**TRƯỜNG ĐẠI HỌC BÁCH KHOA - ĐHQG TP. HCM**

**KHOA ĐIỆN – ĐIỆN TỬ**

**BỘ MÔN ĐIỆN TỬ**

**🙟🕮🙝**

****

**THU TRONG TRUYỀN THÔNG SỐ**

**ĐỀ TÀI:**

**Lập trình FULL-STACK với module nhúng**

**và ứng dụng Marchine Learning**

**GVHD**: PGS TS. Hà Hoàng Kha

**SVTH**: Trần Thanh Đăng Khoa 1411856

Lương Hoàng Phúc 1412944

Nguyễn Sơn Thăng 1413647

**Lớp:** Việt Pháp 2014 – Viễn Thông

TP. HỒ CHÍ MINH, THÁNG 11 NĂM 2018

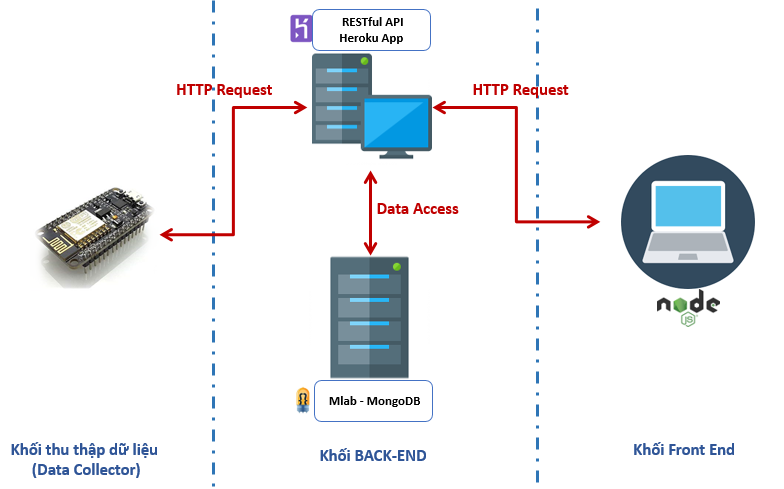
**MỤC LỤC**

1. TỔNG QUAN ĐỀ TÀI3
2. Tổng quan3
3. Nhiệm vụ đề tài3
4. Hợp đồng nhóm3
5. TIẾN HÀNH THỰC NGHIỆM4
6. Thiết kế Back-End4
7. Khối thu thập dữ liệu11
8. Thiết kế Front-End16
9. Machine Learning22
10. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN26
11. TÀI LIỆU THAM KHẢO27
12. **TỔNG QUAN ĐỀ TÀI**
13. **TỔNG QUAN**

Xã hội ngày càng hiện đại, khoa học kỹ thuật càng phát triển thì việc ứng dụng tự động hóa và Internet of Things vào những hoạt động thường ngày lại trở nên phổ biến hơn. Với sự ra đời của ngôi nhà thông minh, và mạng lưới IoT cùng các cảm biến, con người có thể dễ dàng kiểm tra, quan sát điều kiện ở một nơi nào đó thông qua mạng giao diện Web hiển thị dữ liệu thu thập được từ một cảm biến. Thêm vào đó, cùng mới sự phát triển ngày một mạnh mẽ của công nghệ “Marchine Learning”, viễn cảnh con người ứng dụng Marchine Learning vào cuộc sống hằng ngày để dự đoán trước các tình huống có thể xảy ra ngày một rõ ràng hơn trong tương lại không xa.

Trong dự án này, chúng tôi sẽ thiết kế một hệ thống thu thập dữ liệu nhiệt độ và độ ẩm cũng như tọa độ hiện tại của cảm biến, sau đó dữ liệu sẽ được “POST” lên Data Server. Người dùng có thể tương tác với dữ liệu này thông qua dao diện Front-end được lập trình sẵn.

Đề tài **“Lập trình FULL-STACK với module nhúng và ứng dụng Marchine Learning”** do nhóm thiết kế nhìn chung có thể được mô tả bằng mô hình sau:



Hình 1: Thiết kế của nhóm

1. **Khối thu thập dữ liệu (Data Collector)**

Đây là khối lập trình nhúng, đảm nhiệm nhiệm vụ giao tiếp Sensor và các module. Các dữ liệu cần thiết như nhiệt độ, độ ẩm và tọa độ sẽ được xử lý lấy mẫu tại khối này. Sau đó, dữ liệu sẽ được gửi đến Server API bằng phương thức HTTP POST theo đúng data schema được tạo sẵn ở Data Server.

1. **Khối Back-end**

Đây là khối lưu trữ dữ liệu và xử lý các yêu cầu được gửi tới. Đối với đối tượng yêu cầu là khối thu thập dữ liệu với phương thức HTTP POST, RESTful API sẽ kiểm tra schema được gửi tới. Nếu data schema hợp lệ, dữ liệu sau đó sẽ được lưu tại database Mlab của MongoDB. Đối với đối tượng yêu cầu là client với phương thức HTTP Get khi truy cập route được lập trình trước và deploy bằng bên thứ 3 Heroku, RESTful API sẽ trả về dữ liệu dưới dạng bảng, đồ thị và bản đồ.

1. **Khối front-end**

Khối front-end là giao diện cho người sử dụng. Khi người sử dụng truy cập đường dẫn nghĩa là gửi HTTP Get tới khối back-end và sẽ truy cập dữ liệu trong database. Người sử dụng có thể tương tác lựa chọn các dạng hiển thị khác nhau như bảng, đồ thị và bản đồ.

1. **NHIỆM VỤ ĐỀ TÀI**

Nội dung 1: Tìm hiểu lý thuyết

* Lập trình Back-end, front end
* Ngôn ngữ JavaScript
* Các module hỗ trợ của ngôn ngữ Javascript

Nội dung 2: Thiết kế các khối như mô hình đã nêu

Nội dung 3: Nghiên cứu áp dụng Marchine Learning để dự báo thời tiết

1. **HỢP ĐỒNG NHÓM:**

|  |  |  |
| --- | --- | --- |
| **HỢP ĐỒNG NHÓM** | | |
| Tên nhóm: Nhóm X | | Ngày: 10/04/2018 |
| **Thành viên nhóm** | **Vai trò** | **Chữ ký** |
| Trần Thanh Đăng Khoa  (Trưởng nhóm) | Thiết kế phần cứng  Thiết kế Back-End |  |
| Lương Hoàng Phúc | Thiết kế Front-End |  |
| Nguyễn Sơn Thăng | Thiết kế Machine Learning |  |
| **Nhiệm vụ** | | **Thành viên** |
| 1. Tìm tài liệu liên quan | | Tất cả |
| 2. Thiết kế kiến trúc hệ thống | | Trần Thanh Đăng Khoa |
| 3. Thiết kế Back-End | | Trần Thanh Đăng Khoa |
| 4. Thiết kế Front-End | | Lương Hoàng Phúc |
| 5. Lập trình Machine Learning | | Nguyễn Sơn Thăng |
| 6. Tổng hợp và debug | | Tất cả |
| **Quy định nhóm** | 1. Làm việc và tuân theo sự chỉ dẫn của Trưởng nhóm  2. Tham gia đầy đủ các buổi thảo luận, nếu bận cần phải thông báo trước cho Trưởng nhóm ít nhất 1 ngày  3. Các thành viên phải lắng nghe và tôn trọng ý kiến của nhau, cùng thống nhất để đạt được ý kiến chung  4. Hoàn tất các nhiệm vụ được đề ra trước hạn | |

1. **TIẾN HÀNH THỰC NGHIỆM**
2. **THIẾT KẾ BACK-END**

Đến với thiết kế các ứng dụng, ta có thể ví nó như một tảng băng. Khi phần nhỏ những thứ mà người sử dụng có thể thấy, tương tác được, được gọi là Front-end còn lại tất cả những gì diễn ra bên dưới mà chỉ có những người lập trình mới biết được được gọi là Back-end. Hay nói cách khác, Back-end là một hệ thống phức tạp bao gồm rất nhiều các thuật toán, các hàm, các quá trình logic hoạt động ở Background làm nền tảng cho các ứng dụng front-end. Dữ liệu cần được lưu trữ và truy xuất, logic nghiệp vụ và các quy tắc cần phải được tuân theo và kết quả cần được tính toán và tất cả các quá trình đó diễn ra tại back-end.

Với phạm vi đề tài này, nhóm thiết kế và xây dựng hệ thống Back-end để lưu trữ và truy xuất các dữ liệu trên clound bằng một server **RESTful APIs**

Theo đó, REST là từ viết tắt của Representational State Transfer. Đó là kiến trúc tiêu chuẩn web và Giao thức HTTP. Phong cách kiến trúc REST mô tả sáu ràng buộc ban đầu được truyền đạt bởi Roy Fielding trong luận án tiến sĩ của ông và định nghĩa cơ sở của kiểu RESTful là:

1. Giao điện đồng nhất (Uniform interface)
2. Không trạng thái (stateless)
3. Lưu trữ được (cacheable)
4. Client-Server
5. Hệ thống phân lớp (Layered system)
6. Lập trình theo nhu cầu (code on demand) - optional

Các ứng dụng RESTful sử dụng các yêu cầu HTTP để thực hiện bốn hoạt động được gọi là CRUD (C: create, R: read, U: update, và D: delete). Create/Update được sử dụng để đăng dữ liệu (post), nhận dữ liệu (get) bằng read / listing và delete để xóa dữ liệu.

Trong đề tài, tôi sẽ tạo một RESTful API bằng cách sử dụng các tool Node.js, MongoDB để lưu trữ dữ liệu và HeroKu để triển public Server.

Cấu trúc Server sẽ gồm các thành phần chính sau:

* File Package.json
* File Server.js
* Thư mục API bao gồm Controllers, Models và Routes chưa các file JS tương ứng

1. **File Package.json**

Đây là file sẽ được khởi tạo đầu tiên khi ta tiến hành khởi tạo một Server.

**Package.json** sẽ lưu trữ các thông tin tổng quan của project như tên, mô tả, main... đồng thời cũng lưu các Scripts để ta thực thi hệ thống.

Ở đây ta có scripts Start được sử dụng để khởi động node và gọi file **Server.**

{

"name": "todolistapi",

"version": "1.0.0",

"description": "\"tran thanh dang khoa\"",

"main": "index.js",

"scripts": {

"test": "hehe",

"start": "node server.js"

},

"author": "",

"license": "ISC",

"devDependencies": {

"nodemon": "^1.18.4"

},

"dependencies": {

"cors": "^2.8.4",

"express": "^4.16.3",

"mongoose": "^5.2.14",

"socket.io": "^2.1.1"

}

}

Ngoài ra, trong nội dung file còn chứa thông tin về các “dependencies” lưu trữ các module hỗ trợ của hệ thống. Khi ta thực thi command ***Npm install*** trên terminal, hệ thống sẽ tự động cài đặt các module này và người thực hiện sẽ không cần cài đặt thủ công.

1. **Server.js**

Đầu tiên, ta phải tạo các object Express, và Cors để object hóa các module Express và Cors:

var express = require('express');

var cors = require('cors');

var app = express();

app.use(cors());

Express là một framework giành cho nodejs. Nó cung cấp cho chúng ta rất nhiều tính năng mạnh mẽ trên nền tảng web cũng như trên các ứng dụng di động. Express hỗ rợ các phương thức HTTP và midleware tạo ra môt API vô cùng mạnh mẽ và dễ sử dụng. Nhóm sẽ sử dụng module Express để khởi tạo các lớp trung gian để trả về các HTTP request đồng thời định nghĩa router cho phép sử dụng các hành động khác nhau dựa trên phương thức HTTP vì trong phần thiết kế khổi thu thập dữ liệu. Nhóm sẽ giao tiếp giữa cảm biến và Data Server theo giao thức HTTP. Express cugnx cho phép ta tạo nhiều application trong một API

CORS (Cross-Origin Resource Sharing) là một kĩ thuật được sinh ra để làm cho việc tương tác giữa client và server được dễ dàng hơn, nó cho phép JavaScript ở một trang web có thể tạo request lên một REST API được host ở một domain khác.

Một application được định nghĩa là app được khởi tạo và application này ngay lập tức thực thi module cors với code : app.use(cors());

port = process.env.PORT || 3000,

mongoose = require('mongoose'),

Task = require('./api/models/todoListModel'),

bodyParser = require('body-parser');

mongoose.Promise = global.Promise;

mongoose.connect('mongodb://danko:929035xxx@ds125060.mlab.com:25060/hehe');

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

app.use(bodyParser.json());

Module Mongoose được khai báo và sử dụng để tương tác với mlab và database MongoDB. Và hàm connect được sử dụng để liên kết với cloud databasse Mlab của người lập trình BackEnd.

Object Task để gọi các model về Data của hệ thống được tạo ra và định nghĩa trong file todoListModel.

body-parser: Đây là lớp trung gian, xỷ lý JSON, text và mã hóa URL. Ta thiết lập object app sử dụng bodyParser với các thiết lập thích hợp.

var routes = require('./api/routes/todoListRoutes');

routes(app);

app.listen(port);

app.use(express.static('public'));

console.log('todo list RESTful API server started on: ' + port);

app.use(function(req, res) {

res.status(404).send({url: req.originalUrl + ' not found'})

});

Với đoạn code trên, ta gọi các Route đã được khởi tạo với object routes, và cho nó hoạt động khi app được gọi đến. Sau đó ta lập trình mở rộng thêm một p hần nhỏ như: thông báo port hoạt động của hệ thống backend lên terminal, thông báo lỗi khi gặp vấn đề.

1. **todoListModels.js**

Đây là File để tạo các Schema thống nhất giữa Server và client. Khối thu thập dữ liệu khi muốn “post” dữ liệu lên database sẽ phải tuân theo các định dạng dữ liệu này.

'use strict';

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

var TaskSchema = new Schema({

data: {

type: String,

required: 'Kindly enter the name of the task'

},

temp: {

type: Number,

default: 0

},

humid: {

type: Number,

default: 0

},

laitude: {

type: Number,

default: 10.8888

},

longitude: {

type: Number,

default: 106.5777

},

Created\_date: {

type: Number,

default: function(){return new Date().getTime()}

},

});

module.exports = mongoose.model('Tasks', TaskSchema);

Ở đây, Module Mongoose được gọi ra và ta sẽ tạo một “mẫu” dữ liệu (TaskSchema) theo định dạng của Mongoose vì ta sẽ sử dụng could mlab của object này.

Schema bao gồm:

* Name: Tên (mục đích của đợt gửi data)
* Temp: biến Nhiệt độ, mặc định là 0
* Humid: biến Độ ẩm, mặc định là 0
* Longitude: biến số để lưu kinh độ
* Latitude: biến dạng số để lưu vĩ độ
* Created\_date: thời gian ghi nhận dữ liệu, biến này sẽ được tự động khởi tạo khi data gửi tới Database hợp lệ.

1. **todoListRoutes**

Routes quyết định việc xác định cách ứng dụng phản hồi yêu cầu của một client cụ thể, là một URL(hoặc đường dẫn) và một phương thứcHTTP cụ thể (GET, POST, vv).Mỗi routes mà nhóm lập trình có các hàm điều khiển khác nhau được thực thi khi route được khớp với nó.

Dưới đây ta đã xác định hai routes cơ bản (‘/ tasks’ và ‘/ tasks / taskId’). ‘/ Tasks’ thực thi đồng thời cả 2 phương thức ‘GET’ và ‘POST’, trong khi ‘/ tasks / taskId’ thực thi GET, PUT và DELETE.

'use strict';

module.exports = function(app) {

var todoList = require('../controllers/todoListController');

// todoList Routes

app.route('/tasks')

.get(todoList.list\_all\_tasks)

.post(todoList.create\_a\_task);

app.route('/tasks/:taskId')

.get(todoList.read\_a\_task)

.put(todoList.update\_a\_task)

.delete(todoList.delete\_a\_task);

};

Với mỗi phương thức khác nhau, ta gọi các hàm tương ứng được lập trình trong file todoListControllers.

1. **TodoListControllers**

Trong todolistControllers, ta sẽ lập trình 5 hàm khác nhau là: list\_all\_tasks, create\_a\_task, read\_a\_task, update\_a\_task, delete\_a\_task. Và từng hàm sẽ được export để các routes có thể sử dụng chúng

Mỗi hàm này sử dụng các hàm khác nhau trong module Mongoose như find, findById, findOneAndUpdate, save và remove để tương tác với dữ liệu được lưu trên database mlab.

'use strict';

var mongoose = require('mongoose'),

Task = mongoose.model('Tasks');

exports.list\_all\_tasks = function(req, res) {

Task.find({}).sort({Created\_date: 1}).exec(function(err,task) {

if (err)

res.send(err);

res.json(task);

}); };

exports.create\_a\_task = function(req, res) {

var new\_task = new Task(req.body);

new\_task.save(function(err, task) {

if (err)

res.send(err);

res.json(task);

});

};

exports.read\_a\_task = function(req, res) {

Task.findById(req.params.taskId, function(err, task) {

if (err)

res.send(err);

res.json(task);

});

};

exports.update\_a\_task = function(req, res) {

Task.findOneAndUpdate({\_id: req.params.taskId}, req.body, {new: true}, function(err, task) {

if (err)

res.send(err);

res.json(task);

});

};

exports.delete\_a\_task = function(req, res) {

Task.remove({

\_id: req.params.taskId

}, function(err, task) {

if (err)

res.send(err);

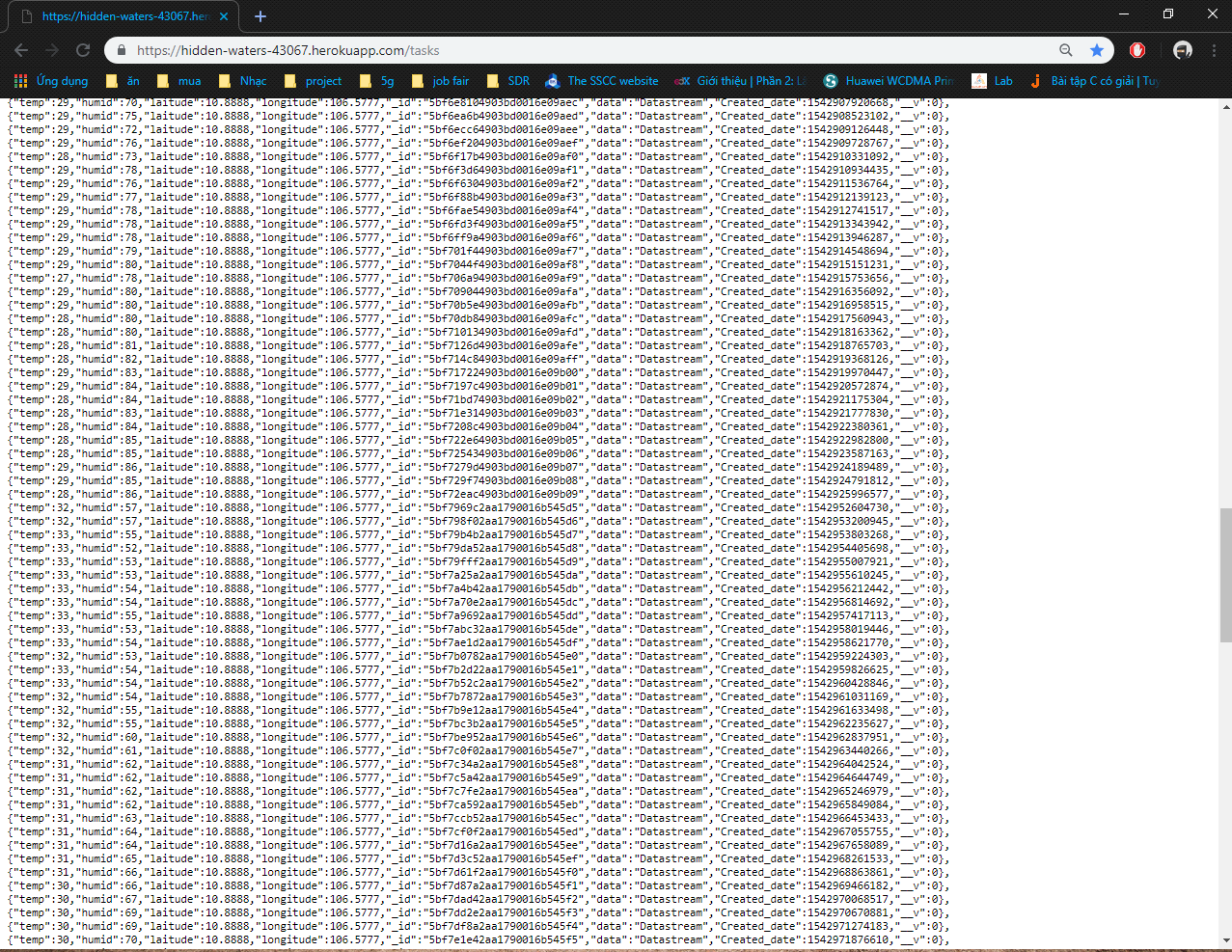
res.json({ message: 'Task successfully deleted' });

});

};

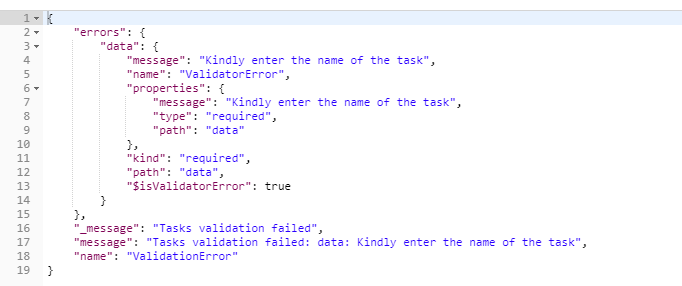
**Kiểm tra kết quả:**

* Khi truy cập tới đường dẫn “https://hidden-waters-43067.herokuapp.com/tasks“ Hệ thống sẽ thực thi phương thức GET và liệt kê các dữ liệu hiện có trong DataBase:



Hình 2: Dữ liệu thể hiện khi truy cập route /tasks

* Tiến hành mô phỏng POST dữ liệu lên Server sử dụng postman :



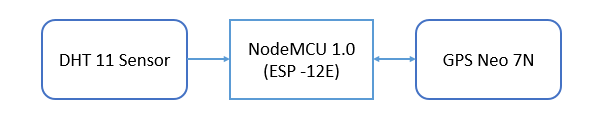
Hình 3: Phản hồi của server khi post sai schema

1. **THIẾT KẾ KHỐI THU THẬP DỮ LIỆU**
2. **PHẦN CỨNG**

Thiết kế phần cứng là một trong ba vấn đề trọng tâm của đề tài lần này, việc lựa chọn các module cũng như phương pháp thu thập dữ liệu ảnh hưởng trực tiếp đến kết quả cũng như độ chính xác của cả hệ thống. Các yêu cầu thiết kế phần cứng được đưa ra như sau:

* Thiết bị đơn giản, nhỏ gọn, phù hợp với phạm vi của đề tài.
* Có khả năng truyền và nhận thông tin
* Cảm biến tích hợp thu thập dữ liệu nhiệt độ và độ ẩm

Từ các yêu cầu thiết kế phần cứng, nhóm đi đến thiết kế như sơ đồ khối sau:

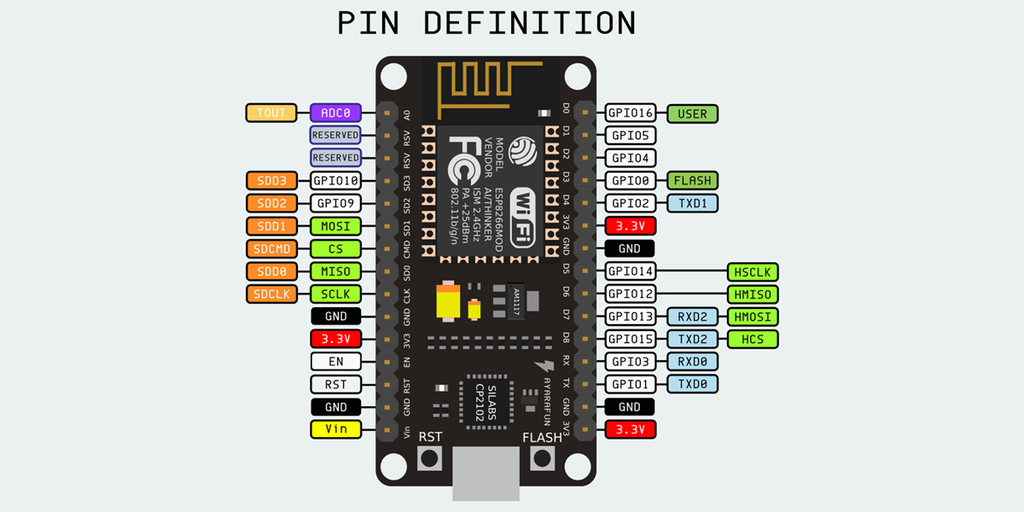


Hình 4: Sơ đồ khổi tổng quát của hệ thống

1. Module NodeMCU 1.0

NodeMCU là kít phát triển dựa trên nền chip Wifi SoC ESP8266 với thiết kế dễ dàng sửa dụng vì tích hợp sẵn mạch nạp sử dụng chíp CP2102 trên borad. Bên trong ESP8266 có sẵn một lõi vi sử lý vì thế ta có thể trực tiếp lập trình cho ESP8266 mà không cần thêm bất kì vi xử lý hỗ trợ. Hiện tại có hai ngôn ngữ có thể lập trình cho ESP8266, sử dụng trực tiếp phần mềm IDE của Arduino để lập trình với bộ thư viện riêng hoặc sử dụng phần mềm node MCU. Trong đề tài này, nhóm thực hiện bằng ArduinoIDE.

Nhóm lựa chọn sử dụng module NodeMCU vì có thể dễ dàng kết nối Wifi mà không cần sử dụng thêm module ngoài như trường hợp các kit khác. Đồng thời việc lập trình, giao tiếp với các module khác cũng như nạp code tương tự như Arduino Uno giúp cho quá trình thực hiện tinh giản và tốn ít thời gian hơn.



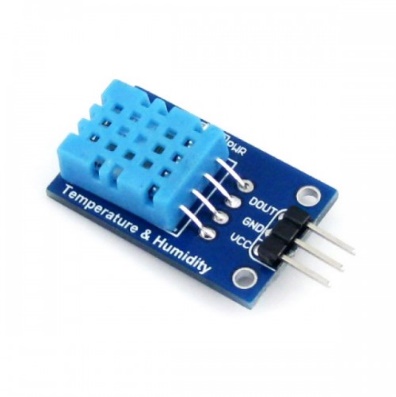
Hình 5: Module NodeMCU và Layout

1. Module DHT 11

DHT11 Là cảm biến nhiệt độ, độ ẩm rất thông dụng hiện nay vì chi phí rẻ và rất dễ lấy dữ liệu thông qua giao tiếp 1-wire ( giao tiếp digital 1-wire truyền dữ liệu duy nhất). Cảm biến được tích hợp bộ tiền xử lý tín hiệu giúp dữ liệu nhận về được chính xác mà không cần phải qua bất kỳ tính toán nào. Với yêu cầu về sai số không quá cao ( ± 1 đơn vị nhiệt độ và độ ẩm), thay vì sử dụng 2 cảm biến riêng biệt cho từng tác vụ ta có thể sử dụng module DHT11 để giảm độ rườm rà của phần cứng.

Đặc điểm:

* Điện áp hoạt động : 3V - 5V (DC)
* Dải độ ẩm hoạt động : 20% - 90% RH, sai số ±5%RH
* Dải nhiệt độ hoạt động : 0°C ~ 50°C, sai số ±2°C
* Tần số lấy mẫu tối đa: 1 Hz
* Khoảng cách truyển tối đa: 20m



Hình 6: Module DHT 11

1. Module GSP NEO 7

Module NEO-7 GPS là module định vị toàn cầu sử dụng hệ thống vệ tinh GPS của Mỹ. Module GPS NEO-7cho tốc độ xác định vị trí nhanh và chính xác, có nhiều mức năng lượng hoạt động, phù hợp với các ứng dụng yêu cầu tiêu thụ ít năng lượng.

Đặc điểm kĩ thuật:

* Điện áp hoạt động: 3.3-5.5V
* Dòng hoạt động bình thường: 50 mA
* Dòng hoạt động ở trạng thái tiết kiệm: 30 mA
* Giao tiếp UART/TTL
* Baud rate: 1200, 2400, 4800, 19200, 38400, 9600 (mặc định), 57600, 115200,…
* Kích cỡ module : 39\*25.5mm



Hình 7: Module Neo-7M

1. **PHẦN MỀM**

**Đặc tả phần mềm:**

* Đọc giá trị nhiệt độ và độ ẩm từ cảm biến
* Đọc giá trị kinh độ và vĩ độ từ module GPS
* Biên dịch các dữ liệu thu được theo đúng định dạng cài đặt sẵn của data server
* Giao tiếp, gửi dữ liệu tới data server bằng giao thức HTTP
* Ghi nhận và hiển thị phản hồi từ Data Server (khi cần debug các vấn đề giao tiếp)

Việc lập trình code được thực hiện bằng ARDUINO IDE. Với các module cho trước, các thư viện hỗ trợ được khai báo như sau:

#include <ESP8266HTTPClient.h>

#include <ESP8266WiFi.h>

#include <ArduinoJson.h>

#include <DHT.h>

#include <SoftwareSerial.h>

#include <TinyGPS.h

* Các thư viện ESP8266HTTPClient.h , ESP8266WiFi.h hỗ trợ việc kết nối Wifi và giao tiếp HTTP với DATA Server.
* Thư Viện ArduinoJson biên dịch các dữ liệu theo chuẩn JSON đồng bộ hoạt động của Dataserver.
* Thư viện DHT và TinyGPS hỗ trợ cho các module tương ứng DHT11 và NEO-7M.
* SoftwareSerial để khai báo các nhân Tx và Rx của kit sang các chân digital\_out

Tiếp theo, NodeMCU được Setup với các thư viện và kết nối với wifi trong hàm Void setup():

void setup() {

dht.begin();

Serial.begin(9600);

WiFi.begin("thanh luong", "nguyetque94"); // Wifi SSID, Wifi password

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.println("Waiting for connection”);

}

}

Sau đó, tất cả các quá trình thu thập dữ liệu, giao tiếp với server được hoạt động liên tục trong Void Loop()

if (WiFi.status() == WL\_CONNECTED) { //Check WiFi connection status

float flat, flon;

unsigned long age;

gps.f\_get\_position(&flat, &flon, &age);

int tempC = dht.readTemperature();;

int Mois = dht.readHumidity();;

String Temp = String(tempC);

String Moi = String(Mois);

String Long = String(flon);

String Lat = String(flat);

Đầu tiên, (WiFi.status() == WL\_CONNECTED) giúp ta kiểm tra trạng thái kết nối với wifi, chỉ khi nào có kết nối với wifi, việc đọc và gửi dữ liệu mới được thực thi

gps.f\_get\_position(&flat, &flon, &age) đây là hàm lấy các giá trị kinh độ, vĩ độ, và số lượng vệ tinh mà module neo quan sát được gán vào các biến được tạo sẵn là flat, flon và ages

2 biến TempC và Mois đùng để lưu các giá trị nhiệt độ và độ ẩm thu được từ cảm biến DHT bằng các hàm tương ứng lần lượt là dht.readTemperature(); và dht.readHumidity();

Ở bước tiếp theo của phần code trên, ta chuyển đổi các giá trị được lưu trong các biến TempC, Mois, flon, flat thành các giá trị kiểu String để các hàm trong thư viện ArduinoJson có thể xử lý.

Tiếp theo, ta sử dụng các hàm trong thư viện ArduinoJson mã hóa các dữ liệu dưới dạng packetJson và gửi dữ liệu lên server bằng giao thức HTTP. Quá trình này được thực thi bằng đoạn code sau:

StaticJsonBuffer<300> JSONbuffer; //Declaring static JSON buffer (1)

JsonObject& JSONencoder = JSONbuffer.createObject(); (2)

JSONencoder["data"] = "Datastream"; (3)

JSONencoder["temp"] = Temp; (4)

JSONencoder["humid"] = Moi; (5)

JSONencoder["longitude"] = Long; (6)

JSONencoder["laitude"] = Lat; (7)

char JSONmessageBuffer[300]; (8)

JSONencoder.prettyPrintTo(JSONmessageBuffer, sizeof(JSONmessageBuffer)); (9)

Serial.println(JSONmessageBuffer); (10)

HTTPClient http; (11)

http.begin("http://hidden-waters-43067.herokuapp.com/tasks "); (12)

http.addHeader("Content-Type", "application/json”); (13)

int httpCode = http.POST(JSONmessageBuffer); (14)

String payload = http.getString(); (15)

Serial.println(httpCode); //Print HTTP return code (16)

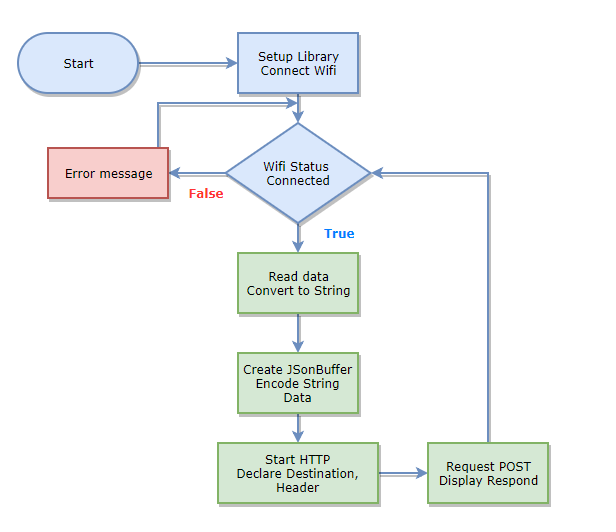
Serial.println(payload); //Print request response payload (17)

http.end(); //Close connection (18)

* (1) và (2) dùng để khởi tạo object Json với độ lớn lưu trữ là 300 char
* (3) -> (7) gán các giá trị từ các biến đã chuyển thành dạng string vào các biến của Data Server theo đúng cú pháp đồng bộ với Server theo định dạng JSON
* (8),(9),(10) Lưu các biến vừa mã hóa ở bước trên vào object buffer đã tạo ở bước (1) và (2), đồng thời xuất ra màn hình để kiểm tra độ chính xác
* (11): khởi tạo object HTTP thuộc class HTTPClient để gửi dữ liệu đến Server
* (12),(13) Khai báo đường dẫn của Server và các thuộc tính content-type Header
* (14) yêu cầu thực hiện thức POST tới server với các dữ liệu đã được lưu vào buffer. Luồng dữ liệu post này cũng được lưu vào biến httpCode.
* (15) Lưu phản hồi trạng thái của Server vào biến payLoad
* (16)->(18): Xuất ra màn hình luồng dữ liệu được yêu cầu post tới server, phản hồi từ server và kết thúc kết nối HTTP

Trong khi đó, đối với trường hợp kết nối wifi xác định là (WiFi.status() ~= WL\_CONNECTED), in ra màn hình thông báo lỗi Serial.println("Error in WiFi connection"); và thực hiện lại vòng lặp với delay được cài đặt dưới cùng.

Nhìn chung, giải thuật của khối thu thập dữ liệu có thể được biểu diễn bằng lưu đồ sau:

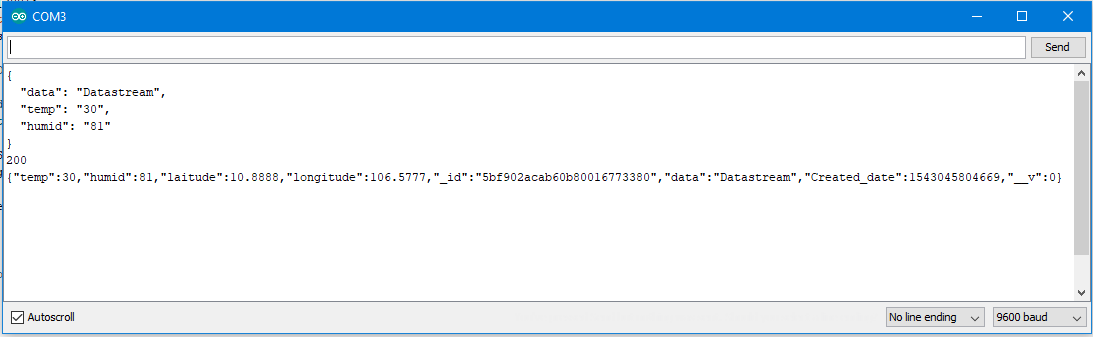


Hình 8: Lưu đồ giải thuật của khối thu thập dữ liệu

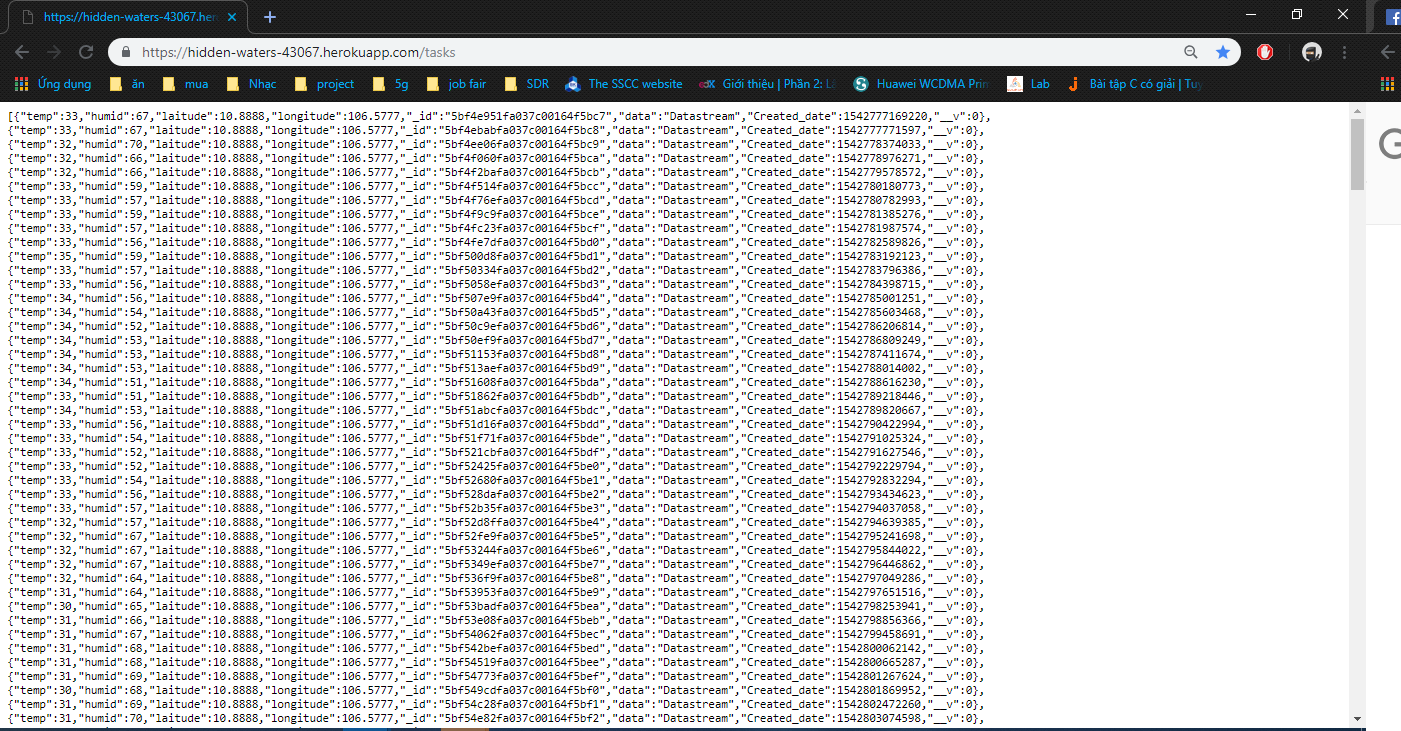
***Kết quả hoạt động :***

HÌnh sau thể hiện dữ liệu được gửi đi và phải hồi từ server: Dữ liệu phản hồi từ server bao gồm

* HTTP status code. (Ở đây status code = 200 )
* Payload đã được lưu trên Data Server



Hình 9: hoạt động của khối và phản hồi của Server

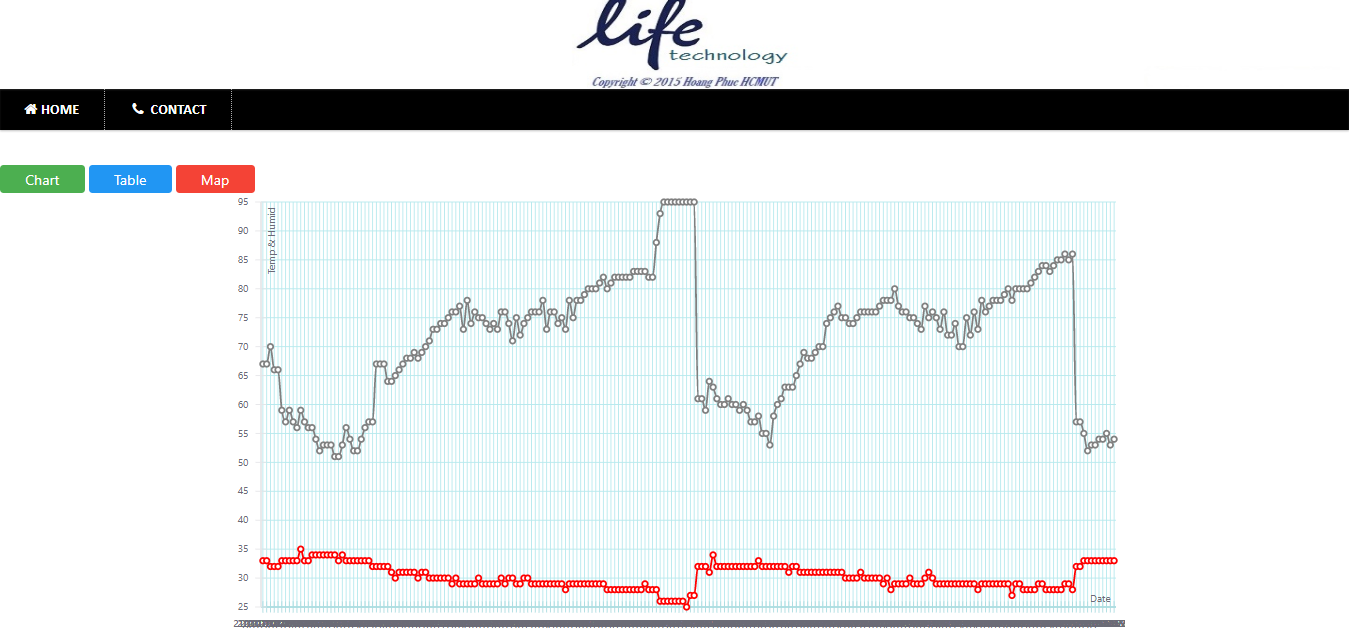


Hình 10: Các tập dữ liệu được lưu trên Data Server

1. **THIẾT KẾ FRONT-END**

**Front-end là gì?**

Front end: bao gồm quá trình xử lý các dữ liệu request nhận từ client và response gửi cho client. Cụ thể là các request của user sẽ được biên dịch thành gói dữ liệu thế nào trên client trước khi gửi đến server, và các gói dữ liệu response nhận được từ server sẽ được browser trình diễn ra trang web trên máy client thế nào. Phần front-end của một trang web là phần tương tác với người dùng. Tất cả mọi thứ bạn nhìn thấy khi điều hướng trên Internet, từ các font chữ, màu sắc cho tới các menu xổ xuống và các thanh trượt, là một sự kết hợp của HTML, CSS, và JavaScript được điều khiển bởi trình duyệt máy tính của bạn.



Hình 11: Front-end

Để phát triển front-end cho trang web này, chúng tôi sử dụng ngôn ngữ lập trình JavaScript cùng thư viện React.js.

**Lý do sử dụng ngôn ngữ lập trình JavaScript:**

JavaScript giúp tăng tính tương tác trên website. Script này chạy trên các trình duyệt của người dùng thay vì trên server và thường sử dụng thư viện của bên thứ 3 nên có thể tăng thêm chức năng cho website mà không phải code lại từ đầu.

**Lý do sử dụng thư viện React.js:**

React.js là một thư viện JavaScript đang nổi lên trong những năm gần đây với xu hướng Single Page Application. Trong khi những framework khác cố gắng hướng đến một mô hình MVC (Model – View- Controller) hoàn thiện thì React nổi bật với sự đơn giản và dễ dàng phối hợp với những thư viện JavaScript khác. Nếu như AngularJS là một framework cho phép nhúng code JavaScript trong code HTML thông qua các attribute như ng-model, ng-repeat... thì **React** là thư viện cho phép nhúng code HTML trong code Javascript nhờ vào JSX, bạn có thể dễ dàng lồng các đoạn HTML vào trong JS. Tích hợp giữa JavaScript và HTML vào trong JSX làm cho các component dễ hiểu hơn.

Đối với phần Front-end, tôi xin được chia thành 4 phần:

* Biểu đồ dữ liệu (Data Chart)
* Bảng dữ liệu (Data Table)
* Bản đồ (Google Map)
* Menu và Button (dùng HTML và CSS)

1. **BIỂU ĐỒ DỮ LIỆU (DATA CHART)**

Ở phần thiết kế front-end cho biểu đồ dữ liệu, chúng tôi sử dụng **React-vis** làm thư viện chính. **React-vis** là thư viện hỗ trợ biểu thị dữ liệu. Các components của **React-vis** được thiết kế để hoạt động như các components của React, chúng đều có properties, children và callback. **React-vis** giúp tạo biểu đồ phức tạp với số lượng code ít và sẵn có, tuy nhiên vẫn có thể tùy chỉnh mọi khía cạnh của biểu đồ.

Trước hết, cần cài đặt react-vis thông qua command line:

npm install react-vis

Dưới đây là code trong **src\components\chart.js** từ project của chúng tôi:

import React from 'react';

import {XYPlot, XAxis, YAxis, VerticalGridLines, Hint, HorizontalGridLines, LineMarkSeries, DiscreteColorLegend} from 'react-vis';

import moment from 'moment';

Trước tiên, tôi sử dụng câu lệnh **import** để nhập module **React** từ thư viện **react**, nhập module **XYPlot, Xasis, …DiscreteColorLegend** từ thư viện **react-vis**, ngoài ra còn có module **moment** từ thư viện **moment**. Chức năng của các module này sẽ được tôi đề cập sau.

class Chart extends React.Component {

constructor(props) {

super(props);

this.state = {

value: null,

items: [

{title: 'Temp', color: 'gray'},

{title: 'Humid', color: 'red'}],

};

}

**Component** cho phép chúng ta chia nhỏ các thành phần UI độc lập mục đích để dễ quản lý và tái sử dụng nó. Trong **React**, chúng ta điều khiển Component bằng cách sử dụng props và state.

**Props** là properties của một component. Các component nhận props từ component cha. Bạn không được thay đổi giá trị của props trong các component này mà chỉ được phép đọc giá trị. Trong React thì dữ liệu sẽ đi theo một chiều, có nghĩa là từ component cha => các component con.

**State** thì hoạt động khác với props. State là dữ liệu nội bộ của một component, trong khi props là dữ liệu được truyền cho Component. Chính vì vậy chúng ta hoàn toàn có thể thay đổi state. Tuy vậy, hãy nhớ rằng đừng bao giờ thay đổi trực tiếp biến this.state. Thay vào đó hãy dùng hàm setState để cập nhật giá trị. Sở dĩ chúng ta cần dùng hàm này là do nó sẽ kích hoạt việc render lại component và tất cả component con nằm trong nó, còn thay đổi this.state thì không. Còn một vấn đề nữa, đó là setState chạy bất đồng bộ, vậy nên nếu bạn tiến hành đọc ra giá trị state ngay sau khi setState thì chưa chắc giá trị sẽ được update lên mới nhất đâu.

**Items** khai báo trong this.state là 2 đường chú thích trong biểu đồ với title là **Temp, Humid** cùng với attribute color.

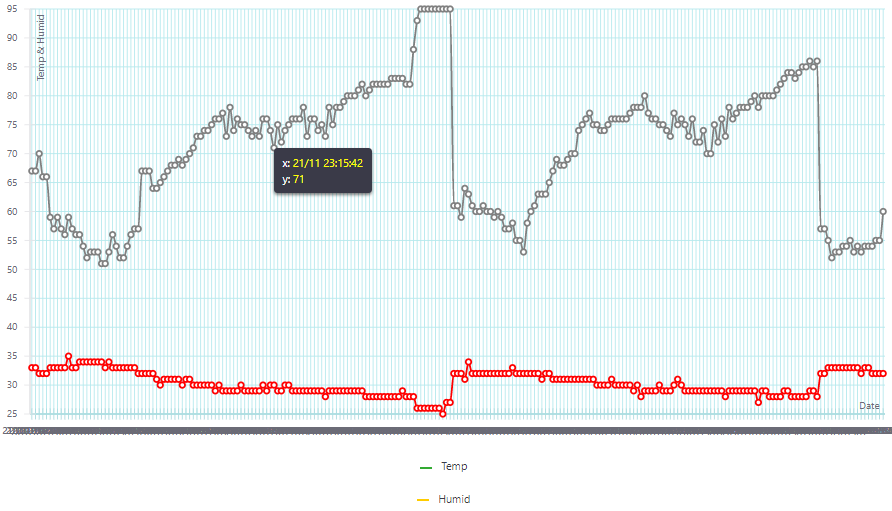
\_forgetValue = () => {this.setState({value: null});

};

\_rememberValue = value => {this.setState({value});

};

Function \_rememberValue được sử dụng để đọc giá trị Nhiệt độ/ Độ ẩm cùng với thời gian khi người dùng rê chuột vào điểm trên chart. Ví dụ như hình dưới đây, ngày 21/11 lúc 23:15 thì độ ẩm là 71.



Hình 12 Hiển thị giá trị khi trỏ chuột lên điểm

Function \_forgetValue dùng để ngưng đọc giá trị Nhiệt độ/ Độ ẩm cùng thời gian khi người dùng rời chuột khỏi điểm đó.

<XYPlot xType="ordinal" width={1000} height={500}>

<HorizontalGridLines style={{stroke: '#B7E9ED'}} />

<VerticalGridLines style={{stroke: '#B7E9ED'}} />

Component XYPlot gọi ra chart với thuộc tính width là bề ngang và height là bề dọc, component HorizontalGridLines và VerticalGridLines gọi ra các đường kẻ dọc và ngang của chart.

<XAxis

title="Date"

style={{

line: {stroke: '#ADDDE1'},

ticks: {stroke: '#ADDDE1'},

text: {stroke: 'none', fill: '#6b6b76', fontWeight: 600}}}/>

<YAxis title="Temp & Humid" />

<DiscreteColorLegend height={200} width={300} items={items}/>

Component XAxis, YAxis với các thuộc tính để định dạng trục tung và trục hoành. Component DiscreteColorLegend là đường chú thích với 2 items đã được khai báo trong this.state.

<LineMarkSeries

onValueMouseOver={this.\_rememberValue}

onValueMouseOut={this.\_forgetValue}

data={this.props.data.map( row=> {return {x: moment(row.Created\_date).format("DD/MM HH:mm:ss"), y: row.temp}})}

fill={'white'}

size={3}

style={{stroke: 'red', strokeWidth: 2}}

curve={'curveMonotoneX'}/>

<LineMarkSeries

onValueMouseOver={this.\_rememberValue}

onValueMouseOut={this.\_forgetValue}

data={this.props.data.map( row=> {return {x: moment(row.Created\_date).format("DD/MM HH:mm:ss"), y: row.humid}})}

fill={'white'}

size={3}

style={{stroke: 'gray', strokeWidth: 2}}

curve={'curveMonotoneX'}/>

LineMarkSeries xuất ra 2 đường Nhiệt độ và Độ ẩm với dữ liệu được lấy từ server. Module moment lấy từ thư viện moment.js giúp định dạng ngày tháng năm và giờ theo ý muốn. Cài đặt thư viện này thông qua command line:

npm install moment

onValueMouseOver và onValueMouseOut sẽ kích hoạt hàm \_rememberValue và \_forgetValue ở trên.

1. **BẢNG DỮ LIỆU (DATA TABLE)**

import React from 'react';

import moment from 'moment';

Tương tự, dùng câu lệnh **import** để nhập module **React**, **moment** từ 2 thư viện **react** và **moment**.

class DataTable extends React.Component {

render() {

const response = this.props.response;

return (

<center>

<div id = "table" style={{'overflow-x':'auto', 'display':"none", 'max-width':"600"}}>

{/\* overflow-x: Tự động co giãn chart,

display none: làm chart mặc định ẩn đi trước khi người dùng toggle bằng cách ấn vào button "Table",

max-width: bề rộng tối đa của chart \*/}

<table class="w3-table-all w3-small">

{/\* Class w3 là css từ src\css\W3.css \*/}

<thead>

<tr class="w3-red">

<th>Date</th>

<th>Temperature</th>

<th>Humid</th>

<th>Latitude</th>

<th>Longitude</th>

</tr>

</thead>

{response {/\* Kết quả từ server trả về cho client \*/

? <tbody>

{response.map((row,i) =>

<tr class="w3-hover-black">

<td>{moment(row.Created\_date).format('DD/MM/YYYY HH:mm')}</td>

<td>{row.temp}</td>

<td>{row.humid}</td>

<td>{row.laitude}</td>

<td>{row.longitude}</td>

</tr>

)}

</tbody>

: <tbody></tbody>

}

</table>

</div>

</center>

);

}

}

Kết quả:

Hình 13: Bảng dữ liệur

1. **BẢN ĐỒ (GOOGLE MAP)**

import React, {Component} from 'react';

import {withGoogleMap, GoogleMap, Marker, InfoBox} from 'react-google-maps';

Dùng **import** để nhập module **các module** từ thư viện **react** và **react-google-maps.**

render() {

var posList = this.props.response;

var center;

if (!posList) {

// posList = false => tạo array, set center là điểm mặc định

posList = [];

center = {lat: 10.800388, lng: 106.576083}

}

else {

// Chọn điểm đầu tiên làm center

const firstPos = posList[0];

center = {lat: parseFloat(firstPos.longitude), lng: parseFloat(firstPos.laitude)}

}

Nếu không có dữ liệu GPS, chọn center làm điểm mặc định trên bản đồ.

<GoogleMap

defaultCenter = {center}

defaultZoom = {16}>

{posList.map((row,i) => {

return (

<Marker position={{lat: parseFloat(row.longitude), lng: parseFloat(row.laitude)}} />

)})}

</GoogleMap>

Module **GoogleMap** với điểm center như khai báo và zoom mặc định là 16. **Marker** xuất hiện tại tọa độ lat và long như trên.

Ngoài ra, để sử dụng database map của Google, cần phải có API key. API key này được gán trong index.html

<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBUm2X9kYwrcRfMsS-F1KsXyDSO7wYBvjQ"></script>

1. **MENU VÀ BUTTON**

<div id = 'header'><img src = "../Pictures/Life technologies.jpg"

height="120" width="100%">

<font face="Calibri">

<div class="main\_menu">

<ul>

<li><a class="active" href="#"><i class="fa fa-home"></i> HOME</a></li>

<li><a class="active" href="contact.html"><i class="fa fa-fw fa-phone"></i> CONTACT</a></li>

<div class="clear"></div>

</div>

</font>

</div>

<button class="btn success" onclick="showchart()">Chart</button>

<button class="btn info" onclick="showtable()">Table</button>

<button class="btn danger" onclick="showmap()">Map</button>

Các class **btn success/ info/ danger** được định dạng trong **src\css\Button.css.**

Khi nhấn các button này, hàm showchart(), showtable() hay showmap() sẽ kích hoạt.

<script>

function showchart() {

var x = document.getElementById("chart");

var y = document.getElementById("table")

var z = document.getElementById("googlemap")

if (x.style.display === "none") {

x.style.display = "block";

y.style.display = "none";

z.style.display = "none";

}

else {

x.style.display = "none";

}

}

Phương thức getElementById() là một trong những phương thức cơ bản của DOM (Document Object Model), truy xuất các phần tử xác định của một tài liệu HTML và trả về tham chiếu đến phần tử đó. Phần tử phải có thuộc tính id để được truy xuất bằng phương thức này.

Dùng lần lượt các biến **x, y** và **z** để chứa phần tử **chart, table** và **googlemap** sau khi sử dụng document.getElementById() để truy xuất phần tử.

Nếu **display** của **x** là **none**, nghĩa là chart chưa hiển thị trên web, thì khi nhấn button, **display** của chart chuyển thành **block** (khối), tức là chart hiển thị. Còn **table** và **googlemap** sẽ chuyển thành **none**, tức là ẩn đi.

**Else** thì ngược lại.

Tương tự với 2 button còn lại ứng với 2 hàm showtable() và showmap().

1. **MACHINE LEARNING**
2. **Tìm hiểu bài toán**

Với bài toán dự báo như thế này, tôi quy nó về Time Series Analysis để có thể dễ hiểu và dễ áp dụng hơn. Bạn có thể tìm thấy một số bài toán thuộc thể loại này, như phân tích tài chính, tỉ số chứng khoán, hay áp dụng nó để phân tích tỉ giá Bitcoin để đầu tư đúng chỗ, đúng thời điểm. Nghĩ ra cái này sớm hơn giờ có phải giàu rồi không (yaoming) Với bài toán này, bạn có thể thử nghiệm trên các thông số thời tiết

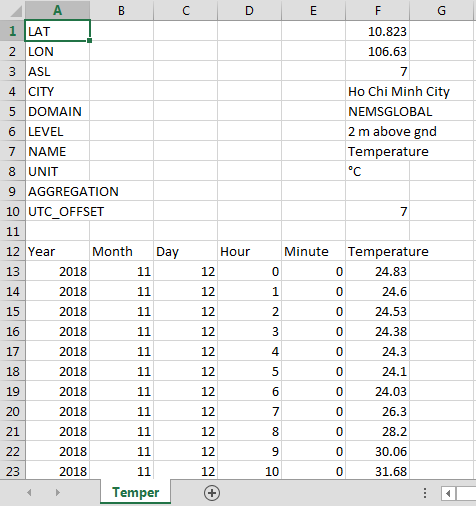
* nhiệt độ (°C)
* độ ẩm (%)

input: thông số thời tiết của ngày hôm trước output: thông số thời tiết của ngày hôm sau.

Sử dụng thư viện tensorflow để hỗ trợ quá trình training, Về cơ bản thì tensorflow đã được tích hợp rất nhiều các thuật toán khác nhau, dễ dàng sử dụng, và giúp giảm thời gian xây dựng các hệ thống deep learning. Đồng thời kết hợp với pandas và numpy để phân tích, và xử lý cấu trúc data, và matplotlib dùng để về đồ thị.

1. **Chuẩn bị dữ liệu**

Chúng tôi đã thu thập dữ liệu và lưu trữ thành file .csv. Tuy nghiên trong quá trình lấy dữ liệu do khó khăn trong việc duy trì nguồn cấp cũng như động bộ với server nên chúng tôi đã không thể thu thập dữ liệu đầy đủ như dự kiến. Chỉ có dữ liệu cho khoảng 15 ngày.



Hình 14: File dữ liệu .csv

1. **Xử lí dữ liệu**

Cũng giống như các bài toán khác, Tôi tiếp tục sử dụng thư viện tensorflow để hỗ trợ quá trình training, Về cơ bản thì tensorflow đã được tích hợp rất nhiều các thuật toán khác nhau, dễ dàng sử dụng, và giúp giảm thời gian xây dựng các hệ thống deep learning. Đồng thời kết hợp với pandas và numpy để phân tích, và xử lý cấu trúc data, và matplotlib dùng để về đồ thị. Việc vẽ đồ thị rất quan trọng đối với các bài toán thuộc dạng Time Series Analysis như thế này. Vì dĩ nhiên việc đoán trước không thể trả về kết quả chính xác 100% được, Kết quả sẽ là tương đối và có thể có một chút sai số không đáng kể. Vì thế việc vẽ đồ thị sẽ giúp bạn dễ dàng so sánh giữa kết quả dự đoán và thực tế.

Đầu tiên import các thư viện sẽ được dùng

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import tensorflow as tf

Load training data

from numpy.core.multiarray import ndarray

from pandas import read\_csv

num\_test = 25

file\_path = "C:/Users/Zizou/Downloads/Temper.csv" #type

data = pd.read\_csv(file\_path, delimiter=',', header=11, skipinitialspace=True)

data = data.head(num\_test)

**header** sẽ là vị trí **header** trong file dữ liệu. Như bạn trong file csv bên trên Header của dữ liệu ở dòng số 12, nói các khách nếu xét theo 1 array thì index=11. Nếu file dữ liệu của bạn không có header thì có thể đổi thành **header=None.**

Như vậy ta đã có được file dữ liệu training. bây giờ bạn chỉ cần gọi data['Temperature'].

temperature = np.array(data, dtype = float)

Bạn có 1 chuỗi các giá trị array, các số tiếp theo sẽ phụ thuộc vào các số trước đó, cũng như vậy chúng ta sẽ có các giá trị thời tiết tại thời điểm trước đó sẽ là X và giá trị thời tiết tại thời điểm trên sau 1 giờ sẽ là Y.

num\_periods = num\_test

f\_horizon = 1

x\_train = temperature[:(len(temperature) - (num\_periods \* 2))]

x\_batches = x\_train.reshape(-1, num\_periods, 1)

y\_train = temperature[1:(len(temperature) - (num\_periods \* 2)) + f\_horizon]

y\_batches = y\_train.reshape(-1, num\_periods, 1)

và dữ liệu test

def test\_data(series, forecast, num):

testX = temperature[-(num + forecast):][:num].reshape(-1, num\_periods, 1)

testY = temperature[-(num):].reshape(-1, num\_periods, 1)

return testX, testY

1. **Traning models**

Do đặc thù của bài toán này là dùng các thông tin trong quá khứ, để dự đoán tương lai, nên thuật toán ở đây tôi lựa chọn là **Recurrent Neeural Network** (RNN) hay trong tiêng Việt chúng ta gọi nó là Mạng Nơ-ron hồi quy



Ý tưởng chính của RNN (Recurrent Neural Network) là sử dụng chuỗi các thông tin. Ví dụ, nếu muốn đoán từ tiếp theo có thể xuất hiện trong một câu thì ta cũng cần biết các từ trước đó xuất hiện lần lượt thế nào. RNN được gọi là hồi quy (Recurrent) bởi lẽ chúng thực hiện cùng một tác vụ cho tất cả các phần tử của một chuỗi với đầu ra phụ thuộc vào cả các phép tính trước đó. Nói cách khác, RNN có khả năng nhớ các thông tin được tính toán trước đó. Trong nhiều mạng neural truyền thống khác, dữ liệu đầu vào và đầu ra hoàn toàn độc lập với nhau, tức là chúng không có liên kết thành chuỗi. Do đó khi áp dụng vào bài toán dự báo thời tiết của tôi sẽ rất khó để đưa ra kết quả dự đoán.

* *x(t)*​ là đầu vào tại bước *t*. Ví dụ, *x*1​ là một vec-tơ one-hot tương ứng với từ thứ 2 của câu.
* *s(t*​) là trạng thái ẩn tại bước *t*. Nó chính là **bộ nhớ** của mạng.  *st*​ được tính toán dựa trên cả các trạng thái ẩn phía trước và đầu vào tại bước đó:*st*​=*f*(*Uxt*​+*Wst*−1​). Hàm  *f* thường là một hàm phi tuyến tính như [tang hyperbolic (tanh)](https://vi.wikipedia.org/wiki/H%C3%A0m_hypebolic) hay [ReLu](https://en.wikipedia.org/wiki/Rectifier_(neural_networks)). Để làm phép toán cho phần tử ẩn đầu tiên ta cần khởi tạo thêm *s*−1​, thường giá trị khởi tạo được gắn bằng 0.
* *o(t)*​ là đầu ra tại bước *t*. Ví dụ, ta muốn dự đoán từ tiếp theo có thể xuất hiện trong câu thì*o(t)*​ chính là một vec-tơ xác xuất các từ trong danh sách từ vựng của ta: *o(t)*​=softmax(*Vst*​)

Ở đây khi khai báo và sử dụng mạng neural, tôi sử dụng mô hình RNN truyền thống.

# Training model

tf.reset\_default\_graph()

rnn\_size = 100

learning\_rate = 0.001

X = tf.placeholder(tf.float32, [None, num\_periods, 1])

Y = tf.placeholder(tf.float32, [None, num\_periods, 1])

rnn\_cells = tf.contrib.rnn.BasicRNNCell(num\_units=rnn\_size, activation=tf.nn.relu)

rnn\_output, states = tf.nn.dynamic\_rnn(rnn\_cells, X, dtype=tf.float32)

output = tf.reshape(rnn\_output, [-1, rnn\_size])

logit = tf.layers.dense(output, 1, name="softmax")

outputs = tf.reshape(logit, [-1, num\_periods, 1])

print(logit)

loss = tf.reduce\_sum(tf.square(outputs - Y))

accuracy = tf.reduce\_mean(tf.cast(tf.equal(tf.argmax(logit, 1), tf.cast(Y, tf.int64)), tf.float32))

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)

train\_step = optimizer.minimize(loss)

epochs = 1000

sess = tf.Session()

init = tf.global\_variables\_initializer()

sess.run(init)

dropout\_keep\_prob = tf.placeholder(tf.float32, name="dropout\_keep\_prob")

for epoch in range(epochs):

train\_dict = {X: x\_batches, Y: y\_batches, dropout\_keep\_prob: 1}

sess.run(train\_step, feed\_dict=train\_dict)

Lưu lại model để sử dụng

saver = tf.train.Saver()

save\_path = saver.save(sess, "models/model.ckpt")

Khi test dữ liệu, bạn cần restore lại models mà mình vừa training lúc nãy để sử dụng

with tf.Session() as sess:

# Restore variables from disk.

saver.restore(sess, "models/model.ckpt")

y\_pred = sess.run(outputs, feed\_dict={X: X\_test})

print(y\_pred)

1. **Kết quả sơ bộ**

Nếu chỉ nhìn vào dữ liệu đầu ra như thế kia tôi cũng không biết liệu kết quả dự đoán của chúng ta sẽ ra sao. Do đó để dễ dàng so sánh, việc vẽ đồ thị sẽ rất cần thiết để giúp chúng ta có cái nhìn chính xác để đánh giá mô hình dự đoán này.

plt.title("Compare Weather Forecast vs Actual", fontsize=14)

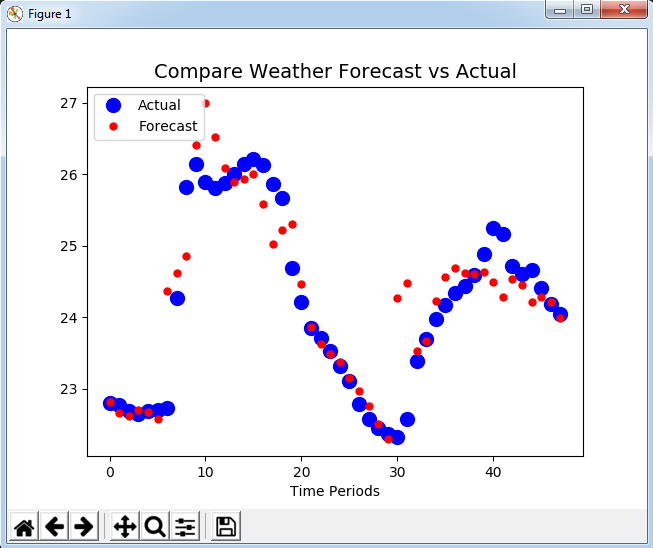
plt.plot(pd.Series(np.ravel(Y\_test)), "bo", markersize=10, label="Actual")

plt.plot(pd.Series(np.ravel(y\_pred)), "r.", markersize=10, label="Forecast")

plt.legend(loc="upper left")

plt.xlabel("Time Periods")

plt.show()



Hình 15: So sánh giữa giải thuật và thực tế

1. **KẾT LUẬN & HƯỚNG PHÁT TRIỂN:**

## KẾT LUẬN

Sau khi hoàn thành Đề tài này, nhóm chúng tôi đã đáp ứng được 3 nội dung chính: Tìm hiểu cách thức hoạt động cơ bản của một hệ thống dự báo thời tiết; tìm hiểu về vi xử lý NodeMCU, module GPS và cách thức hoạt động; tạo ra sản phẩm hoạt động tốt và đạt yêu cầu đề ra.

Đề tài có những ưu và khuyết điểm như sau:

Bảng 1: Ưu và khuyết điểm

|  |  |
| --- | --- |
| **Ưu điểm** | **Khuyết điểm** |
| Giá thành và chi phí thực hiện rẻ.  Giao diện web dễ sử dụng và thao tác. | Chưa áp dụng được Machine Learning vào hệ thống.  Giao diện web chưa thật sự đẹp và chuyên nghiệp, trang web chưa có nhiều function.  Người dùng phải reload trang web để cập nhật lại data chart khi có data mới. |

## HƯỚNG PHÁT TRIỂN

Do thời gian thực hiện và lượng kiến thức của mỗi thành viên trong nhóm là nhất định nên đề tài thực hiện xong chỉ đáp ứng được một phần nhỏ. Vì vậy, để đề tài thêm đa dạng thì có thể phát triển thêm:

* + Áp dụng Machine Learning vào hệ thống để đưa ra những dự báo về thời tiết trong tương lai.
  + Thêm nhiều tính năng hơn cho trang web với từng thành phần như: zoomable chart để có thể xem nhiều dữ liệu hơn trong 1 ngày, data filter table để tìm kiếm dữ liệu cụ thể,..
  + Phát triển một app trên smartphone để tích hợp nhiều tính năng thông minh khác ngoài hệ thống dự báo thời tiết, chẳng hạn: hệ thống tin nhắn SMS báo tới điện thoại của người dùng khi nhận biết khu vực người dùng đang đứng sắp có mưa, bão thông qua GPS, v..v..

Hy vọng với những hướng phát triển nêu trên, những người đi sau sẽ phát triển hơn nữa đề tài này, khắc phục những hạn chế còn tồn tại.

1. **TÀI LIỆU THAM KHẢO**

[1] “ESP8266 (NodeMCU) post request data to website”. [Online] Available at: <https://circuits4you.com/2018/03/10/esp8266-nodemcu-post-request-data-to-website/>

[2] Olatunde Garuba, “Build Node.js RESTful APIs in 10 Minutes”. [Online] Available at: <https://www.codementor.io/olatundegaruba/nodejs-restful-apis-in-10-minutes-q0sgsfhbd>

[3] “Machine Learning dự báo thời tiết”. [Online] Available at: <https://viblo.asia>

# [4] “[Meaning of an Epoch in Neural Networks Training](https://stackoverflow.com/questions/31155388/meaning-of-an-epoch-in-neural-networks-training)”. [Online] Available at: <https://stackoverflow.com>

[5] React-vis Documentation. [Online] Available at: https://uber.github.io/react-vis/documentation

[6] Shyianovska Nataliia, "Data visualization with react-vis". [Online] Available at: <https://medium.com/dailyjs/data-visualization-with-react-vis-bd2587fe1660>

[7] Trần Văn Trọng, "Reactjs - Bài 2 - Components, props, state". [Online] Available at: <https://viblo.asia/p/reactjs-bai-2-component-props-state-07LKXYLkZV4>