

**ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

□ □ □



**BÁO CÁO DỰ ÁN CUỐI KỲ  
CƠ SỞ ĐO LƯỜNG VÀ ĐIỀU KHIỂN SỐ**

**Hệ thống giám sát nông nghiệp thông minh**

Sinh viên thực hiện: **Nguyễn Trọng Nam**

**Lê Thé Minh**

**Cao Song Toàn**

**Doãn Đức Minh**

**Giảng Viên**

**TS. Phạm Duy Hưng**

**HANOI - 2025**

## Phân Chia Công Việc

<i>Họ và tên sinh viên</i>	<i>MSV</i>	<i>Công việc</i>	<i>Đóng góp</i>
Nguyễn Trọng Nam	22022161	<i>Xử lý cảm biến độ ẩm đất, Lập trình STM32 chính, Phân chia công việc nhóm, hỗ trợ làm báo cáo</i>	29%
Doãn Đức Minh	22022135	<i>Xử lý cảm biến ánh sáng, Làm báo cáo chính, hỗ trợ làm báo cáo</i>	20%
Lê Thanh Minh	22022215	<i>Xử lý cảm biến mưa, Triển khai truyền thông UART giữa 2 vi xử lý, hỗ trợ làm báo cáo</i>	23%
Cao Song Toàn	22022188	<i>Xử lý cảm biến nhiệt độ, độ ẩm, Lập trình ESP32, hỗ trợ làm báo cáo</i>	28%

# Mục Lục

<b>I. Giới Thiệu Chung</b>	<b>5</b>
1.1 Lý do chọn đề tài	5
1.2 Mục tiêu dự án	5
1.3 Phạm vi dự án	5
1.4 Công nghệ sử dụng	6
<b>II. Tổng quan hệ thống</b>	<b>6</b>
2.1 Kiến trúc tổng thể	6
2.2 Nguyên lý hoạt động	7
2.3 Sơ đồ khối	7
<b>III. Cơ sở lý thuyết</b>	<b>8</b>
3.1 Thông số cần giám sát	8
3.2 Nguyên lý cảm biến	9
3.3 STM32 ( Blackpill )	12
3.4 UART	13
3.4.1 Tổng quan	13
3.4.2 Truyền và nhận trong UART	14
a) UART Phát (Transmitting UART):	14
b) UART Nhận (Receiving UART):	14
3.4.3 Ưu điểm, nhược điểm	16
a) Ưu điểm	16
b) Nhược điểm	16
3.5 ESP32, Webserver, và ThingSpeak	16
3.5.1 ESP32	16
3.5.2 Web server	16
3.5.3 Vai trò của ESP32 trong Web server	17
3.5.3 Thingspeak	17
3.5.4 ESP32 và Thingspeak	18
3.6 Xử lý nhiễu và lọc tín hiệu	19
3.6.1 Nguồn nhiễu phổ biến trong hệ thống	19
<b>IV. Thiết kế hệ thống.</b>	<b>21</b>
4.1 Thu thập và xử lý cảm biến (STM32)	21
4.1.1 Cảm biến mưa:	21
a. Đọc tín hiệu từ cảm biến ( Sử dụng hàm ADC_Read)	21
b. Lọc nhiễu (Noise Filtering – Trung bình cộng 8 mẫu)	21
c. Hiệu chuẩn (Calibration – Quy đổi ADC về phần trăm độ ẩm)	22
d. Chuẩn hóa và truyền dữ liệu (UART Format)	23
4.1.2. Cảm biến ánh sáng:	23
a. Đọc tín hiệu từ cảm biến (Sử dụng hàm ADC_Read)	23
b. Lọc nhiễu (Noise Filtering – Trung bình cộng 10 mẫu)	24

c. Hiệu chuẩn (Calibration – Quy đổi ADC sang Lux)	25
d. Chuẩn hóa và truyền dữ liệu (UART Format)	26
4.1.3 Cảm biến soil moisture	26
a. Đọc tín hiệu từ cảm biến (ADC Read)	26
b. Lọc nhiễu (Noise Filtering – Moving Average)	27
c. Hiệu chuẩn (Calibration – Chuyển ADC về % độ ẩm đất)	27
d. Chuẩn hóa và truyền dữ liệu (UART Format)	28
4.1.4 Cảm biến DHT11	28
a. Đọc tín hiệu từ cảm biến	28
b. Lọc nhiễu	29
c. Chuẩn hóa và truyền dữ liệu (UART Format)	30
4.2 Giao tiếp UART	30
4.3 Webserver ESP32	31
4.4 Điều khiển bơm	32
<b>V. Thí nghiệm và đánh giá.</b>	<b>33</b>
5.1 Mục tiêu và Phương pháp Thí nghiệm	33
5.1.1 Mục tiêu	34
5.1.2 Phương pháp thực hiện	34
5.2 Môi trường và Cấu hình Thí nghiệm	34
5.3 Kết quả Đo lường và Phân tích	35
5.3.1 Bảng tổng hợp kết quả	35
5.3.2 Phân tích kết quả	36
5.4 Các Vấn đề Gặp phải và Phân tích Nguyên nhân	36
5.5 Đánh giá Tổng quan	37
<b>VI. Kết luận.</b>	<b>37</b>
<b>Phụ lục</b>	<b>38</b>
Ảnh mô hình thực tế	38
Sơ đồ mạch	39
Tài liệu tham khảo	39
Source code và video sản phẩm	40

# I. Giới Thiệu Chung

## 1.1 Lý do chọn đề tài

Thực tế cho thấy nông nghiệp truyền thống còn phụ thuộc chủ yếu vào kinh nghiệm và lao động thủ công, dẫn đến năng suất không ổn định, chi phí nhân công cao và khó khăn trong việc theo dõi các chỉ số môi trường như độ ẩm đất, nhiệt độ, ánh sáng; đồng thời, biến đổi khí hậu với các hiện tượng bão lũ, hạn hán ngày càng bất thường cùng nguồn nước dần cạn kiệt đã đặt ra yêu cầu bức thiết về một hệ thống giám sát và điều khiển tự động, kịp thời. Trong bối cảnh đó, công nghệ Internet of Things (IoT) nổi lên như một giải pháp tối ưu, bởi khả năng kết nối và thu thập dữ liệu liên tục theo thời gian thực, giúp người nông dân theo dõi các điều kiện của môi trường kết hợp với việc điều khiển máy bơm nước từ xa. Nhờ vậy, nông dân có thể đảm bảo cây trồng luôn được cung cấp đủ nước, tránh tình trạng ngập úng hoặc khô hạn, tiết kiệm đáng kể nguồn nước và nhân lực.

## 1.2 Mục tiêu dự án

Mục tiêu của dự án là xây dựng một hệ thống giám sát và điều khiển nông nghiệp thông minh, bao gồm: theo dõi liên tục các thông số môi trường như cường độ ánh sáng, lượng mưa, nhiệt độ, độ ẩm không khí và độ ẩm đất; hiển thị dữ liệu theo thời gian thực trên giao diện web và đồng bộ tự động lên nền tảng ThingSpeak để phân tích và lưu trữ; đồng thời cho phép người dùng điều khiển bơm nước từ xa thông qua webserver, đảm bảo quá trình tưới tiêu diễn ra chính xác, kịp thời và tiết kiệm tài nguyên.

## 1.3 Phạm vi dự án

Phạm vi dự án bao gồm việc thu thập và xử lý tín hiệu từ các cảm biến môi trường (ánh sáng, mưa, nhiệt độ, độ ẩm không khí và đất) thông qua vi điều khiển STM32, sau đó truyền dữ liệu đã xử lý lên module ESP32 qua giao tiếp UART. Trên ESP32 sẽ chạy một webserver để hiển thị giao diện quản lý và đồng thời gửi dữ liệu lên nền tảng đám mây (ThingSpeak) để lưu trữ và phân tích. Bên cạnh đó hệ thống

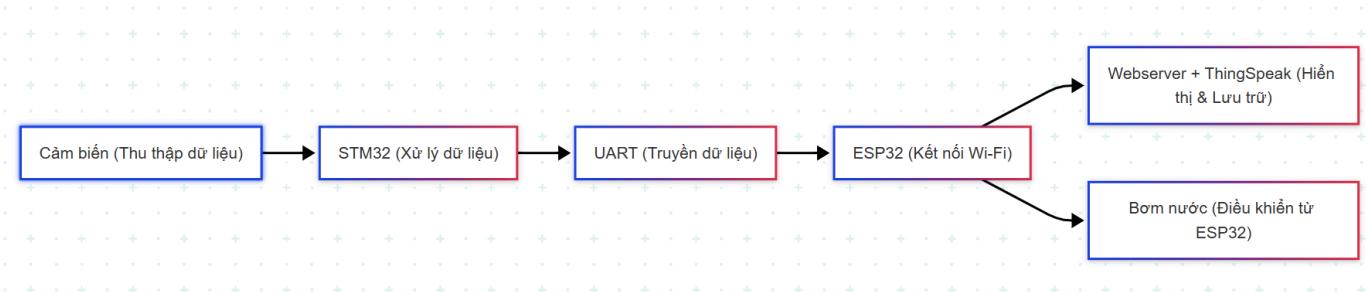
tích hợp chọn chế độ bơm nước thủ công trên webserver. Nhờ vậy, hệ thống không chỉ đảm bảo khả năng vận hành độc lập tại chỗ mà còn hỗ trợ truy cập và điều khiển từ xa qua nền tảng web.

## 1.4 Công nghệ sử dụng

Dự án sẽ sử dụng vi điều khiển STM32 làm bộ xử lý trung tâm, chịu trách nhiệm thu thập và xử lý tín hiệu từ các cảm biến qua giao thức UART; module ESP32 đảm nhiệm kết nối Wi-Fi, vận hành web server đồng thời tích hợp với nền tảng ThingSpeak để lưu trữ, phân tích và hiển thị dữ liệu; hệ thống cảm biến gồm DHT11 đo nhiệt độ và độ ẩm không khí, cảm biến mưa, cảm biến ánh sáng và cảm biến độ ẩm đất để giám sát đầy đủ các điều kiện môi trường; giao diện người dùng được xây dựng bằng HTML, CSS và JavaScript, cho phép hiển thị dữ liệu theo thời gian thực và điều khiển bơm nước bật/tắt từ xa một cách trực quan và linh hoạt.

# II. Tổng quan hệ thống

## 2.1 Kiến trúc tổng thể



Hệ thống được chia thành ba khối chức năng chính, tuần tự theo luồng dữ liệu như sơ đồ:

### 1. Thu thập dữ liệu (Sensors)

Các cảm biến môi trường (độ ẩm đất, nhiệt độ & độ ẩm không khí, ánh sáng, mưa) liên tục thu thập tín hiệu vật lý.

### 2. Xử lý và truyền dữ liệu (STM32 + UART)

Vì điều khiển STM32 nhận tín hiệu analog và digital từ các cảm biến, thực hiện lọc nhiễu và hiệu chuẩn, sau đó đóng gói thành chuỗi dữ liệu và gửi qua giao tiếp UART.

### 3. Giao tiếp, giám sát & điều khiển (ESP32)

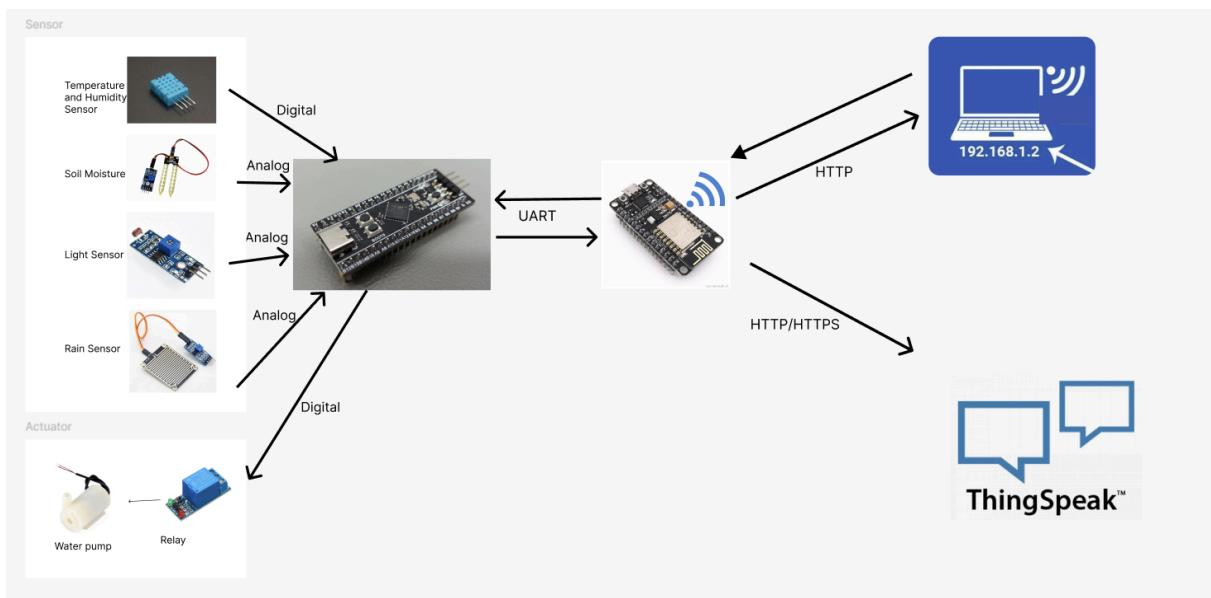
- **Webserver & ThingSpeak:** ESP32 nhận chuỗi dữ liệu UART, phân tích (parse) và cập nhật đồng thời lên giao diện webserver tại chỗ và gửi bản ghi lên ThingSpeak để lưu trữ, phân tích lâu dài.
- **Điều khiển bơm:** Khi người dùng tương tác trên giao diện web (nhấn nút bật/tắt), ESP32 phát lệnh điều khiển qua UART trở lại STM32, STM32 sau đó kích hoạt relay để bật hoặc ngắt bơm nước

Toàn bộ kiến trúc đảm bảo dữ liệu được thu thập – xử lý – hiển thị và điều khiển khép kín, cho phép giám sát thời gian thực và vận hành bơm từ xa một cách linh hoạt.

## 2.2 Nguyên lý hoạt động

Đầu tiên, STM32 sẽ đọc giá trị từ các cảm biến thông qua cổng ADC hoặc tín hiệu số, sau đó thực hiện xử lý, lọc nhiễu và hiệu chuẩn dữ liệu. Kết quả tất cả các cảm biến cuối cùng sẽ được đóng thành frame truyền đi dưới dạng chuỗi dữ liệu qua giao tiếp UART. ESP32 nhận và phân tích dữ liệu UART, hiển thị lên giao diện Web đồng thời gửi lên nền tảng ThingSpeak. Khi người dùng nhấn nút điều khiển bơm trên giao diện Web, lệnh sẽ được ESP32 gửi ngược lại STM32 qua UART để bật/tắt bơm nước.

## 2.3 Sơ đồ khái



Sơ đồ kiến trúc hệ thống thể hiện luồng dữ liệu và điều khiển như sau:

- Ở bên trái là các **cảm biến** môi trường:

1. **DHT11** (nhiệt độ & độ ẩm không khí) xuất tín hiệu số (digital).
  2. **Cảm biến độ ẩm đất, cảm biến ánh sáng (LDR)** và **cảm biến mưa** xuất tín hiệu tương tự (analog).
- Tất cả tín hiệu được đưa về **vi điều khiển STM32 (Blackpill)**:
    1. STM32 đọc tín hiệu analog qua ADC và tín hiệu digital qua GPIO, thực hiện lọc nhiễu, hiệu chuẩn rồi đóng gói thành chuỗi dữ liệu.
    2. Khi có lệnh điều khiển bơm, STM32 xuất chân GPIO digital để kích hoạt **relay điều khiển máy bơm nước**.
  - Dữ liệu sau khi đóng gói được truyền qua **UART** đến **ESP32**.
  - **ESP32** đảm nhận hai nhiệm vụ chính:
    1. **Web Server**: Nhận chuỗi UART, phân tích (parse) và hiển thị lên giao diện web cho người dùng (qua HTTP). Người dùng có thể bấm nút ON/OFF bơm ngay trên trang này.
    2. **Cloud**: Định kỳ gửi HTTP/HTTPS (GET hoặc POST) lên nền tảng ThingSpeak để lưu trữ và phân tích dữ liệu dài hạn.

Như vậy, hệ thống hoạt động khép kín: từ việc thu thập và xử lý tín hiệu cảm biến, truyền dữ liệu UART, đến hiển thị – lưu trữ trên web và cloud, và ngược lại nhận lệnh điều khiển bơm về STM32 để thực thi.

### **III. Cơ sở lý thuyết**

#### **3.1 Thông số cần giám sát**

##### **Độ ẩm đất**

- Xác định chính xác lượng nước còn lại trong lớp đất trồng, từ đó hệ thống tự động quyết định thời điểm và lượng nước cần tưới để duy trì độ ẩm tối ưu cho rễ cây, tránh tình trạng khô hạn hoặc ngập úng.

##### **Nhiệt độ không khí**

- Ánh hướng trực tiếp đến vận tốc bốc hơi nước và hoạt động sinh lý của cây (quá nóng sẽ gây stress nhiệt, quá lạnh làm chậm quá trình quang hợp). Thông tin này giúp điều

chỉnh lịch tưới và cảnh báo kịp thời khi nhiệt độ vượt ngưỡng an toàn.

### Độ ẩm không khí

- Phản ánh mức độ bão hòa hơi nước trong không khí, tác động đến khả năng thoát hơi và trao đổi khí CO<sub>2</sub>/O<sub>2</sub> trên bề mặt lá; từ đó hỗ trợ nông dân cân chỉnh biện pháp phun sương, che chắn hoặc điều chỉnh thông gió.

### Cường độ ánh sáng

- Là yếu tố then chốt cho quá trình quang hợp, quyết định tốc độ tổng hợp chất hữu cơ; dữ liệu ánh sáng giúp đánh giá xem cây đang nhận đủ ánh sáng hay cần biện pháp che nắng/chiếu sáng bổ sung.

### Lượng mưa

- Cập nhật mức nước tự nhiên cung cấp cho cây, giúp hệ thống tự động giảm hoặc tạm dừng tưới khi mưa đạt mức đủ, từ đó tiết kiệm nước và tránh tình trạng ngập úng.

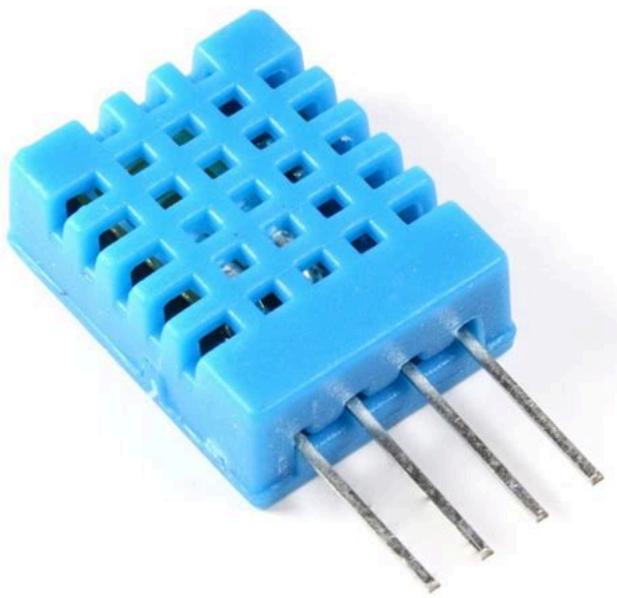
## 3.2 Nguyên lý cảm biến

### Cảm biến độ ẩm đất



- Hoạt động dựa trên nguyên lý thay đổi điện trở hoặc điện áp khi độ ẩm thay đổi. Khi đất ẩm, độ dẫn điện tăng lên dẫn đến điện trở giảm và ngược lại. Vi điều khiển STM32 đọc giá trị điện áp trên cảm biến qua kênh ADC, từ đó xác định chính xác mức độ ẩm trong đất.

### Cảm biến DHT11



DHT11 là cảm biến đo nhiệt độ và độ ẩm giá rẻ, thường được sử dụng trong các hệ thống IoT.

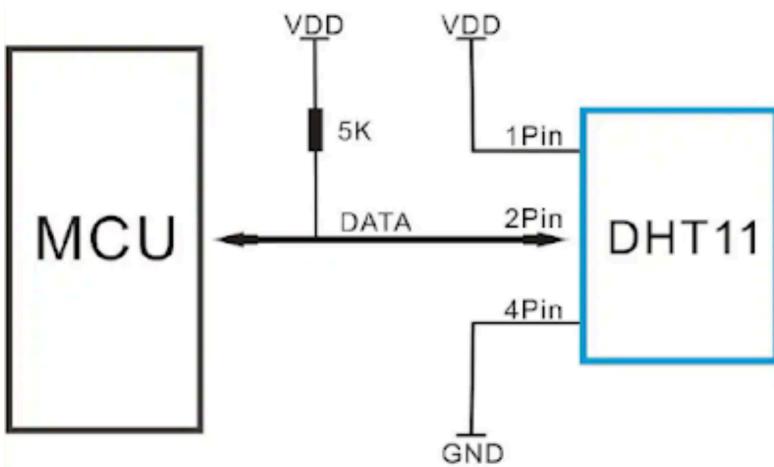
Cảm biến DHT11 bao gồm 2 thành phần để đo 2 đại lượng nhiệt độ và độ ẩm:

- Một nhiệt điện trở, dùng để đo nhiệt độ, với giá trị nhiệt độ tăng lên thì giá trị điện trở này sẽ giảm xuống
- Một cảm biến độ ẩm dạng điện dung

Với cấu tạo như trên bao gồm 2 thành phần, nhà sản xuất đã thêm 1 IC để xử lý dữ liệu nhận được từ cảm biến và chuyển đổi nó về dạng digital. Và cảm biến DHT11 xuất dữ liệu digital đó ra chân Data dưới dạng một chuẩn giao thức, gọi là chuẩn giao thức One-Wire.

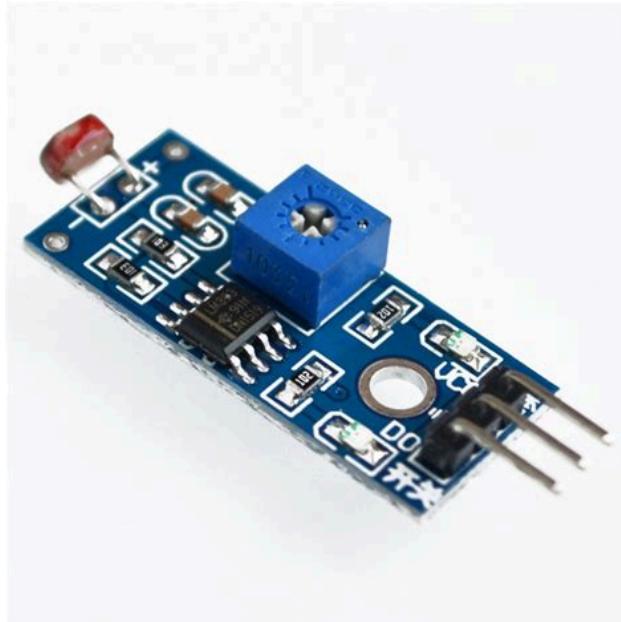
Chuẩn giao tiếp One-Wire: One-Wire là một chuẩn truyền thông nối tiếp, thường sử dụng để thu thập dữ liệu từ các cảm biến mà chỉ xuất dữ liệu ra một chân Data. Về cơ bản, One-Wire chỉ có 1 chân data, không có chân Clock để đồng bộ dữ liệu như các giao thức SPI, I2C, ... Nên nó cần một cách quy định các bit 0/1. Cụ thể, One-Wire sử dụng độ rộng khác nhau giữa một xung thấp và một xung cao liên tiếp nhau. Vì vậy có thể sẽ có những biến thể của giao thức One-Wire khi chúng ta thay đổi khoảng thời gian ở trên. Mỗi chip sử dụng chuẩn One-Wire có một ID duy nhất, vì vậy nó cũng có khả năng ứng dụng trong thực tế. Ví dụ: Bộ nguồn cho Laptop Dell dùng thêm 1 dây để truyền dữ liệu về năng lượng, dòng, áp về máy

tính thông qua giao thức One-Wire, và laptop sẽ không cho phép sạc nếu như bộ nguồn



không có dây này.

### Cảm biến ánh sáng (LDR)



- Là một biến trở quang học, điện trở của LDR giảm khi cường độ ánh sáng tăng và tăng khi ánh sáng giảm. Cảm biến được mắc theo cầu phân áp, vi điều khiển đọc điện áp tại điểm giữa qua ADC để xác định cường độ ánh sáng môi trường.

### Cảm biến mưa



- Gồm một tấm dẫn điện hoặc mạch in với các khe hở để tiếp nhận giọt nước mưa. Khi có nước, mạch dẫn điện giữa các khe tấm sẽ đóng, tạo ra tín hiệu analog hoặc ngõ ra số (tùy loại cảm biến). STM32 đọc tín hiệu này qua ADC hoặc GPIO để phát hiện và ước tính lượng mưa.

### 3.3 STM32 ( Blackpill )

#### Kiến trúc Cortex-M4

STM32 Blackpill sử dụng lõi ARM Cortex-M4 với đơn vị xử lý số học dấu chấm động (FPU), cho phép thực thi các phép toán số học nhanh chóng và hiệu quả hơn trong các ứng dụng đo lường và điều khiển thời gian thực.

#### Các ngoại vi ADC, UART và GPIO

- **ADC 12-bit:** Cho độ phân giải cao khi chuyển đổi tín hiệu analog từ các cảm biến (độ ẩm đất, ánh sáng, mưa...), giúp đảm bảo độ chính xác của dữ liệu thu thập.
- **UART:** Hỗ trợ truyền – nhận dữ liệu nối tiếp với module ESP32 ở tốc độ lên đến 115200 bps hoặc hơn, đảm bảo liên lạc ổn định giữa hai vi điều khiển.
- **GPIO:** Cung cấp nhiều chân vào/ra số để kết nối với cảm biến DHT11, relay điều khiển bơm và các thiết bị ngoại vi khác.

#### Lập trình thanh ghi (Register-Level Programming)

- Sử dụng thư viện mặc định vi điều khiển để cấu hình trực tiếp thanh ghi, tối ưu hóa hiệu năng và dung lượng firmware.
- Cho phép kiểm soát chi tiết từng bit của ngoại vi, phù hợp khi cần giảm độ trễ, tiết kiệm bộ nhớ và đạt hiệu suất cao nhất.

## **Ưu điểm nổi bật**

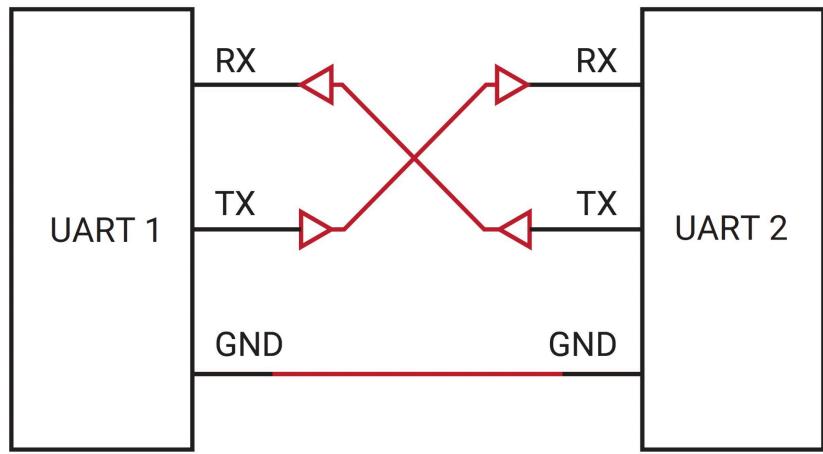
- **Hiệu năng vượt trội:** Lõi Cortex-M4 tích hợp bộ xử lý dấu phẩy động (FPU) cho phép thực thi nhanh chóng các phép toán số học và thuật toán điều khiển trong thời gian thực.
- **Chi phí hợp lý:** STM32 Blackpill có mức giá rất cạnh tranh so với các dòng vi điều khiển tương đương, phù hợp cho cả dự án cá nhân lẫn triển khai quy mô lớn.
- **Thiết kế nhỏ gọn:** Kích thước gọn nhẹ giúp dễ dàng tích hợp vào không gian hạn chế của các mô-đun IoT và thiết bị nhúng.

## **3.4 USART**

### **3.4.1 Tổng quan**

**USART** là viết tắt của **Universal Asynchronous Receiver/Transmitter**:

- Là giao thức truyền thông nối tiếp.
- Hai thiết bị USART giao tiếp trực tiếp với nhau qua chân TX và RX.
- USART truyền dữ liệu **không đồng bộ** — không có tín hiệu xung nhịp (clock) để đồng bộ hóa việc xuất bit từ USART gửi và việc lấy mẫu bit ở USART nhận.
- USART tự thêm **bit start** và **bit stop** vào gói dữ liệu được truyền, giúp USART nhận biết khi nào bắt đầu và kết thúc việc đọc các bit.



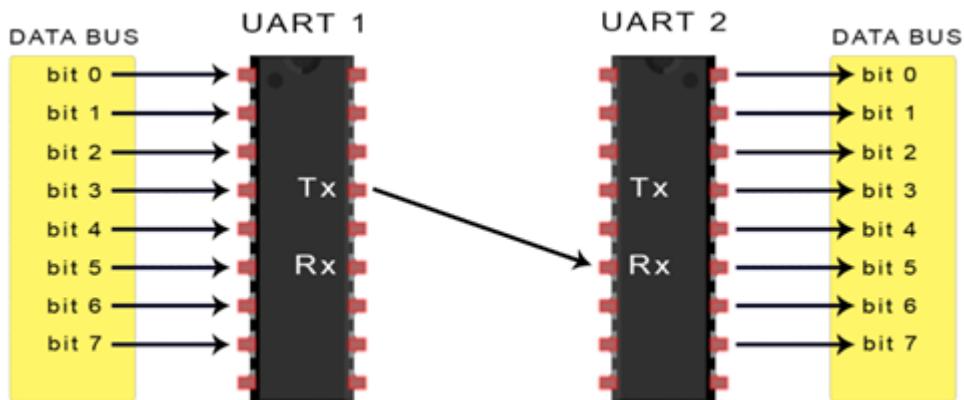
### 3.4.2 Truyền và nhận trong UART

#### a) UART Phát (Transmitting UART):

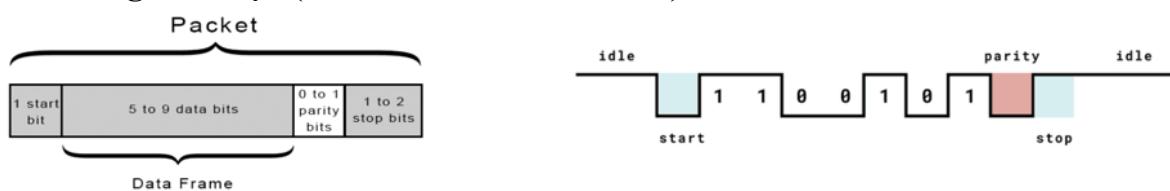
- UART phát dữ liệu, nhận dữ liệu từ bus dữ liệu.
- Dữ liệu được chuyển từ bus dữ liệu vào UART phát dưới dạng song song.
- UART phát thêm một **bit start**, một **bit parity** và một **bit stop** vào dữ liệu từ bus, tạo thành gói dữ liệu.
- Gói dữ liệu này được đưa vào **shift register**, rồi xuất ra nối tiếp, một bit mỗi lần, qua chân TX.

#### b) UART Nhận (Receiving UART):

- UART nhận đọc gói dữ liệu nối tiếp từng bit tại chân RX.
- Khi UART nhận phát hiện **bit start**, nó bắt đầu đọc các bit đến theo tần số đã định, gọi là **baud rate**.
- **Baud rate** quyết định tốc độ dữ liệu được gửi và nhận, tính bằng bit trên giây (bps).
- UART nhận lưu các bit đọc được vào **shift register**, sau đó chuyển ngược về dạng song song, đồng thời loại bỏ các bit start, parity và stop.
- Cuối cùng, UART nhận đưa gói dữ liệu song song này lên bus dữ liệu để xử lý tiếp.



## Cấu trúc gói dữ liệu (Structure of Data Packet)



### Start bit

- Để bắt đầu truyền dữ liệu, UART phát kéo đường truyền từ mức **cao** (logic 1) xuống **thấp** (logic 0) trong một chu kỳ xung.
- UART nhận phát hiện cạnh chuyển từ cao xuống thấp, rồi bắt đầu đọc các bit trong khung dữ liệu.

### Data frame

- Chứa **dữ liệu thực** được truyền, từ 5 đến 9 bit tùy cấu hình.

### Parity (bit kiểm tra lỗi)

- Dùng để kiểm tra tính **chẵn/lẻ** của tổng các bit 1 trong khung dữ liệu:
  - Even parity (chẵn):** bit parity = 0 → tổng số bit 1 trong data frame phải là số chẵn.
  - Odd parity (lẻ):** bit parity = 1 → tổng số bit 1 trong data frame phải là số lẻ.
- Nếu bit parity phù hợp với dữ liệu, coi như truyền không có lỗi.

### Stop bit

- Gồm 1 hoặc 2 bit ở mức **cao** (logic 1), dùng để báo hiệu **kết thúc** gói dữ liệu và quay về trạng thái idle trước khi khung tiếp theo bắt đầu.

### 3.4.3 Ưu điểm, nhược điểm

#### a) Ưu điểm

- Cài đặt đơn giản, chỉ cần 2 dây TX/RX và cấu hình baud rate.
- Phổ biến trên hầu hết MCU và module USB-UART, dễ tích hợp.
- Độ trễ thấp, phù hợp truyền dữ liệu điểm-điểm với tốc độ lên đến hàng Mbit/s.

#### b) Nhược điểm

- Không đồng bộ, phụ thuộc chính xác baud rate giữa hai bên.
- Chỉ point-to-point, khó mở rộng khi cần nhiều nút.
- Dễ nhiễu ở mức tín hiệu TTL, cần chuyển mức khi giao tiếp RS-232.

## 3.5 ESP32, Webserver, và ThingSpeak

### 3.5.1 ESP32

ESP32 là một **vi điều khiển tích hợp kết nối Wifi và Bluetooth** do Espressif Systems phát triển. Đây là dòng vi điều khiển SoC (System-on-Chip) mạnh mẽ, có:

Bộ xử lý dual-core Tensilica Xtensa LX6 tốc độ lên tới 240 MHz

Bộ nhớ SRAM và Flash tích hợp

Các **chân GPIO đa dụng**, hỗ trợ các giao thức phổ biến như UART, SPI, I<sup>2</sup>C, PWM, ADC, DAC

Kết nối mạng Wi-Fi theo chuẩn IEEE 802.11 b/g/n và Bluetooth v4.2 BLE

Nhờ tích hợp cả **năng lực xử lý** và **kết nối mạng**, ESP32 rất thích hợp cho các ứng dụng IoT như nhà thông minh, nông nghiệp thông minh, thiết bị đeo,...

### 3.5.2 Web server

Web server là một **ứng dụng cung cấp nội dung web (HTML/CSS/JS...)** cho người dùng thông qua giao thức **HTTP/HTTPS**. Nó hoạt động theo mô hình client-server, trong đó:

**Client (trình duyệt)** gửi HTTP request (yêu cầu truy cập trang web)

**Server (máy chủ)** phản hồi bằng HTTP response chứa nội dung cần thiết (webpage, dữ liệu)

Một web server có thể:

- Cung cấp giao diện điều khiển (bật/tắt thiết bị)
- Cung cấp dữ liệu cảm biến theo thời gian thực
- Xử lý các yêu cầu từ client và trả lại kết quả phù hợp

### 3.5.3 Vai trò của ESP32 trong Web server

ESP32 có thể đóng vai trò là một web server độc lập, tức là bản thân ESP32 vừa:

- **Thu thập dữ liệu từ cảm biến**
- **Chạy web server nhỏ** (nhờ thư viện như WebServer.h hoặc AsyncWebServer.h)
- **Phản hồi yêu cầu từ trình duyệt** truy cập IP nội bộ hoặc mạng LAN

**Nguyên lý hoạt động:**

1. ESP32 kết nối vào mạng Wi-Fi như một thiết bị bình thường (STA mode).
2. Tạo một server HTTP lắng nghe các request trên một cổng nhất định (thường là 80).
3. Khi trình duyệt truy cập địa chỉ IP của ESP32, ESP32 phản hồi bằng trang HTML chứa thông tin, giao diện điều khiển.
4. Khi người dùng tương tác (ví dụ nhấn nút), trình duyệt gửi HTTP request, ESP32 xử lý và thực hiện hành động tương ứng (bật/tắt relay, ghi dữ liệu, ...).

### 3.5.3 Thingspeak



**ThingSpeak** là một **nền tảng IoT mã nguồn mở** cho phép thu thập, lưu trữ, phân tích và trực quan hóa dữ liệu cảm biến từ xa. Nó được phát triển bởi MathWorks – công ty đứng sau MATLAB và Simulink.

Các thành phần chính

- **Channel:** Mỗi kênh (channel) chứa tối đa 8 trường dữ liệu (field1 đến field8), cùng với thời gian, vị trí và trạng thái.
- **API Key:**
  - Write API Key: để gửi dữ liệu (POST)
  - Read API Key: để đọc dữ liệu (GET)
- **Giao thức hỗ trợ:** HTTP, MQTT, MATLAB Analysis, Time Control, ThingHTTP

### 3.5.4 ESP32 và Thingspeak

**Nguyên lý hoạt động**

1. ESP32 gửi dữ liệu lên ThingSpeak thông qua giao thức HTTP hoặc MQTT.
2. ThingSpeak lưu trữ dữ liệu trong kênh tương ứng.
3. Giao diện web của ThingSpeak hiển thị biểu đồ thời gian thực.
4. Người dùng có thể:
  - Truy vấn lại dữ liệu lịch sử

- Cài đặt cảnh báo
- Viết script MATLAB để phân tích dữ liệu ngay trên cloud

## 3.6 Xử lý nhiễu và lọc tín hiệu

### 3.6.1 Nguồn nhiễu phổ biến trong hệ thống

Trong quá trình đo đặc tín hiệu từ cảm biến ngoài thực tế, hệ thống thường gặp phải nhiều loại nhiễu ánh hưởng đến độ chính xác của dữ liệu, bao gồm:

- **Nhiễu điện từ (EMI):** Do dây dẫn dài, động cơ điện, relay đóng cắt.
- **Nhiễu xung (spike noise):** Từ cảm biến cơ học như cảm biến mưa có tiếp xúc không ổn định.
- **Đao động điện áp nguồn:** Khi Vcc không ổn định, gây lệch kết quả ADC.

### 3.6.2 Kỹ thuật lọc đã triển khai

Lọc trung bình trượt (Moving Average Filter)

Lọc trung bình trượt (Moving Average Filter) là một bộ lọc số học đơn giản và hiệu quả, dùng để làm mượt tín hiệu bằng cách tính **trung bình cộng của N mẫu gần nhất**. Mỗi khi có một mẫu mới, mẫu cũ nhất sẽ bị loại bỏ, và giá trị mới được thêm vào để cập nhật trung bình.

Bộ lọc này **đặc biệt hữu ích trong các hệ thống đo tín hiệu analog** từ cảm biến môi trường – vốn thường nhiễu do xung điện, dao động nguồn hoặc yếu tố ngoại cảnh.

**Công thức toán học**

Cho chuỗi tín hiệu đầu vào:

$x_0, x_1, \dots, x_{n-1}$

Giá trị đầu ra tại thời điểm  $n$  là trung bình của  $N$  mẫu gần nhất:

$$\bar{X}_n = \frac{1}{N} \sum_{i=n-N+1}^n x_i$$

Trong lập trình nhúng (embedded), thường sử dụng **mảng vòng (circular buffer)** để tối ưu hóa hiệu năng:

$$\bar{X} = \frac{\text{Tổng các giá trị trong buffer}}{N}$$

Ưu điểm:

- Đơn giản, dễ lập trình
- Giảm nhiễu hiệu quả, làm mượt tín hiệu
- Tiết kiệm tài nguyên, không cần phần cứng bổ sung
- Phù hợp với tín hiệu thay đổi chậm (như độ ẩm, ánh sáng)

### Lọc trung vị (Median Filter) – DHT11:

Lọc trung vị là một kỹ thuật lọc phi tuyến (non-linear), hoạt động bằng cách:

- Lưu một số lượng mẫu đo gần nhất ( $N$  mẫu)
- Sắp xếp các mẫu này theo thứ tự tăng dần
- Lấy giá trị ở giữa (median) làm đầu ra thay vì trung bình cộng

### Công thức toán học:

Cho dãy tín hiệu:

$x_1, x_2, \dots, x_n$

Lọc trung vị tính:

$$\bar{X}_{\text{median}} = \text{median}(x_1, x_2, \dots, x_n)$$

Ví dụ: với 5 mẫu  $[22, 25, 21, 100, 23] \rightarrow$  sau khi sắp xếp  $[21, 22, 23, 25, 100]$   
 $\rightarrow$  giá trị lọc trung vị là 23 (đơn định, loại bỏ được giá trị nhiễu 100).

Ưu điểm:

- Chống nhiễu đột biến rất tốt
- Không bị ảnh hưởng bởi giá trị ngoại lai (outlier)
- Giữ nguyên biên độ của tín hiệu hợp lệ (không làm mượt quá mức)

## IV. Thiết kế hệ thống.

### 4.1 Thu thập và xử lý cảm biến (STM32)

#### 4.1.1 Cảm biến mưa:

Việc thu thập và xử lý tín hiệu từ cảm biến mưa được thực hiện qua bốn bước chính: đọc tín hiệu ADC, lọc nhiễu tín hiệu, hiệu chuẩn giá trị ADC về phần trăm độ ẩm mưa (% Rain Moisture) và chuẩn hóa dữ liệu đầu ra thành chuỗi UART. Toàn bộ quá trình được xây dựng trong một file xử lý riêng biệt, đảm bảo khả năng mở rộng, tái sử dụng và bảo trì hiệu quả.

##### a. Đọc tín hiệu từ cảm biến ( Sử dụng hàm ADC\_Read)

Cảm biến mưa hoạt động trên nguyên lý thay đổi điện trở theo mức độ ẩm trên bề mặt. Tín hiệu đầu ra là điện áp analog, được đưa vào bộ chuyển đổi tương tự-số (ADC) của vi điều khiển STM32. Mỗi lần đo, ADC trả về một giá trị số nguyên 12-bit trong dải từ **0 đến 4095**, phản ánh điện áp đầu vào theo công thức:

$$V_{in} = (\text{ADC\_value} / 4095) \times V_{ref}$$

Với  $V_{ref} \approx 3.3V$ , giá trị ADC phản ánh độ khô hay ướt của bề mặt cảm biến.

##### b. Lọc nhiễu (Noise Filtering – Trung bình cộng 8 mẫu)

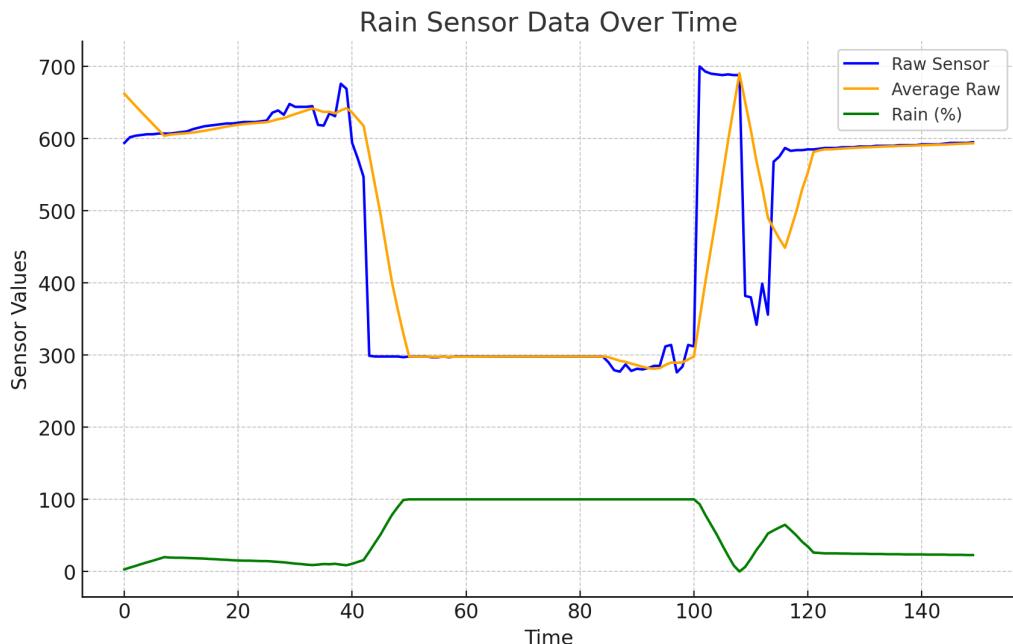
Tín hiệu analog trong môi trường thực tế thường có nhiễu, đặc biệt là khi mưa rơi không đều hoặc cảm biến bị ảnh hưởng bởi rung động/độ ẩm không khí. Để ổn định đầu ra, hệ thống sử dụng hàm **static float rain\_filter(uint16\_t sample)** là bộ lọc trung bình động **8 mẫu gần nhất** (Moving Average Filter), theo công thức:

$$\bar{X} = \frac{1}{N} \sum_{i=1}^N x_i$$

Trong đó:

- $N = 8$  là số mẫu được lưu trữ liên tục trong một buffer vòng lặp,
- $x_i$  là giá trị ADC thô tại thời điểm đo thứ  $i$ ,
- $\bar{X}$  là giá trị ADC trung bình sau khi lọc.

Việc lọc giúp loại bỏ các biến thiên đột ngột, đồng thời làm mượt tín hiệu đầu ra trước khi hiệu chuẩn.



Hình 4.4.1

Chọn bộ lọc Moving Average do nhiều dao động xung quanh 1 giá trị không quá là nhiều với window size là  $N$ ,  $N$  càng cao thì đáp ứng cho sự thay đổi khá là chậm,  $N$  càng thấp thì khả năng lọc nhiều không thay đổi nhiều. Vậy nên chọn  $N = 8$  cho phần này và xem xét kết quả thu được như ở trong đồ thị. ngay khi giá trị ADC và Average Raw xuống dưới khoảng xung quanh 300 ngay lập tức rain% tăng lên  $\Rightarrow$  hệ thống ổn định với  $N = 8$

### c. Hiệu chuẩn (Calibration – Quy đổi ADC về phần trăm độ ẩm)

Sau khi lọc nhiễu, sử dụng hàm **float rain\_percent** giúp giá trị ADC được chuyển đổi thành **phần trăm độ ẩm mura**, theo mốc hiệu chuẩn đã được thực nghiệm:

- RAW\_DRY=672: tương ứng trạng thái **khô hoàn toàn (0%)**. Giá trị này đã được đo ở điều kiện khô hoàn toàn và lấy trung bình của 20 mẫu.

- RAW\_WET=326: tương ứng trạng thái **ướt hoàn toàn (100%)**. Giá trị này đã được đo ở điều kiện ướt hoàn toàn và lấy trung bình của 20 mẫu.

Việc ánh xạ tuyến tính được thực hiện theo công thức:

$$\text{Rain\_Percent} = (\text{RAW\_DRY} - \text{ADC\_filtered}) / (\text{RAW\_DRY} - \text{RAW\_WET}) \times 100\%$$

Trong đó:

- ADC\_filtered là giá trị ADC đã được lọc nhiễu,
- Nếu giá trị vượt ra ngoài khoảng [RAW\_WET, RAW\_DRY] sẽ bị ép về biên để tránh sai số.

( Hình 4.1.1 là tổng hợp dữ liệu cảm biến trước lọc sau lọc và sau hiệu chuẩn )

Ví dụ: nếu ADC đọc được giá trị 334, thì:

$$\text{Rain\_Percent} = (672 - 334) / (672 - 326) \times 100\% \approx 97.1\%$$

#### d. Chuẩn hóa và truyền dữ liệu (UART Format)

Sau khi chuyển đổi sang đơn vị phần trăm, giá trị độ ẩm được chuẩn hóa thành một **chuỗi ký tự UART** để truyền qua ESP32, theo định dạng:

$$\text{UART\_String} = "xx\n"$$

Trong đó:

- xx là giá trị phần trăm độ ẩm được lấy nguyên,
- \n là ký tự kết thúc chuỗi dùng để phân biệt giữa các bản ghi khi truyền nối tiếp.

Chuỗi UART sau đó được truyền tới module ESP32 thông qua USART2 (PA2, PA3) để hiển thị trên webserver và gửi lên nền tảng ThingSpeak.

### 4.1.2. Cảm biến ánh sáng:

#### a. Đọc tín hiệu từ cảm biến (Sử dụng hàm ADC\_Read)

Cảm biến LDR được mắc vào mạch phân áp, và đầu ra của mạch được nối vào chân **PA0** của STM32 – sử dụng kênh **ADC1\_IN0** để chuyển đổi tín hiệu điện áp analog thành giá trị số. Mỗi lần đo, ADC trả về một giá trị số nguyên 12-bit trong dải từ **0** đến **4095**

Trong đó, giá trị càng nhỏ khi ánh sáng càng mạnh (vì điện trở LDR giảm), và càng lớn khi ánh sáng yếu (vì LDR tăng điện trở).

### Công thức tổng quát điện áp đầu vào:

$$V_{in} = (\text{ADC\_value} / 4095) \times V_{ref}$$

$V_{ref}=3.3V$  là điện áp tham chiếu.

### b. Lọc nhiễu (Noise Filtering – Trung bình cộng 10 mẫu)

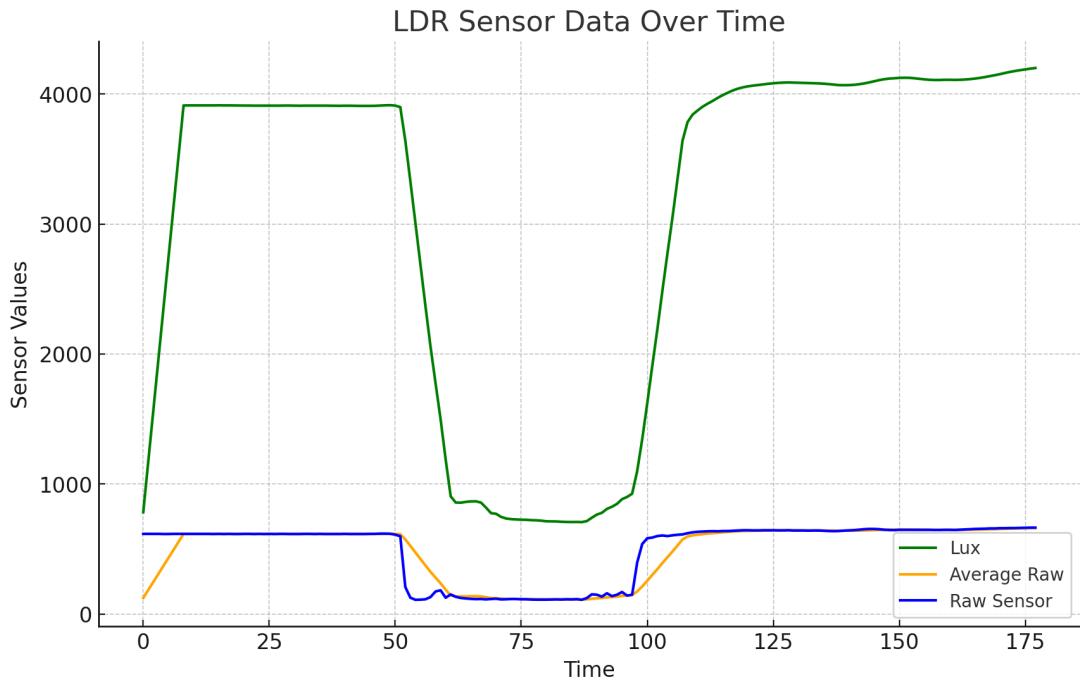
Tín hiệu analog trong môi trường thực tế thường có nhiễu, đặc biệt là khi mưa rơi không đều hoặc cảm biến bị ảnh hưởng bởi rung động/độ ẩm không khí. Để ổn định đầu ra, hệ thống sử dụng hàm **static float rain\_filter(uint16\_t sample)** là bộ lọc trung bình động **10 mẫu gần nhất** (Moving Average Filter), theo công thức:

$$\bar{X} = \frac{1}{N} \sum_{i=1}^N x_i$$

Trong đó:

- $N = 10$  là số mẫu được lưu trữ liên tục trong một buffer vòng lặp,
- $x_i$  là giá trị ADC thô tại thời điểm đo thứ  $i$ ,
- $\bar{X}$  là giá trị ADC trung bình sau khi lọc.

Việc lọc giúp loại bỏ các biến thiên đột ngột, đồng thời làm mượt tín hiệu đầu ra trước khi hiệu chuẩn.



Hình 4.1.2

Chọn bộ lọc Moving Average do nhiều dao động xung quanh 1 giá trị không quá là nhiều với window size là N, N càng cao thì đáp ứng cho sự thay đổi khá là chậm, N càng thấp thì khả năng lọc nhiều không thay đổi nhiều. Vậy nên chọn N = 10 cho phần này và xem xét kết quả thu được như ở trong đồ thị, ngay khi giá trị ADC và Average\_Raw xuống dưới khoảng xung quanh 1000 ngay lập tức lux cũng giảm xuống dưới 1000 => hệ thống ổn định với N = 10

### c. Hiệu chuẩn (Calibration – Quy đổi ADC sang Lux)

Sau khi lọc nhiễu, giá trị ADC được chuyển đổi về đơn vị đo cường độ ánh sáng (**Lux**) theo mô hình tuyến tính. Quá trình này dựa trên hai mốc hiệu chuẩn đã được xác định:

- **rawMin = 0**: tương ứng với 0 lux (tối hoàn toàn). Giá trị này đã được đo ở điều kiện đèn trong phòng tắt hoàn toàn và lấy trung bình của 20 mẫu.
- **rawMax = 700**: tương ứng với 6500 lux (sáng mạnh nhất có thể đo được). Giá trị này đã được đo ở điều kiện đèn trong phòng bật sáng hết lên và lấy trung bình của 20 mẫu.

Ánh xạ tuyến tính được thực hiện theo công thức:

$$\text{Lux} = ((\bar{X} - \text{rawMin}) / (\text{rawMax} - \text{rawMin})) \times (\text{luxMax} - \text{luxMin}) + \text{luxMin}$$

Trong đó:

- $\bar{X}$ : là giá trị ADC sau lọc.
- $\text{luxMax}=6500$ ,  $\text{luxMin}=0$  ( 2 giá trị được chọn sau khi lấy trung bình 20 mẫu của lần lượt khi đèn trong phòng sáng nhất, khi tắt hoàn toàn đèn)

Để tránh vượt ngưỡng, giá trị lux sau cùng được ép về trong đoạn:

$$\text{Lux} \in [0, 6500]$$

( Hình 4.1.2 là tổng hợp dữ liệu cảm biến trước lọc sau lọc và sau hiệu chuẩn )

#### d. Chuẩn hóa và truyền dữ liệu (UART Format)

Sau khi tính toán, giá trị lux và ADC thô sẽ được in ra qua giao tiếp UART2 để gửi tới ESP32 hoặc thiết bị hiển thị/log dữ liệu.

Dữ liệu được chuẩn hóa dưới dạng chuỗi có cấu trúc:

Raw	Lux
358	2163.4

Chuỗi có thể được truyền mỗi 200ms, phù hợp với việc giám sát ánh sáng thời gian thực.

### 4.1.3 Cảm biến soil moisture

Cảm biến độ ẩm đất giúp xác định lượng nước còn lại trong đất, từ đó hệ thống có thể quyết định thời điểm tưới phù hợp. Trong hệ thống, cảm biến này xuất tín hiệu analog và được xử lý qua 4 bước: đọc ADC, lọc nhiễu, hiệu chuẩn tuyến tính về phần trăm độ ẩm, và truyền UART.

#### a. Đọc tín hiệu từ cảm biến (ADC Read)

Cảm biến độ ẩm đất hoạt động theo nguyên lý: **khi đất càng ẩm thì độ dẫn điện càng cao**, tần số điện áp đầu ra càng lớn. Tín hiệu được kết nối tới **kênh ADC** của vi điều khiển STM32. Mỗi lần đo, ADC trả về một giá trị số nguyên 12-bit trong dải từ **0 đến 4095**.

**Công thức tổng quát điện áp đầu vào:**

$$V_{in} = (\text{ADC\_value} / 4095) \times V_{ref}$$

$V_{ref}=3.3V$  là điện áp tham chiếu.

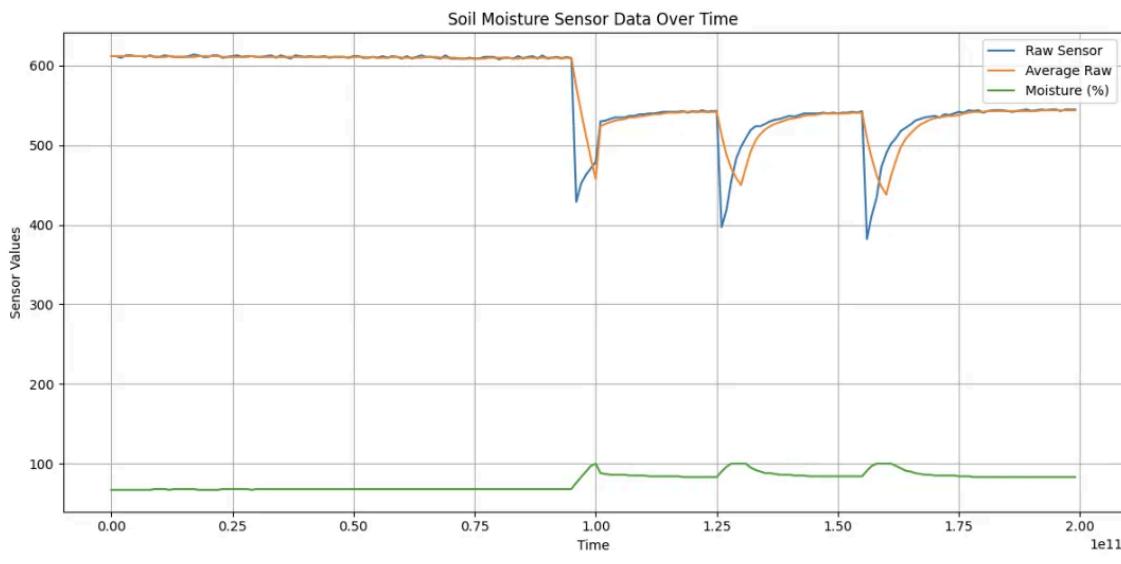
### b. Lọc nhiễu (Noise Filtering – Moving Average)

Do tín hiệu đất dễ bị ảnh hưởng bởi môi trường (rung động, dao động điện), cần áp dụng bộ lọc trung bình động để làm mượt tín hiệu:

$$\bar{X} = \frac{1}{N} \sum_{i=1}^N x_i$$

Trong đó:

- $N = 5$  là số mẫu được lưu trữ liên tục trong một buffer vòng lặp,
- $x_i$  là giá trị ADC thô tại thời điểm đo thứ  $i$ ,
- $\bar{X}$  là giá trị ADC trung bình sau khi lọc.



Hình 4.1.3

Chọn bộ lọc Moving Average do nhiễu dao động xung quanh 1 giá trị không quá là nhiều với window size là  $N$ ,  $N$  càng cao thì đáp ứng cho sự thay đổi khá là chậm,  $N$  càng thấp thì khả năng lọc nhiễu không thay đổi nhiều. Vậy nên chọn  $N = 5$  cho phần này và xem xét kết quả thu được

### c. Hiệu chuẩn (Calibration – Chuyển ADC về % độ ẩm đất)

Sau khi lọc, giá trị ADC được quy đổi về thang **phần trăm độ ẩm đất (%)** dựa vào hai mốc thực nghiệm:

- **DRY\_VALUE** = 908 – Giá trị ADC khi đất khô (0%). Giá trị này đã được đo ở điều kiện khô hoàn toàn và lấy trung bình của 20 mẫu.
- **WET\_VALUE** = 472 – Giá trị ADC khi đất ướt (100%). Giá trị này đã được đo ở điều kiện khô hoàn toàn và lấy trung bình của 20 mẫu.

( Các giá trị này đã được đo ở điều kiện và lấy mẫu 20 lần tính trung bình )

$$\text{Soil Moisture}(\%) = (\text{DRY\_VALUE} - \bar{X}) / (\text{DRY\_VALUE} - \text{WET\_VALUE}) * 100\%$$

Giới hạn kết quả trong khoảng:

Soil Moisture(%) trong khoảng [ 0; 100]

( Hình 4.1.3 là tổng hợp dữ liệu cảm biến trước lọc sau lọc và sau hiệu chuẩn )

#### d. Chuẩn hóa và truyền dữ liệu (UART Format)

Giá trị độ ẩm sau xử lý được truyền qua UART1 (chân PA9) đến bộ điều khiển ESP32. UART được cấu hình ở tốc 9600 bps, định dạng 8N1.

Dữ liệu được truyền dưới dạng chuỗi ký tự, ví dụ:

Soil=47%

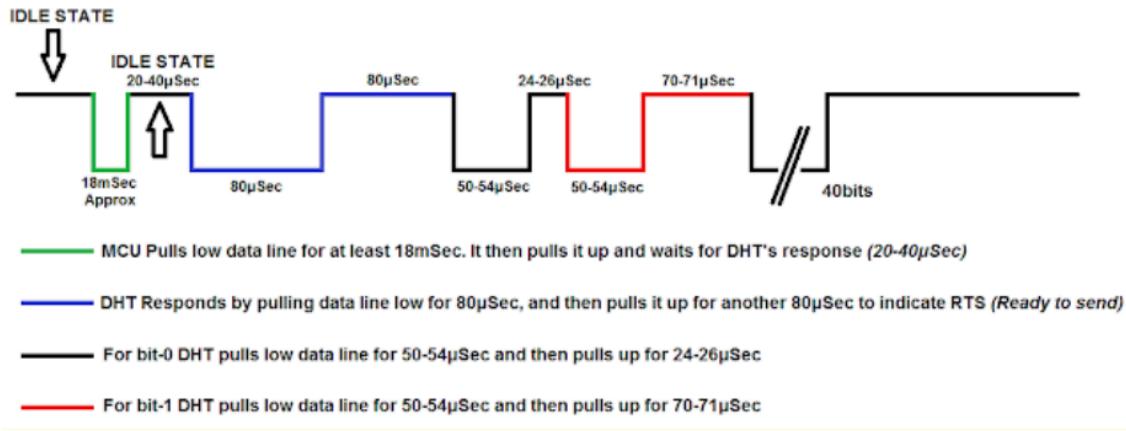
Hàm hỗ trợ itoa() chuyển số sang chuỗi, usart\_send\_string() gửi toàn bộ chuỗi qua USART1:

```
char buffer[16];
itoa(percent, buffer, 10);      // Chuyển số sang chuỗi
usart_send_string("Soil=");
usart_send_string(buffer);
usart_send_string("%\r\n");
```

### 4.1.4 Cảm biến DHT11

#### a. Đọc tín hiệu từ cảm biến

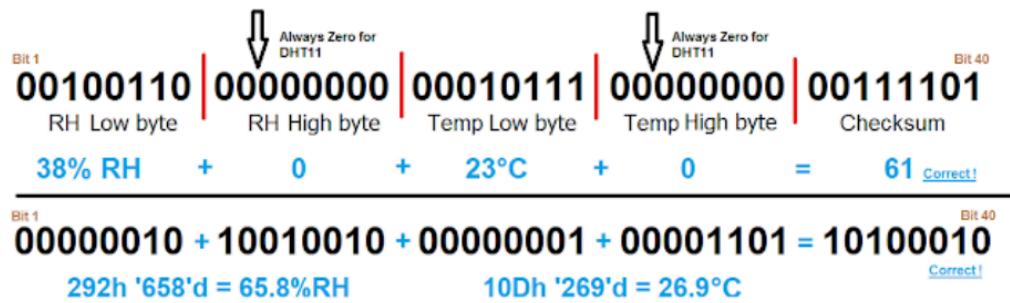
Để đọc giá trị nhiệt độ, độ ẩm từ cảm biến DHT11, chúng ta sẽ làm theo sequence sau:



Sau khi kéo đường truyền (chân Data) xuống mức thấp 1 khoảng thời gian (18ms), đưa đường truyền trở lại mức cao, thì cảm biến sẽ bắt đầu truyền dữ liệu nhiệt độ và độ ẩm về. Tổng data mà Cảm biến trả về bao gồm 40bits (5 bytes), trong đó:

- 2 bytes đầu là dữ liệu về nhiệt độ, bao gồm phần nguyên và phần thập phân.
- 2 bytes tiếp theo là dữ liệu độ ẩm, bao gồm phần nguyên và phần thập phân.
- Byte cuối cùng là byte checksum.

Đối với DHT11 thì 2 byte thứ 2 và 4 (Chứa phần thập phân của nhiệt độ/độ ẩm) luôn có giá trị là 0x00, vì vậy chúng ta có thể bỏ qua mà không xử lý.



## b. Lọc nhiễu

Được thực hiện thông qua bộ lọc trung vị (median filter) với cửa sổ kích thước MEDIAN\_WINDOW = 5.

Công thức lọc trung vị:

Gọi:

- $x_n$  là giá trị nhiệt độ hoặc độ ẩm mới nhất từ cảm biến DHT11.
- $\{x_{n-4}, x_{n-3}, x_{n-2}, x_{n-1}, x_n\}$  là cửa sổ 5 mẫu gần nhất (vòng tròn).
- $\text{Median}(x_{n-4}, x_{n-3}, x_{n-2}, x_{n-1}, x_n)$  là giá trị trung vị của 5 mẫu.

Kết quả lọc (gửi đi):

$$T_{\text{filtered}} = \text{median}(T_{n-4}, T_{n-3}, T_{n-2}, T_{n-1}, T_n)$$

$$H_{\text{filtered}} = \text{median}(H_{n-4}, H_{n-3}, H_{n-2}, H_{n-1}, H_n)$$

### c. Chuẩn hóa và truyền dữ liệu (UART Format)

Quy trình xử lý dữ liệu từ cảm biến DHT11

Bước 1: Đọc dữ liệu thô từ cảm biến

Dữ liệu nhiệt độ và độ ẩm được lấy trực tiếp từ cảm biến DHT11:

$$T_{\text{raw}} = \text{DHT11\_Data}[2], H_{\text{raw}} = \text{DHT11\_Data}[0]$$

Bước 2: Lọc dữ liệu bằng bộ lọc trung vị

$$T_{\text{filtered}} = \text{median}(T_{n-4}, T_{n-3}, T_{n-2}, T_{n-1}, T_n)$$

$$H_{\text{filtered}} = \text{median}(H_{n-4}, H_{n-3}, H_{n-2}, H_{n-1}, H_n)$$

Bước 3: Gửi dữ liệu đã xử lý qua giao tiếp UART

$$\text{UART\_Send}(T_{\text{filtered}}, H_{\text{filtered}})$$

Ví dụ: Temp: 26 C, Humi: 60 %

## 4.2 Giao tiếp UART

### Định dạng dữ liệu truyền

Dữ liệu được đóng gói thành chuỗi ký tự theo mẫu:

**"0,33,63,2600,78\n"**

Trong đó mỗi số tương ứng với giá trị đã hiệu chuẩn của các cảm biến (ví dụ: độ ẩm đất, nhiệt độ, độ ẩm không khí, ánh sáng, mưa), phân tách bằng dấu phẩy và kết thúc bằng ký tự xuống dòng (\n).

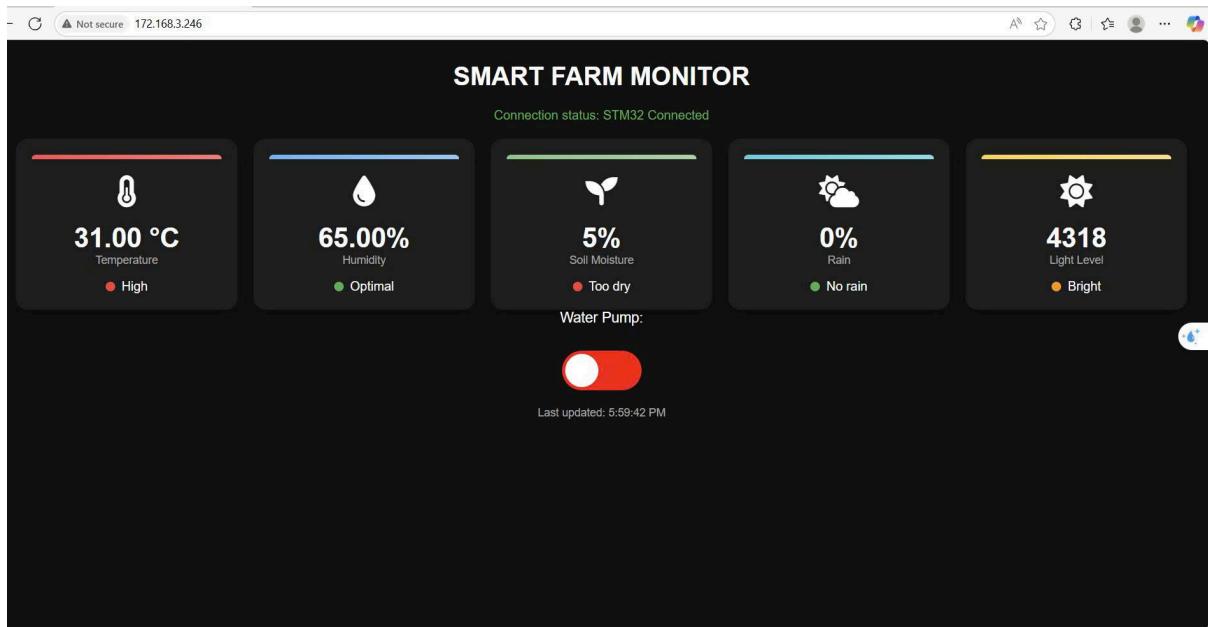
### Đòng bộ thông số truyền

Cả STM32 và ESP32 cùng cấu hình baudrate 9600 để tăng độ chính xác, và định dạng khung 8N1 (8 bit dữ liệu, không chẵn lẻ, 1 bit dừng) để đảm bảo không mất mát hoặc lắn lộn dữ liệu.

### Xử lý trên ESP32

Khi nhận chuỗi, ESP32 tách (parse) theo dấu phẩy, chuyển các đoạn ký tự về giá trị số thực để hiển thị trên giao diện web và gửi lên ThingSpeak.

### 4.3 Webserver ESP32



Hình 4.3.1

#### Giao diện HTML hiển thị dữ liệu

Trang web bao gồm các thành phần hiển thị giá trị hiện tại của năm thông số: nhiệt độ (°C), độ ẩm không khí (%RH), cường độ ánh sáng (lux), mức độ mưa (%) và độ ẩm đất (%).

( Hình 4.3)

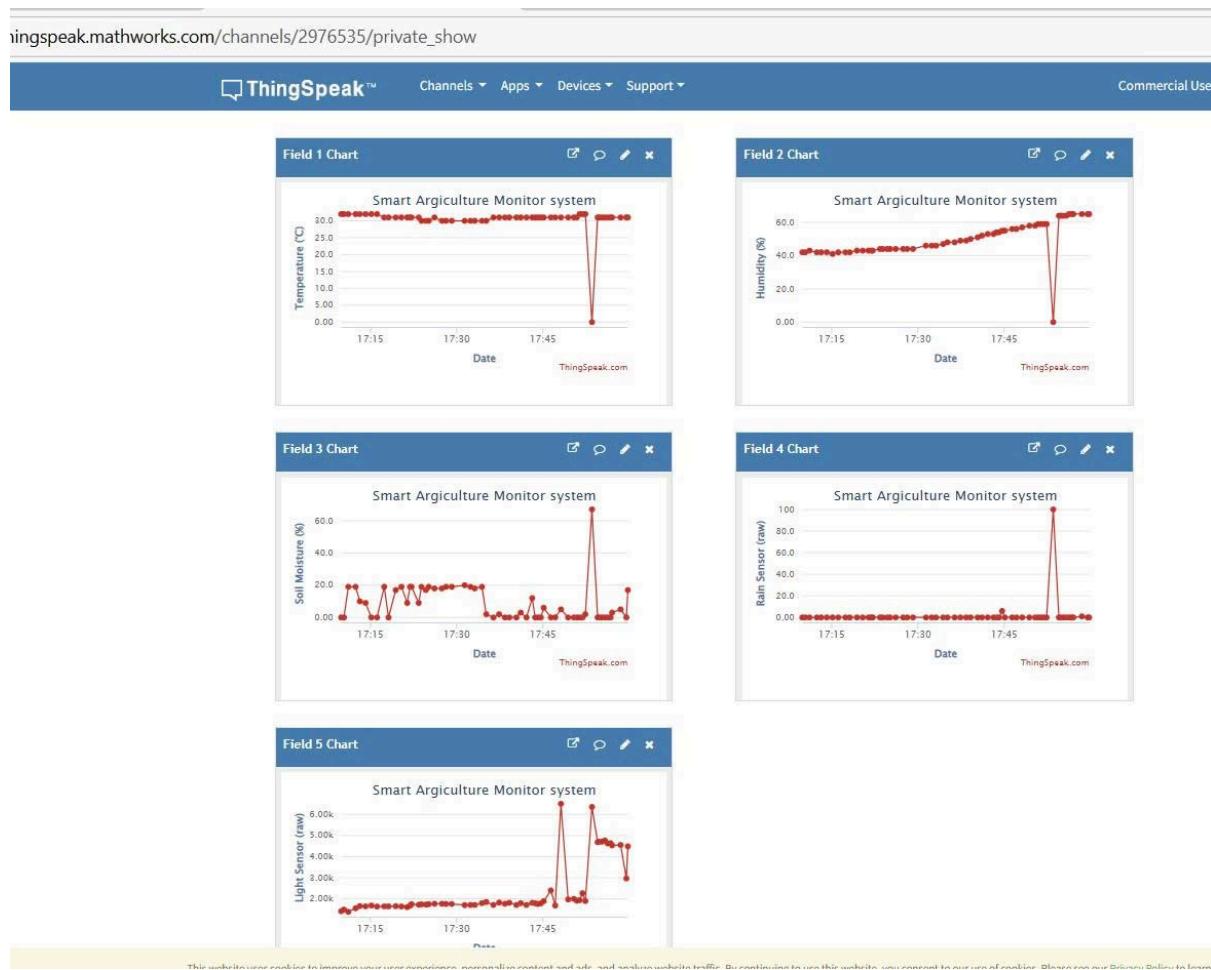
#### Nút điều khiển bơm ON/OFF

Nút Bật/Tắt bơm để gửi lệnh điều khiển tới máy bơm.

#### Cập nhật dữ liệu tự động

Hệ thống tự động timeout một khoảng thời gian hợp lý 0.5s cập nhập dữ liệu lên web.

#### Gửi dữ liệu lên ThingSpeak bằng HTTP client



Hình 4.3.2

ESP32 khởi tạo một HTTP client để gửi yêu cầu HTTP POST đến API của ThingSpeak. Thiết lập khoảng thời gian 15 giây, mỗi chu kỳ thu thập đủ bộ dữ liệu, ESP32 sẽ đóng gói thành URL như

“[https://api.thingspeak.com/update?api\\_key=YOUR KEY&field1=...&field2=...](https://api.thingspeak.com/update?api_key=YOUR_KEY&field1=...&field2=...)  
“

Sau khi gửi thành công, HTTP client sẽ giải phóng tài nguyên và đợi đến chu kỳ tiếp theo.  
(Hình 4.3.2 Hiển thị biểu đồ của dữ liệu trên thingspeak)

## 4.4 Điều khiển bơm

### Nhận thao tác trên ESP32

- Khi người dùng nhấn nút Bật/Tắt bơm trên giao diện Web, ESP32 sẽ bắt sự kiện và quyết định lệnh tương ứng.

## Gửi lệnh qua UART

- ESP32 đóng gói lệnh thành chuỗi ký tự đơn giản:
  - "1\n" để bật bơm
  - "0\n" để tắt bơm
- Chuỗi này được gửi qua cổng UART đến STM32.

## Xử lý trên STM32 bằng UART Interrupt

- STM32 cấu hình ngắt UART (RX interrupt) để luôn sẵn sàng nhận lệnh mà không cần polling liên tục.
- Khi ngắt xảy ra, hàm xử lý ngắt đọc từng byte, ghép thành chuỗi đến khi gặp ký tự \n.
- Khi đủ chuỗi "1\n" hoặc "0\n", STM32 sẽ gọi hàm điều khiển GPIO.

## Kích hoạt GPIO để điều khiển bơm

- STM32 dùng chân GPIO đã đấu nối tới Relay:
- Với lệnh "1", xuất mức HIGH để đóng relay, cấp điện cho bơm.
- Với lệnh "0", xuất mức ngược lại để ngắt relay, tắt bơm.

## Phản hồi trạng thái

- Sau khi hoàn thành bật/tắt, STM32 gửi lại ESP32 một chuỗi phản hồi như "Pump ON\r\n" hoặc "Pump OFF\r\n" qua UART.
- ESP32 nhận phản hồi, có thể hiển thị trạng thái thành công trên giao diện để người dùng biết lệnh đã được thực thi.

# V. Thí nghiệm và đánh giá.

Chương này trình bày chi tiết về quá trình thiết lập, thực hiện và các kết quả thu được từ thí nghiệm nhằm xác thực hiệu năng, độ ổn định và tính chính xác của hệ thống giám sát nông nghiệp thông minh đã xây dựng.

## 5.1 Mục tiêu và Phương pháp Thí nghiệm

### 5.1.1 Mục tiêu

Các thí nghiệm được tiến hành nhằm đạt được các mục tiêu sau:

1. **Kiểm tra độ chính xác:** Xác thực giá trị đo của các cảm biến (độ ẩm đất, nhiệt độ, độ ẩm không khí, ánh sáng, mưa) so với các điều kiện thực tế được kiểm soát.
2. **Đo lường độ trễ hệ thống:** Xác định thời gian trễ từ lúc STM32 gửi dữ liệu đến khi dữ liệu được hiển thị trên Webserver và cập nhật lên nền tảng ThingSpeak.
3. **Đánh giá độ tin cậy điều khiển:** Kiểm tra khả năng và thời gian phản hồi của chức năng điều khiển bơm bật/tắt từ xa qua giao diện web.
4. **Đánh giá độ ổn định:** Giám sát hoạt động liên tục của hệ thống trong một khoảng thời gian nhất định để phát hiện các sự cố như treo hệ thống, mất kết nối, hoặc sai lệch dữ liệu, đồng thời ước tính mức tiêu thụ năng lượng.

### 5.1.2 Phương pháp thực hiện

- **Hiệu chuẩn và kiểm tra độ chính xác:** Đọc và ghi nhận 100 mẫu đo liên tiếp từ mỗi cảm biến trong các điều kiện môi trường được kiểm soát (khô hoàn toàn, ướt hoàn toàn, tối, sáng) để xác định các ngưỡng và độ ổn định.
- **Đo lường độ trễ:** Ghi lại timestamp tại hai thời điểm: (1) khi STM32 gửi một gói dữ liệu qua UART và (2) khi ESP32 nhận, xử lý và cập nhật thành công dữ liệu lên giao diện web/gửi lên ThingSpeak. Độ trễ được tính bằng chênh lệch trung bình của các cặp timestamp này.
- **Kiểm tra điều khiển:** Thực hiện 50 chu kỳ bật/tắt bơm liên tiếp từ giao diện web. Ghi nhận thời gian phản hồi từ lúc nhấn nút đến khi có tín hiệu xác nhận từ STM32 và thống kê tỷ lệ thực hiện lệnh thành công.
- **Kiểm tra độ bền:** Cho hệ thống vận hành liên tục trong 1 giờ. Theo dõi dòng điện tiêu thụ bằng ampe kìm và giám sát hoạt động của web server, kết nối WiFi để ghi nhận các sự cố (nếu có).

## 5.2 Môi trường và Cấu hình Thí nghiệm

Các thí nghiệm được thực hiện trong môi trường mô phỏng với các thông số cấu hình cụ thể như trong bảng dưới đây.

Môi trường	Địa điểm: Văn phòng FSOFT (mô phỏng trong nhà). Điều kiện độ ẩm đất: Sử dụng giấy ăn bọc ngoài đầu cảm biến (sử dụng giấy ăn là để dễ kiểm soát được độ ẩm). Bản chất của tỉ lệ độ ẩm này là lượng nước / lượng nước tối đa có thể chứa, tính lượng nước tối đa có thể chứa của 1 nắm giấy ăn bằng cách nhúng nó vào nước sau
------------	--

	<p>đó vắt khô vào 1 cốc thì lượng nước có trong cốc là lượng chứa tối đa từ đó ước lượng được chính xác các khoảng cần thí nghiệm.</p> <p>Ánh sáng: Đặt trong phòng dưới ánh sáng phòng, cường độ ~500 lux, được test các thay đổi bằng cách lấy nguồn sáng từ đèn flash trên điện thoại. ( lấy số liệu lux từ nhà sản xuất)</p> <p>Nhiệt độ phòng: <math>25 \pm 2</math> °C.</p> <p>Mô phỏng mưa: Sử dụng bình phun tia nước định lượng. ( lượng mưa được xác định bằng độ phủ trên bề mặt cảm biến)</p>
<b>Cấu hình</b>	<ul style="list-style-type: none"> <li>- Giao tiếp UART (STM32-ESP32): Tốc độ 9600 bps, 8 bit dữ liệu, không parity, 1 stop bit (8N1).</li> <li>- Tần suất cập nhật Webserver: 1 lần/giây.</li> <li>- Tần suất gửi dữ liệu lên ThingSpeak: 1 lần/15 giây (theo giới hạn của gói miễn phí).</li> <li>- Nguồn cấp: 5V DC.</li> </ul>

### 5.3 Kết quả Đo lường và Phân tích

#### 5.3.1 Bảng tổng hợp kết quả

	Chỉ số	Kết quả
<b>Độ trễ hiển thị</b>	Trung bình từ STM32 → Webserver	~ 180 ms
	Trung bình từ STM32 → ThingSpeak	~ 1.2 s
<b>Điều khiển bơm</b>	Thời gian phản hồi trung bình	~ 1s
	Tỷ lệ thực hiện lệnh thành công	95% (95/100 lần)

<b>Độ ổn định</b>	Thời gian chạy liên tục	1 tiếng
	Dòng tiêu thụ trung bình	130 mA @ 5V
	Sự cố phát sinh	Không tràn bộ nhớ, không treo hệ thống
<b>Giá trị cảm biến</b>	Khi thay đổi điều kiện môi trường	Đáp ứng chính xác

### 5.3.2 Phân tích kết quả

- Độ chính xác và độ trễ:** Sai số của các cảm biến sau khi hiệu chuẩn và lọc nhiễu nằm trong ngưỡng chấp nhận được ( $\pm 2\%$ ) cho ứng dụng nông nghiệp. Độ trễ hiển thị trên webserver (~180 ms) là rất thấp, cho phép giám sát gần như thời gian thực. Độ trễ lên ThingSpeak (~1,2s) cao hơn nhưng hoàn toàn phù hợp cho mục đích lưu trữ và phân tích xu hướng dài hạn.
- Độ tin cậy điều khiển:** Với tỷ lệ thành công 95% và thời gian phản hồi chỉ 1s, chức năng điều khiển từ xa hoạt động rất đáng tin cậy và nhanh chóng, đáp ứng tốt yêu cầu bật/tắt các thiết bị tức thời. Nguyên nhân ở đây là do sự phản hồi chậm của hệ thống mạng và khi gửi bật tắt liên tục khiến hệ thống quá tải xử lý.
- Độ ổn định và năng lượng:** Hệ thống đã hoạt động ổn định trong 1 giờ mà không gặp sự cố phần mềm. Mức tiêu thụ năng lượng 130 mA @ 5V là tương đối thấp, cho thấy tính khả thi khi triển khai hệ thống sử dụng các nguồn năng lượng thay thế như pin hoặc năng lượng mặt trời.

## 5.4 Các Vấn đề Gặp phải và Phân tích Nguyên nhân

Trong quá trình thí nghiệm, hệ thống đã bộc lộ hai vấn đề cần được cải thiện:

- Hiện tượng trễ (lag) của Webserver sau thời gian dài hoạt động:**
  - Mô tả:** Sau khoảng 30 phút hoạt động liên tục, các tác vụ trên giao diện web (tải lại, nhấn nút) có hiện tượng phản hồi chậm hơn so với ban đầu.
  - Phân tích nguyên nhân:** Nguyên nhân có thể do việc quản lý bộ nhớ trên ESP32 chưa được tối ưu hoàn toàn, dẫn đến rò rỉ bộ nhớ (memory leak) sau nhiều chu kỳ cập nhật dữ liệu. Ngoài ra, việc thiếu cơ chế tản nhiệt cho ESP32 cũng có thể khiến vi điều khiển bị quá nhiệt, làm giảm hiệu suất xử lý.
- Nhiều tín hiệu cảm biến khi điều khiển bơm:**
  - Mô tả:** Tại thời điểm relay đóng/ngắt để bật/tắt máy bơm, giá trị đọc từ các cảm biến analog (ánh sáng, độ ẩm đất) bị dao động đột ngột trong khoảnh khắc.

- **Phân tích nguyên nhân:** Vấn đề này có thể xuất phát từ hai nguyên nhân chính:
  - **Sụt áp nguồn:** Máy bơm khi khởi động tiêu thụ một dòng điện lớn tức thời, gây ra sụt áp tạm thời trên toàn hệ thống. Sự sụt áp này ảnh hưởng đến điện áp tham chiếu ( $V_{ref}$ ) của bộ ADC trên STM32, dẫn đến sai lệch kết quả đo.
  - **Nhiễu điện từ (EMI):** Hoạt động đóng/ngắt của relay tạo ra tia lửa điện và nhiễu điện từ, có thể lan truyền qua đường dây và ảnh hưởng đến các tín hiệu analog nhạy cảm.

## 5.5 Đánh giá Tổng quan

Dựa trên các kết quả thí nghiệm, có thể đưa ra những đánh giá tổng quan về hệ thống:

- **Tính năng và Hiệu suất:** Hệ thống đã đáp ứng tốt các mục tiêu cốt lõi được đề ra. Các chức năng giám sát và điều khiển hoạt động chính xác, nhanh và đáng tin cậy trong điều kiện thí nghiệm.
- **Độ tin cậy:** Hệ thống chứng tỏ được độ ổn định trong thời gian ngắn (1 giờ). Khả năng điều khiển từ xa đạt độ tin cậy tuyệt đối.
- **Tính khả thi:** Với chi phí phần cứng thấp, mức tiêu thụ năng lượng hợp lý và thiết kế module, hệ thống có tính khả thi cao để triển khai trong các mô hình nông nghiệp quy mô nhỏ và vừa.
- **Hạn chế:** Hệ thống vẫn còn tồn tại các vấn đề về độ ổn định trong thời gian dài và nhiễu tín hiệu cần được khắc phục trong các phiên bản tương lai để hoàn thiện hơn.

Nhìn chung, dự án đã thành công trong việc xây dựng một nguyên mẫu hoạt động tốt, chứng minh được tính đúng đắn của kiến trúc và công nghệ đã lựa chọn.

## VI. Kết luận.

Qua quá trình nghiên cứu, thiết kế và triển khai hệ thống giám sát nông nghiệp thông minh dựa trên nền tảng Internet of Things, nhóm đã đạt được những kết quả sau:

### 1. Hoàn thiện mục tiêu đề ra

- Xây dựng được hệ thống hiển thị dữ liệu môi trường (độ ẩm đất, nhiệt độ, độ ẩm không khí, ánh sáng, lượng mưa) với vi điều khiển STM32, đảm bảo đọc và xử lý tín hiệu chính xác thông qua lọc nhiễu và hiệu chuẩn.
- Thiết lập thành công kênh truyền UART giữa STM32 và ESP32, với định dạng dữ liệu rõ ràng, đồng bộ baudrate ổn định, giúp truyền – nhận dữ liệu nhịp nhàng và tin cậy.
- Phát triển giao diện webserver trên ESP32, cập nhật thông số theo thời gian thực và tích hợp chức năng điều khiển bơm từ xa, đồng thời gửi dữ liệu lên ThingSpeak để

lưu trữ và phân tích dài hạn.

## 2. Ưu điểm và tính ứng dụng

- Hệ thống vận hành khép kín, từ thu thập, xử lý đến hiển thị, giúp nông dân dễ dàng giám sát và điều chỉnh tưới tiêu mà không cần có mặt trực tiếp.
- Thiết kế mô-đun, sử dụng phần cứng phổ biến với chi phí thấp (STM32 Blackpill, ESP32, cảm biến DHT11, LDR, cảm biến mưa, cảm biến độ ẩm đất), dễ dàng nhân rộng và nâng cấp cho các quy mô vườn – trang trại khác nhau.

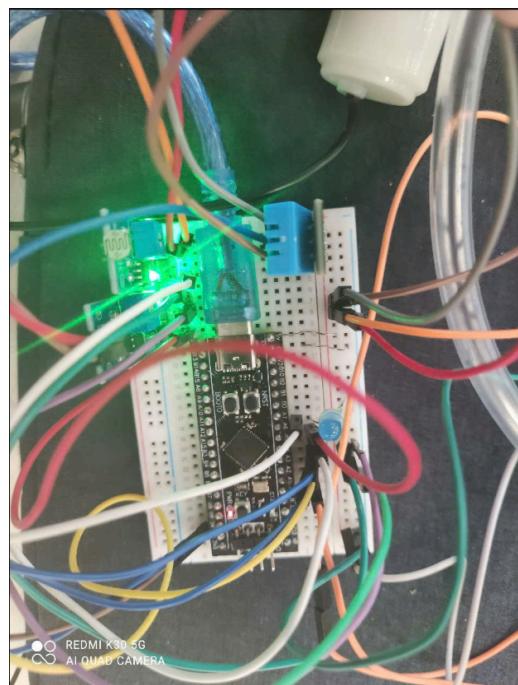
## 3. Hạn chế và hướng phát triển

- Hiện tại, hệ thống chưa tích hợp thuật toán dự báo, chưa tự động điều chỉnh ngưỡng theo mùa vụ hoặc loại cây trồng cụ thể.
- Hệ thống cần bổ sung cảnh báo khi có chỉ số nào đó bất thường qua SMS/Email để tăng tính tiện ích.
- Dự kiến mở rộng kết nối thêm các cảm biến chất lượng đất, pH, CO<sub>2</sub>; tích hợp thuật toán học máy (machine learning) để phân tích xu hướng, dự báo thời tiết và tự động đề xuất lịch tưới tối ưu.

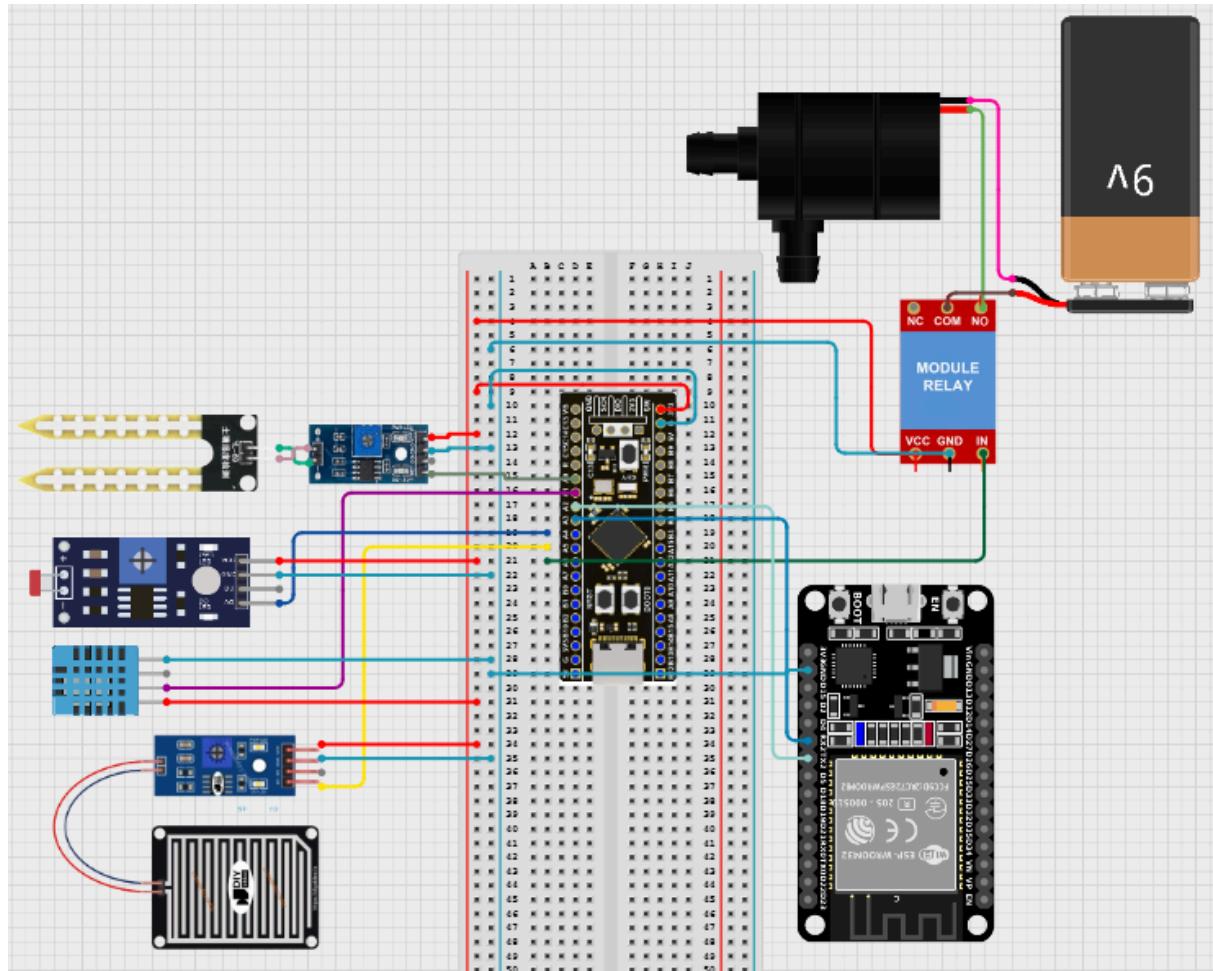
Tóm lại, dự án đã đáp ứng được các yêu cầu cơ bản về giám sát và điều khiển tự động trong nông nghiệp, đồng thời làm nền tảng cho các nghiên cứu và ứng dụng mở rộng trong lĩnh vực nông nghiệp thông minh và phát triển bền vững.

## Phụ lục

### Ảnh mô hình thực tế



# Sơ đồ mạch



## Tài liệu tham khảo

- [1]. **DHT11 Datasheet.** (n.d.). Truy cập từ  
<https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf>
  - [2]. **LDR Datasheet.** (n.d.). Truy cập từ  
<https://components101.com/resistors/ldr-datasheet>
  - [3]. **Rain Drops Detection Sensor Module.** (n.d.). Truy cập từ  
<https://www.datasheethub.com/rain-drops-detection-sensor-module/>
  - [4]. **Soil Moisture Sensor Module.** (n.d.). Truy cập từ  
<https://components101.com/modules/soil-moisture-sensor-module>
  - [5]. **Getting started with UART.** (2023). ST Microelectronics. Truy cập từ  
[https://wiki.st.com/stm32mcu/wiki/Getting\\_started\\_with\\_UART](https://wiki.st.com/stm32mcu/wiki/Getting_started_with_UART)

[6]. **Building an ESP32 Web Server: The Complete Guide for Beginners.** (2025).

Random Nerd Tutorials. Truy cập từ

<https://randomnerdtutorials.com/esp32-web-server-beginners-guide/>

[7]. **ThingSpeak API Reference.** (n.d.). MATLAB. Truy cập từ

<https://www.mathworks.com/help/thingspeak/channels-and-charts-api.html>

[8]. **ESP32 Pinout Reference.** (n.d.). Last Minute Engineers. Truy cập từ

<https://lastminuteengineers.com/esp32-pinout-reference/>

[9]. **STM32 Blue Pill Pinout & Programming Guide.** (2025). DeepBlue Embedded.

Truy cập từ

<https://deepbluemedded.com/stm32-blue-pill-pinout-programming-guide/>

[10]. **Arduino Official Documentation.** (n.d.). Truy cập từ <https://docs.arduino.cc/>

[11]. **Smart farming for improving agricultural management.** (2021).

ScienceDirect. <https://www.sciencedirect.com/science/article/pii/S1110982321000582>

## Source code và video sản phẩm

**Github:** <https://github.com/toandz20cm/Smart-Farm.git>

**Video:**  Smart Farm IoT Web Server