

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

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**🙞🕮🙜**



**BÁO CÁO**

**MÔN: ĐIỆN TOÁN ĐÁM MÂY**

**ĐỀ TÀI: XÂY DỰNG HỆ THỐNG DỰ BÁO NHIỆT ĐỘ SAU MỘT GIỜ DỰA TRÊN MÔ HÌNH HỒI QUY TUYẾN TÍNH ĐA BIẾN**

**GVHD: ThS. Huỳnh Nam**

**SVTH:**

**Huỳnh Anh Thế Vinh - 17110255**

**Nguyễn Văn Hà - 17110130**

**Đinh Ngọc Nhi - 17110200**

**Trần Văn Tuấn - 16110247**

**Võ Châu Nhuận Trường - 16110236**

**TP. Hồ Chí Minh, Tháng 7 năm 2020**

**ĐIỂM SỐ**

|  |  |  |  |
| --- | --- | --- | --- |
| **TIÊU CHÍ** | **NỘI DUNG** | **TRÌNH BÀY** | **TỔNG** |
| **ĐIỂM** |  |  |  |

NHẬN XÉT CỦA GIẢNG VIÊN

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**Giảng viên hướng dẫn**

(*ký và ghi họ tên*)

…………………….....

MỤC LỤC

[PHẦN MỞ ĐẦU 1](#_Toc7292)

[1. Lý do chọn đề tài 1](#_Toc30690)

[2. Mục đích nghiên cứu đế tài: 1](#_Toc10669)

[3. Phương pháp nghiên cứu: 1](#_Toc2377)

[PHẦN NỘI DUNG 2](#_Toc2220)

[1. Kế hoạch làm việc 2](#_Toc30797)

[2. Bảng đánh giá các thành viên và tỉ lệ đóng góp 2](#_Toc18110)

[3. Mô tả bài toán 3](#_Toc3939)

[4. Kiến trúc tổng quát của hệ thống 3](#_Toc13751)

[5. Quá trình xây dựng 3](#_Toc11617)

[a. Mô đun IoT 3](#_Toc1063)

[i. Mô tả thiết bị 3](#_Toc3446)

[ii. Mô hình triển khai 7](#_Toc13270)

[iii. Mã nguồn 8](#_Toc30784)

[iv. Cài đặt và vận hành 12](#_Toc18708)

[v. Kết quả 16](#_Toc28237)

[b. Mô đun AI 17](#_Toc8628)

[i. Mô tả phương pháp xử lý, xây dựng dữ liệu huấn luyện 17](#_Toc8123)

[ii. Mô hình huấn luyện 17](#_Toc6308)

[iii. Mã nguồn 17](#_Toc25055)

[iv. Kết quả 20](#_Toc896)

[v. Đánh giá mô hình 21](#_Toc6933)

[c. Mô đun Restful API 21](#_Toc16793)

[i. Mô tả 21](#_Toc7726)

[ii. Mô hình triển khai 21](#_Toc4160)

[iii. Mã nguồn 21](#_Toc31242)

[iv. Cài đặt và vận hành 22](#_Toc13909)

[v. Kết quả 25](#_Toc10094)

[d. Mô đun Restful API (AWS) 25](#_Toc26985)

[i. Cài đặt và vận hành 25](#_Toc3600)

[ii. Kết quả 29](#_Toc8450)

[e. Mô đun Mobile (React) 30](#_Toc25450)

[i. Mô tả 30](#_Toc28845)

[ii. Mã nguồn 30](#_Toc14557)

[iii. Cài đặt và vận hành 32](#_Toc2064)

[iv. Kết quả 34](#_Toc31780)

[6. Kết quả hệ thống 36](#_Toc12309)

[7. Ưu điểm và nhược điểm 36](#_Toc1065)

[PHẦN KẾT LUẬN 37](#_Toc5860)

[1. Những điều đã đạt 37](#_Toc21947)

[2. Những hạn chế 37](#_Toc2127)

[TÀI LIỆU THAM KHẢO 38](#_Toc26293)

DANH MỤC CÁC BẢNG

*[Bảng 1. Bảng kế hoạch làm việc 2](#_Toc24833)*

*[Bảng 2. Bảng đánh giá và phần trăm đóng góp 2](#_Toc23177)*

*[Bảng 3. Bảng giá danh sách thiết bị 4](#_Toc7978)*

*[Bảng 4. Thông số kĩ thuật NodeMCU Esp8266 5](#_Toc1607)*

*[Bảng 5. Chức năng các chân của NodeMCU Esp8266 7](#_Toc6111)*

DANH MỤC CÁC HÌNH

*[Hình 1. Kiến trúc tổng quát của hệ thống](#_Toc25535)* [3](#_Toc25535)

*[Hình 2. NodeMCU CP2102 ESP8266](#_Toc7198)* [4](#_Toc7198)

*[Hình 3. Các chân NodeMCU Esp8266](#_Toc2221)* [4](#_Toc2221)

*[Hình 4. Cảm biến DHT11](#_Toc23528)* [7](#_Toc23528)

*[Hình 5. Sơ đồ lắp đặt](#_Toc18866)* [8](#_Toc18866)

*[Bảng 6. Sơ đồ lắp đặt](#_Toc29520)* [8](#_Toc29520)

*[Hình 6. Database Firebase](#_Toc24879)* [14](#_Toc24879)

*[Hình 7. Setting Firebase](#_Toc19546)* [14](#_Toc19546)

*[Hình 8. New channel ThingSpeak](#_Toc19893)* [15](#_Toc19893)

*[Hình 9. WriteAPIKey ThingSpeak](#_Toc7550)* [16](#_Toc7550)

*[Hình 10. Kết quả nhiệt độ thu được từ Firebase](#_Toc7492)* [16](#_Toc7492)

*[Hình 11. Biểu đồ kết nhiệt độ và độ ẩm ThingSpeak](#_Toc20612)* [17](#_Toc20612)

*[Hình 12. Đồ thị ví dụ hồi quy đa biến](#_Toc24381)* [21](#_Toc24381)

*[Hình 13. Dịch vụ Heroku](#_Toc14616)* [23](#_Toc14616)

*[Hình 14. Ứng dụng React Native](#_Toc23336)* [35](#_Toc23336)

PHẦN MỞ ĐẦU

1. **Lý do chọn đề tài**

Nhằm củng cố kiến thức đã học, áp dụng vào thực tế những đề tài gắn liền với cuộc sống hằng ngày. Hướng tới việc sử dụng các kiến thức về PaaS và SaaS trong Cloud Computing nhằm triển khai hệ thống phần mềm tương tác hướng dịch vụ SOA theo công nghệ Restful. Nhóm chúng em đã quyết định chọn đề tài xây dựng gói thu thập dữ liệu nền dựa trên IoT ghi nhận nhiệt độ và độ ẩm theo thời gian, sau đó đưa dữ liệu lên FireBase và lưu trữ dạng Real Time Data.

1. **Mục đích nghiên cứu đế tài:**

Sử dụng các kiến thức đã học trong môn Cloud computing để xây dựng được một gói thu thập dữ liệu dựa trên IoT ghi nhận nhiệt độ và độ ẩm theo thời gian, sau đó đưa dữ liệu lên FireBase và lưu trữ dạng Real Time Data, có thể áp dụng được trong thực tế đời sống hiện nay. Sau đó xây dựng mô hình AI để dự đoán nhiệt độ thông qua các dữ liệu có được và triển khai Resful API. Xây dựng ứng dụng React Native Expo để hiển thị nhiệt độ hiện tại và nhiệt độ dự đoán sau 60 phút.

1. **Phương pháp nghiên cứu:**

Nghiên cứu tham khảo tra cứu thông tin và tài liệu.

Hoàn thiện đề tài với sự hướng dẫn của giảng viên.

PHẦN NỘI DUNG

1. **Kế hoạch làm việc**

|  |  |
| --- | --- |
| Thành viên | Công việc thực hiện |
| Huỳnh Anh Thế Vinh | * Bảo trì hệ thống fire-base và IoT. * Xây dựng AI Services và deploy lên Heroku. * Xây dựng mô hình hồi quy đa biến có sự tham gia của tham số dộ ẩm. |
| Võ Châu Nhuận Trường | * Triển khai ứng dụng lên AWS. * Xây dựng AI Services và deploy lên Heroku. |
| Nguyễn Văn Hà | * Tìm hiểu và xây dựng ứng dụng tiêu thụ React Native. |
| Đinh Ngọc Nhi | * Viết báo cáo. * Nghiên cứu xây dựng hoàn chỉnh kiến trúc hệ thống tự động hóa. |
| Trần Văn Tuấn | * Điều chỉnh báo cáo. * Nghiên cứu xây dựng hoàn chỉnh kiến trúc hệ thống tự động hóa. |

Bảng 1. Bảng kế hoạch làm việc

1. **Bảng đánh giá các thành viên và tỉ lệ đóng góp**

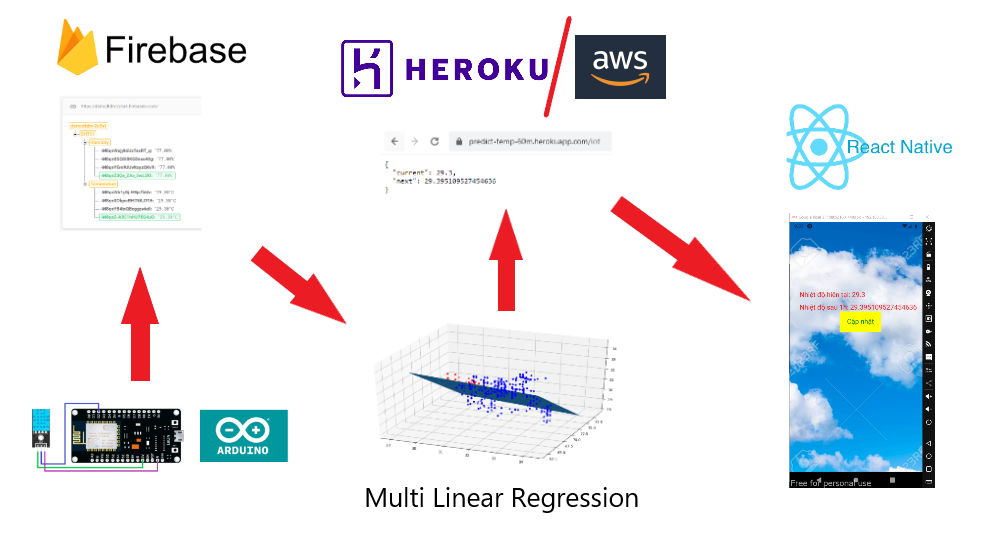
|  |  |  |  |
| --- | --- | --- | --- |
| MSSV | Họ và Tên | Đánh giá hoàn thành | Đóng góp |
| 17110255 | Huỳnh Anh Thế Vinh | Tốt | 100% |
| 17110130 | Nguyễn Văn Hà | Tốt | 100% |
| 17110200 | Đinh Ngọc Nhi | Tốt | 100% |
| 16110247 | Trần Văn Tuấn | Tốt | 100% |
| 16110236 | Võ Châu Nhuận Trường | Tốt | 100% |

Bảng 2. Bảng đánh giá và phần trăm đóng góp

1. **Mô tả bài toán**

Ứng dụng sử dụng cảm biến được lập trình bằng arduino để đo dữ liệu nhiệt độ và độ ẩm ứng với thời gian mỗi 30s và gửi lên lên cơ sở dữ liệu đám mây Firebase. Sau khi Firebase nhận được dữ liệu và sắp xếp, xây dựng mô đun AI để thu thập dữ liệu rồi sử dụng mô hình hồi quy tuyến tính đa biến để huấn luyện và dự đoán cho nhiệt độ 60p tiếp theo tính từ nhiệt độ hiện tại đo được. Sau đó gửi lên dịch vụ web Heroku với 2 dữ liệu từ mô đun AI sử dụng công cụ Flask. Cuối cùng là xây dựng ứng dụng React Native để hiện thị dữ liệu nhiệt độ hiện thời và nhiệt độ dữ đoán với giao diện thân thiện với người dùng.

1. **Kiến trúc tổng quát của hệ thống**



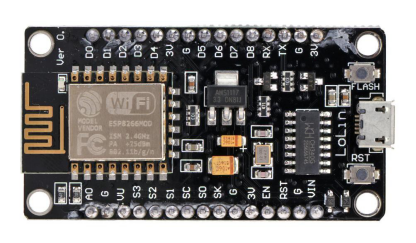
*Hình 1. Kiến trúc tổng quát của hệ thống*

1. **Quá trình xây dựng**
2. **Mô đun IoT**
3. **Mô tả thiết bị**

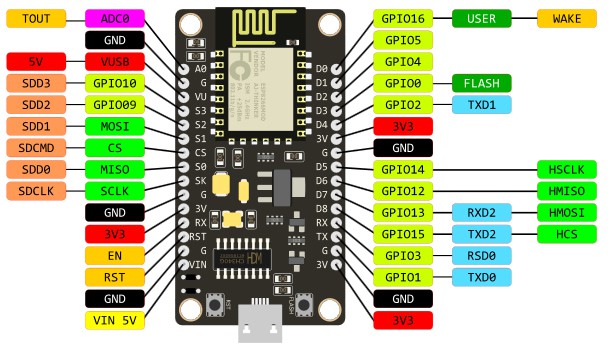
|  |  |  |
| --- | --- | --- |
| Số thứ tự | Tên thành phần | Giá tiền |
| 1 | Module NodeMCU Esp8266 | 85,000 VND |
| 2 | Cảm biến nhiệt độ và độ ẩm DHT11 | 30,000 VND |

Bảng 3. Bảng giá danh sách thiết bị

**NodeMCU CP2102 ESP8266**



*Hình* *2. NodeMCU CP2102 ESP8266*



*Hình* *3. Các chân NodeMCU Esp8266*

**Thông số kĩ thuật**

|  |  |
| --- | --- |
| IC chính | ESP8266 |
| Phiên bản firmware | NodeMCU Lua |
| Chip nạp và giao tiếp UART | CH340 |
| Số chân GPIO (General Purpose Input Output) | 16 chân |
| Cấp nguồn | 5V MicroUSB hoặc Vin. |
| GIPO giao tiếp mức | 3.3V |
| Kích thước | 59 x 32 mm |

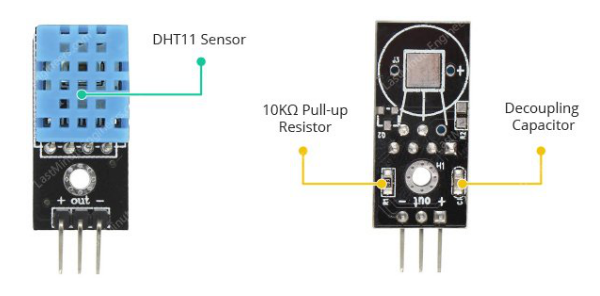
*Bảng 4. Thông số kĩ thuật NodeMCU Esp8266*

**Chức năng các chân**

|  |  |  |
| --- | --- | --- |
| Pin Category | Pin Name | Details |
| Power | USB Jack, Vin, VUSB, 3.3V, GND | USB Jack: Cấp nguồn, nạp chương trình, giao tiếp với máy tính (sử dụng cáp micro usb)  Vin: điện áp đầu vào khi sử dụng nguồn điện bên ngoài.  VUSB: điện áp đầu vào khi sử dụng nguồn điện bên ngoài (5V).  5V: Nguồn cung cấp được điều chỉnh dùng để cấp nguồn cho con chip và các thành phần khác trên bo mạch.  3.3V: Nguồn cung cấp 3.3V GND: ground pins. |
| Serial communication | RX, TX | Sử dụng để nạp chương trình vào giao tiếp với máy tính. |
| Reset | Reset | Khởi động lại mô đun. |
| Flash | Flash | Ít sử dụng, dành cho các nhà phát triển. |
| Analog | A0 | Được dùng để đọc điện áp bên ngoài hay đọc điển áp VCC.  Để đọc điện áp bên ngoài bằng chân ADC, sử dụng hàm analogRead(A0) |
| Input/Output | D0 – D8 | Digital (chân kĩ thuật số) dùng để đọc, viết và các chức năng thông dụng khác.  D0 có thể sử dụng để đọc viết, không hỗ trợ interrupt, không hỗ trợ pwm/ic2/ow. |

*Bảng 5. Chức năng các chân của NodeMCU Esp8266*

**Cảm biến nhiệt độ và độ ẩm DHT11**

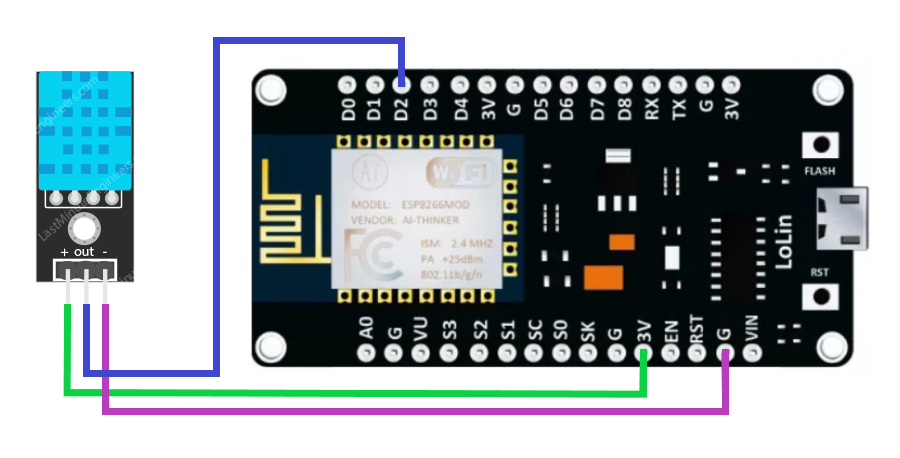


*Hình* *4. Cảm biến DHT11*

* Chức năng: Đo nhiệt độ môi trường và độ ẩm không khí với độ ổn định cao, có khả năng hoạt động liên tục trong thời gian dài. Giá thành rẻ.
* Thông số kĩ thuật
* Điện áp hoạt động: 3V - 5V (DC)
* Dãi độ ẩm hoạt động: 20% - 90% RH, sai số ±5%RH
* Dãi nhiệt độ hoạt động: 0°C ~ 50°C, sai số ±2°C
* Khoảng cách truyển tối đa: 20m
* Kích thước: 15.5mm x 12mm x 5.5mm
* Tần số lấy mẫu: 1Hz, nghĩa là 1 giây DHT11 lấy mẫu một lần.
* 4 chân: VCC (cực (+) nguồn), DATA (chân tín hiệu), NC, GND (cực (-) nguồn)

1. **Mô hình triển khai**

**Sơ đồ mạch lắp đặt**



*Hình* *5. Sơ đồ lắp đặt*

|  |  |
| --- | --- |
| ESP8266 | Sensor |
| 3V | Vcc |
| GND | GND |
| D2 | OUT |

*Bảng 6. Sơ đồ lắp đặt*

1. **Mã nguồn**

#include <ESP8266WiFi.h> // esp8266 library

#include <FirebaseArduino.h> // firebase library

#include <DHT.h> // dht11 temperature and humidity sensor library

#include <NTPClient.h>

#include <WiFiUdp.h>

//DHT config

#define DHTPIN D2 // what digital pin we're connected to

#define DHTTYPE DHT11 // select dht type as DHT 11 or DHT22

DHT dht(DHTPIN, DHTTYPE);

// Wi-Fi Settings

#define WIFI\_SSID "network\_name" // input your home or public wifi name

#define WIFI\_PASSWORD "password" // password of wifi ssid

// FireBase Settings

#define FIREBASE\_HOST "your-project.firebaseio.com" // the project name address from firebase id

#define FIREBASE\_AUTH "UpmxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxhHa" // the secret key generated from firebase

WiFiClient client;

WiFiUDP ntpUDP;

// You can specify the time server pool and the offset (in seconds, can be

// changed later with setTimeOffset() ). Additionaly you can specify the

// update interval (in milliseconds, can be changed using setUpdateInterval() ).

//NTPClient timeClient(ntpUDP, "europe.pool.ntp.org", 3600, 60000);

NTPClient timeClient(ntpUDP, "vn.pool.ntp.org");

// ThingSpeak Settings

const int channelID = YOUR-CHANNEL-ID; //

String writeAPIKey = "YOUR-API-KEY"; // write API key for your ThingSpeak Channel

const char\* server = "api.thingspeak.com";

const int postingInterval = 29 \* 1000; // post data every 29 seconds

void setup() {

Serial.begin(115200);

delay(1000);

WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD); //try to connect with wifi

Serial.print("Connecting to ");

Serial.print(WIFI\_SSID);

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

Serial.print(".");

delay(100);

}

Serial.println();

Serial.print("Connected to ");

Serial.println(WIFI\_SSID);

Serial.print("IP Address is : ");

Serial.println(WiFi.localIP()); //print local IP address

Firebase.begin(FIREBASE\_HOST, FIREBASE\_AUTH); // connect to firebase

dht.begin(); //Start reading dht sensor

timeClient.begin();

}

void loop() {

// wait and then post again

delay(postingInterval);

float humi = dht.readHumidity(); // Reading temperature or humidity takes about 250 milliseconds!

float temp = dht.readTemperature(); // Read temperature as Celsius (the default)

timeClient.update();

if (isnan(humi) || isnan(temp)) { // Check if any reads failed and exit early (to try again).

Serial.println(F("Failed to read from DHT sensor!"));

return;

}

String fireTime = String(timeClient.getEpochTime());

String fireTemp = String(temp); //convert integer temperature to string temperature

String fireHumid = String(humi); //convert integer humidity to string humidity

// String fireTemp = String(temp) + String("°C"); //convert integer temperature to string temperature

// String fireHumid = String(humi) + String("%"); //convert integer humidity to string humidity

Serial.print("Timestamp: "); Serial.print(fireTime);

Serial.print(" Temperature: "); Serial.print(temp); Serial.print("°C ");

Serial.print(" Humidity: "); Serial.print(humi); Serial.println("% ");

// Serial.println(timeClient.getFormattedTime());

Firebase.pushString("/DHT11/Timestamp", fireTime); //setup path and send readings

Firebase.pushString("/DHT11/Temperature", fireTemp); //setup path and send readings

Firebase.pushString("/DHT11/Humidity", fireHumid); //setup path and send readings

if (client.connect(server, 80)) {

// Construct API request body

String body = "field1=" + String(temp, 1) + "&field2=" + String(humi, 1);

client.print("POST /update HTTP/1.1\n");

client.print("Host: api.thingspeak.com\n");

client.print("Connection: close\n");

client.print("X-THINGSPEAKAPIKEY: " + writeAPIKey + "\n");

client.print("Content-Type: application/x-www-form-urlencoded\n");

client.print("Content-Length: ");

client.print(body.length());

client.print("\n\n");

client.print(body);

client.print("\n\n");

}

client.stop();

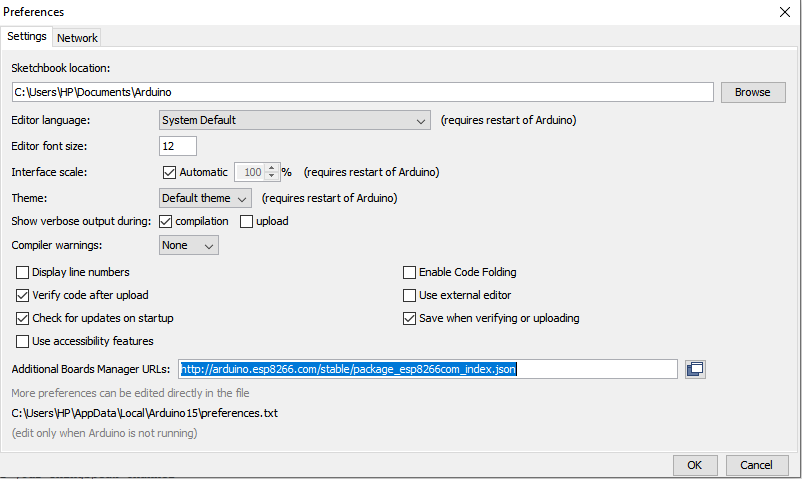
}

1. **Cài đặt và vận hành**

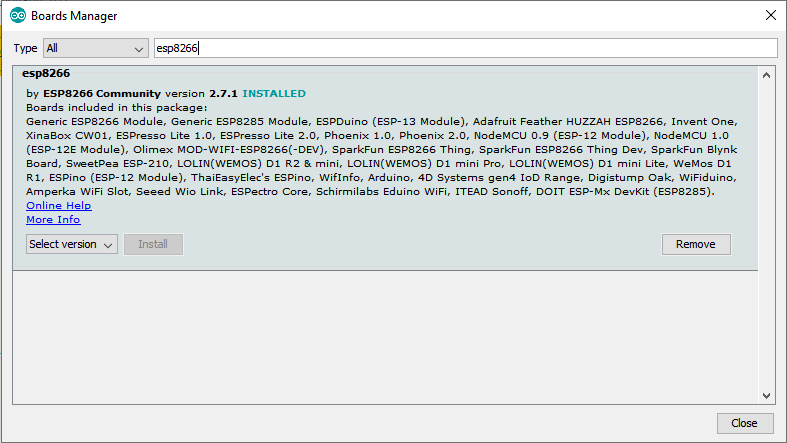
* Cài đặt Adruino IDE
* Cài đặt thư viện

#include <ESP8266WiFi.h>

Vào File --> Preferences --> Thêm <http://arduino.esp8266.com/stable/package_esp8266com_index.json> vào Addtionals Board Manager URLs.



Sau đó vào Tools --> Board Manager --> Gõ tìm kiếm “esp8266" rồi cài đặt.



Cuối cùng vào Tools --> Board --> Chọn “NodeMCU 1.0 (ESP-12E Module)” để thêm thư viện esp8266 vào project.

#include <FirebaseArduino.h>

Tải xuống và cài đặt thư viện theo liên kết:

Firebase-arduino: <https://github.com/FirebaseExtended/firebase-arduino>

ArduinoJson-5.x: <https://github.com/bblanchon/ArduinoJson>

Sau khi tải hoàn tất, vào Sketch --> Include Library --> Add .ZIP Library… --> trỏ tới 2 file .zip vừa tải.

#include <DHT.h>

Vào Sketch --> Include Library --> Cài đặt thư viện “DHT sensor library”

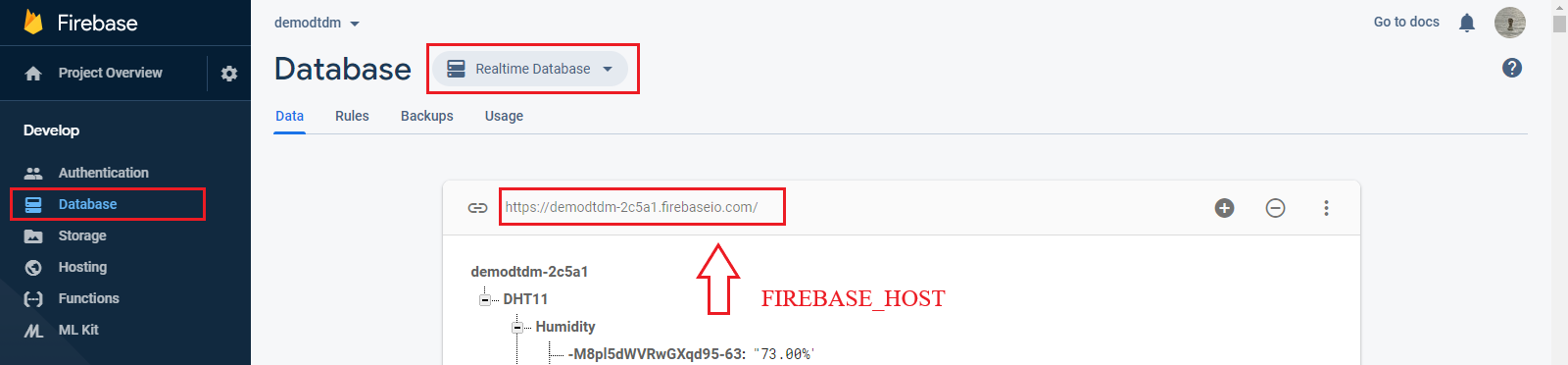
* Lập trình

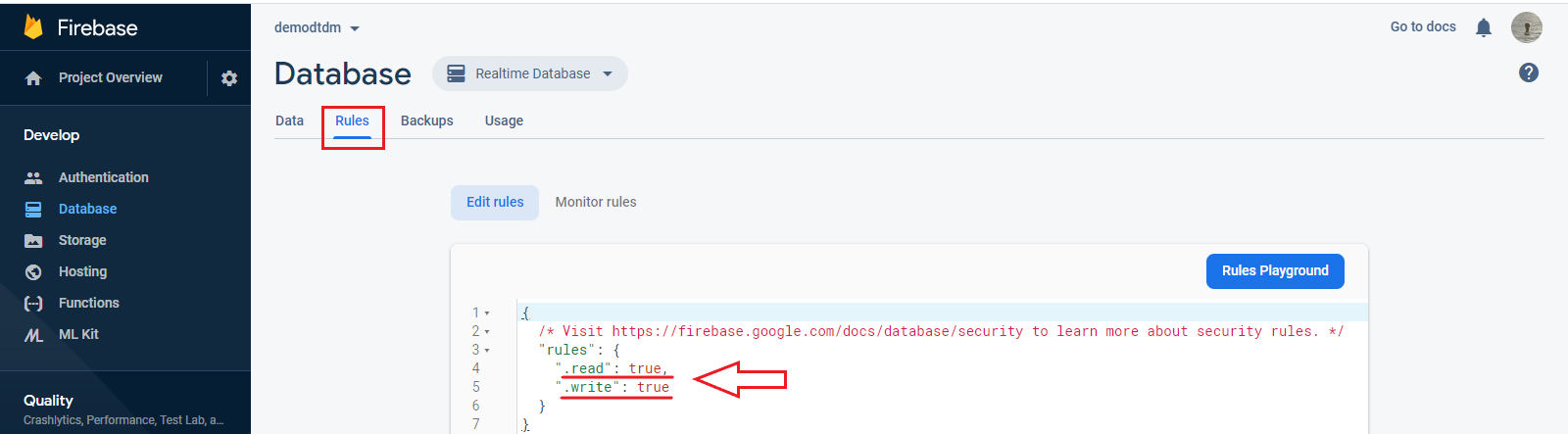
Source Code: sử dụng mã nguồn ở trên.

* Tạo Firebase

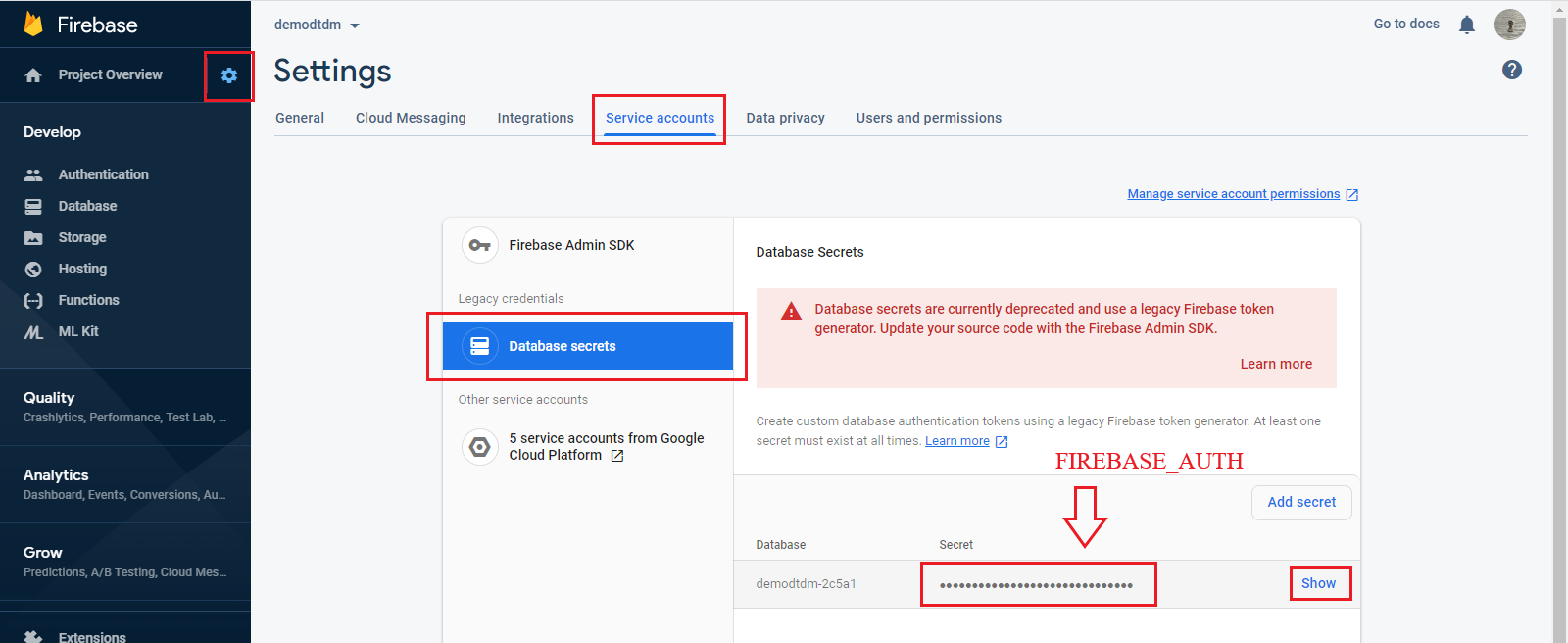
Trang chủ: Firebase - <https://firebase.google.com>

- Sử dụng tài khoản google đăng nhập Firebase Google và tạo mới 1 database để lấy FIREBASE\_HOST và FIREBASE\_AUTH





*Hình 6. Database Firebase*



*Hình 7. Setting Firebase*

* Tạo ThinhSpeak

- Đăng nhập vào ThingSpeak™ bằng tài khoản MathWorks®, hoặc tạo mới một tài khoản tại MathWorks account.

- Nhấp vào Channels > MyChannels.

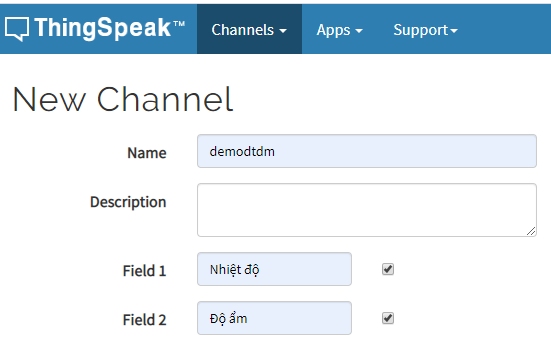
- Trên trang Channels, chọn New Channel.

- Nhập tên cho Channel. Đánh dấu vào các ô bên cạnh Fields 1 và 2. Nhập tên các giá trị cài đặt channel này

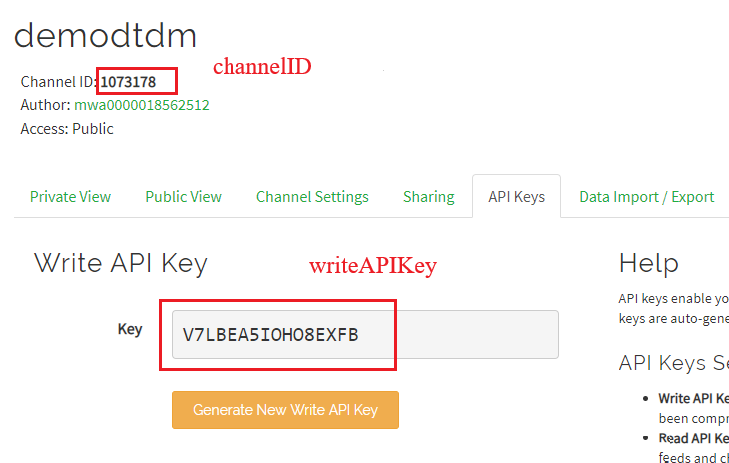
- Nhấp vào Save Channel ở phía dưới sẽ thấy các tab sau:

Private View: Tab này hiển thị thông tin về channel của bạn mà chỉ bạn mới có thể nhìn thấy.

* Public View: Nếu bạn chọn để công khai kênh của mình, hãy sử dụng tab này để hiển thị các trường đã chọn và trực quan hóa kênh.
* Channel Settings: Tab này hiển thị tất cả các tùy chọn kênh bạn đặt khi tạo. Bạn có thể chỉnh sửa, xoá dữ liệu hoặc xóa kênh khỏi tab này.
* Sharing: Tab này hiển thị các tùy chọn chia sẻ kênh. Bạn có thể đặt kênh ở chế độ riêng tư, chia sẻ với mọi người (công khai) hoặc chia sẻ với những người dùng cụ thể.
* API Keys: Tab này hiển thị các khóa API kênh của bạn. Sử dụng các phím để đọc và ghi vào kênh của bạn.
* Data Import/Export: Tab này cho phép bạn nhập và xuất dữ liệu kênh.

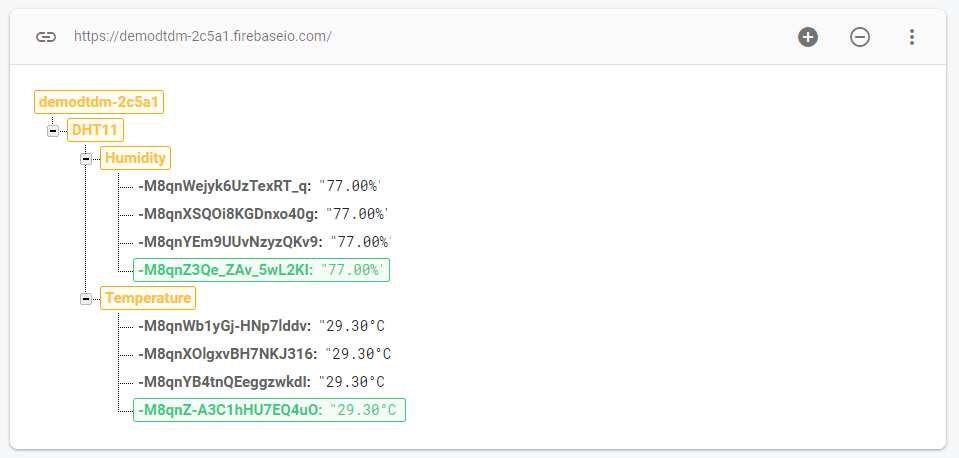


*Hình 8. New channel ThingSpeak*

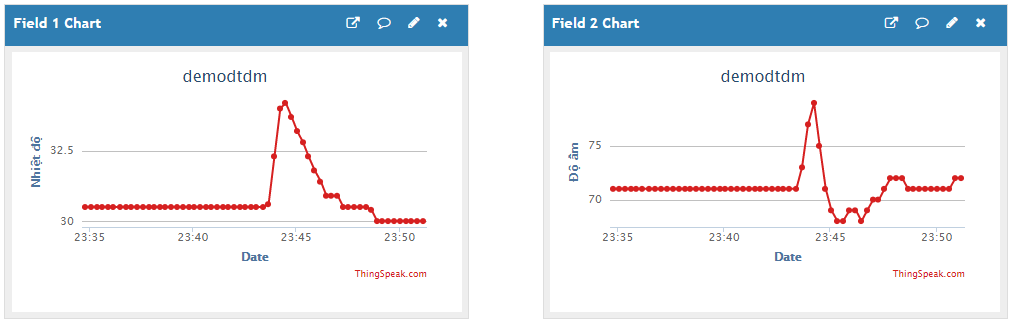


*Hình* *9. WriteAPIKey ThingSpeak*

1. **Kết quả**



*Hình* *10. Kết quả nhiệt độ thu được từ Firebase*



*Hình* *11. Biểu đồ kết nhiệt độ và độ ẩm ThingSpeak*

1. **Mô đun AI**
2. **Mô tả phương pháp xử lý, xây dựng dữ liệu huấn luyện**

* Dữ liệu thu được là nhiệt độ và độ ẩm mỗi 30 giây. Để có thể dự đoán được nhiệt độ sau 60 phút ta cần xây dựng được dữ liệu huấn luyện 60 phút. Đầu tiên đối với dữ liệu đo được với các khoảng thời gian rời rạc do không phải lúc nào cảm biến cũng đo và gửi về chính xác mỗi 30 giây. Để làm được điều đó ta xét nhiệt tại thời điểm *t*, nếu không tồn tại dự liệu nhiệt độ tại *t + 3600s* ta sẽ thay thế bằng nhiệt độ tại thời điểm liền kề dưới *t + 3600s.*
* Xây dựng dữ liệu huấn luyện lấy 80% bộ dữ liệu để huấn luyện, 20% còn lại là dữ liệu kiểm tra.

1. **Mô hình huấn luyện**

* Sử dụng mô hình hồi quy đa biến với biến nhiệt độ và biến độ ẩm để dự đoán nhiệt độ sau 60 phút
  + X1: là nhiệt độ tại thời điểm t
  + X2: là độ ẩm tại thời điểm t
  + Y: là nhiệt độ tại thời điểm t+3600

1. **Mã nguồn**

#============================= Packages ================================#

from firebase.firebase import FirebaseApplication

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from scipy import stats

from sklearn import linear\_model

from mpl\_toolkits.mplot3d import axes3d

from flask import Flask, jsonify

#========================= Data =================================#

# app = FirebaseApplication('https://fir-1f5ec.firebaseio.com/', authentication=None)

app = FirebaseApplication('https://dht11-1389c.firebaseio.com/', authentication=None)

result = app.get('/DHT11', '')

df = pd.DataFrame(result)

# print(df)

time\_list = df['Timestamp'].dropna().tolist()

temp\_list = df['Temperature'].dropna().tolist()

humi\_list = df['Humidity'].dropna().tolist()

time\_list = [int(i) for i in time\_list]

temp\_list = [float(i) for i in temp\_list]

humi\_list = [float(i) for i in humi\_list]

df\_data = pd.DataFrame()

#======================= Bộ dữ liệu 60p ===========================#

df\_data['t'] = time\_list[0:int(len(time\_list)-120)]

df\_data['x'] = temp\_list[0:int(len(time\_list)-120)]

df\_data['x2'] = humi\_list[0:int(len(time\_list)-120)]

y = []

for i in range(len(df\_data['x'])):

    t = time\_list[i] + 3600

    if t in time\_list:

        y.append(temp\_list[i])

    else:

        for j in range(len(time\_list)):

            if time\_list[j] > t - 30:

                y.append(temp\_list[j])

                break

df\_data['y'] = y

# print(df\_data)

#================= Mô hình huấn luyện ==========================#

''' 80% tran

    20% test'''

size\_train = int(len(df\_data)\*0.8)

df\_train = df\_data[0:size\_train]

df\_test = df\_data[size\_train:]

#================== Hồi quy tuyến tính đa biến ========================#

''' X1 là nhiệt độ tại thời điểm t

    X2 là độ ẩm tại thời điểm t

    Y là nhiệt dộ tại thời điểm t + 3600s'''

reg = linear\_model.LinearRegression()

reg.fit(df\_train[['x','x2']], df\_train['y'])

print('Intercept: \n', reg.intercept\_)

print('Coefficients: \n', reg.coef\_)

def myfun2(x1, x2):

    return reg.intercept\_ + reg.coef\_[0]\*x1 + reg.coef\_[1]\*x2

fig = plt.figure()

ax = fig.gca(projection='3d')

ax.scatter(df\_train['x'], df\_train['x2'], df\_train['y'], c='b')

ax.scatter(df\_test['x'], df\_test['x2'], df\_test['y'], c='r')

# Plot the surface.

X1 = df\_train['x']

X2 = df\_train['x2']

X1, X2 = np.meshgrid(X1, X2)

Y = myfun2(X1, X2)

ax.plot\_surface(X1, X2, Y)

plt.show()

#========================== API ===================================#

'''API lấy nhiệt độ hiện tại và dự đoán 60p'''

def nhietDoDuDoan(X1, X2):

    return myfunc(X1, X2)

1. **Kết quả**

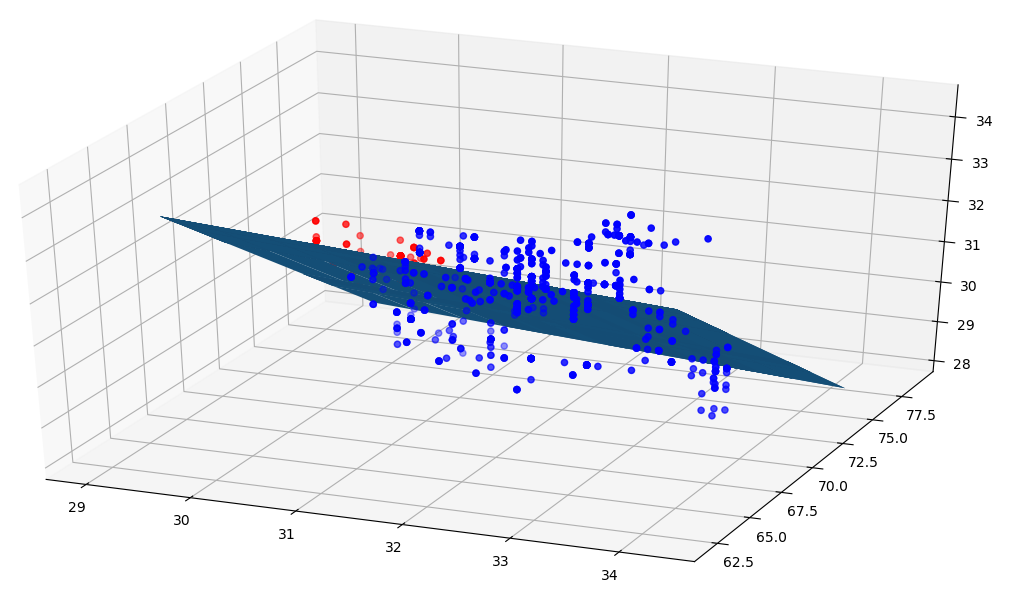
Ví dụ cho 1000 dữ liệu nhiệt độ và độ ẩm:

Intercept:

61.865733964398046

Coefficients:

[-0.17125273 -0.36614653]



*Hình* *12. Đồ thị ví dụ hồi quy đa biến*

1. **Đánh giá mô hình**

* Hồi quy tuyến tính là mô hình đơn giản dễ hiểu, dễ xây dựng nhưng bản thân nó cũng có nhiều hạn chế. Để có thể dự đoán chính xác nhiệt độ cần phải có nhiều yếu tố ngoài độ ẩm và thời gian. Kết quả dự đoán còn ảnh hưởng do mùa mà nhiều biến động khác. Nên mô hình dự đoán hồi quy tuyến tính có thể mang lại sai số rất lớn.

1. **Mô đun Restful API**
2. **Mô tả**

* Lấy dữ liệu nhiệt độ hiện tại và nhiệt độ 60 phút tiếp theo gửi lên dịch vụ web dưới dạng dữ liệu Json

1. **Mô hình triển khai**

* Sử dụng dịch vụ Heroku và Flask để triển khai Restfull API.

1. **Mã nguồn**

from firebase.firebase import FirebaseApplication

import pandas as pd

from flask import Flask, jsonify

from dht11 import nhietDoDuDoan

app = FirebaseApplication('https://dht11-curr.firebaseio.com/', authentication=None)

result = app.get('/DHT11', '')

df = pd.DataFrame(result)

temp\_list = df['Temperature'].dropna().tolist()

humi\_list = df['Humidity'].dropna().tolist()

temp\_list = [float(i) for i in temp\_list]

humi\_list = [float(i) for i in humi\_list]

nhietDoHienTai = temp\_list[-1]

doAmHienTai = humi\_list[-1]

app = Flask(\_\_name\_\_)

app.config["DEBUG"] =  True

@app.route('/', methods=['GET'])

def home():

    return "HELLO MR.NAM"

@app.route('/iot', methods=['GET'])

def getNextFromCurrent():

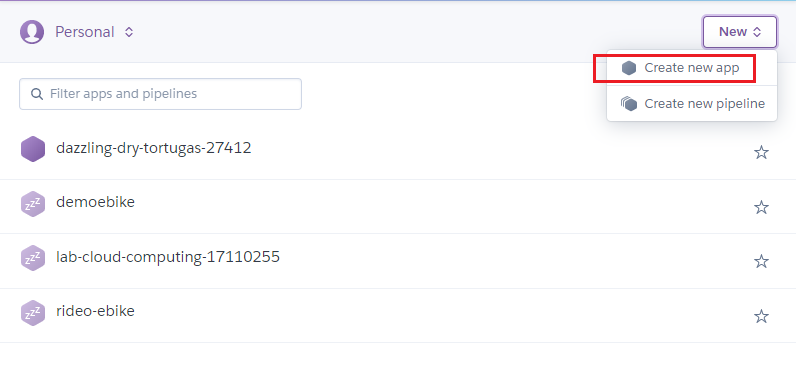
    return jsonify({'current': nhietDoHienTai, 'next': nhietDoDuDoan(nhietDoHienTai, doAmHienTai)})

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

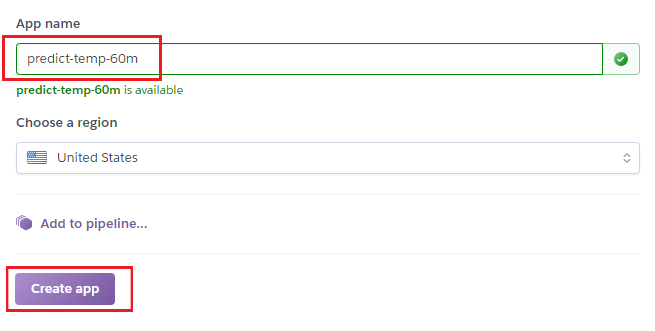
1. **Cài đặt và vận hành**

* Tạo tài khoản và đăng nhập và Heroku.



*Hình 13. Dịch vụ Heroku*

* Tạo tên miền cho ứng dụng.



* Cài đặt công cụ Git và nắm các thao tác cơ bản (*link cài đài:* <https://git-scm.com/>).
* Cài đặt môi trường ảo

>pip install virtualenv

* Tạo thư mục và thêm mô đun AI với tên *dht11.py* và mô đun API với tên *app.py*

>mkdir predict-temp-60m

>cd predict-temp-60m

predict-temp-60m>dir

Volume in drive C has no label.

Volume Serial Number is 0601-B880

Directory of C:\Users\HP\Desktop\predict-temp-60m

07/05/2020 07:43 PM <DIR> .

07/05/2020 07:43 PM <DIR> ..

07/05/2020 07:26 PM 876 app.py

07/05/2020 07:26 PM 3,323 dht11.py

* Khởi chạy môi trường ảo.

predict-temp-60m>virtualenv venv

predict-temp-60m>cd venv/Scripts

predict-temp-60m\venv\Scripts>activate

(venv) predict-temp-60m\venv\Scripts>

* Cài đặt tất cả packages mà ứng dụng yêu cầu.

predict-temp-60m>pip install flask gunicorn jinja2

predict-temp-60m>pip install requests python-firebase

predict-temp-60m>pip install pandas numpy matplotlib

predict-temp-60m>pip install scipy scikit-learn mplot3d-dragger

* Tạo file *Procfile* với nội dung

web: gunicorn app:app

* Khai báo các packages sử dụng

predict-temp-60m>pip freeze > requirements.txt

* Tạo file *.gitinore*

venv

* Triển khai lên Heroku

predict-temp-60m>heroku login

predict-temp-60m>git init

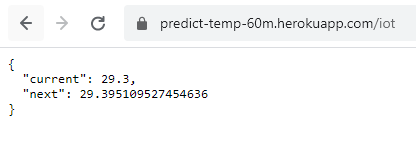
predict-temp-60m>heroku git:remote -a predict-temp-60m

predict-temp-60m>git add .

predict-temp-60m>git commit -am "make it better"

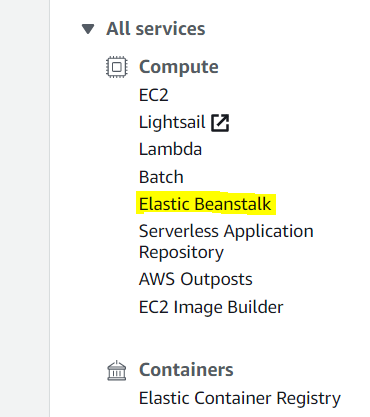
predict-temp-60m>git push heroku master

1. **Kết quả**

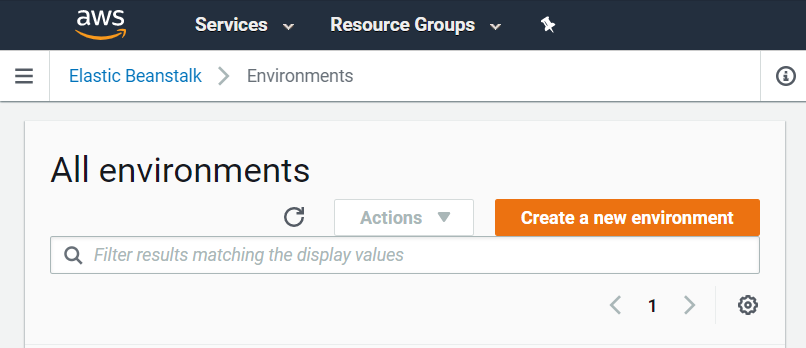


1. **Mô đun Restful API (AWS)**
2. **Cài đặt và vận hành**

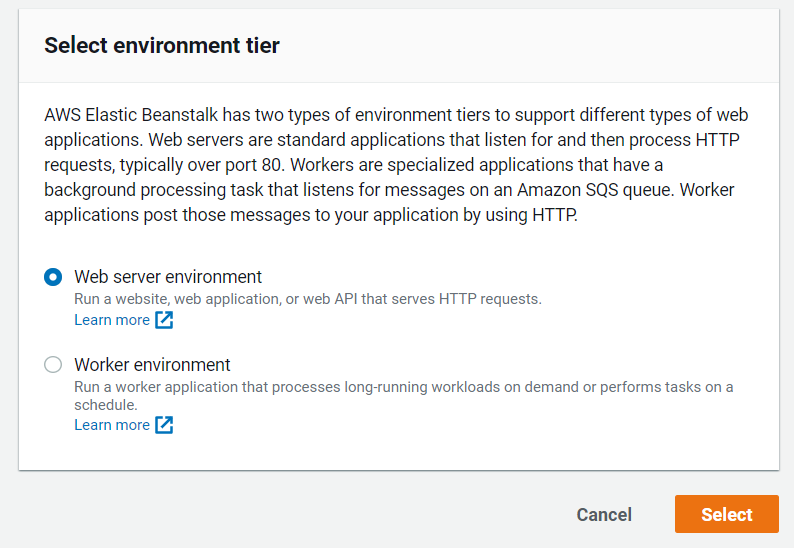
* Truy cập console.aws.amazon.com, chọn Elastic Beanstalk.



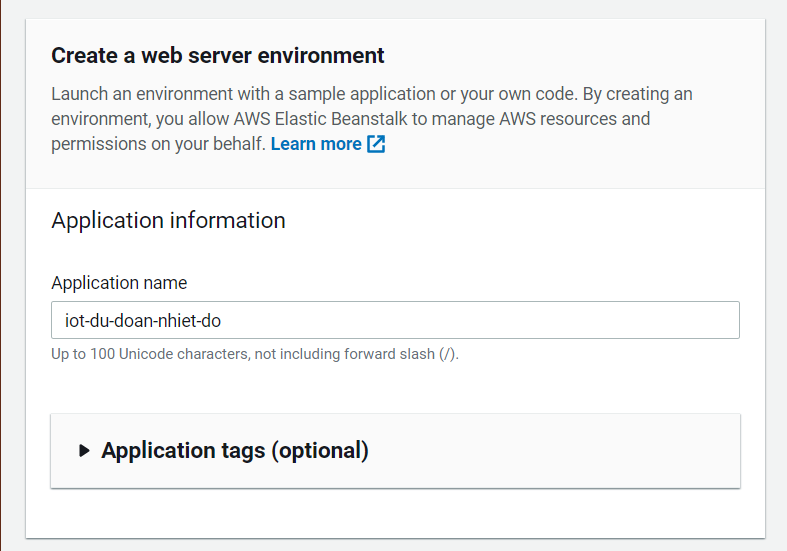
* Chọn Create a new environment



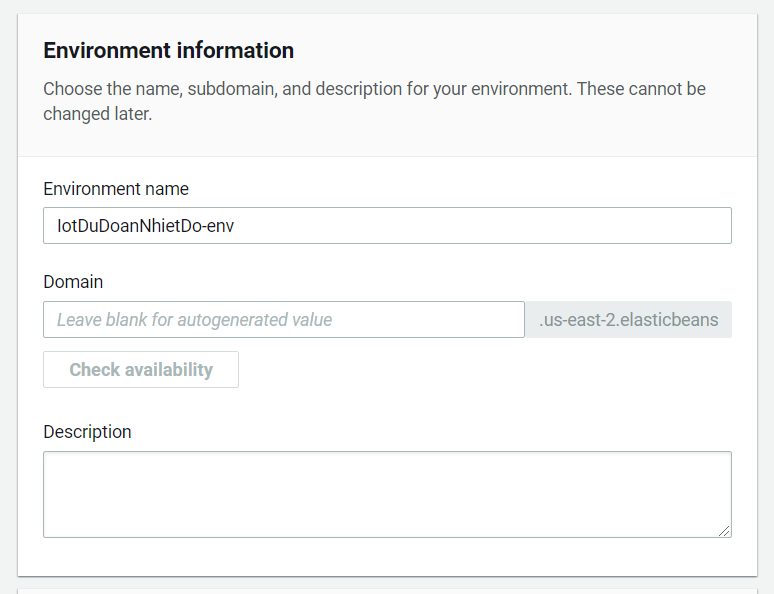
* Chọn Web server environment, sau đó click Select.



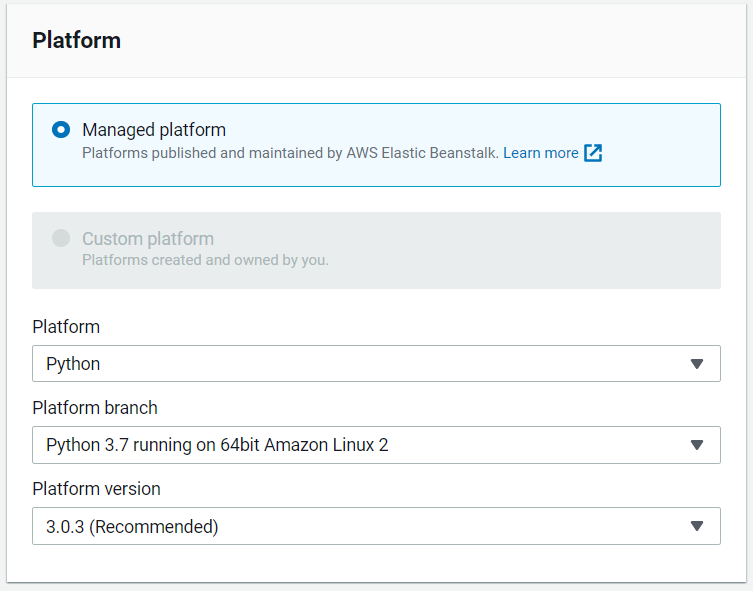
* Điền tên application



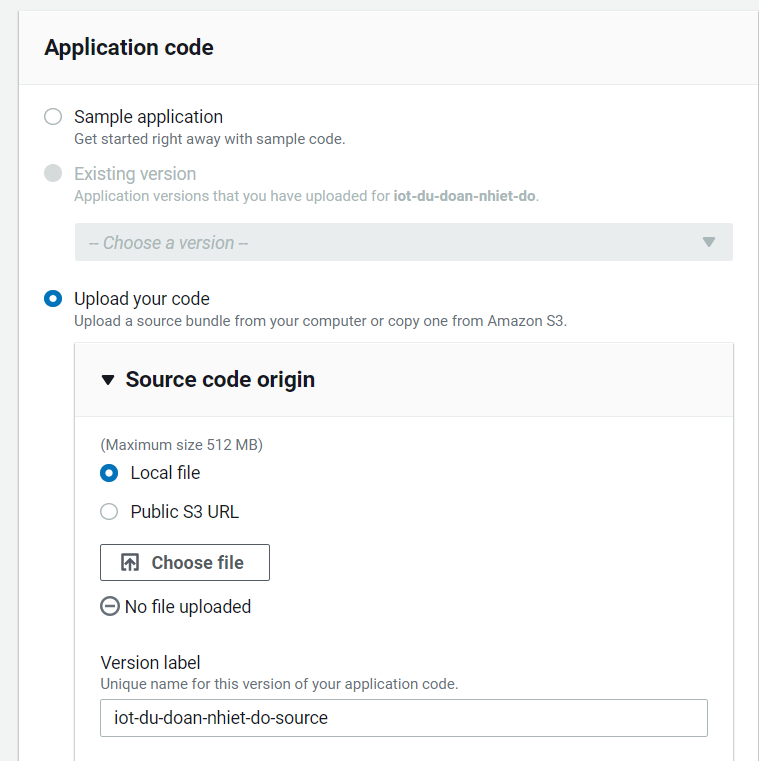
* Điền tên môi trường



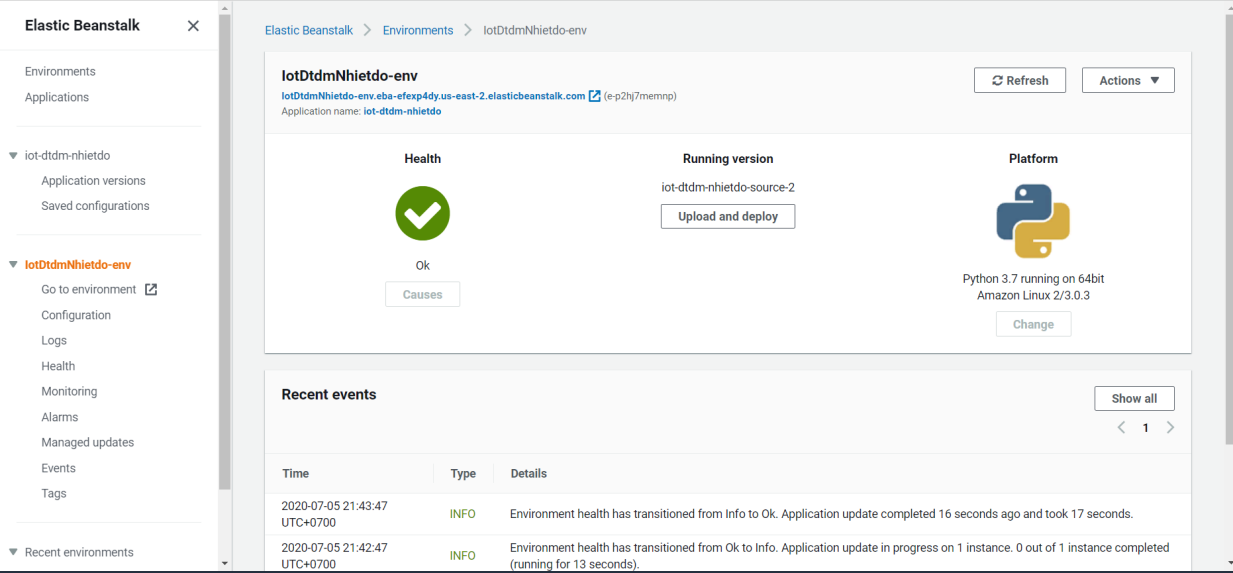
* Chọn platfỏm



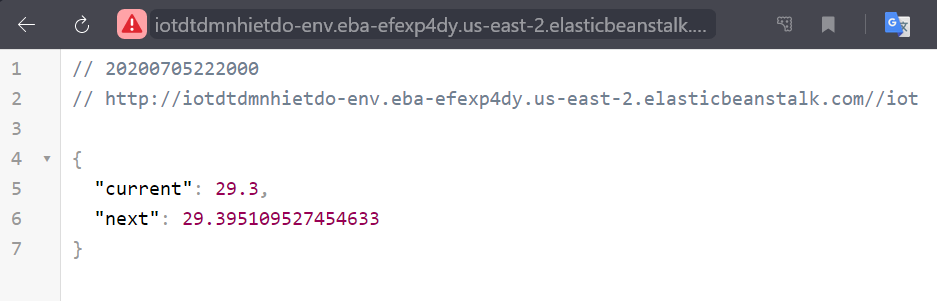
* Chọn tải lên source code



* Chờ khoảng vài phút, ứng dụng sẽ được deploy lên



1. **Kết quả**



1. **Mô đun Mobile (React)**
2. **Mô tả**

* Tiêu thụ dữ liệu từ dịch vụ web, thu thập 2 dữ liệu trả về là nhiệt độ hiện tại và nhiệt độ 60 phút sau. Hiển thị kết quả với giao diện người dùng.

1. **Mã nguồn**

* **Api.js**

export const getTemCurrent = async weather =>

{

  const response = await fetch(`https://predict-temp-60m.herokuapp.com/iot`);

  let {current, next} = await response.json();

  return {

    current: current,

    next: next,

  }

}

* **App.js**

import React from 'react';

import { StyleSheet, Text, View, ImageBackground, TouchableOpacity } from 'react-native';

// Utils

import { getTemCurrent } from './utils/api';

export default class App extends React.Component {

  constructor(props) {

    super(props);

    this.state={

      current: 0,

      next: 0,

    };

  }

  // Life cycle

  componentDidMount() {

    this.handleUpdateWeather();

  }

  // Update current

  handleUpdateWeather= async weather => {

    this.setState( async () => {

      try {

        const {current, next }  = await getTemCurrent();

        this.setState({

          current,

          next

        });

      } catch (e) { }

    });

  };

  render() {

    const { current, next } = this.state;

    return (

        <ImageBackground source={require('./assets/wbg.jpg')} style={styles.imageContainer}>

          <View style={{paddingTop: 200, paddingLeft: 30}}>

            <Text style={styles.text}>Nhiệt độ hiện tại: {this.state.current}</Text>

          </View>

          <View style={{paddingTop:10, paddingLeft: 30}}>

            <Text style={styles.text}>Nhiệt độ sau 1h: {this.state.next}</Text>

          </View>

          <TouchableOpacity onPress={()=>{this.handleUpdateWeather()}} style={styles.bt }>

            <View >

              <Text style={{paddingLeft:20, paddingTop: 15, fontSize: 20, color: 'green'}}>Cập nhật</Text>

            </View>

          </TouchableOpacity>

        </ImageBackground>

    );

  }

}

const styles = StyleSheet.create({

  imageContainer: {

    flex: 1,

  },

  text: {

    fontSize: 20,

    color: 'red',

  },

  bt: {

    width:120,

    height:60,

    backgroundColor: 'yellow',

    marginLeft: 150,

  }

});

1. **Cài đặt và vận hành**

* Cài đặt expo: npm install expo-cli
* Tạo project: expo init WeatherApp
* Cấu hình file package.json

{

  "main": "node\_modules/expo/AppEntry.js",

  "scripts": {

    "start": "expo start",

    "android": "expo start --android",

    "ios": "expo start --ios",

    "web": "expo start --web",

    "eject": "expo eject"

  },

  "dependencies": {

    "expo": "~38.0.1",

    "expo-status-bar": "^1.0.0",

    "react": "~16.11.0",

    "react-dom": "~16.11.0",

    "react-native": "https://github.com/expo/react-native/archive/sdk-38.0.0.tar.gz",

    "react-native-elements": "^2.0.2",

    "react-native-web": "~0.11.7",

    "moment": "^2.22.2"

  },

  "devDependencies": {

    "@babel/core": "^7.8.6",

    "babel-preset-expo": "~8.1.0"

  },

  "private": true

}

* Cài đặt các package cho project: npm i
* Run project: expo start



* Run android device

1. **Kết quả**

* **Android app**



*Hình 14. Ứng dụng React Native*

1. **Kết quả hệ thống**

Hệ thống có thể dự đoán được nhiệt độ 60 phút tiếp theo tính từ nhiệt độ hiện tại đo được. Từ đây có thể thực hiện dự đoán với các khoản thời gian mong muốn nhờ vào mô hình huấn luyện hồi quy tuyến tính đa biến. Hệ thống có thể tự động cập nhật nhiệt độ đo được lần cuối nhờ vào cảm biến gửi liên tục dữ liệu lên Firebase và ứng dụng React có thể thông qua đó tạo mới lại trạng thái nhiệt độ hiện tại để có nhiệt độ dự đoán mới.

1. **Ưu điểm và nhược điểm**

**Ưu điểm**

* Đơn giản, dễ hiểu, dễ xây dựng với mô hình hồi quy tuyến tính đa biến.
* Các dịch vụ đều miễn phí phù hợp với mọi đối tương muốn xây dựng mô hình dự báo nhiệt độ đơn giản.

**Nhược điểm**

* Hạn chế của mô hình hồi quy đa biến gây ra sai số lớn.
* Ứng dụng chưa được tối ưu gây trễ dữ liệu, chỉ phù hợp với mô hình nhỏ.
* Dịch vụ đơn giản nên khả năng bảo mật kém.
* Độ chính xác không tuyệt đối.
* Không có tiên liệu trước là độ trễ sẽ là bao nhiêu.
* Mô hình có thêm một độ trễ thì mất đi một bậc tự do, nếu dữ liệu chuỗi thời gian không đủ dài thì ý nghĩa thống kê của mô hình ngày càng kém.
* Các biến giải thích thực chất là giá trị của một biến X theo thời gian, điều này gây ra sự tương quan giữa các biến giải thích trong mô hình, tức là có hiện tượng đa cộng tuyến. Ước lượng các tham số của mô hình trong trường hợp có đa cộng tuyến sẽ cho kết quả kém chính xác.

PHẦN KẾT LUẬN

1. **Những điều đã đạt**

Sau khi hoàn thành xong project, chúng em đã tự tích lũy được cho chính mình những kiến thức cơ bản về lập trình qua các thiết bị IoT, hệ thống cloud và lập trình nói chung, dịch vụ SOA, FireBase và các lưu trữ dạng Real Time Data.

1. **Những hạn chế**

Trong lúc thực hiện project, chúng em cũng còn 1 số thiếu sót như bài làm đã đạt yêu cầu nhưng vẫn chưa hoàn hảo cho mọi trường hợp. Các nhiệt độ và độ ẩm chỉ được tương đối chưa có sự chính xác tuyệt đối. Khoảng thời gian cập nhật vẫn tương đối lâu.

TÀI LIỆU THAM KHẢO

1. Arduino.vn> Thư viện> Serial> read() [online], viewed 04 June 2020, from:<<http://arduino.vn/reference/library/serial/1/huong-dan-ham/read>>
2. Advancecad.edu.vn> Gửi dữ liệu cảm biến nhiệt độ và độ ẩm đến cơ sở dữ liệu thời gian thực Google Firebase bằng NodeMCU ESP8266 [online], viewed 04 June 2020, from:<<https://advancecad.edu.vn/gui-du-lieu-cam-bien-nhiet-do-va-do-am-den-co-so-du-lieu-thoi-gian-thuc-google-firebase-bang-nodemcu-esp8266/>>
3. Arduino.esp8266.vn> Đo nhiệt độ, độ ẩm và gởi lên Thingspeak [online], viewed 04 June 2020, from:<<https://arduino.esp8266.vn/projects/dht11-thingspeak.html>>
4. George Lambert (14 Jan 2016)> Deploying a Flask app to Heroku (Windows). Viewed from <http://www.gtlambert.com/blog/deploy-flask-app-to-heroku>
5. w3schools.com> Machine Learning. Viewed from <https://www.w3schools.com/python/python_ml_getting_started.asp>