|  |
| --- |
| **HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  **Khoa: Công nghệ thông tin 1**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Logo_PTIT_University**  **BÁO CÁO GIỮA KỲ**  **Đề tài: HỆ THỐNG TƯỚI CÂY THÔNG MINH**  **ỨNG DỤNG AI ĐỂ LÊN LỊCH TƯỚI HÀNG TUẦN**  **GVHD : TS. Kim Ngọc Bách**  **SV thực hiện : Trịnh Quang Lâm -**  **Vũ Nhân Kiên -**  **Vũ Thế Văn -**  **Cao Thị Thu Hương -**  **Nhóm môn học :**  **Môn học : IoT và Ứng dụng**  **Hà Nội - 10/2025** |

MỤC LỤC

# Giới thiệu đề tài

## Mô tả dự án

Dự án này nhằm xây dựng một hệ thống tưới cây tự động sử dụng vi điều khiển ESP32 kết hợp AI phân tích dữ liệu thời tiết. Dự án được thiết kế nhằm mang lại giải pháp nông nghiệp thông minh, hiện đại và hiệu quả. Hệ thống sử dụng ESP32 làm bộ xử lý trung tâm, kết nối với các cảm biến môi trường như cảm biến độ ẩm đất, cảm biến nhiệt độ và độ ẩm không khí để thu thập dữ liệu theo thời gian thực. Các dữ liệu này sẽ được truyền về hệ thống Backend để lưu trữ, phân tích và đưa ra quyết định tưới tiêu phù hợp

Điểm đặc biệt của dự án là việc tích hợp AI phân tích dữ liệu thời tiết. AI sẽ dự đoán điều kiện điều kiện khí hậu trong tương lai từ đó xây dựng lịch tưới tự động tối ưu cho từng tuần. Điều này giúp hạn chế tình trạng tưới thừa khi sắp có mưa hoặc thiếu nước khi thời tiết khô hạn kéo dài

Ngoài ra, hệ thống còn cung cấp giao diện web trực quan phát triển bằng ReactJS, cho phép người dùng theo dõi trạng thái thiết bị, lịch tưới, dữ liệu cảm biến và can thiệp thủ công khi cần thiết. Phần Backend được xây dựng trên NodeJS với ExpressJS, kết nối với cơ sở dữ liệu MongoDB Atlas để lưu trữ thông tin. Nhờ đó, toàn bộ hệ thống hoạt động dựa trên mô hình IoT kết hợp AI, vừa tự động vừa cho phép giám sát và điều khiển từ xa thông qua Internet.

## Mục tiêu của hệ thống

Mục tiêu chính của dự án là tối ưu hóa quá trình tưới tiêu cho cây trồng bằng cách áp dụng công nghệ và trí tuệ nhân tạo. Thay vì người trồng phải theo dõi và tưới thủ công hằng ngày, hệ thống sẽ tự động phân tích các yếu tố môi trường và dự báo thời tiết để điều chỉnh lượng nước tưới phù hợp cho cây.

1. **Phạm vi của hệ thống**

Hệ thống được triển khai trong phạm vi diện tích nhỏ, bao gồm các cảm biến đo nhiệt độ, độ ẩm, áp suất và độ ẩm đất, kết hợp với bơm nước mini điều khiển tự động thông qua vi điều khiển ESP32. Người dùng có thể giám sát và điều khiển hệ thống từ xa thông qua giao diện web. Phạm vi hiện tại chưa bao gồm các chức năng mở rộng như bón phân tự động hay giám sát sâu bệnh, hoặc khả dụng/tối ưu cho diện tích khu vực lớn.

1. **Kết quả mong đợi**

Hệ thống kỳ vọng đạt được các kết quả sau:  
- Tối ưu lịch tưới cây theo điều kiện thời tiết và độ ẩm thực tế.  
- Giảm tối thiểu 25–40% số lần tưới trùng mưa.  
- Tiết kiệm từ 15–30% lượng nước sử dụng.  
- Cung cấp giao diện trực quan giúp người dùng dễ dàng theo dõi và điều khiển hệ thống.

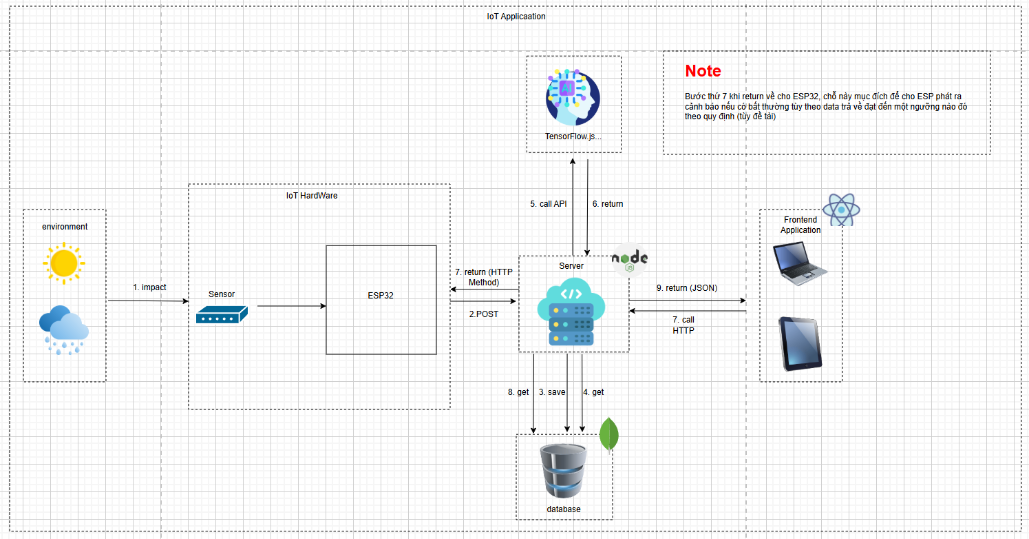
# Mô tả tổng quan hệ thống

1. **Mô tả tổng quan**

Hệ thống tưới cây thông minh bao gồm ba phần chính:

* Phần cảm biến và điều khiển: ESP32 kết nối với các cảm biến môi trường (DHT22, BME280, cảm biến độ ẩm đất) để thu thập dữ liệu.
* Phần xử lý dữ liệu và trí tuệ nhân tạo: Backend (NodeJS + ExpressJS) xử lý dữ liệu cảm biến, AI dự đoán mưa bằng mô hình XGBoost.
* Phần giao diện người dùng: Ứng dụng web (ReactJS) hiển thị dữ liệu, lịch tưới và cho phép điều khiển bơm từ xa.

***\* Sơ đồ tổng quan hệ thống:***



1. **Môi trường phát triển và hoạt động**
2. ***Môi trường phát triển:***  
   - Arduino IDE: Lập trình và nạp chương trình cho ESP32.  
   - Visual Studio Code: Phát triển frontend (ReactJS) và backend (NodeJS).  
   - MongoDB Atlas: Cơ sở dữ liệu NoSQL lưu trữ dữ liệu cảm biến.  
   - Git/Github: Quản lý mã nguồn và làm việc nhóm.  
   ***b) Môi trường hoạt động:***  
   - Kết nối Wi-Fi nội bộ.  
   - Trình duyệt web hiển thị giao diện người dùng.  
   - Máy chủ NodeJS hoặc Cloud lưu trữ và xử lý dữ liệu.
3. **Ràng buộc và giả định**

- Cảm biến hoạt động ổn định với chu kỳ đo 5 phút.  
- Hệ thống yêu cầu kết nối Internet liên tục.  
- AI cần ít nhất 60 ngày dữ liệu để học và dự đoán chính xác.  
- ESP32 có giới hạn tài nguyên nên chỉ đảm nhận nhiệm vụ thu thập và gửi dữ liệu, không chạy mô hình AI trực tiếp.  
- Nếu mất kết nối, hệ thống tự chuyển sang chế độ tưới thủ công theo lịch mặc định, hoặc cho phép người dùng điều khiển thủ công/điều khiển từ xa.

1. **Yêu cầu chức năng**

- Thu thập và gửi dữ liệu cảm biến (nhiệt độ, độ ẩm, áp suất, độ ẩm đất).  
- Lưu trữ dữ liệu và phân tích trên server.  
- AI dự đoán khả năng mưa và lượng mưa trong 60 phút tới.  
- Điều khiển tự động bơm tưới theo ngưỡng độ ẩm và kết quả dự báo.  
- Cho phép người dùng theo dõi và điều khiển thủ công qua giao diện web.  
- Ghi lại lịch sử hoạt động và nhật ký tưới cây.

1. **Yêu cầu phi chức năng**

- Thời gian phản hồi < 300 ms cho mỗi yêu cầu dự đoán.  
- Hệ thống ổn định, tự phục hồi khi kết nối lại.  
- Có khả năng mở rộng, thêm cảm biến hoặc khu vực tưới.  
- Giao diện thân thiện, dễ sử dụng.  
- Đảm bảo bảo mật cơ bản (xác thực truy cập web).

1. **Công nghệ và phần cứng sử dụng**

## Phần cứng

### Vi điều khiển ESP32

**\* Giới thiệu về ESP32-WROOM-32D DevKit:**

ESP32-WROOM-32D là một module Wi-Fi + Bluetooth tích hợp, phát triển bởi Espressif Systems, được sử dụng rộng rãi trong IoT (Internet of Things), tự động hóa, các thiết bị nhúng thông minh nhờ khả năng xử lý mạnh mẽ và kết nối không dây. Trên thực tế, người ta thường sử dụng bo mạch phát triển ESP32 DevKit V1 tích hợp sẵn module ESP32-WROOM-32D để dễ dàng lập trình và kết nối.

Các thông số chính:

* CPU: Xtensa® dual-core 32-bit LX6 (tốc độ tối đa 240 MHz).
* Bộ nhớ: 448 KB ROM, 520 KB SRAM, 4 MB Flash (trên module).
* Kết nối: Wi-Fi 802.11 b/g/n, Bluetooth v4.2 (Classic + BLE).
* GPIO: 34 chân I/O, hỗ trợ PWM, ADC (12-bit), DAC (8-bit), SPI, I2C, UART.
* Điện áp hoạt động: 3.0V – 3.6V (DevKit hỗ trợ cấp từ 5V qua cổng USB).

**\* Các linh kiện chính trên bo mạch ESP32 DevKit 32D:**

Module ESP32-WROOM-32D:

* Thành phần trung tâm (SoC ESP32 + Flash 4 MB).
* Chứa bộ xử lý, bộ nhớ, Wi-Fi/Bluetooth, và mạch RF.

USB-to-UART Bridge (CP2102 hoặc CH340):

* Cho phép nạp chương trình và giao tiếp với máy tính qua cổng USB.
* Chuyển đổi tín hiệu USB ↔ UART.

Ổn áp AMS1117-3.3V:

* Giúp chuyển đổi điện áp từ 5V (USB) hoặc Vin xuống 3.3V để nuôi ESP32.

Cổng Micro-USB:

* Dùng để cấp nguồn (5V) và nạp code.

Nút nhấn:

* EN (Reset): Reset lại vi điều khiển.
* BOOT (IO0): Giữ để đưa ESP32 vào chế độ nạp chương trình.

Thạch anh (Crystal 40 MHz):

* Tạo xung clock ổn định cho chip ESP32.

LED chỉ thị (thường nối với GPIO2):

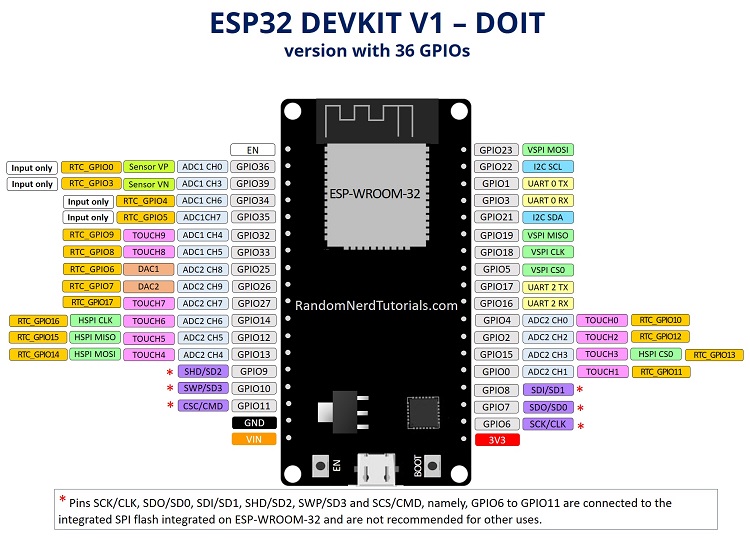
* Báo hiệu nguồn hoặc được dùng test xuất tín hiệu.

Hàng chân header (Male header pins):

* Xuất các chân GPIO, nguồn (3.3V, 5V), GND để dễ dàng kết nối với breadboard hoặc module khác.

Các tụ điện, điện trở dán (SMD):

* Dùng để lọc nhiễu, ổn định nguồn, kéo lên/kéo xuống cho các chân tín hiệu.



***Hình 1: Sơ đồ chân ESP32 DEVKIT ver 36 chân***

| **Chân GPIO** | **Chức năng chính** | **Ghi chú đặc biệt** |
| --- | --- | --- |
| VIN (5V) | Cấp nguồn 5V từ USB hoặc ngoài |  |
| 3V3 | Nguồn 3.3V từ ổn áp |  |
| GND | Mass (Ground) |  |
| EN | Reset (Enable) | Kéo xuống GND để reset chip |
| GPIO0 | ADC2\_CH1, Touch0 | Dùng BOOT mode, cẩn thận khi sử dụng |
| GPIO1 (TX0) | UART0 TX | Dùng cho USB nạp code |
| GPIO2 | ADC2\_CH2, Touch2 | Ảnh hưởng boot, nên tránh kéo xuống GND |
| GPIO3 (RX0) | UART0 RX | Dùng cho USB nạp code |
| GPIO4 | ADC2\_CH0, Touch4 | GPIO đa năng |
| GPIO5 | VSPI CS | GPIO đa năng |
| GPIO6 – 11 | Kết nối Flash | ⚠️ Không dùng |
| GPIO12 | ADC2\_CH5, Touch5, HSPI MISO | Ảnh hưởng boot, nên tránh khi khởi động |
| GPIO13 | ADC2\_CH4, Touch4, HSPI MOSI |  |
| GPIO14 | ADC2\_CH6, Touch6, HSPI CLK |  |
| GPIO15 | ADC2\_CH3, Touch3, HSPI CS | Ảnh hưởng boot |
| GPIO16 | UART2 RX | GPIO đa năng |
| GPIO17 | UART2 TX | GPIO đa năng |
| GPIO18 | VSPI CLK | GPIO đa năng |
| GPIO19 | VSPI MISO | GPIO đa năng |
| GPIO21 | I2C SDA | GPIO đa năng |
| GPIO22 | I2C SCL | GPIO đa năng |
| GPIO23 | VSPI MOSI | GPIO đa năng |
| GPIO25 | ADC2\_CH8, DAC1 | GPIO đa năng |
| GPIO26 | ADC2\_CH9, DAC2 | GPIO đa năng |
| GPIO27 | ADC2\_CH7, Touch7 | GPIO đa năng |
| GPIO32 | ADC1\_CH4, Touch9 | GPIO đa năng |
| GPIO33 | ADC1\_CH5, Touch8 | GPIO đa năng |
| GPIO34 | ADC1\_CH6 | Chỉ Input |
| GPIO35 | ADC1\_CH7 | Chỉ Input |
| GPIO36 (VP) | ADC1\_CH0 | Chỉ Input |
| GPIO39 (VN) | ADC1\_CH3 | Chỉ Input |

**\* Sơ đồ khối hoạt động cơ bản của DevKit 32D:**

Máy tính (USB)

↓

USB-to-UART (CP2102/CH340)

↓

ESP32-WROOM-32D ←→ Wi-Fi / Bluetooth

↓

GPIO / ADC / DAC / UART / I2C / SPI

### Cảm biến nhiệt độ độ ẩm DHT22

**\* Giới thiệu chung:**

DHT22 (còn gọi là AM2302) là cảm biến nhiệt độ và độ ẩm kỹ thuật số, được sử dụng nhiều trong các ứng dụng IoT, nông nghiệp thông minh, nhà thông minh, trạm thời tiết mini,… So với DHT11, DHT22 có độ chính xác cao hơn, phạm vi đo rộng hơn.

**\* Thông số kỹ thuật chính:**

* Điện áp hoạt động: 3.3V – 5.5V
* Dòng tiêu thụ: 1.5 mA khi đo, < 100 µA khi chờ
* Giao tiếp: 1 dây (Single-bus)
* Khoảng đo nhiệt độ: -40°C → +80°C
* Độ chính xác nhiệt độ: ±0.5°C
* Khoảng đo độ ẩm: 0% → 100% RH
* Độ chính xác độ ẩm: ±2–5% RH
* Chu kỳ đo tối thiểu: 2 giây (0.5 Hz)

**\* Cấu tạo phần cứng:**

Cảm biến DHT22 gồm 2 phần chính:

* Cảm biến độ ẩm điện dung (Capacitive humidity sensor): Hoạt động dựa trên sự thay đổi điện dung của vật liệu nhạy ẩm khi độ ẩm không khí thay đổi.
* Cảm biến nhiệt độ (Thermistor hoặc nhiệt điện trở): Thay đổi điện trở theo nhiệt độ môi trường.
* Chip xử lý tín hiệu tích hợp: Chuyển đổi tín hiệu analog từ 2 cảm biến thành dữ liệu số (digital) và xuất dữ liệu ra ngoài qua giao tiếp 1 dây.

**\* Sơ đồ chân DHT22:**

Thông thường DHT22 có 4 chân (hoặc module 3 chân đã tích hợp điện trở kéo lên).

| **Chân** | **Ký hiệu** | **Chức năng** |
| --- | --- | --- |
| 1 | VCC | Cấp nguồn (3.3V – 5V) |
| 2 | DATA | Truyền dữ liệu 1 dây (cần điện trở kéo lên 4.7k – 10kΩ) |
| 3 | NC | Không kết nối |
| 4 | GND | Nối đất |

**\* Nguyên lý hoạt động và giao tiếp:**

ESP32 (hoặc vi điều khiển khác) kéo chân DATA xuống mức thấp trong vài ms để khởi động cảm biến. Sau đó, DHT22 phản hồi bằng một chuỗi xung dữ liệu.

Dữ liệu gửi đi có tổng cộng 40 bit:

* 16 bit cho độ ẩm
* 16 bit cho nhiệt độ
* 8 bit checksum (kiểm tra lỗi)

Vi điều khiển đọc tín hiệu này, giải mã thành giá trị độ ẩm (%) và nhiệt độ (°C).

**\* Ưu điểm và nhược điểm:**

Ưu điểm:

* Độ chính xác cao hơn DHT11.
* Đo được dải nhiệt độ và độ ẩm rộng.
* Giao tiếp đơn giản (1 dây).

Nhược điểm:

* Tốc độ đọc chậm (2 giây mới cập nhật dữ liệu).
* Giá thành cao hơn DHT11.
* Dễ hỏng khi làm việc trong môi trường ẩm quá lâu mà không có bảo vệ.

### Cảm biến độ ẩm đất

**\* Giới thiệu chung:**

Cảm biến độ ẩm đất là loại cảm biến dùng để đo hàm lượng nước trong đất, được ứng dụng trong nông nghiệp thông minh, hệ thống tưới tự động, giám sát cây trồng. Nguyên lý hoạt động dựa trên việc thay đổi điện trở hoặc điện dung của đất khi hàm lượng nước thay đổi.

Có 2 loại phổ biến:

* Cảm biến độ ẩm đất loại điện trở (Soil Moisture Sensor – Resistive):
* Đo sự thay đổi điện trở của đất.
* Rẻ, đơn giản nhưng dễ bị ăn mòn điện cực.
* Cảm biến độ ẩm đất loại điện dung (Capacitive Soil Moisture Sensor):
* Đo sự thay đổi hằng số điện môi của đất.
* Bền hơn, ít bị ăn mòn, kết quả ổn định hơn.

→ Ở đây sử dụng loại dựa trên điện trở.

**\* Cấu tạo và nguyên lý hoạt động:**

Gồm 2 que kim loại cắm xuống đất.

* Khi đất khô → điện trở giữa 2 que cao → dòng điện nhỏ.
* Khi đất ẩm → điện trở giảm → dòng điện lớn.
* Module đi kèm thường có mạch so sánh LM393, xuất tín hiệu Analog và Digital.

**\* Sơ đồ chân:**

| **Chân** | **Ký hiệu** | **Chức năng** |
| --- | --- | --- |
| VCC | 3.3V – 5V | Nguồn cấp |
| GND | GND | Mass |
| AO | Analog Output | Xuất tín hiệu analog (0 – 1023) |
| DO | Digital Output | Xuất mức logic (0/1) khi vượt ngưỡng đặt trên chiết áp |

**\* Nguyên lý đo lường:**

* Vi điều khiển (ví dụ ESP32) đọc tín hiệu từ chân AO (analog).
* Khi đất khô → điện áp AO cao (ít nước → điện dung/điện trở thay đổi).
* Khi đất ẩm → điện áp AO thấp hơn (nhiều nước → điện dung tăng hoặc điện trở giảm).
* Tín hiệu DO chỉ có 2 trạng thái (khô/ướt), phù hợp cho ứng dụng đơn giản (bật/tắt bơm).

**\* Ứng dụng thực tế:**

* Hệ thống tưới cây tự động (khi đất khô → bật bơm, khi ẩm → tắt bơm).
* Đo lường và giám sát độ ẩm đất trong nông nghiệp thông minh.
* Kết hợp với ESP32 để đưa dữ liệu lên IoT Cloud (Blynk, MQTT, Firebase…).

**\* Ưu điểm và nhược điểm:**

* Giá rẻ, dễ tìm, dễ sử dụng.
* Nhanh bị ăn mòn điện cực → tuổi thọ ngắn.

### Cảm biến môi trường BME280

**\* Giới thiệu chung:**

BME280 là cảm biến môi trường tích hợp của hãng Bosch Sensortec, được thiết kế để đo nhiệt độ, độ ẩm và áp suất khí quyển với độ chính xác cao. Đây là phiên bản nâng cấp của BMP280 (chỉ đo nhiệt độ và áp suất).

Ứng dụng trong:

* Trạm thời tiết mini.
* Hệ thống IoT giám sát môi trường.
* Thiết bị đeo thông minh (smartwatch, fitness tracker).
* Nông nghiệp thông minh, nhà thông minh.

**\* Thông số kỹ thuật chính:**

Điện áp hoạt động: 1.8V – 3.6V (module breakout thường hỗ trợ 3.3V và 5V).

Dòng tiêu thụ: cực thấp (~3.6 µA khi đo, ~0.1 µA ở chế độ sleep).

Đo nhiệt độ:

* Dải: -40°C → +85°C
* Sai số: ±1.0°C
* Đo độ ẩm:
* Dải: 0% → 100% RH
* Sai số: ±3% RH

Đo áp suất khí quyển:

* Dải: 300 → 1100 hPa
* Sai số: ±1 hPa
* Giao tiếp: I²C (tối đa 3.4 MHz) hoặc SPI (tối đa 10 MHz).

**\* Cấu tạo và nguyên lý hoạt động:**

BME280 tích hợp 3 cảm biến:

* Cảm biến nhiệt độ (Temperature sensor): Dùng để bù sai số cho độ ẩm và áp suất.
* Cảm biến độ ẩm (Humidity sensor – điện dung): Đo sự thay đổi hằng số điện môi của vật liệu nhạy ẩm.
* Cảm biến áp suất (Pressure sensor – áp điện trở màng mỏng): Dựa trên sự thay đổi điện trở khi màng cảm biến biến dạng do áp suất khí quyển.
* Chip BME280 có bộ xử lý tín hiệu tích hợp, dữ liệu được hiệu chỉnh sẵn và xuất ra dạng số (digital) qua giao tiếp I²C hoặc SPI.

**\* Sơ đồ chân (trên module breakout phổ biến):**

| **Chân** | **Ký hiệu** | **Chức năng** |
| --- | --- | --- |
| VCC | 3.3V – 5V | Nguồn cấp |
| GND | GND | Mass |
| SDA | I²C data | Dữ liệu (có thể dùng MISO trong SPI) |
| SCL | I²C clock | Xung clock (có thể dùng SCK trong SPI) |
| CS | Chip Select (SPI) | Kéo lên 1 = I²C, kéo xuống 0 = SPI |
| SDO | Địa chỉ I²C (0/1) hoặc MOSI (SPI) |  |

Thông thường khi dùng với ESP32 qua I²C:

* SDA (BME280) → GPIO21 (ESP32)
* SCL (BME280) → GPIO22 (ESP32)

**\* Ưu điểm và nhược điểm:**

Ưu điểm:

* Đo được cả nhiệt độ, độ ẩm và áp suất → tiện lợi, tích hợp cao.
* Kích thước nhỏ, tiêu thụ điện năng thấp.
* Giao tiếp linh hoạt: I²C hoặc SPI.
* Độ chính xác cao hơn so với DHT22 và BMP280.

Nhược điểm:

* Giá thành cao hơn so với cảm biến đơn chức năng.
* Cần thư viện để xử lý dữ liệu (ví dụ Adafruit BME280, SparkFun BME280 cho Arduino/ESP32).

### Bơm nước tự mồi 12V sử dụng động cơ 365/385

**\* Thông số kỹ thuật chính:**

* Điện áp hoạt động: 9–12V DC.
* Dòng điện:
* Không tải: ~0.23A.
* Tải làm việc: 0.5 – 0.7A.
* Công suất tiêu thụ: ~6–8W.
* Lưu lượng nước tối đa: 2 – 3 lít/phút.
* Áp suất tối đa: 1 – 2.5 kg/cm².
* Chiều cao đẩy tối đa: 1 – 2.5 mét.
* Chiều cao hút tối đa: 1 – 2 mét.
* Tuổi thọ trung bình: 2 – 3 năm (trong điều kiện sử dụng bình thường).
* Kích thước tổng thể: 90 × 40 × 35 mm.
* Đường kính ống nước: 8 mm (ngoài).
* Khối lượng: 111 g.

**\* Nguyên lý hoạt động:**

Bơm sử dụng động cơ DC 365/385 gắn với cơ cấu cánh quạt/guồng bơm. Khi cấp điện 9–12V, động cơ quay tạo lực hút chất lỏng từ đường ống vào (IN) và đẩy ra đường ống ra (OUT). Do có khả năng tự mồi, bơm có thể hút nước từ mực thấp hơn (tối đa 1–2 mét) mà không cần đổ nước vào ống trước khi khởi động.

**\* Ứng dụng trong mạch:**

* Tưới cây thông minh: Kết hợp với ESP32 + cảm biến độ ẩm đất, bơm sẽ được điều khiển bật/tắt thông qua MOSFET/Relay tùy vào ngưỡng độ ẩm cài đặt.
* Hệ thống IoT: Bơm có thể kết hợp với cảm biến DHT22, BME280 để kiểm soát môi trường (nhiệt độ, độ ẩm, áp suất).
* Hệ thống làm mát hoặc bơm dung dịch cho các mô hình thí nghiệm.

**\* Ưu điểm và hạn chế:**

Ưu điểm:

* Kích thước nhỏ gọn, giá thành rẻ.
* Lắp đặt dễ dàng, nguồn cấp đơn giản.
* Lưu lượng nước đủ cho các mô hình nhỏ và hệ thống tưới cây mini.
* Có khả năng tự mồi, tiện lợi khi hút nước từ bể chứa thấp hơn.

Hạn chế:

* Không phù hợp với chất lỏng đặc hoặc chứa nhiều tạp chất.
* Tuổi thọ động cơ có giới hạn (2–3 năm).
* Lưu lượng nhỏ, chỉ thích hợp cho quy mô mini/mô hình.

### Module MOSFET 1 kênh F5305S có cách ly

**\* Giới thiệu chung:**

Module MOSFET 1 kênh F5305S là một mạch đóng/ngắt tải DC công suất lớn, được thiết kế để thay thế relay cơ học truyền thống. MOSFET có ưu điểm đóng cắt nhanh, bền, ít phát sinh nhiệt, điều khiển được tải dòng lớn và đặc biệt là có opto cách ly để chống nhiễu, giúp bảo vệ vi điều khiển như ESP32, Arduino, STM32….

**\* Cấu tạo và thành phần chính:**

* MOSFET F5305S: linh kiện bán dẫn chính, chịu trách nhiệm đóng/ngắt dòng tải DC.
* Opto cách ly: tách biệt mạch điều khiển (ESP32) với mạch công suất, giúp an toàn và chống nhiễu.
* Mạch điều khiển tín hiệu: nhận tín hiệu 3V–24V từ vi điều khiển.
* LED báo trạng thái: hiển thị khi MOSFET đang bật/tắt.
* Chân kết nối IN/OUT: ngõ vào điều khiển, ngõ ra tải.

**\* Thông số kỹ thuật:**

* Tín hiệu điều khiển: 3V – 24V DC (tương thích với ESP32 3.3V).
* Điện áp tải: 5V – 80V DC.
* Dòng tải tối đa: 18A (khuyến nghị gắn tản nhiệt khi dùng dòng lớn).
* Kích thước: 44 × 25 × 22 mm.

**\* Chức năng và nguyên lý hoạt động:**

Khi vi điều khiển (ESP32) xuất tín hiệu HIGH (3.3V hoặc 5V) đến chân IN, opto sẽ dẫn và kích hoạt MOSFET F5305S → tải DC được cấp nguồn. Khi tín hiệu LOW (0V), MOSFET ngắt → tải ngừng hoạt động. Nhờ có opto cách ly, mạch điều khiển và mạch công suất không bị ảnh hưởng trực tiếp bởi dòng tải lớn.

**\* Ứng dụng:**

* Thay thế Relay cơ học: đóng ngắt thiết bị DC như bơm mini, motor, quạt, đèn LED.
* Điều khiển PWM: cho phép chỉnh tốc độ động cơ, độ sáng đèn LED, tốc độ bơm.
* Hệ thống IoT: kết hợp với ESP32, Arduino để điều khiển thiết bị từ xa.
* Các ứng dụng cần đóng/ngắt nhanh và tần suất cao (relay cơ học không đáp ứng được).

**\* Ưu điểm và hạn chế:**

Ưu điểm:

* Đóng cắt nhanh, bền, tuổi thọ cao hơn relay cơ học.
* Điều khiển được tải công suất lớn (dòng cao).
* Có cách ly opto, an toàn cho vi điều khiển.
* Hỗ trợ tín hiệu điều khiển 3.3V → dùng trực tiếp với ESP32 không cần mạch trung gian.

Hạn chế:

* Chỉ sử dụng cho tải DC (không dùng cho AC).
* Khi dòng tải lớn (>10A) cần tản nhiệt bổ sung.
* MOSFET có thể hỏng nếu nối sai cực tải hoặc cấp quá dòng định mức.

**\* Điều chế độ rộng xung (PWM) bằng MOSFET:**

- Khái niệm PWM:

PWM (Pulse Width Modulation – Điều chế độ rộng xung) là phương pháp điều khiển bằng cách thay đổi tỷ lệ thời gian ON/OFF của tín hiệu vuông trong một chu kỳ. Thay vì cấp điện áp analog liên tục, vi điều khiển (ESP32, Arduino, STM32, …) phát ra xung số HIGH/LOW rất nhanh → tải (động cơ, LED, bơm) “cảm nhận” như mức điện áp trung bình.

Ví dụ:

* Chu kỳ PWM = 100%.
* Nếu duty cycle = 50%, tín hiệu ON 50% thời gian, OFF 50% thời gian → tải nhận “điện áp trung bình ≈ 0.5 × Vcc”.

- Vai trò của MOSFET trong PWM:

MOSFET đóng vai trò như một công tắc điện tử tốc độ cao, có thể bật/tắt tải DC hàng ngàn lần mỗi giây. Khi ESP32 xuất tín hiệu PWM đến chân Gate của MOSFET, MOSFET sẽ đóng/ngắt nguồn của tải theo đúng chu kỳ PWM. Nhờ khả năng đóng cắt nhanh, MOSFET thích hợp cho điều khiển tốc độ động cơ DC, bơm nước, điều chỉnh độ sáng đèn LED, quạt làm mát…

- Nguyên lý hoạt động PWM với MOSFET:

* ESP32 tạo tín hiệu PWM (ví dụ 1 kHz – 10 kHz).
* Gate MOSFET nhận tín hiệu PWM → MOSFET lần lượt đóng (ON) và mở (OFF) theo duty cycle.
* Tải (motor/đèn/bơm) nhận điện áp trung bình tỉ lệ với duty cycle.

Ví dụ:

* Duty cycle 20% → bơm chạy chậm.
* Duty cycle 50% → bơm chạy trung bình.
* Duty cycle 100% → bơm chạy hết công suất.

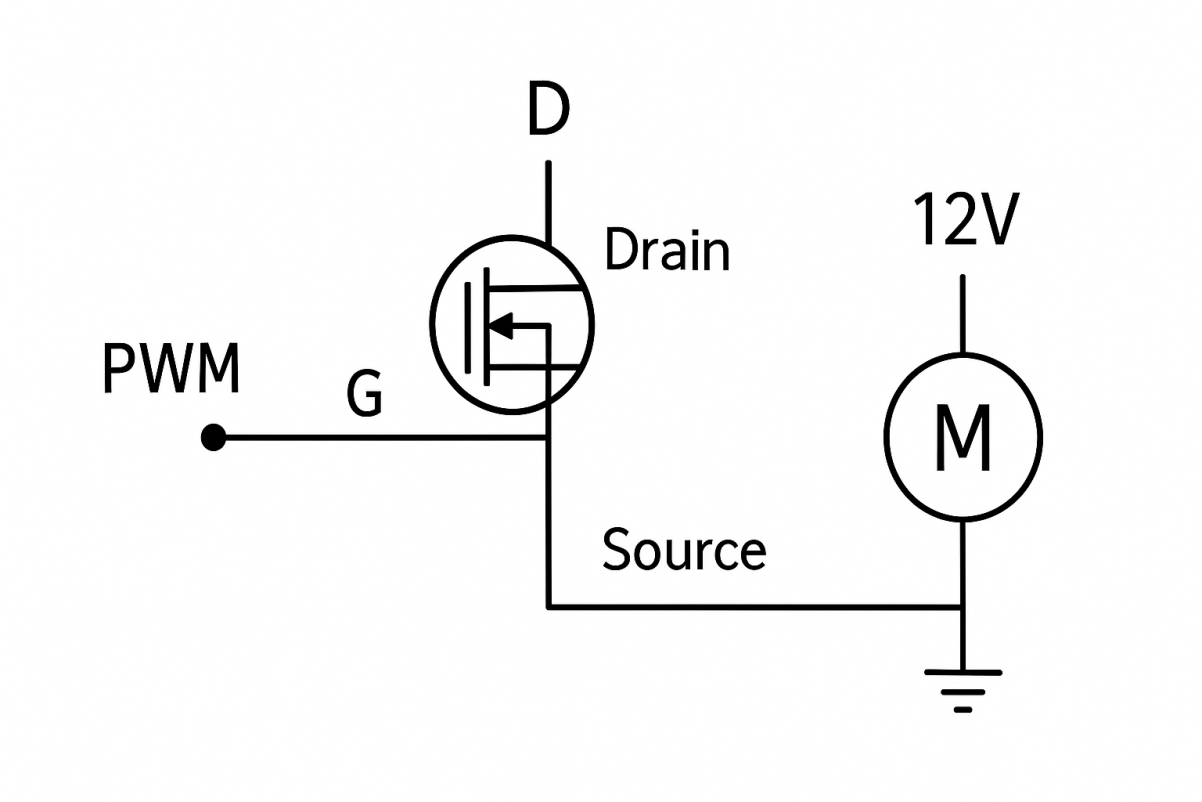
- Ưu điểm PWM bằng MOSFET:

* Hiệu suất cao, ít hao tổn nhiệt so với phương pháp giảm điện áp bằng điện trở.
* Có thể điều khiển mượt tốc độ động cơ DC, độ sáng LED.
* MOSFET đóng cắt nhanh hơn relay → cho phép PWM ở tần số cao (hàng kHz).
* Dễ kết hợp với vi điều khiển như ESP32 (vì GPIO xuất trực tiếp PWM).

- Ứng dụng PWM bằng MOSFET trong hệ thống:

* Điều chỉnh tốc độ bơm mini 12V: tùy vào độ ẩm đất, ESP32 có thể không chỉ bật/tắt mà còn điều chỉnh tốc độ bơm (ví dụ: đất hơi khô thì chạy nhẹ, rất khô thì chạy mạnh).
* Điều khiển LED chiếu sáng: chỉnh độ sáng theo PWM.
* Điều khiển quạt làm mát: thay đổi tốc độ quạt để tiết kiệm năng lượng.

**- Sơ đồ nguyên lý:**



## Phần mềm

### Server/Client

- Môi trường phát triển:

* Arduino IDE: Được sử dụng để lập trình và nạp chương trình điều khiển và vi điều khiển (ESP32/Arduino). Đây là công cụ chính để viết mã, biên dịch và quản lý kết nối với phần cứng IoT.
* Visual Studio Code: Trình soạn thảo mã nguồn hiện đại, hỗ trợ nhiều ngôn ngữ lập trình và có nhiều tiện ích (extension). VS Code được sử dụng để phát triển cả frontend(ReactJS) và backend(NodeJS), đồng thời dễ dàng quản lý dự án với Git.

- Frontend:

* ReactJS: Dùng để xây dựng  giao diện người dùng (UI), cho phép hiển thị trực quan trạng thái của các thiết bị IoT, đồng thời cung cấp các nút điều khiển để tương tác trực tiếp với hệ thống.
* Ngôn ngữ: JavaScript - Ngôn ngữ chính để viết Logic cho giao diện, xử lý sự kiện và gọi API đến Backend Server.

- Backend:

* NodeJS(22.17.1) với ExpressJS: NodeJS cung cấp môi trường chạy JavaScript phia Server. ExpressJS là framework nhẹ trên NodeJS, được sử dụng để xây dựng các API RESTful, giúp kết nối frontend với cơ sở dữ liệu và các thiết bị.
* Ngôn ngữ: JavaScript: Dùng để viết các dịch vụ Backend, xử lý yêu cầu từ Frontend, giao tiếp với cơ sở dữ liệu, và quản lý luồng dữ liệu từ các thiết bị IoT.

- Database:

* MongoDB Atlas: Dịch vụ cơ sở dữ liệu NoSQL trên nền tảng đám mây. MongoDB Atlas giúp lưu trữ dữ liệu từ các thiết bị IoT (ví dụ: nhiệt độ, độ ẩm, trạng thái thiết bị), đồng bộ với Backend và hỗ trợ truy xuất nhanh chóng để hiển thị trên frontend.

- Version Control:

* Git (Github Server): Dùng để quản lý mã nguồn, theo dõi thay đổi và hỗ trợ làm việc nhóm. Github Server đóng vai trò là kho lưu trữ trung tâm, giúp các thành viên dễ dàng chia sẻ, kiểm soát phiên bản, và cộng tác hiệu quả.

### AI

1. Bài toán, giả định và tiêu chí đánh giá

1.1 Bài toán

- Đầu vào: Chuỗi đo cảm biến tại vườn theo bước 5’: nhiệt độ không khí temp\_c, độ ẩm không khí rh\_pct, áp suất khí quyển pressure\_hpa, độ ẩm đất soil\_moist\_pct, mưa 5 phút rain\_mm\_5min (nếu có); thông tin vận hành tưới (irrigation\_events); đặc trưng thời gian (giờ, thứ); tùy chọn bổ trợ từ API thời tiết khu vực.

- Đầu ra AI (hai “đầu” mô hình):

* Nowcast nhị phân: xác suất mưa trong 60’ tới:

p = P(rain\_next\_60=1 | x) → quy ra label 0/1 theo ngưỡng.

* Hồi quy lượng mưa: ước lượng rain\_amount\_next\_60\_mm (mm/60’).

1.2 Giả định

* Tần suất đo ổn định 5’/bản ghi; mỗi device/zone là một chuỗi thời gian riêng.
* Có tối thiểu 30–60 phút lịch sử gần nhất để tạo đặc trưng trễ/rolling.
* Nhãn mưa được suy ra từ rain\_mm\_5min hoặc nguồn ngoại lực đã chuẩn hóa về mm.

1.3 Chỉ số đánh giá (KPI)

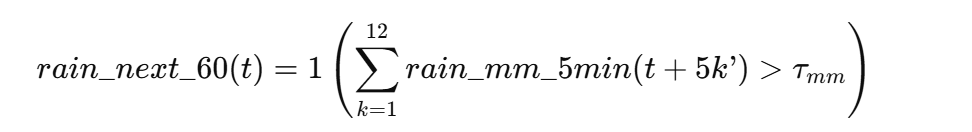
* Nowcast: AUC-ROC, PR-AUC, Precision/Recall/F1 tại ngưỡng lựa chọn; ưu tiên Recall (hạn chế bỏ lỡ mưa).
* Hồi quy: RMSE/MAE; ràng buộc đầu ra ≥ 0.
* Vận hành: giảm ≥25–40% lần tưới trùng mưa; độ trễ suy luận < 300 ms/yêu cầu.

2. Thiết kế dữ liệu & gán nhãn

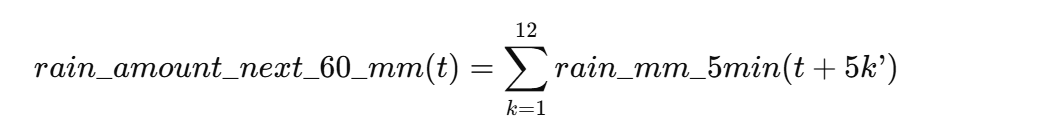
2.1 Lược đồ dữ liệu huấn luyện (tối thiểu)

* sensor\_raw\_60d.csv : ts(UTC), device\_id, temp\_c, rh\_pct, pressure\_hpa, soil\_moist\_pct, rain\_mm\_5min
* irrigation\_events\_60d.csv : device\_id, start\_ts, end\_ts
* labels\_rain\_60d.csv : ts, device\_id, rain\_next\_60 (0/1), tùy chọn rain\_amount\_next\_60\_mm
* external\_weather\_60d.csv (tùy chọn) : các cột như api\_rain\_prob\_60, api\_rain\_mm\_60, api\_cloud, …

2.2 Quy tắc gán nhãn

- Nhãn nhị phân 60’:  


Trong đó τmm là ngưỡng mm (ví dụ 0.1–0.5 mm), chọn theo thực nghiệm để cân bằng PR-AUC/Recall.

- Hồi quy lượng mưa 60’:  


2.3 Làm sạch & đồng bộ thời gian

* Chuẩn UTC, sort theo (device\_id, ts).
* Kiểm tra trùng (device\_id, ts); khử/ghép bản ghi; nội suy bảo thủ nếu đứt nhịp ngắn (≤1–2 bước).
* Chuẩn kiểu số float32, giá trị bất thường (áp suất/độ ẩm) → winsorize/clamp theo IQR.

3. Thiết kế đặc trưng (Feature Engineering)

Được áp dụng riêng theo từng thiết bị/zone theo trật tự thời gian:

- Trễ & trung bình trượt:

* \*\_lag15 = shift(3) với temp\_c, rh\_pct, pressure\_hpa, soil\_moist\_pct.
* \*\_mean30 = rolling(6).mean() (30’).

- Biến thiên/gradient:

* pressure\_delta15 = pressure\_hpa - pressure\_hpa.shift(3)
* temp\_delta15, rh\_delta15.

- Dấu hiệu mưa gần: rain\_in\_last\_15m = rolling(3).sum().gt(0)

- Đặc trưng thời gian: hour\_of\_day, day\_of\_week.

- Đặc trưng tưới:

* is\_irrigating\_now (1 nếu ts ∈ [start, end] một sự kiện tưới),
* irrig\_total\_min\_last\_3h/6h = rolling tổng phút tưới 36/72 bước (5’/bước).

- Ngoại lực (tùy chọn): api\_rain\_prob\_60, api\_rain\_mm\_60, mây gió… sau khi đồng bộ thời gian.

\* Nguyên tắc quan trọng: Bộ tạo đặc trưng FeatureBuilder dùng chung cho train và infer. Thứ tự cột được cố định, lưu vào models/metadata.json để chống “trôi đặc trưng”.

4. Kiến trúc mô hình

4.1 Lựa chọn thuật toán

XGBoost cho cả nhị phân và hồi quy: ổn định với dữ liệu dạng bảng, tận dụng tốt đặc trưng FE, suy luận nhanh & giải thích được (feature importance). So với LSTM/CNN: XGBoost phù hợp hơn khi mỗi bước thời gian đã được “mã hoá” bởi các cửa sổ trễ/rolling; giảm yêu cầu dữ liệu dài & tài nguyên.

4.2 Hai “đầu” mô hình

- Đầu phân loại – xgb\_nowcast:

* objective = binary:logistic, đầu ra là p ∈ (0,1).
* Mất cân bằng lớp: scale\_pos\_weight = neg/pos hoặc reweighting.
* Tham số khởi tạo khuyến nghị:
* eta 0.03–0.05, max\_depth 5–7, subsample 0.8–0.9, colsample\_bytree 0.8–0.9,
* n\_estimators 400–1200, early\_stopping\_rounds 50–100.

- Đầu hồi quy – xgb\_amount

* Dự đoán mm/60’; tiêu chí RMSE/MAE; clamp đầu ra ≥ 0 để tránh giá trị âm.

5. Quy trình huấn luyện & đánh giá

5.1 Chia tập & giữ trật tự thời gian

Không shuffle; tách train:val/test = 85:15 theo mốc thời gian (hoặc TimeSeriesSplit k-fold). Nếu nhiều thiết bị, đảm bảo mỗi device có mặt ở cả train và test theo dòng thời gian để phản ánh vận hành thật.

5.2 Huấn luyện

* Khởi tạo tham số như 4.2; tính scale\_pos\_weight từ tỉ lệ lớp.
* Sử dụng early stopping trên tập validation để tránh overfit; lưu best\_iteration.
* Lưu mô hình .pkl (joblib) và metadata.json (phiên bản mô hình, danh sách FEATURES, ngưỡng mặc định threshold\_default, thông tin chuẩn hoá).

5.3 Tối ưu ngưỡng phân loại

* Theo F1: quét ngưỡng ∈ [0,1] chọn thr\_f1\* tối đa F1.
* Ưu tiên Recall: chọn ngưỡng nhỏ nhất sao cho Recall ≥ mục tiêu (ví dụ 0.80), rồi trong miền đó lấy ngưỡng có Precision cao nhất.
* Ghi threshold\_default vào metadata.json; cho phép chỉnh trên UI.

5.4 Chỉ số & báo cáo

* Nowcast: AUC-ROC, PR-AUC, bảng Precision/Recall/F1 tại các ngưỡng (0.5, thr\_f1\*, thr\_recall\*), confusion matrix.
* Hồi quy: RMSE, MAE, biểu đồ residuals; phần trăm dự báo âm (sau clamp = 0).
* Giải thích mô hình: feature importance theo gain/cover; ablation (loại nhóm đặc trưng để đo mức giảm hiệu năng).

6. Suy luận (infer) & ra quyết định tưới

6.1 Suy luận

* Lấy bản ghi gần nhất (hoặc timestamp người dùng chọn) cho từng device\_id.
* Dùng FeatureBuilder dựng đặc trưng từ lịch sử gần (≥30 phút).
* Nowcast: trả p = predict\_proba(x).
* Hồi quy (tùy chọn): mm = max(0, predict(x)).

6.2 Chính sách quyết định (mặc định – tinh chỉnh theo vườn)

* Nếu p ≥ 0.70 → hoãn tưới 60–90 phút.
* Nếu 0.50 ≤ p < 0.70 và soil\_moist\_pct ≥ mục tiêu → giảm 30–60% thời lượng/lưu lượng.
* Nếu p < 0.50 → tưới theo lịch.
* Nếu mm ≥ 3 mm dù p thấp → cảnh báo xem xét (mưa giông cục bộ).
* Ghi log: device, ts, p, thr, mm, action để hậu kiểm.

7. MLOps cho phần AI (tối thiểu)

* Huấn luyện theo lịch: Cron mỗi tuần; điều kiện kích hoạt nếu có ≥N ngày dữ liệu mới hoặc drift.
* Theo dõi drift:
* Data drift: phân phối pressure\_hpa, rh\_pct, temp\_c, soil\_moist\_pct theo tuần/tháng.
* Prediction drift: trung bình p, tỉ lệ cảnh báo; lệch so với lịch sử.
* Hậu kiểm: khi có mưa thật, tính Precision/Recall theo tháng/season.
* Quản lý phiên bản: model\_version, dataset\_hash, feature\_schema; giữ ≥3 bản để rollback.
* Độ tin cậy: timeout suy luận < 300 ms; nếu không đủ lịch sử để dựng FE → dùng rule-based an toàn (tưới theo lịch + cảnh báo)

8. Kế hoạch thí nghiệm & đường cơ sở

- Thiết lập baseline:

* Đặc trưng: chỉ dùng temp\_c, rh\_pct, pressure\_hpa, soil\_moist\_pct, rain\_mm\_5min.
* Mô hình: Logistic Regression (chuẩn hoá z-score).
* Mục tiêu: làm mốc AUC/PR-AUC để so sánh cải tiến.

- Bản cải tiến 1 (FE cơ bản):

* Thêm lag15, mean30, delta15, rain\_in\_last\_15m, thời gian.
* Huấn luyện XGBoost; so sánh PR-AUC, Recall@0.5.

- Bản cải tiến 2 (tích hợp tưới + ngoại lực):

* Thêm is\_irrigating\_now, rolling tưới 3/6h; api\_rain\_prob\_60.
* Đo mức tăng điểm và phân tích trọng số đặc trưng.

- Ablation & chọn ngưỡng:

* So sánh khi bỏ nhóm áp suất/độ ẩm đất/ngoại lực.
* Chọn threshold\_default theo mục tiêu vận hành (Recall cao).

9. Phân tích rủi ro & cách giảm thiểu (riêng phần AI)

* Thiếu nhãn mưa chính xác: giai đoạn đầu có thể dựa vào tổng rain\_mm\_5min và API; khuyến nghị lắp cảm biến mưa để nâng độ tin cậy.
* Sai lệch FE giữa train và infer: đóng gói FeatureBuilder thành module dùng chung; kiểm thử bằng “golden samples”.
* Mất cân bằng lớp (ít mưa): dùng scale\_pos\_weight, oversample có kiểm soát, tối ưu ngưỡng theo Recall.
* Chuỗi dữ liệu đứt mạch: rolling với min\_periods và cảnh báo mất dữ liệu; bỏ suy luận nếu thiếu lịch sử.
* Khí hậu theo mùa: tinh chỉnh ngưỡng theo tháng/mùa; huấn luyện mô hình theo mùa hoặc thêm one-hot season.

10. Kết quả mong đợi & định hướng mở rộng

* Ngắn hạn: đạt PR-AUC ≥ 0.50, Recall@thr ≥ 0.75; dashboard hiển thị xác suất & quyết định tưới minh bạch; tiết kiệm ≥ 15–30% nước.
* Trung hạn:
* Thêm nowcast đa mốc (15/30/60’) để ra quyết định linh hoạt theo khung giờ.
* Tự động lịch tưới tuần dựa trên dự báo lăn, ưu tiên giờ mát/ít gió.
* Đóng gói ONNX để suy luận nhẹ trên edge (Raspberry Pi/mini-PC).
* Dài hạn: thử nghiệm mô hình lai: XGBoost + encoder chuỗi ngắn (TabNet/Temporal Convolution) khi dữ liệu đủ dài; học chuyển giao giữa các khu vườn.

11. Kết luận

Phần AI của hệ thống tập trung vào nowcasting mưa cục bộ với XGBoost và bộ đặc trưng thời gian/hoạt động tưới được thiết kế cẩn thận. Cách tiếp cận này phù hợp bối cảnh IoT nông nghiệp: nhẹ – nhanh – dễ giải thích, đồng thời đủ linh hoạt để mở rộng. Mấu chốt bảo đảm chất lượng là: (i) FE nhất quán giữa train và infer, (ii) tối ưu ngưỡng theo mục tiêu vận hành (ưu tiên Recall), và (iii) MLOps tối thiểu để giám sát drift, tái huấn luyện định kỳ.  
Khi triển khai thực địa vài tuần, mô hình sẽ học “dấu vân tay vi khí hậu” của khu vườn, từ đó cải thiện khả năng phát hiện mưa sớm và tự động hoá quyết định tưới một cách an toàn, tiết kiệm và minh bạch.

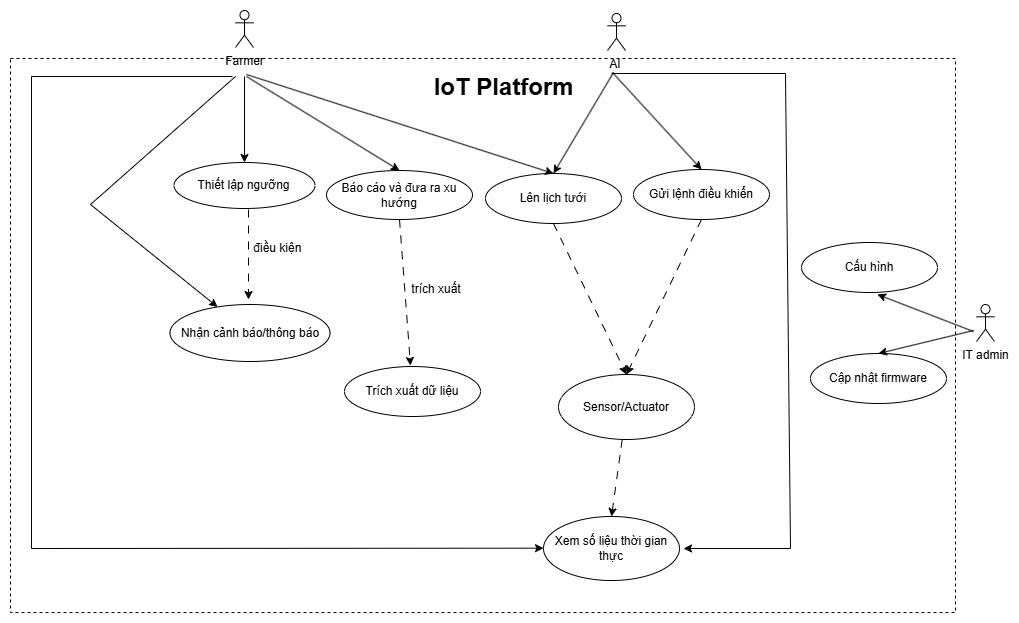
# Biểu diễn chức năng

## Các tác nhân trong hệ thống

- Người dùng (nông dân, người quản lý hệ thống).

- Hệ thống tưới thông minh (IoT + AI).

1. **Use Case tổng quan hệ thống**



1. **Kế hoạch triển khai**
2. **Phân công công việc**

### Công việc chính:

* Làm báo cáo giữa kỳ
* Kiểm tra báo cáo cuối kỳ
* Test sản phẩm sau toàn bộ quá trình xây dựng

### Phân chia công việc

|  |  |  |
| --- | --- | --- |
| B22DCCN482 | Trịnh Quang Lâm | Thuyết trình |
| Code Backend + Frontend |
| B22DCCN434 | Vũ Nhân Kiên | Làm Slide |
| Code Backend + Frontend |
| B22DCCN422 | Cao Thị Thu Hương | Làm báo cáo cuối kỳ |
| Code phần cứng |
| B22DCCN889 | Vũ Thế Văn | Code AI |
| Làm báo cuối kỳ |

## Tiến độ dự kiến

|  |  |  |
| --- | --- | --- |
| **Tuần** | **Nội dung chính** | **Kết quả dự kiến** |
| Tuần 1  (17/09 - 23/09) | - Xác định đề tài, mục tiêu, phạm vi hệ thống. - Phân công nhiệm vụ trong nhóm. - Tìm hiểu các thành phần phần cứng (ESP32, cảm biến DHT22, BME280, Soil Sensor, bơm nước). | Hoàn thiện đề cương ý tưởng, chọn mô hình cảm biến và bơm |
| Tuần 2  (24/09 - 01/10) | - Nghiên cứu nguyên lý hoạt động các cảm biến. - Lập sơ đồ nguyên lý và sơ đồ kết nối phần cứng. - Chuẩn bị linh kiện, kiểm tra mạch cơ bản. | Sơ đồ kết nối ESP32 với cảm biến và bơm hoạt động ổn định. |
| Tuần 3  (30/09 - 06/10) | - Viết chương trình ESP32 đọc dữ liệu cảm biến - Gửi dữ liệu qua Serial / Wi-Fi đến server thử nghiệm - Kiểm tra tín hiệu từ cảm biến | ESP32 đọc và gửi được dữ liệu nhiệt độ, độ ẩm, độ ẩm đất. |
| Tuần 4  (07/10 - 13/10) | - Xây dựng backend (NodeJS + ExpressJS) - Thiết kế API nhận dữ liệu từ ESP32 và lưu MongoDB - Kiểm thử API với dữ liệu mẫu | Backend hoạt động, lưu dữ liệu cảm biến thành công. |
| Tuần 5  (14/10 - 20/10) | - Phát triển frontend (ReactJS) - Hiển thị dữ liệu cảm biến theo thời gian thực - Thiết kế giao diện hiển thị độ ẩm, nhiệt độ, trạng thái bơm | Giao diện web cơ bản hiển thị dữ liệu IoT. |
| Tuần 6  (21/10 - 27/10) | - Thiết kế và huấn luyện mô hình AI (XGBoost) dự báo mưa - Chuẩn bị dữ liệu huấn luyện, xử lý feature - Đánh giá độ chính xác ban đầu (AUC, Recall) | Mô hình AI huấn luyện được, có thể dự đoán xác suất mưa |
| Tuần 7  (28/10 - 03/11) | - Tích hợp AI với backend - Xây dựng API cho dự đoán thời tiết - Thử nghiệm điều kiện tưới tự động dựa trên kết quả AI | Hệ thống tự động tưới theo dự báo mưa hoạt động |
| Tuần 8  (04/11 - 10/11) | - Hoàn thiện giao diện web (hiển thị lịch tưới, biểu đồ, điều khiển thủ công) - Kiểm thử toàn hệ thống (phần cứng – backend – frontend – AI) | Toàn bộ luồng dữ liệu hoạt động ổn định, giao diện hoàn chỉnh |
| Tuần 9  (11/11 - 17/11) | - Viết báo cáo giữa kỳ (theo mục P1–P2) - Tổng hợp hình ảnh, kết quả kiểm thử - Chuẩn bị slide thuyết trình | Báo cáo giữa kỳ hoàn chỉnh, slide trình bày sẵn sàng |
| Tuần 10  (18/11 - 24/11) | - Nộp và trình bày báo cáo giữa kỳ - Thu thập phản hồi từ giảng viên - Lên kế hoạch cải tiến giai đoạn cuối kỳ (bổ sung AI hoặc mở rộng cảm biến) | Báo cáo giữa kỳ được chấm và phản biện xong. |

1. **Kết luận**

Hệ thống Tưới cây thông minh ứng dụng AI để lên lịch tưới hàng tuần là một bước tiến nhỏ nhưng có ý nghĩa trong việc ứng dụng IoT và Trí tuệ nhân tạo vào nông nghiệp hiện đại.  
 Bằng cách kết hợp vi điều khiển ESP32, các cảm biến môi trường (DHT22, BME280, cảm biến độ ẩm đất), bộ điều khiển bơm tự động và mô hình AI dự báo mưa (XGBoost), nhóm đã xây dựng được một giải pháp có khả năng tự động hóa việc tưới tiêu, tiết kiệm nước, và tăng hiệu quả chăm sóc cây trồng.