

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP.HCM**

**VIỆN KỸ THUẬT HUTECH**

**BÁO CÁO KHOA HỌC**

**HỆ THỐNG GIẾNG TRỜI TỰ ĐỘNG**

**TRẠM THỜI TIẾT MINI**

Chuyên ngành**: Robot – Trí tuệ nhân tạo**

**Sinh viên thực hiện/MSSV:** Nguyễn Đình Khiêm / 2186300559

Nguyễn Văn Thành Đạt / 2186300021

**Lớp:** 21DRTA1

*TP.HCM, Ngày…Tháng…Năm*

Mục Lục

[I. Đặt vấn đề: 4](#_Toc130462638)

[II. Tổng quan công trình: 4](#_Toc130462639)

[III. Mục tiêu – Phương pháp: 4](#_Toc130462640)

[ MỤC TIÊU: 4](#_Toc130462641)

[ PHƯƠNG PHÁP: 4](#_Toc130462642)

[IV. Kết quả - Thảo luận: 5](#_Toc130462643)

[1. Thiết kế phần cứng: 5](#_Toc130462644)

[1.1. Mô hình phần cứng: 5](#_Toc130462645)

[1.2. Động cơ: 5](#_Toc130462646)

[1.3. Cảm biến – Thiết bị: 6](#_Toc130462647)

[1.3.1. Cảm biến nhiệt độ - độ ẩm: 6](#_Toc130462648)

[1.3.2. Cảm biến áp suất: 7](#_Toc130462649)

[1.3.3. Cảm biến mưa: 8](#_Toc130462650)

[1.3.4. Cảm biến cường độ ánh sáng: 9](#_Toc130462651)

[1.3.5. Cảm biến chạm: 10](#_Toc130462652)

[1.3.6. Đèn led: 10](#_Toc130462653)

[1.3.7. Vi điều khiển: 10](#_Toc130462654)

[1.4. Sơ đồ kết nối hệ thống: 12](#_Toc130462655)

[2. Thiết kế phần mềm: 12](#_Toc130462656)

[2.1. Tổng quan hệ thống: 12](#_Toc130462657)

[2.2. Nguyên lý hoạt động: 13](#_Toc130462658)

[2.3. Lập trình điều khiển: 16](#_Toc130462659)

[2.3.1. Thư viện 16](#_Toc130462665)

[2.3.2. Hàm dùng cho hệ thống: 17](#_Toc130462666)

[2.3.3. Kết nối tới Wifi: 19](#_Toc130462667)

[2.3.4. Lấy API từ Server khác: 19](#_Toc130462668)

[2.3.5. Upload dữ liệu Front End lên mạng và Xử lý Resquest: 20](#_Toc130462669)

[2.3.5.1.Upload dữ liệu Web Config: 20](#_Toc130462680)

[2.3.5.2.Upload dữ liệu Weather Web: 21](#_Toc130462703)

[2.3.6. Xử lý Logic: 22](#_Toc130462704)

[3. Tổng hợp hệ thống: 23](#_Toc130462705)

[3.1. Chế tạo phần cứng: 23](#_Toc130462706)

[3.2. Thực nghiệm: 23](#_Toc130462707)

[3.3. Phân tích và đánh giá kết quả: 23](#_Toc130462708)

[4. Kết luận – hướng phát triển: 23](#_Toc130462709)

# Đặt vấn đề:

Trong cuộc sống ngày nay, con người ta luôn bận rộn. Những vấn đê như giặt đồ, dọn dẹp nhà cửa, hay đơn giản là điều chỉnh cửa sổ trời vào mỗi buổi sáng dường như là 1 việc tốn thời gian. Tại sao chúng ta không tự động hóa vấn đề ấy? Nhiều kỹ sự đã đáp ứng điều ấy, minh chứng là ta đã thấy được máy giặt, robot hút bụi, lau nhà, ... Để tiếp nối ý tưởng ấy, chúng tôi đã đưa ra sáng kiến tự động hóa việc điều khiển giếng trời, tích hợp thêm vào đó là hệ thống trạm thời tiết mini nhằm kiểm soát tốt hơn cho ngôi nhà của bạn.

# Tổng quan công trình:

Công trình Hệ thống giếng trời tự động tích hợp trạm thời tiết mini được dung để điều khiển cửa sổ trời một cách tự động: Hệ thống sẽ phân tích dữ liệu nhận từ các cảm biến và cho quyết định về việc đóng/mở giếng trời. Bên cạnh đó, ta cũng có thể theo dõi những thông số thời tiếttại nhà qua 1 trang web thời tiết do chip xử lý host lên.

# Mục tiêu – Phương pháp:

## MỤC TIÊU:

Gồm các mục tiêu chính sau:

* Điều khiển cửa sổ trời 1 cách tự động
* Giám sát các thông số thời tiết bên trong nhà

Hướng tới giải pháp được đưa ra bởi đề tài này sẽ ứng dụng thực tiễn được trong bất cứ căn nhà nào có tích hợp cửa sổ trời, nhưng việc phát triển lên những thứ khác thì vẫn còn là 1 thách thức lớn bởi vì so với mặt bằng chung thì đề tài vẫn còn đơn giản. Tuy nhiên, đây chính là sườn cơ bản để ta nắm bắt và phát triển.

## PHƯƠNG PHÁP:

Nhóm đã thực hiện 1 số phương pháp nghiên cứu để hình thành nên đề tài nên đề tài này. Cụ thể:

* Điều khiển giếng trời:

Tìm hiểu quá trình điều khiển giếng trời 1 cách thủ công, điều kiện để hầu hết mọi người đóng/mở giếng trời.

* Thông số thời tiết:

Tìm hiểu các giá trị đo lường thời tiết, mức độ ảnh hưởng của các gíá trị đó đến con người

* Thiết kế phần cứng:

Chúng tôi thiết kê mô phỏng ra 1 ngôi nhà mini có tích hợp các hệ thống trình bày trong bài báo cáo này.

* Viết chương trình phần mềm:

Chúng tôi viết 1 chương trình phần mềm để điều khiển cửa sổ trời, đèn trong nhà và liên kết dữ liệu của các cảm biến và trạng thái đèn, giếng trời với web thời tiết.

* Chạy thử nghiệm:

Chạy thử nghiệm, đánh giá sản phẩm làm được hoạt động thế nào, có đáp ứng nhu cầu người dùng hay không, có bị lỗi hay không, có hoạt động theo đúng trình tự được đưa ra hay không, … Xong việc chạy thử nghiệm giúp ta đưa ra sự hạn chế của sản phẩm từ đó chỉnh sửa lại, hoàn thiện máy.

# Kết quả - Thảo luận:

## Thiết kế phần cứng:

### Mô hình phần cứng:

### Động cơ:

Chúng tôi sử dụng động cơ RC Servo SG90 của hãng TOWER PRO

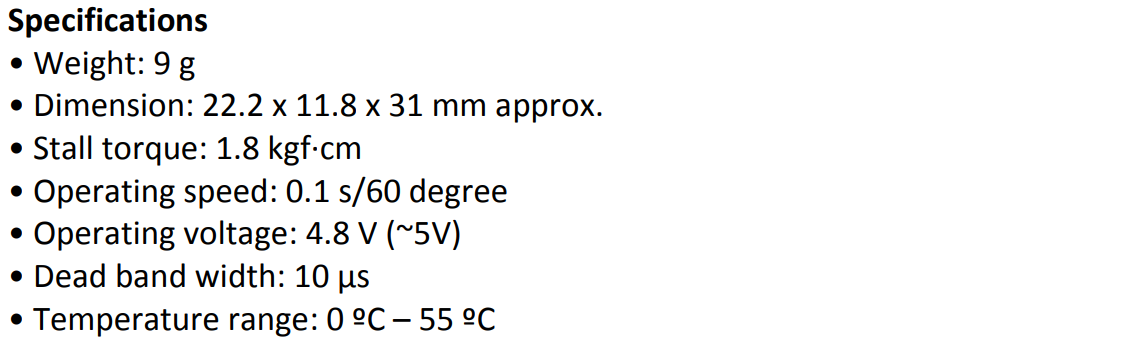
* Thông số kỹ thuật:
* Chức năng: mô phỏng động cơ điều khiển cửa sổ trời.

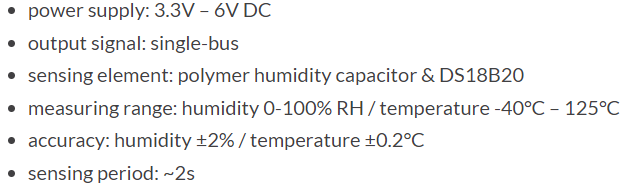


Figure 1 – Động cơ Servo

### Cảm biến – Thiết bị:

#### Cảm biến nhiệt độ - độ ẩm:

Chúng tôi sử dụng cảm biến DHT22.

* Thông số kỹ thuật:
* Chức năng: cung cấp dữ liệu nhiệt độ - độ ẩm.

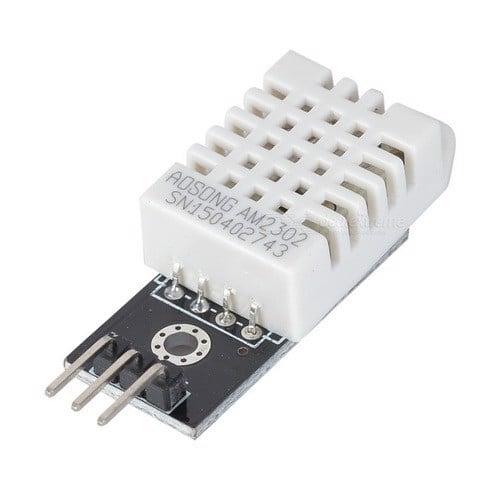
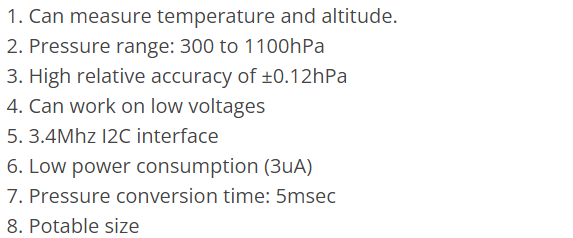


Figure 2 - Cảm biến nhiệt độ - độ ẩm

#### Cảm biến áp suất:

Chúng tôi sử dụng cảm biến GY 68 BMP180

* Thông số kỹ thuật:
* Chức năng: Cung cấp thông số áp suất.

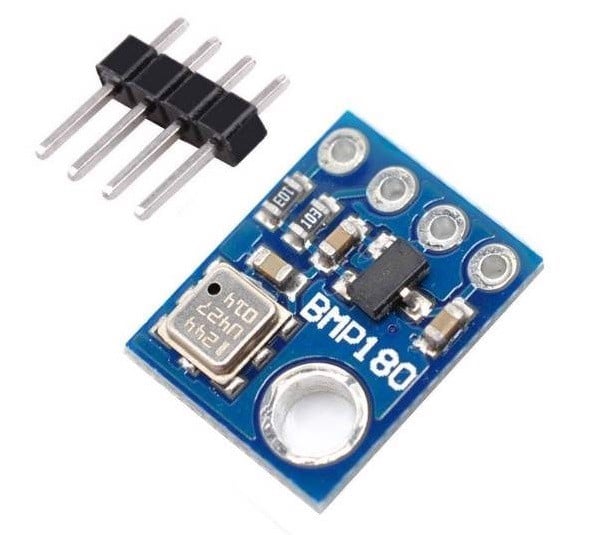


Figure 3 - Cảm biến áp suất

#### Cảm biến mưa:

Chúng tôi dùng cảm biến mưa loại module có tích hợp mạch chuyển đổi tín hiệu.

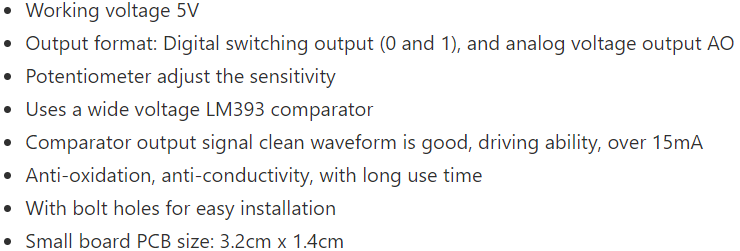
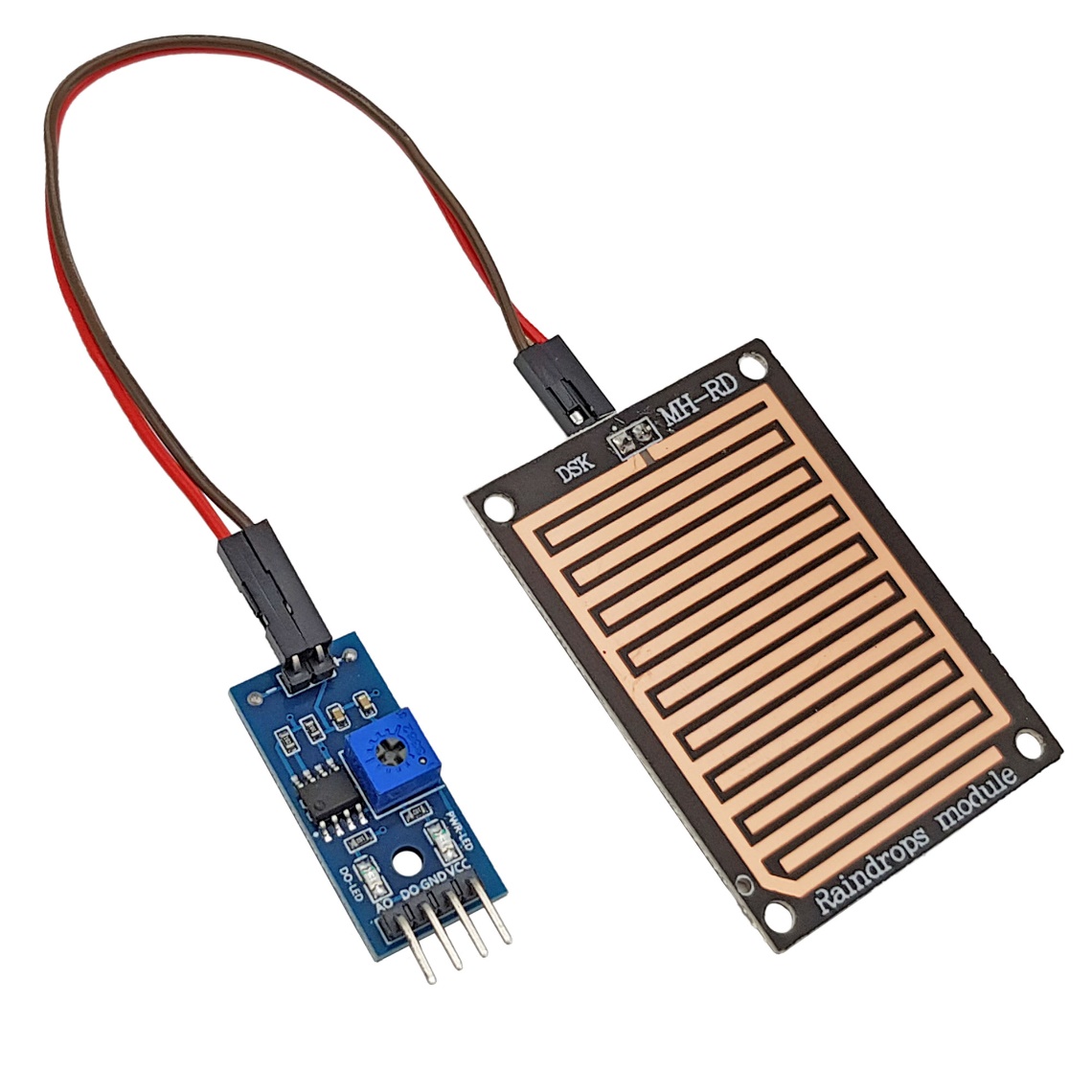
* Thông số kỹ thuật:
* Chức năng: Cung cấp thông số về lượng mưa.

Figure 4- Cảm biến lượng mưa

#### Cảm biến cường độ ánh sáng:

Chúng tôi dùng cảm biến ánh sáng GY-30 BH1750VI.

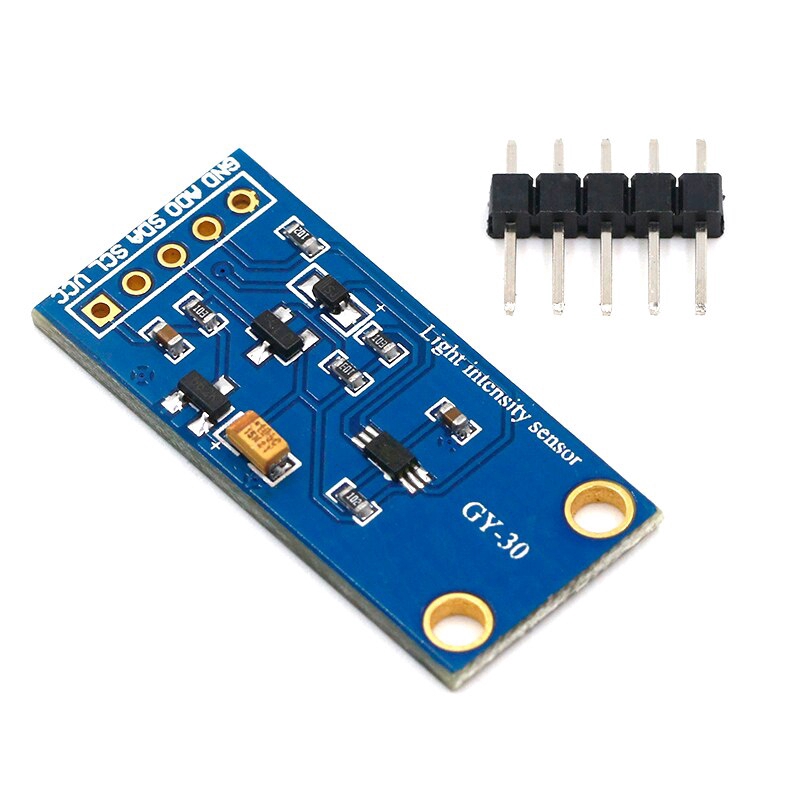
* Thông số kỹ thuật:
  + Working Voltage: 3.3V~6V
  + I2C bus Interface with 2 alternative address (f/s Mode Support)
  + Spectral responsibility is approximately human eye response
  + Wide range and High resolution (1 ~ 65535 lx)
  + Light source dependency is little. (ex. Incandescent Lamp. Fluorescent Lamp. Halogen Lamp. White LED. Sun Light)
  + Adjustable measurement result for influence of optical window (It is possible to detect min. 0.11 lx, max. 100000 lx by using this function.) (1 ~ 65535 lx)
  + The influence of infrared is very small.
  + Small measurement variation (+/- 20%)
* Chức năng: Cung cấp thông số về cường độ ánh sáng (lux)

Figure 5-Cảm biến cường độ ánh sáng

#### Cảm biến chạm:

#### Đèn led:

#### Vi điều khiển:

Chúng tôi sử dụng hệ thống chip điều khiển tích hợp phát Wifi 2.4Hz ESP32. Đây là một hệ thống vi điều khiển trên chip (SoC) giá rẻ của Espressif Systems, nhà phát triển của ESP8266 SoC. Do được cấu thành từ các thành phần RF tích hợp của nó như bộ khuếch đại công suất, bộ khuếch đại nhận tiếng ồn thấp, công tắc ăng-ten, bộ lọc và Balun RF, điều này làm cho việc thiết kế phần cứng xung quanh ESP32 rất dễ dàng vì bạn cần rất ít thành phần bên ngoài. Một điều quan trọng khác cần biết về ESP32 là nó được sản xuất bằng công nghệ 40 nm công suất cực thấp của TSMC. Vì vậy, việc thiết kế các ứng dụng hoạt động bằng pin như thiết bị đeo, thiết bị âm thanh, đồng hồ thông minh, ..., sử dụng ESP32 sẽ rất dễ dàng.

* Thông số kỹ thuật:
  + Bộ vi xử lý LX6 32-bit lõi đơn hoặc lõi kép với xung nhịp lên đến 240 MHz.
  + 520 KB SRAM, 448 KB ROM và 16 KB SRAM RTC.
  + Hỗ trợ kết nối Wi-Fi 802.11 b / g / n với tốc độ lên đến 150 Mbps.
  + Hỗ trợ cho cả thông số kỹ thuật Bluetooth v4.2 và BLE cổ điển.
  + 34 GPIO có thể lập trình.
  + 18 kênh SAR ADC 12 bit và 2 kênh DAC 8 bit
  + Kết nối nối tiếp bao gồm 4 x SPI, 2 x I2C, 2 x I2S, 3 x UART.
  + Ethernet MAC cho giao tiếp mạng LAN vật lý (yêu cầu PHY bên ngoài).
  + 1 bộ điều khiển host cho SD / SDIO / MMC và 1 bộ điều khiển slave cho SDIO / SPI.
  + Động cơ PWM và 16 kênh LED PWM.
  + Khởi động an toàn và mã hóa Flash.
  + Tăng tốc phần cứng mật mã cho AES, Hash (SHA-2), RSA, ECC và RNG.
* Chức năng:
  + Xử lý thuật toán điều khiển tự động đèn led và cửa sổ trời
  + Kết nối Wifi và host lên 1 trang web thời tiết bằng phương thức HTTP
  + Xử lý tín hiệu từ những cảm biến kết nối với nó và đưa dữa liệu lên web hiển thị.
  + Cấp nguồn cho các thiết bị ngoại vi.



Figure 6-Vi điều khiển ESP32

### Sơ đồ kết nối hệ thống:

## Thiết kế phần mềm:

### Tổng quan hệ thống:

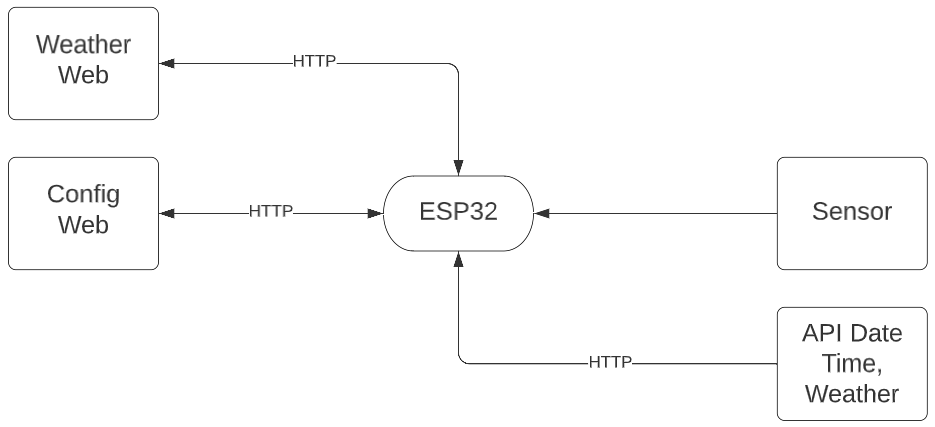
Chúng tôi đã xây dựng hệ thống dựa trên mô hình đơn giản như sau:

Figure 7- Mô hình hoạt động

Trong hệ thống này, ESP32 đóng vai trò như là bộ não xử lý tất cả những gì nhận được. Ta nói sơ qua 1 chút về thuật ngữ dùng trong sơ đồ trên:

* **Giao thức HTTP** (HyperText Transfer Protocol): nghĩa là **Giao thức Truyền tải Siêu Văn Bản** được sử dụng trong **www**. HTTP là 1 giao thức cho phép tìm nạp tài nguyên, chẳng hạn như HTML doc. Nó là nền tảng của bất kỳ sự trao đổi dữ liệu nào trên Web và cũng là giao thức giữa client (thường là các trình duyệt hay bất kỳ loại thiết bị, chương trình nào) và server (thường là các máy tính trên đám mây). 1 doc hoàn chỉnh được tái tạo từ các doc con khác nhau được fetch – tìm nạp, chẳng hạn như văn bản, mô tả layout, hình ảnh, video, script, …
* **API (**Application Programming Interface**)**: là các phương thức, giao thức kết nối với các thư viện và ứng dụng khác. API cung cấp khả năng truy xuất đến một tập các hàm hay dùng. Và từ đó có thể trao đổi dữ liệu giữa các ứng dụng. Trong đó web API là một phương thức dùng để cho phép các ứng dụng khác nhau có thể giao tiếp, trao đổi dữ liệu qua lại. Dữ liệu được Web API trả lại thưởng ở dưới dạng JSON hoặc XML thông qua giao thức HTTP hoặc HTTPS.

### Nguyên lý hoạt động:

Phần mềm của máy thanh toán gồm các phần chính sau:

* Giao diện người dùng: Giao diện của hệ thống được ESP32 đưa lên 1 web thông qua IP của router. Giao diện này được xây dựng dựa trên công nghệ web thông qua các ngôn ngữ như HTML, CSS.
* Back-end: Được xây dựng dựa trên ngôn ngữ C++, có chức năng điều hướng trang web, xử lý dữ liệu, giao tiếp với API
* Firmware (điều khiển phần cứng): Đây là một phần của back-end, cũng được viết bằng C++, chủ yếu được dùng để điều khiển các thiết bị phần cứng của máy như đầu đọc tiền, cảm biến, rơ-le,...

Nguyên lý chính:

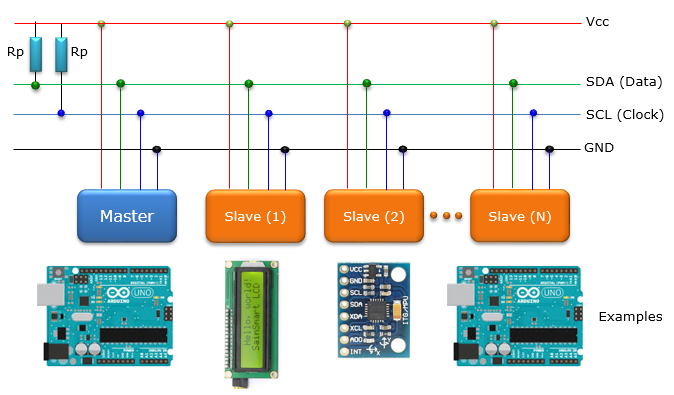
* **Giao tiếp giữa các cảm biến và ESP32:** Các cảm biến sẽ thu nhập những tác nhân từ bên ngoài như: ánh sáng, độ ẩm, … để xử lý chúng thành tín hiệu số. Sau đó chúng truyền những dữ liệu này thông qua những giao tiếp đặc trưng như I2C, SPI, … truyền đến chip xử lý.

Figure 8 - Chuẩn giao tiếp I2C

* **Giao tiếp giữa Front end (Weather Web) – Back end thông qua giao thức HTTP:**

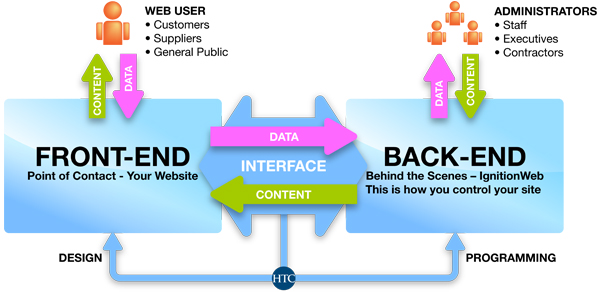


Figure 9 - Giao tiếp Front End - Back End

Esp32 sẽ tạo ra 1 máy chủ HTTP ảo và đưa UI (Giao diện người dùng) lên đó, ở hệ thống này là Weather Web. Nhiệm vụ chính của giao diện này là dùng để hiển thị thông tin về thời tiết, đồng thời dùng để lấy những Request(yêu cầu từ người dùng) truyền đến Back end, Back end sẽ xử lý logic và Response (phản hồi) về lại ngay sau đó.

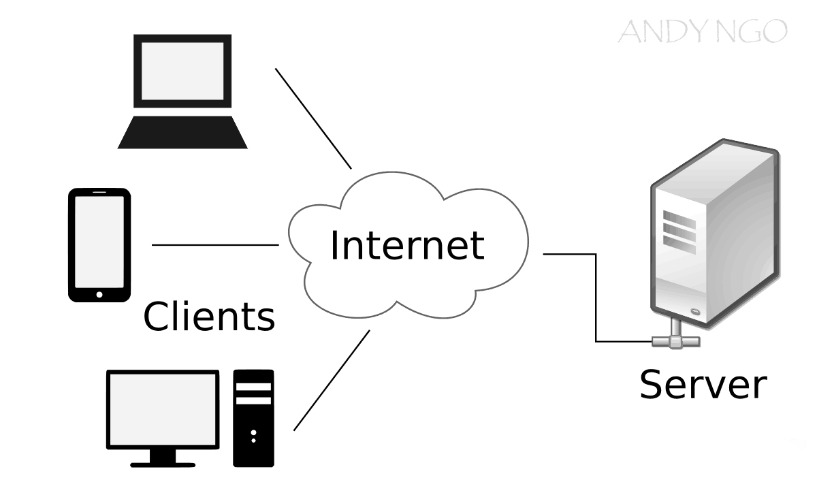
* **Giao tiếp back end – server thông qua API:**

Figure 10 - Mô hình Client - Server

Các dữ liệu như description (mô tả về thời tiết) hay real time (thời gian thực) được lấy từ những Server (máy chủ) khác. Những máy chủ này cung cấp những Phương thức để hiển thị những dữ liệu ấy, đồng thời cho phép những máy tính hay vi xử lý lên mạng được truy cập vào để lấy chúng, đây được gọi là API.

Các web API đầu vào thường là những URI, URL dẫn đến 1 địa chỉ nào đó và đầu ra sẽ trả về dữ liệu dưới dạng XML hay Json. Ở hệ thông này, ESP 32 sẽ lấy những dữ liệu cần thiết để hiện thị thêm thông tin về thời tiết lên Weather Web.

* **Lưu đồ thuật toán điều khiển đèn và Của sổ trời tự động:**

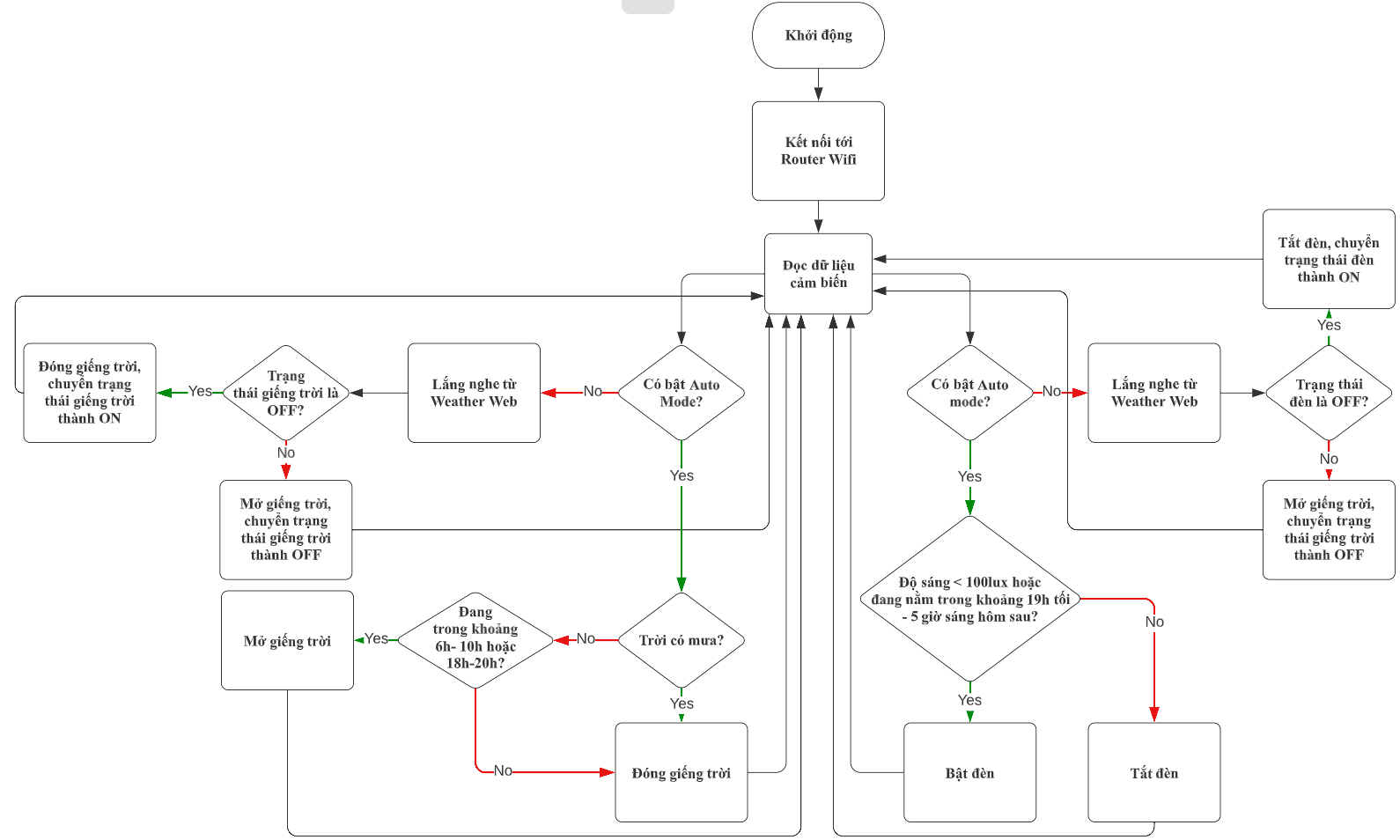


Figure 11- Lưu đồ giải thuật điều khiển đèn và cửa sổ trời tự động

### Lập trình điều khiển:

Mã nguồn được chúng tôi upload lên GitHub theo đường dẫn: [**https://github.com/sieunhan223/do-an-CB\_CCCH.git**](https://github.com/sieunhan223/do-an-CB_CCCH.git)

Thư mục của chúng tôi được phân chia như hình bên dưới:



#### Thư viện

Chúng tôi sử dụng những thư viện sau cho dự án:

* #include <Servo.h> : thư viện điều khiển động cơ Servo
* #include <HTTPClient.h> : Thư viện cho phép giao tiếp HTTP
* #include <DHT.h> : Thư viện cảm biến nhiệt đọ độ ẩm
* #include <SPIFFS.h> : Thư viện Upload file vào ESP32
* #include <ArduinoJson.h> : thư viện làm việc với Json
* #include "ESPAsyncWebServer.h" : Thư viện cho Web Server
* #include <WiFiClient.h> : Thư viện kết nối Wifi
* #include <WiFiAP.h> : Thư viện cho chế đọ Access Point
* #include <Arduino.h>: Thư viện Arduino
* #include <WiFi.h>: Thư viện kết nối Wifi
* #include <Wire.h>: Thư viện cho chuẩn kết nối I2C
* #include <BH1750.h>: Thư viện cho cảm biến ánh sáng
* #include <Adafruit\_BMP085.h>: Thư viện cho cảm biến áp suất

#### Hàm dùng cho hệ thống:

* **Hàm processor**: Thư viện ESPAsyncWebServer.h yêu cầu 1 hàm có tác dụng thay thế những cú pháp “%<giá trị>%” trong file **.html** là các biến có trong file .**ino**, cho nên ta cần tạo ra hàm như vậy và trả về kết quả là biến nào sẽ tương ứng với cú pháp nào trong file .html

String processor(const String &var)

{

  if (var == "nhiet")

    return String((nhiet));

  if (var == "doam")

    return String((doam));

  if (var == "id")

    return idWeather;

  if (var == "descriptionWeather")

    return descriptionWeather;

  if (var == "dateTime")

    return dateTime;

  if (var == "rainRate")

    return String(rainRate);

  if (var == "apsuat")

    return String(apsuat);

  if (var == "skylight")

    return String(skylightStatus);

  if (var == "light")

    return String(lightStatus);

  if (var == "modeColor")

    return modeColor;

  if (var == "mode")

    return mode;

  return String();

}

* **Hàm Đọc file:** Ta cần có 1 hàm có thể đọc được những dữ liệu có trong các file mà thư viện SPIFFS upload lên esp32:

String readFile(fs::FS &fs, const char \*path)

{

  File file = fs.open(path, "r");

  if (!file || file.isDirectory())

  {

    Serial.println("- empty file or failed to open file");

    return String();

  }

  String fileContent;

  while (file.available())

    fileContent = file.readStringUntil('\n');

  return fileContent;

}

* **Hàm viết file:** Ta cần có 1 hàm có thể viết được những dữ liệu có trong file .ino lên những file khác mà thư viện SPIFFS upload lên esp32:

void writeFile(fs::FS &fs, const char \*path, const char \*message)

{

  Serial.printf("Writing file: %s\r\n", path);

  File file = fs.open(path, "w");

  if (!file)

  {

    Serial.println("- failed to open file for writing");

    return;

  }

  if (file.print(message))

    Serial.println("- file written");

  else

    Serial.println("- frite failed");

  file.close();

}

* **Hàm điều khiển động cơ Servo**:

void controlServo(String status){

  if (status == "ON")

  {

    for (pos = curPos; pos <= 130; pos++)

    {

      myservo.write(pos);

      delay(10);

    }

    curPos = 130;

  }

  else

  {

    for (pos = curPos; pos >= 0; pos--)

    {

      myservo.write(pos);

      delay(10);

    }

    curPos = 0;

  }

}

#### Kết nối tới Wifi:

Chúng tôi kết nối ESP32 đến Wifi theo cách sau:

  Serial.println("Connecting...");

  WiFi.begin(ssid.c\_str(), password.c\_str());

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

  {

    Serial.print(".");

    delay(200);

  }

  Serial.println("Connected!");

  Serial.print("My IP: ");

  Serial.println(WiFi.localIP());

#### Lấy API từ Server khác:

// Lấy API từ http://worldtimeapi.org/api/timezone/Asia/Ho\_Chi\_Minh

  http.begin("http://worldtimeapi.org/api/timezone/Asia/Ho\_Chi\_Minh");

  cod = http.GET();

  if (cod == HTTP\_CODE\_OK)

  {

    String s\_timeApi = http.getString();

    DynamicJsonDocument doc1(1000);

    DeserializationError error1 = deserializeJson(doc1, s\_timeApi);

    dateTime\_data = doc1["datetime"];

    // format date time:

    String temp = "";

    for (int i = 0; i < strlen(dateTime\_data); i++)

    {

      if (dateTime\_data[i] == 'T')

        temp += " | ";

      else if (dateTime\_data[i] == '.')

        break;

      else if (dateTime\_data[i] == '-')

        temp += " ";

      else

        temp += dateTime\_data[i];

    }

    dateTime = temp;

  }

  else

    Serial.printf("date time-[HTTP] GET... failed, error: %s\n", http.errorToString(cod).c\_str());

  http.end();

  // Lấy API từ http://api.openweathermap.org/data/2.5/weather?lat=10.8230&lon=106.6296&appid=8e1880f460a20463565be25bc573bdc6

  http.begin("http://api.openweathermap.org/data/2.5/weather?lat=10.8230&lon=106.6296&appid=8e1880f460a20463565be25bc573bdc6");

  cod = http.GET();

  if (cod == HTTP\_CODE\_OK)

  {

    String s\_weatherApi = http.getString();

    // Serial.println(s\_weatherApi);

    DynamicJsonDocument doc2(1000);

    DeserializationError error2 = deserializeJson(doc2, s\_weatherApi);

    JsonObject obj = doc2.as<JsonObject>();

    String weather = obj["weather"][0]; // lấy dữ liệu tên Data

    deserializeJson(doc2, weather);     // Vì Data là 1 JSON cho nên phải fix data thêm 1 lần nữa

    JsonObject obj2 = doc2.as<JsonObject>();

    String descriptionWeatherg = obj2["description"];

    String idWeatherg = obj2["id"];

    descriptionWeather = descriptionWeatherg;

    descriptionWeather.toUpperCase();

    idWeather = idWeatherg;

    Serial.println(descriptionWeather);

  }

  else

    Serial.printf("weather-[HTTP] GET... failed, error: %s\n", http.errorToString(cod).c\_str());

  http.end();

#### Upload dữ liệu Front End lên mạng và Xử lý Resquest:



##### Upload dữ liệu Web Config:

  // Upload nội dung index2.html

  server.on("/config", HTTP\_GET, [](AsyncWebServerRequest \*request)

            { request->send(SPIFFS, "/index2.html", String(), false, processor); });

  // Upload nội dung file style2.css

  server.on("/style2.css", HTTP\_GET, [](AsyncWebServerRequest \*request)

            { request->send(SPIFFS, "/style2.css", "text/css"); });

  // Nếu có sự kiện cập nhật ssid và pwd:

  server.on("/config/Ok", HTTP\_POST, [](AsyncWebServerRequest \*request)

            {

      int count = request->params();

      for (int i = 0; i < count; i++)

      {

        AsyncWebParameter\* param = request->getParam(i);

        //HTTP POST get value ssid

        if (param->name() == PARAM\_INPUT\_SSID)

          ssid = param->value();

        writeFile(SPIFFS,"/ssid.txt",ssid.c\_str());

        //HTTP POST get value password

        if (param->name() == PARAM\_INPUT\_PWD)

          password = param->value();

        writeFile(SPIFFS,"/pwd.txt",password.c\_str());

      }

      restart = true;

      request->send(200, "text/plain", "Done. ESP will restart."); });

  server.begin();



##### Upload dữ liệu Weather Web:

  // Upload nội dung file index.html

  server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request)

            { request->send(SPIFFS, "/index.html", String(), false, processor); });

  // Nếu có sự kiện change mode:

  server.on("/checked", HTTP\_GET, [](AsyncWebServerRequest \*request)

            {

    if (request->hasParam(PARAM\_INPUT\_MODE)){

      if (mode == "ON"){

        mode = "OFF";

        modeColor = "Red";

        writeFile(SPIFFS,"/mode.txt","OFF");

        writeFile(SPIFFS,"/modeColor.txt","Red");

      }

      else{

        mode = "ON";

        modeColor = "Green";

        writeFile(SPIFFS,"/mode.txt","ON");

        writeFile(SPIFFS,"/modeColor.txt","Green");

      }

    }

    request->redirect("/"); });

  // Nếu có sự kiện skylight, light:

  server.on("/checkeddivice", HTTP\_GET, [](AsyncWebServerRequest \*request)

            {

        //Get value light, skylight status:

    if (request->hasParam(PARAM\_INPUT\_SKYLIGHT)) {

      skylightStatus = request->getParam(PARAM\_INPUT\_SKYLIGHT)->value();

      if (skylightStatus == "OFF"){

        skylightStatus = "ON";

        controlServo("OFF");

      }

      else if (skylightStatus == "ON"){

        skylightStatus = "OFF";

        controlServo("ON");

      }

      Serial.println(skylightStatus);

    }

    if (request->hasParam(PARAM\_INPUT\_LIGHT)) {

      lightStatus = request->getParam(PARAM\_INPUT\_LIGHT)->value();

      if (lightStatus == "OFF"){

        lightStatus = "ON";

        digitalWrite(RELAY,HIGH);

      }

      else if (lightStatus == "ON"){

        lightStatus = "OFF";

        digitalWrite(RELAY,LOW);

      }

      Serial.println(lightStatus);

    }

    request->redirect("/"); });

  // Upload nội dung file css

  server.on("/style.css", HTTP\_GET, [](AsyncWebServerRequest \*request)

            { request->send(SPIFFS, "/style.css", "text/css"); });

  server.begin();

#### Xử lý Logic:

  // Xử lý:

  if (mode == "ON")

  {

    processTime = dateTime.substring(13, 15).toInt();

    // Serial.println(processTime);

    if (rainRate < 40.0)

    {

      // Xử lý giếng trời

      if ((processTime >= 6 && processTime <= 10) || (processTime >= 18 && processTime <= 22))

      {

        Serial.println("OK");

        skylightStatus = "ON";

        controlServo("ON");

        // delay(10000);

      }

      else

      {

        Serial.println("No OK");

        skylightStatus = "OFF";

        controlServo("OFF");

        // delay(10000);

      }

    }

    else if (rainRate > 40.0)

    {

      Serial.println("No OK");

      skylightStatus = "OFF";

      controlServo("OFF");

      // delay(10000);

    }

    // Xử lý Đèn

    if ((processTime <= 5 && processTime >= 19) || lux < 100.0)

    {

      Serial.println("No OK");

      lightStatus = "ON";

      digitalWrite(RELAY, LOW);

    }

    else

    {

      Serial.println("OK");

      lightStatus = "OFF";

      digitalWrite(RELAY, HIGH);

    }

  }

## Tổng hợp hệ thống:

### Chế tạo phần cứng:

Để mô phỏng cho hệ thống, chúng tôi đã tạo ra 1 hình ngôi nhà đơn giản bằng cách ghép các que lưỡi gỗ lại với nhau. Kết nối các cảm biến và đặt chúng xung quanh nhà. Các cảm biến như cảm biến lượng mưa, cường độ ánh sáng được chúng tôi để ngoài trời, còn lại đều được đặt trong nhà nhằm lấy dữ liệu 1 cách tốt nhất và tránh hư hỏng vè lâu dài.

### Thực nghiệm:

### Phân tích và đánh giá kết quả:

Theo những gì thu được từ thực nghhiệm, chúng tôi đánh giá khách quan sản phẩm như sau:

* Ưu điểm:
  + Là hệ thống điều khiển tự động nên giúp ít cho con người trong việc thay vì phải đóng/ mở đèn hay giếng trời 1 cách thủ công thì ta có thể nhờ hệ thống làm thay việc ấy.
  + Hệ thống nhỏ gọn cho nên ta có thể tháo lắp đem đi kháp nơi 1 cách dễ dàng, có thể custom lại theo nhiều cách phù hợp với những nhu cầu khác nhau miễn là kết nối được với mạng.
  + Do có thể lên được mạng cho nên ta có thể lấy thêm nhiều thông tin về thời tiết trong khu vực (sức gió, tầm nhìn,…) mở rộng thêm cho hệ thống giúp ta có thể theo dõi cụ thể hơn về tình hình thời tiết mà không cần phải tốn chi phí cho việc mua thêm các cảm biến.
* Khuyết điểm:
* Tốc độ xử lý của chip phụ thuộc khá nhiều vào tốc độ phản hồi từ API của server.
* Phụ thuộc khá nhiều vào internet vô dụng nếu đem đi vào những môi trường không có sóng.
* Các cảm biến có giá thành rẻ nên chất lượng dữ liệu lấy được không cao, độ bền kém.

## Kết luận – hướng phát triển:

Nói tóm lại, Hệ thống điều khiển cửa sổ trời tự động tích hợp trạm thời tiết mini có đem lại những hiệu quả, lợi ích cho con người. Hệ thống giúp ta theo dõi trực tiếp những chỉ số về thời tiết ngay tại khu vực mà ta đặt chúng. Mặc khác hệ thống cong giúp ta theo dỏi và tự động điều khiển các thiết bị trong nhà 1 cách tự động mà ta không cần phải trực tiếp làm.

Hệ thống này cũng là nền tảng để chúng tôi phát triển lên 1 hệ thống có tính quy mô hơn đó là Hệ thống nhà thông minh – 1 Hệ thống quản lý, chăm sóc ngôi nhà 1 cách tự động ví dụ như: tưới cây tự động, vệ sinh nhà, bắt tắt các thiết bị trong nhà theo 1 tín hiệu nào đó từ chủ nhà, tự động hóa cho hệ thống pin mặt trời,…

Do vậy, Hệ thống điều khiển cửa sổ trời tự động tích hợp trạm thời tiết mini là 1 trong những công cụ, nền tảng để đưa ta tới thế giới 4.0 – Thế giới của tự động hóa và AI!