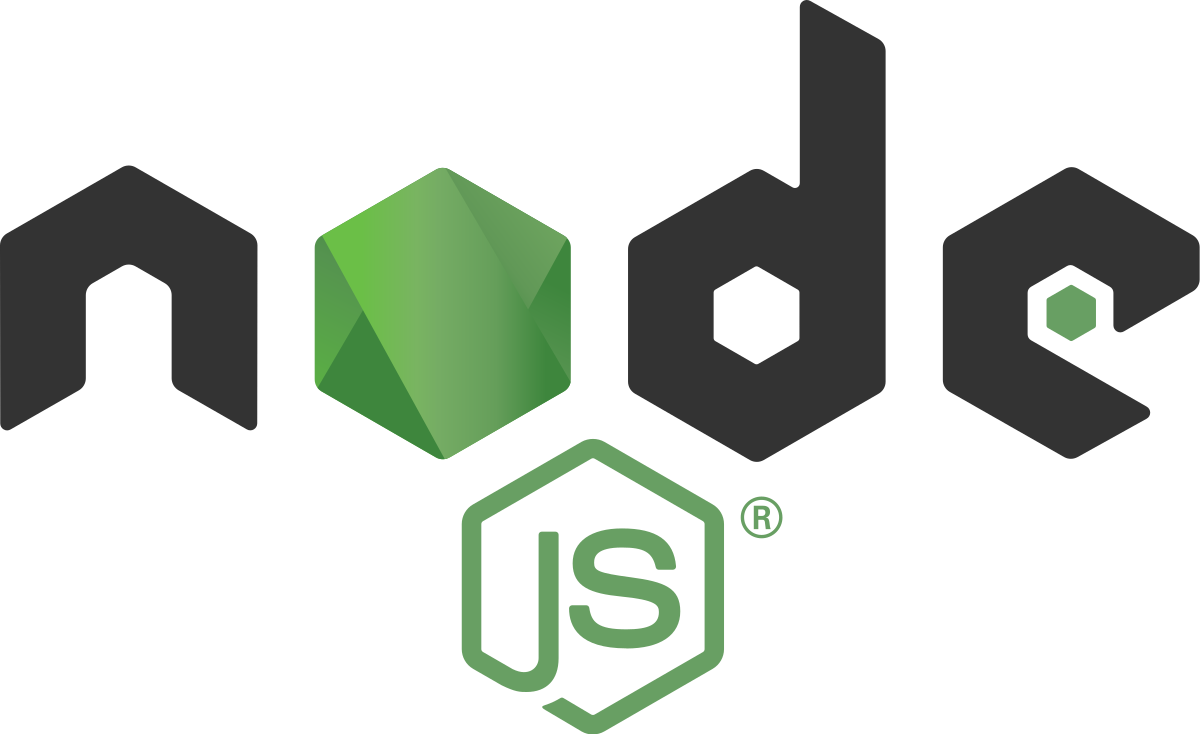
**Tìm hiểu về Nodejs**

1. **NodeJS là gì?**

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.



Bài viết này hướng dẫn cài đặt và sử dụng NodeJS trên Ubuntu 18.04.

Đầu tiên ta chạy lệnh sau :

$ sudo apt-get update

Cài đặt NodeJS bằng cách sử dụng apt install:

$ sudo apt-get install nodejs npm

Kiểm tra lại phiên bản NodeJS sau khi cài đặt:

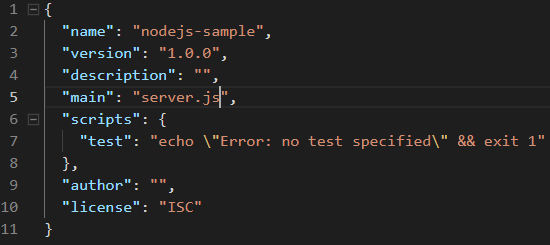
$ node –v

1. **Viết ứng dụng huyền thoại: Hello World**

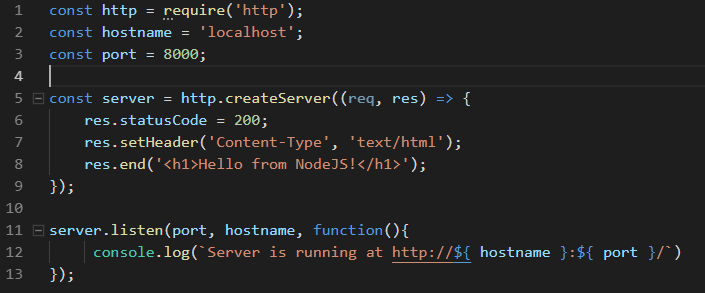
* Trước bạn cần tạo một thư mục có tên **MyProject**, hoặc một tên nào đó mà bạn muốn. Mở cửa sổ **Terminal** và **CD** tới thư mục mà bạn vừa tạo ra
* Tiếp theo ta sẽ tạo file **package.json**, đây là file lưu trữ thông tin cấu hình cũng như các gói packages đang sử dụng trong Project. Để tạo file package.json thì bạn mở **Terminal** lên và nhập vào lệnh:

$ npm init

Chấp nhận các tùy chọn mặc định và nhấn Enter cho tới khi hoàn thành.

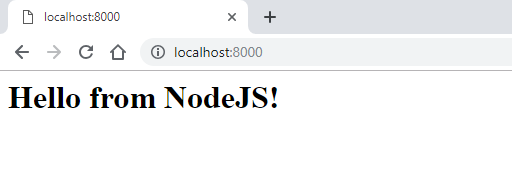


Tiếp theo, tạo một file có tên **server.js**, và nhập vào nội dung cho file này. Đây là file cấu hình cho project của bạn. Chúng ta sẽ viết theo 2 cách:

* **Cách thứ nhất** là dùng module http của Nodejs, đây là module được tích hợp sẵn khi cài Nodejs không sử dụng Framework.

Để chạy chương trình trên, các bạn cũng đứng từ **Terminal** như lúc init project và chạy lệnh **node server.js** hoặc **npm start**, sau đó mở trình duyệt và gõ **localhost:8000/**

**Kết quả:**



* **Cách thứ hai** là sử dụng **Framework Express**, một Framework khá phổ biến để xây dựng ứng dụng Nodejs.

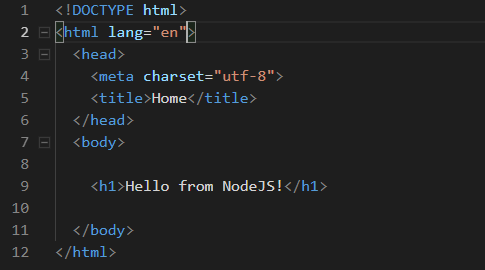
Cài đặt **Express**: npm install express

Cài đặt **EJS**: npm install ejs

Trên project tạo 2 thư mục con là **public** & **views**:

* + **public**: Là thư mục để chứa tất cả các tập tin mà người dùng có thể truy cập vào được, chẳng hạn image, video, file css, …
  + **views**: Là thư mục để chứa các trang (page) của bạn.

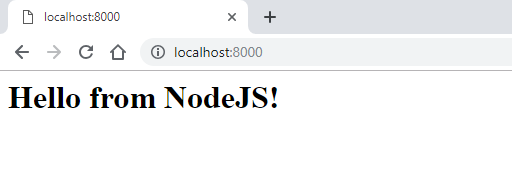
Trong thư mục **views** chúng ta tạo tập tin **home.ejs**:



Trong file **server.js** ta cấu hình như sau:



**Kết quả:**



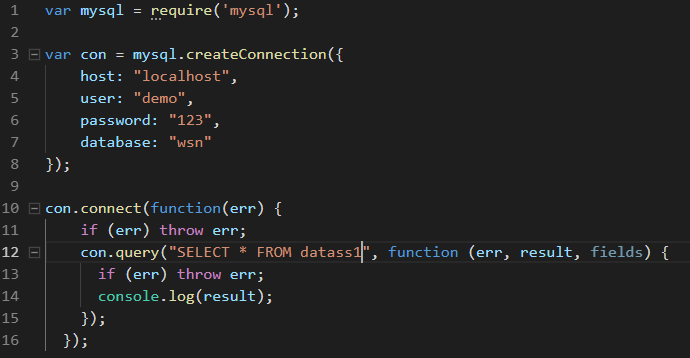
1. **Hiển thị giá trị cảm biến lên website**

**Yêu cầu đề ra :**

Viết code Nodejs lấy các giá trị cảm biến từ databases (Humidity, Temperature, …) và hiển thị các giá trị lên web.

1. **Lấy đữ liệu từ databases**

Cài đặt **mysql**: $ npm install mysql



Dữ liệu trả về có dạng:

[

{ id: 1, temp: 15, hum: 71, light: 230},

{ id: 2, temp: 42, hum: 64, light: 230},

{ id: 3, temp: 20, hum: 86, light: 230},

{ id: 4, temp: 35, hum: 24, light: 230},

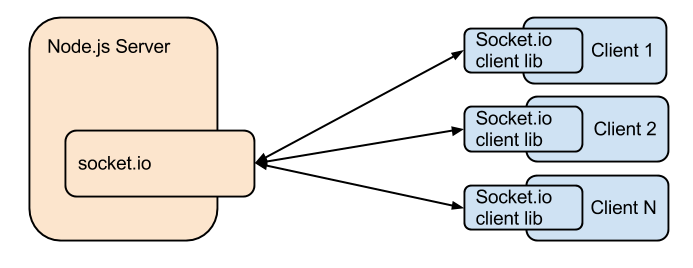
{ id: 5, temp: 24, hum: 68, light: 230}

]

1. **Gửi dữ liệu lên web**

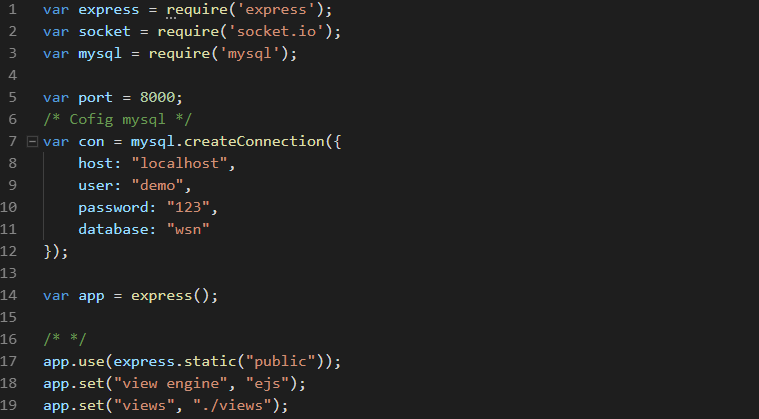
Cài đặt **Socket.IO**: $ npm install socket.io

* Là một module của NodeJs
* Được xây dựng nhằm mục đích tạo ra real time NodeJS application.
* Khi chúng ta include Socket.io module vào trong ứng dụng của mình nó sẽ cung cấp cho chúng ta hai object đó là: socket server quản lý functionality phía server và socket client điều khiển funtionality phía client.
* Khi client muốn kết nối tới Socket.io server, nó sẽ gửi cho server một “handshake HTTP request”. Server sẽ phân tích request đó với những thông tin cần thiết trong suốt quá trình kết nối. Nó sẽ tìm cấu hình của middleware mà đã được đăng ký với server và thực thi chúng trước khi đưa ra sự kiện kết nối. Khi kết nối thành công thì connection event listener được thực thi, tạo ra một instance mới của socket có thể coi như định danh của client mà mỗi một client kết nối tới sẽ có 1 định danh. Các bạn có thể thấy rõ khi xem hình dưới đây

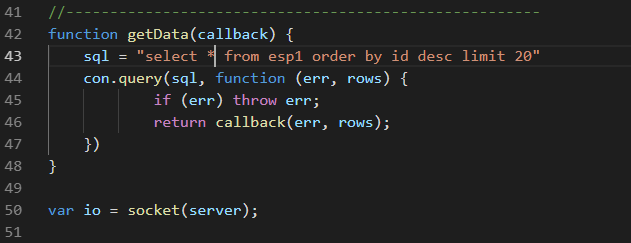


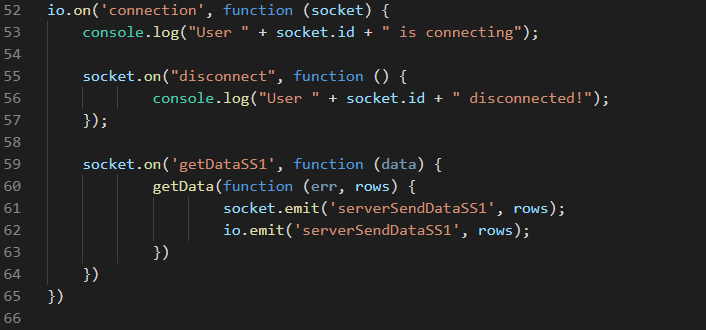
**Sau khi đã có một số khái niệm cơ bản như đã nói ở trên bây giờ ta có thể bắt đầu vào xây dựng một ứng dụng nhỏ sử dụng NodeJs và Socket.io**

**Phía Server:**



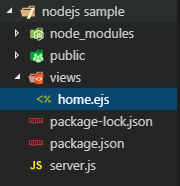






**Phía client:**

* **Hiển thị các giá trị cảm biến:**
  + **Bước 1**: Tạo một file mới trong mục **views** có đuôi **.ejs**:



* **Bước 2**: Tạo các vùng để hiển thị dữ liệu bằng html

<div id="displaySensorVal">

          <div class="card">

            <div class="labelOfVal">

              <h5>Temperature</h5>

            </div>

            <hr>

            <div class="showValue">

              <div id="temp">

                <div id="showTempC">

                  <span id="tempC1"></span><sup>&#8451</sup>

                </div>

                <div id="showTempF">

                  <span id="tempF1"></span><sup>&#8457</sup>

                </div>

              </div>

              <i class="fas fa-temperature-low"></i>

            </div>

          </div>

          <div class="card">

            <div class="labelOfVal">

              <h5>Humidity</h5>

            </div>

            <hr>

            <div class="showValue">

              <div id="showHum">

                <span id="humVal1"></span><span>%</span>

              </div>

              <i class="fas fa-tint"></i>

            </div>

          </div>

          <div class="card">

            <div class="labelOfVal">

              <h5>Light</h5>

            </div>

            <hr>

            <div class="showValue">

              <div id="showLight">

                <span id="lighVal1"></span>

              </div>

              <i class="fas fa-sun"></i>

            </div>

          </div>

        </div> <!-- end displaySensorVal -->

Chúng ta viết thêm một đoạn css cho giao diện:

#displaySensorVal {

width: 100%;

height: 200px;

padding: 15px 0;

}

.card {

background-color: rgba(15, 156, 230, .05);

float: left;

margin: 5px 20px;

border: 1px solid rgba(15, 156, 230, .35);

box-shadow: inset 0 0 20px rgba(15, 156, 230, .25);

border-radius: 5px;

}

.labelOfVal {

width: 270px;

height: 40px;

color: #0f9ce6;

padding-top: 10px;

padding-left: 15px;

}

/\* Show value \*/

.showValue {

width: 270px;

height: 100px;

transition: all 0.7s;

-webkit-transition: all 0.7s;

}

/\* div-Temp \*/

#temp {

width: 60%;

float: left;

}#showTempC {

width: 70%;

margin: 2px 10px;

font-size: 40px;

}

#showTempF {

width: 70%;

margin: 2px 10px;

font-size: 20px;

color: #0f9ce6;

}

/\* div-Hum \*/

#showHum {

float: left;

width: 50%;

margin-top: 5%;

margin-left: 7%;

font-size: 40px;

}

/\* div-Light \*/

#showLight {

float: left;

width: 50%;

margin-top: 5%;

margin-left: 7%;

font-size: 40px;

}

/\* div-icon \*/

.card .showValue i {

margin: 18px 15px;

float: right;

font-size: 60px;

color: #0f9ce6;

}

* **Bước 3:** Thêm đoạn **script** để đọc đữ liệu từ server và hiển thị lên vùng đã tạo:

<script>

function updateData() {

var xhttp = new XMLHttpRequest();

// Đọc đữ liệu trả về từ server

xhttp.onreadystatechange = function () {

if (this.readyState == 4 && this.status == 200) {

var data = JSON.parse(this.responseText);

// In ra giá trị nhiệt độ

document.getElementById("tempC1").innerHTML = data["temp"];

document.getElementById("tempF1").innerHTML = c2fcovertsion(data["temp"]);

// In ra giá trị độ ẩm

document.getElementById("humVal1").innerHTML = data["hum"];

// In ra giá trị ánh sáng

document.getElementById("lighVal1").innerHTML = data["light"];

}

}

// Gửi req đến server

xhttp.open("GET", "/tempVal", true);

xhttp.send();

}

// Chuyển từ độ C -> F

function c2fcovertsion(degree) {

f = Math.round(degree \* 1.8 + 32);

return f;

}

</script>

Dùng hàm **setInterval** để cập nhập giá trị sau mỗi 1s.

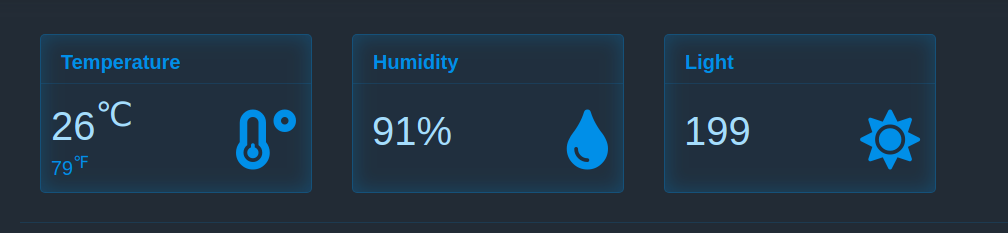
setInterval(refresh, 1000);

function refresh() {

updateData();

}

**Kết quả:**



**\*Note:** Nếu muốn đổi màu sắc các tùy thuộc vào giá trị cảm biến (vd như giá trị **Temperature**, các giá trị khác làm tương tự**)** ta chỉnh đoạn **script** bên trên như sau:

xhttp.onreadystatechange = function () {

if (this.readyState == 4 && this.status == 200) {

var data = JSON.parse(this.responseText);

// In ra giá trị nhiệt độ

document.getElementById("tempC1").innerHTML = data["temp"];

// Phần thay đổi màu sắc ----------------------------------

if(data["temp"] >= 20){

// Nếu nhiệt độ dưới 20\*C ta in ra màu #0f9ce6

document.getElementById("tempC1").style.color = "#0f9ce6";

} else if(data["temp"] > 20 && data["temp"] < 30){

// Nếu nhiệt độ trong 20~30\*C ta in ra màu #f4c741

document.getElementById("tempC1").style.color = "#f4c741";

} else if(data["temp"] >= 30){

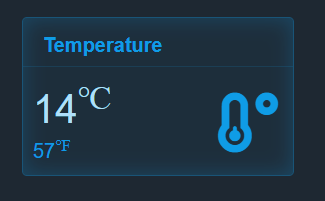
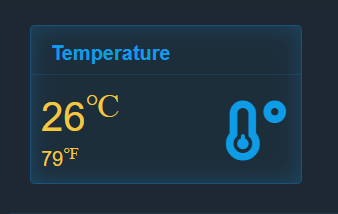
// Nếu nhiệt độ trên 30\*C ta in ra màu đỏ

document.getElementById("tempC1").style.color = "red";

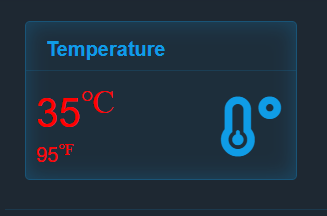
}

// --------------------------------------------------------

**Kết quả:**

*(Nhiệt độ dưới 20\*C) (Nhiệt độ trong khoảng 20~30\*C)*



*(Nhiệt độ trên 30\*C)*

* **Vẽ đồ thị từ các tham số cảm biến đọc được**
  + **Bước 1: Khai báo thư viện**

<script src="/socket.io/socket.io.js"></script>

<script src="javascript/jquery-3.3.1.min.js"></script>

<script type=text/javascript src="javascript/Chart.min.js"></script>

* **Bước 2: Tạo vùng hiển thị trong html:**

<div class="displayChart">

<canvas id="myChart"></canvas>

</div>

* **Bước 3: Viết đoạn script để đọc và vẽ đồ thị**

// Khai báo socket - port 8000

var socket = io('http://localhost:8000');

// Yêu cầu server gửi đữ liệu

function sendReqData() {

socket.emit('getDataSS1');

}

Server sẽ chờ yêu cầu 'getDataSS1' từ phía clien và trả về data. Ta sẽ viết hàm để client chờ nhận data từ phía server .

var socket = io('http://localhost:8000');

    // Require sensor data

    function sendReqData() {

      socket.emit('getDataSS1');

    }

    // Receive sensor data for chart

    socket.on('serverSendDataSS1', function (data) {

      var temp\_data = {

        temp: [],

        hum: [],

        time: []

      }

      for (var i = data.length - 1; i >= 0; i -= 1) {

        temp\_data.temp.push(data[i].temp);

        temp\_data.hum.push(data[i].hum);

        temp\_data.time.push(data[i].datetime);

      }

      //Update chart

      myChart.data.labels = temp\_data.time;

      myChart.data.datasets[0].data = temp\_data.temp;

      myChart.data.datasets[1].data = temp\_data.hum;

      myChart.update();

    });

Sau khi nhận, dữ liệu sẽ được phân tích thành từng giá trị nhiệt độ, độ ẩm,.. rồi được đẩy vào đồ thị.

* **Bước 4: Vẽ đồ thị**

<script>

    var options = {

      responsive: true,

      title: {

        display: false,

        text: 'Chart of Temp & Hum',

        fontSize: 20,

        fontColor: '#0f9ce6',

      },

      scales: {

        xAxes: [{

          gridLines: {

            display: false

          },

          ticks: {

            autoSkip: true,

            maxTicksLimit: 20 // Max 20

          },

          scaleLabel: {

            display: false,

            labelString: "Times",

            fontSize: 15,

            fontColor: '#3cba9f'

          }

        }],

        yAxes: [{

          id: "val\_temp",

          position: 'left',

          type: 'linear',

          scaleLabel: { // Set lable Oy on left

            display: true,

            labelString: "Temperature (℃)",

            fontSize: 15,

            fontColor: '#c45850',

          },

          ticks: {

            min: 5,

            max: 40

          }

        }, {

          id: "val\_hum",

          position: 'right',

          type: 'linear',

          scaleLabel: { //Set lable Oy on right

            display: true,

            labelString: "Humidity (%)",

            fontSize: 15,

            fontColor: '#3e95cd'

          }

        }]

      }

    }

    //-----------------------------------------------------------

    var ctx = document.getElementById('myChart').getContext('2d');

    var myChart = new Chart(ctx, {

      title: "Hear",

      type: 'line',

      data: {

        labels: [],

        datasets: [{

          label: "Temperature",

          yAxisID: 'val\_temp',

          data: [],

          borderColor: "#c45850",

          backgroundColor: '#c45850',

          fill: false

          //borderWidth: 2

        }, {

          label: "Humidity",

          yAxisID: 'val\_hum',

          data: [],

          borderColor: "#3e95cd",

          backgroundColor: '#3e95cd',

          fill: false

        }]

      },

      options: options

    });

  </script>

1. **Điều khiển thiết bị**

Cài đặt **mqtt** : $ npm install mqtt

**Phía Server**

// Khai báo thêm thư viện mqtt cho NodeJS

var mqtt = require('mqtt')

//Khai báo thông số cho mqtt

var options = {

port: 1883,

keepalive: 40,

username: 'mcb1',

password: '123'

}

var client = mqtt.connect('mqtt://localhost/', options);

Trong phần Socket.IO ta thêm đoạn code sau:

io.on('connection', function (socket) {

console.log("User " + socket.id + " is connecting");

socket.on("disconnect", function () {

console.log("User " + socket.id + " disconnected!");

});

// Lắng nghe sự kiện ledChange từ Client

socket.on('ledChange', function (data) {

console.log(data);

var jsonData = JSON.stringify(data);

// gửi data qua mqtt với topic 'home/gpioChange'

client.publish('home/gpioChange', jsonData);

io.emit('broadcastChange', jsonData);

});

// Lắng nghe sự kiện pwmChange từ Client

socket.on('pwmChange', function (data) {

var pwmVal = JSON.stringify(data);

console.log(pwmVal);

// gửi data qua mqtt với topic 'home/gpioChange'

client.publish('home/gpioChange', pwmVal);

});

})

**Phía Client:**

* **Bước 1: Tạo button và range sliders trong html:**

Có rất nhiều kiểu button và sliders để chúng ta lựa chọn. Vd như kiểu này:

<input type="range" min="0" max="255" value="125" id="pwmled1">

<span id="pwmVal1"></span>



<label class="switch">

<input type="checkbox" name="button" id="led1" onchange="stageChange(this)">

<span class="slider1 round"></span>

</label>



* **Bước 2: Đọc sự kiện button**

Để phát hiện sự kiện ấn nút ta sử dụng **onchange**. Khi đó hàm **stageChange()** sẽ được gọi.

// Check change stage

function stageChange(element) {

if (element.checked) {

var msg = { 'name': 'esp1', 'id': element.id, 'sst': 'on' };

// Emnit ledChange

socket.emit('ledChange', msg);

} else if (!element.checked) {

var msg = { 'name': 'esp1', 'id': element.id, 'sst': 'off' };

// Emnit ledChange

socket.emit('ledChange', msg);

}

}

* **Bước 3: Đọc giá trị** **sliders**

Để phát hiện sự kiện dùng **sliders** ta sử dụng **oninput.**

<script>

var slider = document.getElementById("pwmled1");

var output = document.getElementById("pwmVal1");

output.innerHTML = slider.value;

slider.oninput = function() {

output.innerHTML = this.value;

// Send slider value

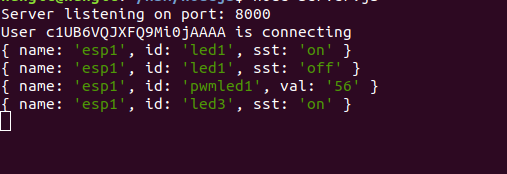
var msg = { 'name': 'esp1', 'id': this.id, 'val': this.value }

// emit ledChange with data msg

socket.emit('ledChange', msg);

}

</script>



*Console log server khi ta điều khiển thiết bị*

1. **Tạo trang login**

Chúng ta sẽ xây đựng một trang login đơn giản:

* Chức năng xác thực người dùng với username/password được lưu trong databases
* Sử dụng session để lưu phiên làm việc

**Phương thức GET và POST**

**GET** và **POST** là hai phương thức của giao thức HTTP, đều là gửi dữ liệu về server xử lí sau khi người dùng nhập thông tin vào form và thực hiện submit.

Đặc điểm:

* Khi lấy dữ liệu nên dùng **GET** để truy xuất và xử lí nhanh hơn.
* Khi tạo dữ liệu nên dùng **POST** để bảo mật dữ liệu hơn. Một ví dụ cụ thể như trong trường hợp bạn cần tạo một form để người dùng upload avatar hay video thì chúng ta thường sẽ sử dụng phương thức POST. Tuy nhiên khi tải về avatar hoặc video thì bạn lại nên sử dụng phương thức GET. Ngoài ra khi cần xử lý các thông tin nhạy cảm ví dụ như email, password thì bạn cần sử dụng POST thay vì GET.

**Giới thiệu về Session**

Một session hay còn gọi là một phiên làm việc. Trong khoa học máy tính, Nó đơn giản là cách giao tiếp giữa client (ở đây là trình duyệt web hoặc ứng dụng trên thiết bị của bạn) với server. Một session bắt đầu khi client gửi request đến sever, nó tồn tại xuyên suốt từ trang này đến trang khác trong ứng dụng và chỉ kết thúc khi hết thời gian timeout hoặc khi bạn đóng ứng dụng. Giá trị của session sẽ được lưu trong một tệp tin trên máy chủ. Thông thường chúng ta chỉ nên lưu trữ những thông tin tạm thời trong session VD như: thông tin đăng nhập, thông tin các sản phẩm trong giỏ hàng ...

Cài đặt các thư viện:

Cài đặt **Session:** $ npm install express-session

Cài đặt **body-parser:** $ npm install body-parser

Trong database chúng ta cần tạo ra một bảng **accounts** gồm 2 trường: **username** và **password**, sau đó ta insert user va pass vào. Vd như user: ‘mcb’, pass: ‘1’ . Đây chính là để ta xác thực tài khoản khi đăng nhập.

* **Bước 1: Phía Server**

// Khai báo thêm thư viện session và bodyParser

var session = require('express-session');

var bodyParser = require('body-parser');

app.use(session({

secret: 'secret',

resave: true,

saveUninitialized: true

}));

app.use(bodyParser.urlencoded({ extended: true }));

app.use(bodyParser.json());

// Authentication

app.post('/auth', function (req, res) {

// Lấy giá trị user và pass từ data qua phương thức POST

var username = req.body.username;

var password = req.body.password;

// Kiểm tra trong database

if (username && password) {

sql = 'SELECT \* FROM accounts WHERE username = "' + username + '" AND password = "' + password + '"'

con.query(sql, function (error, results) {

if (results.length > 0) {

// Lưu vào session khi đăng nhập thành công

req.session.loggedin = true;

req.session.username = username;

res.redirect('/');

} else {

//Quay về trang login nếu ko thành công

res.render("login", { notice: '\*Wrong user or password'});

}

});

} else {

res.render("login", { notice: '\*Wrong user or password !!!' });

}

});

Ở phần route ta thêm điều kiện truy cập vào trang:

// Trả về trang login khi truy cập /login

app.get("/login", function (req, res) {

if (!req.session.loggedin) {

res.render("login", { notice: '' });

} else {

res.redirect('/');

}

});

app.get("/", function (req, res) {

// Kiểm tra xem đã đăng nhập chưa

if (req.session.loggedin) {

// Nếu đã đăng nhập thì đc phép truy cập

res.render("index", { pageTitle: "Home" });

} else {

//Quay về trang login nếu chưa đăng nhập

res.redirect('/login');

}

});

* **Bước 2: Xây dựng trang login**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Login</title>

<link rel="stylesheet" type="text/css" href="css/style.css" />

<script src="javascript/script.js"></script>

</head>

<body>

<div class="login-content">

<div class="login-form">

<h1>Login</h1>

<form action="auth" method="POST">

<input type="text" name="username" placeholder="Username" autocomplete="off" required>

<input type="password" name="password" placeholder="Password" required>

<input type="submit">

</form>

<p style="color:red;"> <%= notice %> </p>

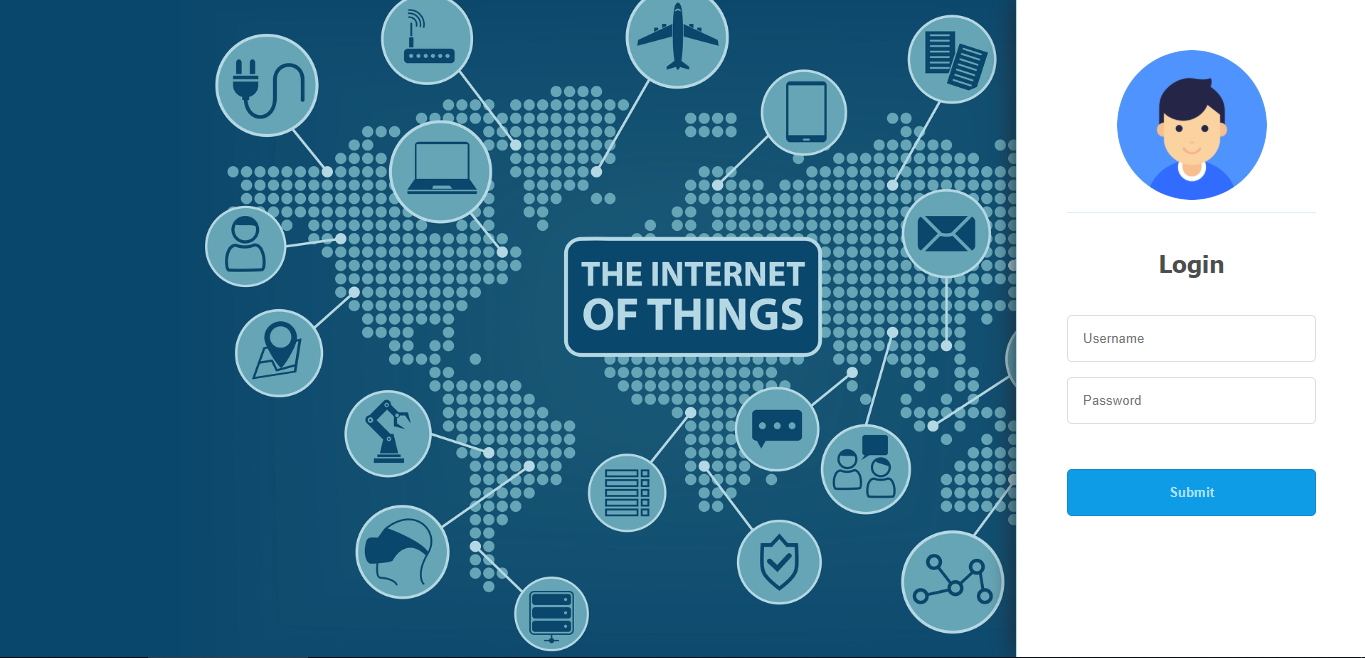
</div>

</div>

</body>

</html>

Khi truy cập vào đường link: <http://localhost:8000/login>. Chúng ta sẽ được giao diện trang đăng nhập sẽ như thế này:



Đây là giao điên trang chủ khi đăng nhập thành công

