HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN I

BÁO CÁO BÀI TẬP LỚN

MÔN: IOT VÀ ỨNG DỤNG

ĐỀ TÀI: CỬA CHỐNG NGẬP TỰ ĐỘNG

Nhóm bài tập lớn:	**04**
Lớp:	**E22HTTT**
Giảng viên giảng dạy:	**Kim Ngọc Bách**
Thành viên:	Phạm Anh Minh – B22DCAT192
	Nguyễn Đức Trí – B22DCAT302
	Khuất Quang Đông – B22DCVT146
	Nguyễn Đại Phát – B22DCVT393

Hà Nội – 11/2025

Mục lục

LỜI CẢM ƠN

Trước tiên, nhóm chúng em xin gửi lời cảm ơn chân thành và sâu sắc đến thầy Kim Ngọc Bách, giảng viên phụ trách môn IOT và ứng dụng, người đã tận tâm truyền đạt kiến thức, định hướng chuyên môn và luôn nhiệt tình hỗ trợ chúng em trong suốt quá trình học tập và thực hiện đề tài.

Bên cạnh đó, nhóm xin chân thành cảm ơn sự hợp tác tích cực và tinh thần trách nhiệm cao của các thành viên trong nhóm: Nguyễn Đức Trí, Khuất Quang Đông, Nguyễn Đại Phát và Phạm Anh Minh. Mỗi thành viên đều đã nỗ lực đóng góp ý tưởng, kiến thức và thời gian để cùng nhau xây dựng, phân tích, thiết kế và hoàn thiện hệ thống một cách hiệu quả nhất. Nhờ sự đồng lòng và nghiêm túc trong quá trình làm việc, nhóm đã hoàn thành báo cáo đúng thời hạn với nội dung và chất lượng phù hợp với yêu cầu môn học.

Mặc dù nhóm đã cố gắng hết sức trong việc nghiên cứu và triển khai, nhưng do giới hạn về thời gian cũng như kinh nghiệm thực tế còn hạn chế, báo cáo chắc chắn vẫn còn những thiếu sót nhất định. Rất mong thầy sẽ thông cảm và góp ý để nhóm có thể rút kinh nghiệm và hoàn thiện hơn trong những dự án sau này. Một lần nữa, nhóm xin chân thành cảm ơn sự tận tâm của thầy cùng tinh thần hợp tác trách nhiệm của các thành viên đã góp phần tạo nên kết quả của bài báo cáo này.

I. Giới thiệu đề tài

1. Tên đề tài

Hệ thống cửa chống ngập thông minh (Smart Flood Barrier System) - Một giải pháp IoT tích hợp Machine Learning để tự động hóa việc chống ngập lụt tại các khu vực:

2. Bối cảnh thực tế và vấn đề

Tình trạng ngập lụt cục bộ tại các đô thị Việt Nam, đặc biệt là tại các hầm gửi xe chung cư, nhà phố thấp trũng, xảy ra thường xuyên và gây thiệt hại lớn về tài sản. Các giải pháp hiện tại có những hạn chế:

- Phụ thuộc vào con người: Cần người trực để đóng/mở cửa chắn, không hoạt động 24/7.
- Phản ứng chậm: Chỉ phát hiện khi nước đã vào, không có cảnh báo sớm.
- Không có giám sát liên tục: Không theo dõi được mực nước và điều kiện môi trường.
- Thiếu dự báo: Không có khả năng dự đoán nguy cơ ngập dựa trên điều kiện thời tiết.

3. Mục tiêu dự án

Dự án nhằm xây dựng một hệ thống hoàn chỉnh với các mục tiêu cụ thể:

- Tự động hóa hoàn toàn: Hệ thống tự động đo mực nước và điều khiển cửa chắn mà không cần can thiệp con người.
- Tích hợp Machine Learning: Sử dụng mô hình dự đoán để cảnh báo sớm nguy cơ ngập dựa trên dữ liệu thời tiết.
- Giám sát từ xa: Web Dashboard cho phép theo dõi và điều khiển hệ thống từ bất kỳ đâu.
- Chi phí thấp: Sử dụng các linh kiện phổ biến, giá rẻ (ESP32, cảm biến HC-SR04, DHT22).

4. Phạm vi thực hiện

- Phần cứng: Xây dựng mô hình prototype mini với ESP32, cảm biến, và servo motor.
- Mô phỏng: Sử dụng Wokwi để mô phỏng và kiểm thử trước khi triển khai thực tế.
- Web Application: Phát triển Dashboard Next.js (MVP) với đầy đủ tính năng giám sát và điều khiển.
- Machine Learning: Huấn luyện và nhúng mô hình Logistic Regression vào ESP32.

II. Phân công chi tiết nhiệm vụ từng thành viên

Process_checklist

Task	Detail	Status	Priority	Dev
Sơ đồ phần cứng và lắp ráp	Thiết kế sơ đồ + chốt BOM Đấu dây, lắp mạch trên breadboard Lắp cơ khí cửa	Đã hoàn thành	Cao	Đông
Firmware ESP32 + mô phỏng Wokwi	Hoàn thiện sơ đồ Wokwi Code đọc HC-SR04, DHT22, điều khiển servo, nút, LED Viết cơ chế đóng/mở tùy thuộc vào điều kiện từ sensor	Đã hoàn thành	Cao	Minh
Mô hình giả lập hệ thống thực tế	Thiết kế + lắp ráp mô hình bằng tấm alu	Không hoàn thành	Trung bình	Phát
Machine Learning (Notebook Kaggle)	Train Logistic Regression, đánh giá model Tích hợp hàm ML vào code ESP32	Đã hoàn thành	Trung bình	Minh
Logic code và cấu hình các phần cứng thiết bị		Đã hoàn thành	Trung bình	Trí + Đông + Phát
Web/App + tích hợp Telegram	Tạo web dashboard cho sản phẩm Tạo API nhận dữ liệu Kết nối ESP32 với web Tích hợp Telegram Bot gửi cảnh báo	Đã hoàn thành	Cao	Minh
Tích hợp & test tổng thể	Viết kịch bản test Kiểm thử End to End toàn bộ hệ thống	Đang kiểm thử	Cao	Đông
Slide		Đã hoàn thành	Trung bình	Đông + Phát
Nội dung báo cáo		Đã hoàn thành	Trung bình	Minh
Làm báo cáo		Đã hoàn thành	Trung bình	Đông

III. Công nghệ, lý thuyết áp dụng

1. Kiến trúc hệ thống IoT 3 lớp

Hệ thống được thiết kế theo mô hình IoT 3 lớp chuẩn:

1.1. Lớp 1: Lớp thiết bị (Device Layer)

- **MCU: ESP32 DevKit v1:**
  - Vi xử lý dual-core 240MHz.
  - Hỗ trợ WiFi 802.11 b/g/n.
  - Bộ nhớ: 520KB SRAM, 4MB Flash.
  - GPIO đa dạng, hỗ trợ PWM cho servo.
- **Cảm biến:**
  - **HC-SR04 (Ultrasonic Sensor): Đo khoảng cách mực nước:**
    - Nguyên lý: Phát sóng siêu âm 40kHz, đo thời gian phản hồi.
    - Công thức tính: d = (t x 0.0343) / 2 (cm).
    - Tầm đo: 2cm - 400cm.
    - Độ chính xác: ± 3mm.
  - **DHT22 (Temperature & Humidity Sensor): Đo nhiệt độ & độ ẩm:**
    - Đo nhiệt độ: -40°C đến 80°C, độ chính xác ± 0.5°C.
    - Đo độ ẩm: 0-100% RH, độ chính xác ±2% RH.
    - Tần số đọc: 2 giây/lần.
- **Cơ cấu chấp hành:**
  - **Servo Motor (SG90) - điều khiển cửa chắn**
    - Góc quay: 0° (mở hoàn toàn) ⟶ 90° (đóng).
    - Độ phân giải: 1°.
    - Mô-men xoắn: 1.8 kg/cm .
    - Điều khiển PWM: 50Hz, pulse width 500-2400µs.
- **Giao diện người dùng tại chỗ:.**

- LED cảnh báo (GPIO 2) với điện trở 220Ω.
- 2 công tắc gạt (Slide Switch) cho điều khiển thủ công.

1.2. Lớp 2: Lớp mạng (Network Layer)

- **Giao thức:** HTTP/HTTPS (REST API).
- **Kết nối:** WiFi 802.11 b/g/n.
- **Giao tiếp:**
  - ESP32 → Server: POST /api/status (gửi dữ liệu cảm biến).
  - ESP32 ← Server: GET /api/control/latest (polling lệnh điều khiển).
- **Xác thực:** Device API Key trong headers (x-device-id, x-api-key).

1.3. Lớp 3: Application Layer (Lớp Ứng dụng)

- **Web Dashboard**: Next.js 14 với App Router.
- **Database**: SQLite (development) / MySQL (production).
- **Authentication**: NextAuth.js cho người dùng, API Key cho thiết bị.

2. Lý thuyết Machine Learning

2.1. Thuật toán: Logistic Regression

- Dự án sử dụng Logistic Regression (Hồi quy Logistic) cho bài toán phân loại nhị phân dự đoán mưa lớn.
- Lý do lựa chọn:
  - Mô hình nhẹ: Chỉ cần 2 trọng số và 1 bias, phù hợp với ESP32 có bộ nhớ hạn chế.
  - Tính toán nhanh: Interface chỉ cần vài phép nhân và hàm sigmoid.
  - Dễ diễn giải: Có thể hiểu được ý nghĩa của từng tham số.
  - Dễ nhúng: Có thể implement trực tiếp bằng C++ trên ESP32.
- Công thức toán học:

$$z = -18.496 + 0.235 \times TempMean + 0.151 \times Humidity$$

$$P(HeavyRain) = \frac{1}{1 + e^{-z}}$$

Trong đó:

- TempMean: Nhiệt độ trung bình ngày (°C).
- Humidity: Độ ẩm trung bình (%).
- w_1, w_2: Trọng số (weights) được học từ dữ liệu.
- bias: Hệ số điều chỉnh (intercept).

- **Hàm Sigmoid:** Chuyển đổi giá trị tuyến tính z thành xác suất trong khoảng [0, 1].

2.2. Quy trình huấn luyện mô hình

a. Dataset: Vietnam Weather Data từ Kaggle

- 181,960 bản ghi thời tiết các tỉnh/thành phố Việt Nam.
- Thời gian: 2009-2019.
- Các đặc trưng: Nhiệt độ max/min, độ ẩm, lượng mưa, v.v.

b. Feature Engineering:

- Tính $TempMean = \frac{MaxTemp + MinTemp}{2}$.
- Tạo label HeavyRainTomorrow: 1 nếu mưa ≥ 20mm, 0 nếu ngược lại.
- Sử dụng shift(-1) để lấy lượng mưa ngày hôm sau.

c. Chia dữ liệu (Chia theo thời gian (time-based split) để tránh data leakage):

- Train: 80% (116,428 mẫu).
- Validation: 20% của train (29,108 mẫu).
- Test: 20% (36,384 mẫu).
- **Lưu ý**: Chia theo thời gian (time-based split) để tránh data leakage.

d. Huấn luyện:

- Sử dụng sklearn.linear_model.LogisticRegression.
- class_weight="balanced" để xử lý class imbalance.
- max_iter=1000 để đảm bảo hội tụ.

e. Kết quả:

- **Intercept (bias):** -18.496.
- **Weight TempMean:** 0.235.
- **Weight Humidity:** 0.151.
- **Test Accuracy:** 69.4%.
- **Test AUC-ROC:** 0.771.

3. Công nghệ phần mềm

3.1. Firmware (ESP32)

- **Ngôn ngữ: C++ (Arduino Framework).**
- **Thư viện chính:**
  - DHT.h: Đọc cảm biến DHT22.
  - ESP32Servo.h: Điều khiển servo motor.
  - math.h: Hàm expf() cho sigmoid.
- **Kiến trúc code:**
  - **State Machine:** Quản lý chế độ AUTO/MANUAL.
  - **Smoothing Algorithm:** Cập nhật góc servo dần dần (1°/loop) để tránh giật.
  - **Backup Plan:** Xử lý khi cảm biến siêu âm lỗi, dùng nhiệt độ/độ ẩm.

3.2. Web Application

a. Frontend:

- **Framework:** Next.js 14 (App Router).
- **UI Library:** shadcn/ui (dựa trên Radix UI).
- **Styling:** Tailwind CSS.
- **Charts:** Recharts (React charting library).
- **Icons:** Lucide React.

b. Backend:

- **API Routes:** Next.js API Routes (Serverless).
- **ORM:** Prisma (type-safe database client).
- **Database:** SQLite (dev) / MySQL (production).

- **Authentication:** NextAuth.js.
- **Password Hashing:** bcryptjs.

c. Database Schema (Prisma):

- **User:** Thông tin người dùng, phân quyền (ADMIN/VIEWER).
- **Device:** Thông tin thiết bị, API keys.
- **DeviceStatus:** Lịch sử trạng thái (mực nước, nhiệt độ, độ ẩm, ML prediction).
- **ControlCommand:** Lệnh điều khiển từ web.
- **AlertLog:** Lịch sử cảnh báo Telegram.

IV. Phân tích yêu cầu

1. Tính năng cơ bản (Basic Features)

1.1. Đo mực nước tự động

- **Yêu cầu**: Hệ thống phải đo mực nước liên tục với tần suất cao (mỗi 100ms trong loop chính).
- **Triển khai:**
  - Sử dụng cảm biến HC-SR04.
  - Đọc giá trị mỗi chu kỳ loop.
  - Xử lý lỗi: Nếu distance > 404cm (max_distance), coi là lỗi và chuyển sang Backup Plan.
- **Code thực tế** (từ src/main.cpp):

```
float readDistanceCm() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH, 30000);
  if (duration == 0) return 9999.0f; // Lỗi

  float distance = duration * 0.0343f / 2.0f;
  return distance;
}
```

1.2. Điều khiển cửa tự động (Auto Mode)

- **Logic điều khiển:**
  - **Ngưỡng cơ bản:**
    - LEVEL_HIGH_CM_BASE = 20cm: Nếu mực nước ≤ 20cm → Mở cửa (SERVO_OPEN_ANGLE = 0°).
    - LEVEL_LOW_CM_BASE = 40cm: Nếu mực nước ≥ 40cm → Đóng cửa (SERVO_CLOSE_ANGLE = 90°).

- Khoảng giữa (20-40cm): Cửa ở vị trí giữa (SERVO_MID_ANGLE = 30°).

■ **Điều chỉnh ngưỡng động dựa trên ML:**

```
// Nếu xác suất mưa ≥ 80% (P_HEAVY_RAIN_HIGH)
if (pHeavyRainNext24h >= 0.8f) {
  levelHigh = 20.0f + 10.0f;  // = 30cm (mở sớm hơn)
  levelLow  = 40.0f + 20.0f;  // = 60cm (đóng muộn hơn)
}
// Nếu xác suất mưa ≥ 60% (P_HEAVY_RAIN_WARN)
else if (pHeavyRainNext24h >= 0.6f) {
  levelHigh = 20.0f + 5.0f;   // = 25cm
  levelLow  = 40.0f + 10.0f;  // = 50cm
}
```

- **Ý nghĩa**: Khi ML dự đoán mưa lớn, hệ thống tự động hạ ngưỡng kích hoạt để phòng ngừa sớm hơn.

1.3. Điều khiển thủ công (Manual Mode)

- **Cơ chế:**
  - ■ 2 công tắc gạt vật lý:
    - SW_OPEN_PIN (GPIO 26): Gạt ON → Mở cửa.
    - SW_CLOSE_PIN (GPIO 27): Gạt ON → Đóng cửa.
  - ■ Khi bất kỳ công tắc nào được kích hoạt → manualOverride = true.
  - ■ Khi cả 2 công tắc đều OFF → Trở về AUTO mode
- **Code:**

```
bool swOpenActive  = (digitalRead(SW_OPEN_PIN)  == LOW);
bool swCloseActive = (digitalRead(SW_CLOSE_PIN) == LOW);

if (swOpenActive && !swCloseActive) {
  manualOverride = true;
  targetAngle = SERVO_OPEN_ANGLE;
}
else if (!swOpenActive && swCloseActive) {
  manualOverride = true;
  targetAngle = SERVO_CLOSE_ANGLE;
}
else {
  manualOverride = false; // Trở về AUTO
}
```

- **Điều khiển từ Web:**
  - ■ Admin có thể gửi lệnh OPEN, CLOSE, hoặc AUTO qua Dashboard.
  - ■ ESP32 polling /api/control/latest mỗi 5-10 giây để lấy lệnh.

1.4. Hệ thống cảnh báo

- **Cảnh báo tại chỗ:**

- LED (GPIO 2) bật khi: pHeavyRainNext24h >= 0.6 (60%).

- **Cảnh báo Telegram:**
  - Điều kiện kích hoạt (một trong hai điều kiện dưới đây):
    1. pHeavyRainNext24h >= 0.8 (80% xác suất mưa lớn).
    2. waterLevelCm <= 20 (mực nước nguy hiểm).
  - Nội dung cảnh báo:
    - Device ID.
    - Mực nước hiện tại.
    - Xác suất mưa lớn (%).
    - Trạng thái cửa và chế độ.
    - Nhiệt độ và độ ẩm.
    - Timestamp.
- **Code từ Web App** (App/app/api/status/route.js):

```
const shouldAlert =
  status.pHeavyRainNext24h >= 0.8 || status.waterLevelCm <= 20;

if (shouldAlert) {
  const severity =
    status.waterLevelCm <= 20 || status.pHeavyRainNext24h >= 0.9
      ? 'CRITICAL' : 'WARN';

  await sendTelegramAlert(alertMessage);
}
```

2. Tính năng nâng cao (Advanced features)

2.1. Dự báo thông minh bằng Machine Learning

- **Tính năng**: Hệ thống tính toán xác suất mưa lớn trong 24h tới ngay trên ESP32.
- **Implementation** (từ src/main.cpp):

```
float predictHeavyRainNext24h(float temp_c, float humidity_pct) {
  const float BIAS   = -18.49602386139671f;
  const float W_TEMP = 0.235201f;
  const float W_HUM  = 0.151022f;

  float z = BIAS
          + W_TEMP * temp_c
          + W_HUM  * humidity_pct;

  float p = 1.0f / (1.0f + expf(-z));   // sigmoid
  return p;                             // 0..1
}
```

- **Sử dụng trong loop:**

float pHeavyRainNext24h = predictHeavyRainNext24h(temperature, humidity);

- **Kết quả**: Giá trị từ 0.0 (0%) đến 1.0 (100%), được hiển thị trên Dashboard và sử dụng để điều chỉnh ngưỡng.

2.2. Ngưỡng đo động (Adaptive Thresholds)

- Như đã trình bày ở mục 3.1.2, hệ thống tự động điều chỉnh ngưỡng mực nước dựa trên xác suất mưa từ ML model. Đây là điểm sáng tạo chính của dự án: **không chỉ phản ứng với nước, mà còn phòng ngừa dựa trên dự báo**

2.3. Backup plan (Kế hoạch dự phòng)

- **Vấn đề**: Cảm biến siêu âm có thể lỗi hoặc bị che khuất.
- **Giải pháp**: Sử dụng nhiệt độ và độ ẩm để phát hiện điều kiện bất thường.
- **Logic** (từ src/main.cpp):

```
void backup_plan(float last_hum, float now_hum,
                 float last_temp, float now_temp) {
  // Tính delta (thay đổi) trong khoảng thời gian (mặc định 5 phút)
  float delta_hum = now_hum - last_hum;
  float delta_temp = last_temp - now_temp; // Nhiệt độ giảm = delta dương

  // Ngưỡng critical (mở cửa ngay)
  if (last_hum >= 98) {
    targetAngle = SERVO_OPEN_ANGLE;
  }
  else if (delta_hum >= lim_hum && delta_temp >= lim_temp) {
    // Độ ẩm tăng nhanh + nhiệt độ giảm nhanh → Mưa sắp đến
    targetAngle = SERVO_OPEN_ANGLE;
  }
  // Ngưỡng warning (chỉ cảnh báo)
  else if (delta_hum >= warn_hum && delta_temp >= warn_temp) {
    gui_canh_bao();
  }
}
```

- **Lưu trữ lịch sử**: Hệ thống lưu giá trị nhiệt độ/độ ẩm mỗi 5 phút để so sánh.

2.4. Dashboard realtime

2.4.1. Tính năng chính:

- **Summary Cards** (4 thẻ thông tin):
  - **Mực nước**: Hiển thị khoảng cách (cm), mã màu theo mức độ nguy hiểm.
  - **Trạng thái cửa & Chế độ**: OPEN/MID/CLOSE, AUTO/MANUAL.
  - **Rủi ro mưa lớn 24h**: Xác suất từ ML model (0-100%), mã màu.

- **Môi trường**: Nhiệt độ (°C) và độ ẩm (%).
- **Biểu đồ thời gian thực:**
  - **Line Chart mực nước**: Hiển thị 50 điểm dữ liệu gần nhất.
  - **Line Chart xác suất mưa**: Theo dõi xu hướng rủi ro.
- **Bảng điều khiển** (chỉ ADMIN):
  - 3 nút: OPEN, CLOSE, AUTO.
  - Hiển thị lệnh cuối cùng và thời gian.
- **Bảng sự kiện (Event Log):**
  - Trang /logs hiển thị lịch sử đầy đủ.
  - Các cột: Timestamp, Water Level, Door State, Mode, Rain Risk, Temperature, Humidity.
- **Cập nhật tự động:**
  - Polling mỗi 5 giây.
  - Phát hiện thiết bị offline nếu không nhận dữ liệu > 60 giây.

2.4.2. Công nghệ sử dụng:

- **Recharts**: Thư viện biểu đồ React.
- **shadcn/ui**: UI components (Card, Button, Badge, Table).
- **Next.js API Routes**: /api/status/latest, /api/logs, /api/control.

2.5. Hệ thống xác thực và phân quyền

- **Xác thực người dùng:**
  - NextAuth.js với email/password.
  - Mật khẩu được hash bằng bcryptjs.
  - Session management tự động.
- **Phân quyền:**
  - **ADMIN**: Xem dashboard, gửi lệnh điều khiển, quản lý thiết bị.
  - **VIEWER**: Chỉ xem (read-only), không thể điều khiển.
- **Xác thực thiết bị:**
  - Mỗi thiết bị có API key duy nhất.
  - ESP32 gửi headers: x-device-id, x-api-key.
  - Server verify và chỉ cho phép thiết bị hợp lệ.

2.6. Luồng phân quyền thiết bị & onboarding người dùng mới

Đây là phần mở rộng quan trọng được hoàn thiện trong giai đoạn cuối của đồ án. Mục tiêu là biến hệ thống từ "prototype một thiết bị" thành một nền tảng có thể giao thiết bị cho khách hàng, kết nối vào mạng thực tế và phân quyền sử dụng trên Dashboard một cách dễ dàng – tương tự trải nghiệm khi cài đặt các thiết bị IoT thương mại:

2.6.1. Chuẩn bị & kết nối thiết bị mới

- **Cấp Device ID + API Key**: quản trị viên đăng nhập Dashboard (role ADMIN), tạo bản ghi Device mới trong database (qua Prisma Studio hoặc form nội bộ). Mỗi thiết bị có id duy nhất (ví dụ esp32-b3) và chuỗi API key ngẫu nhiên 64 ký tự hex.

- **Cấu hình firmware:** kỹ thuật viên flash firmware ESP32 và điền cặp DEVICE_ID, DEVICE_API_KEY, cùng địa chỉ SERVER_BASE = http://<IP-server>:3000. Tại hiện trường, thiết bị phát một WiFi AP tạm (FloodBarrier-Setup). Người cài đặt kết nối vào AP này, nhập WiFi thật + IP server thông qua trang cấu hình 192.168.4.1, sau đó thiết bị tự kết nối về mạng nội bộ.
- **Kiểm tra kết nối:** ESP32 gửi bản tin POST /api/status đầu tiên kèm headers x-device-id, x-api-key. Server xác thực, lưu bản tin vào bảng DeviceStatus và hiển thị trạng thái "Online" trên Dashboard.

2.6.2. Onboarding người dùng trên ứng dụng web

- **Tạo tài khoản:** ADMIN tạo user mới (ví dụ baove1@toa-nha.com) hoặc sử dụng tài khoản seed (viewer@example.com). NextAuth quản lý mật khẩu và session.
- **Gán quyền truy cập thiết bị:** ADMIN mở trang "Permissions" (đã bổ sung trong sidebar). Các bước thao tác:
  - Panel bên trái liệt kê toàn bộ thiết bị (được lấy từ API /api/devices?all=true).
  - Chọn thiết bị cần phân quyền → bảng bên phải hiển thị danh sách user đang có quyền (UserDevicePermission).
  - Nhấn Add Permission, chọn user, tick *Can View* (xem Dashboard) và/hoặc *Can Control* (gửi lệnh OPEN/CLOSE/AUTO), sau đó Grant.
  - Form gọi POST /api/permissions, Prisma upsert bản ghi trong bảng UserDevicePermission (unique theo userId_deviceId). Giao diện reload ngay để phản ánh quyền mới.
- **Người dùng đăng nhập & sử dụng:** tài khoản Viewer sau khi được gán quyền sẽ nhìn thấy đúng thiết bị của họ trong Dashboard/Logs. Nếu chưa có quyền, server trả 403 và ẩn toàn bộ điều khiển, đảm bảo "đúng người – đúng thiết bị".

2.6.3. Luồng vận hành sau khi gán quyền

- **GET /api/devices:** trả về danh sách thiết bị mà user có canView. Dashboard dựa vào API này để lọc dữ liệu hiển thị.
- **GET /api/status/latest và GET /api/logs:** trước khi trả dữ liệu, API gọi helper canViewDevice(deviceId) để bảo vệ thông tin.
- **POST /api/control:** chỉ hoạt động với user có canControl hoặc ADMIN. Mọi lệnh được lưu vào bảng ControlCommand kèm requestedByUserId, cho phép audit "ai mở/đóng cửa vào thời điểm nào".
- Khi ADMIN bỏ tick hoặc Revoke, bản ghi UserDevicePermission bị xóa; người dùng mất quyền ngay lập tức mà không cần đăng xuất.

2.6.4. Trải nghiệm triển khai thực tế

1. Lắp đặt thiết bị → cấp nguồn → thiết bị phát WiFi setup.
2. Kỹ thuật viên cấu hình WiFi & IP server → thiết bị kết nối cloud.

3. ADMIN tạo Device + API key tương ứng trong Dashboard.
4. Thiết bị bắt đầu gửi dữ liệu → xuất hiện trong danh sách.
5. ADMIN gán quyền cho tổ vận hành khu vực đó.
6. Nhân viên bảo vệ đăng nhập ứng dụng, thấy ngay thiết bị của mình để giám sát/điều khiển.

⇨ Nhờ quy trình trên, hệ thống đã chuyển từ dạng demo sang một giải pháp có thể bàn giao: khi triển khai thêm một "cửa chống ngập thông minh", chỉ cần cấp mã thiết bị, cấu hình WiFi, rồi phân quyền trên web là thiết bị đã sẵn sàng hoạt động và được kiểm soát đầy đủ

V. Phân tích thiết kế

1. Thiết kế pipeline Machine Learning.

1.1. Data collection & preprocessing

- **Dataset**: Vietnam Weather Data (Kaggle)
  - **Nguồn**: Dữ liệu thời tiết các tỉnh/thành phố Việt Nam.
  - **Thời gian**: 2009-2019.
  - **Số lượng**: 181,960 bản ghi.
  - **Các cột**: province, max (nhiệt độ max), min (nhiệt độ min), rain (lượng mưa), humidi (độ ẩm), date.
- **Preprocessing** (từ notebook ML/iot-smart-flood-barrier.ipynb):

```
# Rename columns
df = df.rename(columns={
    'province': 'Province',
    'max': 'MaxTemp',
    'min': 'MinTemp',
    'rain': 'Rainfall',
    'humidi': 'Humidity',
    'date': 'Date'
})

# Tính nhiệt độ trung bình
df["TempMean"] = (df["MaxTemp"] + df["MinTemp"]) / 2.0

# Tạo label: RainTomorrow (lượng mưa ngày hôm sau)
df["RainTomorrow"] = df.groupby("Province")["Rainfall"].shift(-1)

# Label nhị phân: HeavyRainTomorrow (≥ 20mm)
HEAVY_RAIN_THRESHOLD = 20.0
df["HeavyRainTomorrow"] = (df["RainTomorrow"] >=
HEAVY_RAIN_THRESHOLD).astype(int)
```

- **Phân phối label:**
  - Class 0 (không mưa lớn): 91.46%.
  - Class 1 (mưa lớn): 8.54%.

→ **Class imbalance**, cần dùng class_weight="balanced"

1.2. Feature selection

- **Features được chọn:**
  - Nhiệt độ trung bình ngày.
  - Humidity: Độ ẩm trung bình.
- **Lý do:**
  - **Dễ đo trên ESP32**: DHT22 có thể đo cả 2 giá trị này.
  - **Đủ thông tin**: Nhiệt độ và độ ẩm là 2 yếu tố quan trọng nhất để dự đoán mưa.
  - **Nhẹ**: Chỉ 2 features → mô hình đơn giản, phù hợp ESP32
- **Features không sử dụng** (có trong dataset nhưng không dùng):
  - wind, wind_d: Hướng và tốc độ gió (không có cảm biến).
  - cloud: Mây (không có cảm biến).
  - pressure: Áp suất khí quyển (không có cảm biến).

1.3. Model training & evaluation

- **Chia dữ liệu** (time-based split):

```
# Sắp xếp theo thời gian
df = df.sort_values("Date").reset_index(drop=True)

n = len(df)
test_size = int(n * 0.2)

trainval_df = df.iloc[:-test_size]  # 145,536 mẫu
test_df     = df.iloc[-test_size:]  # 36,384 mẫu

# Chia train/val từ trainval
X_train, X_val, y_train, y_val = train_test_split(
    X_trainval, y_trainval,
    test_size=0.2,
    random_state=42,
    stratify=y_trainval
)
```

- **Huấn luyện:**

```
log_reg = LogisticRegression(
    max_iter=1000,
    class_weight="balanced",  # Xử lý class imbalance
    n_jobs=-1
)
log_reg.fit(X_train, y_train)
```

- **Kết quả đánh giá:**

Metric	Train	Validation	Test
Accuracy	64.8%	65.1%	69.4%
AUC-ROC	0.758	0.763	0.771

- **Classification Report (Test):**

	precision	recall	f1-score	Support
0	0.97	0.69	0.81	33894
1	0.15	0.71	0.24	2490

- **Nhận xét:**
  - Model có **recall cao** (71%) cho class 1 → Phát hiện tốt mưa lớn (ít bỏ sót).
  - **Precision thấp** (15%) → Có nhiều false positive (cảnh báo nhầm).
  - **Phù hợp với mục đích**: Trong bài toán chống ngập, **ưu tiên recall** (không bỏ sót) hơn precision (cảnh báo nhầm không sao).

1.4. Model export & deployment

- **Trích xuất tham số:**

```
coef = log_reg.coef_.reshape(-1)
intercept = float(log_reg.intercept_[0])

# Kết quả:
# Intercept (bias): -18.49602386139671
# Weight TempMean: 0.235201
# Weight Humidity: 0.151022
```

- **Nhúng vào ESP32:**
  - Hardcode các tham số vào firmware.
  - Implement hàm predictHeavyRainNext24h() bằng C++.
  - Sử dụng expf() từ math.h cho sigmoid.

2. Thiết kế firmware (ESP32)

2.1. Kiến trúc state machine

- Hệ thống quản lý 2 trạng thái chính:

AUTO MODE	← Mặc định, dựa trên cảm biến + ML

(Công tắc được kích hoạt)

MANUAL MODE	← Điều khiển bằng công tắc hoặc Web

- **Biến trạng thái:**

```
bool manualOverride = false;  // false = AUTO, true = MANUAL
int targetAngle;              // Góc mục tiêu của servo (0-90°)
int currentAngle;             // Góc hiện tại (được cập nhật dần)
```

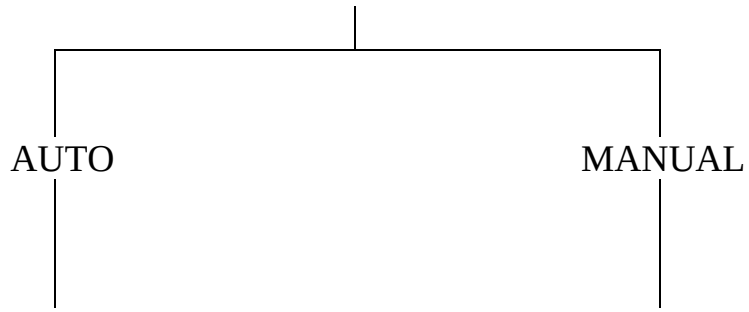
2.2. Servo control với smoothing

- **Vấn đề**: Nếu thay đổi góc servo đột ngột (ví dụ: 90° → 0°), servo sẽ giật mạnh, gây hư hỏng cơ khí.
- **Giải pháp**: Cập nhật góc dần dần (1°/loop):

```
void updateServo() {
  if (currentAngle < targetAngle) currentAngle++;
  else if (currentAngle > targetAngle) currentAngle--;
  writeServoAngle(currentAngle);
}
```

- **Kết quả**: Servo quay mượt mà, không giật.

2.3. Main loop flow

1. Đọc cảm biến	→ DHT22 (temp, humidity), HC-SR04 (distance)
↓	
2. Tính ML prediction	→ predictHeavyRainNext24h
↓	
3. Điều chỉnh ngưỡng	→ levelHigh, levelLow (dựa trên ML)
↓	
4. Kiểm tra công tắc	→ manualOverride?
	
5. Xác định targetAngle	
↓	
6. Cập nhật servo	→ updateServo() (smoothing)
↓	
7. Điều khiển LED	→ Bật nếu pHeavyRain >= 0.6
↓	
8. Log debug	→ Serial output
↓	

delay(100ms) → Lặp lại

2.4. Xử lý lỗi và backup plan

- **Phát hiện lỗi cảm biến siêu âm:**

```
if (distance > max_distance) {  // max_distance = 404cm
  // Cảm biến lỗi → Dùng Backup Plan
  backup_plan(historyHum, humidity, historyTemp, temperature);
}
```

- **Backup plan logic:**
  - Lưu giá trị nhiệt độ/độ ẩm mỗi 5 phút.
  - Tính delta (thay đổi) so với giá trị trước.
  - Nếu độ ẩm tăng nhanh + nhiệt độ giảm nhanh → Mưa sắp đến → Mở cửa.

3. Thiết kế Web Application

3.1. Kiến trúc Next.js App Router

```
App/
├── app/
│   ├── api/                # API Routes
│   │   ├── auth/           # NextAuth
│   │   ├── status/         # POST /api/status, GET /api/status/latest
│   │   ├── logs/           # GET /api/logs
│   │   └── control/        # POST /api/control, GET /api/control/latest
│   ├── login/              # Trang đăng nhập
│   ├── logs/               # Trang lịch sử
│   ├── settings/           # Trang cài đặt
│   ├── page.jsx            # Dashboard chính
│   └── layout.jsx          # Root layout
├── components/
│   ├── ui/                 # shadcn/ui components
│   ├── dashboard/          # Dashboard components
│   └── layout/             # Layout components
└── lib/
    ├── auth.js             # User authentication
    ├── authDevice.js       # Device authentication
    ├── prisma.js           # Prisma client
    ├── telegram.js         # Telegram bot
    └── utils.js            # Utilities
```

3.2. Database Schema (Prisma)

```
model User {
  id          String   @id @default(cuid())
  name        String
  email       String   @unique
  passwordHash String
  role        String   @default("VIEWER")  // "ADMIN" | "VIEWER"
```

```
  createdAt    DateTime @default(now())
  updatedAt    DateTime @updatedAt
}

model Device {
  id          String   @id @default("esp32-001")
  name        String
  apiKey      String   @unique  // API key cho ESP32
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt
}

model DeviceStatus {
  id              String   @id @default(cuid())
  deviceId        String
  waterLevelCm    Float
  doorState       String   // "OPEN" | "MID" | "CLOSE"
  mode            String   // "AUTO" | "MANUAL"
  pHeavyRainNext24h Float    // Xác suất từ ML (0-1)
  temperatureC    Float
  humidityPct     Float
  timestamp       DateTime
  createdAt       DateTime @default(now())

  device Device @relation(fields: [deviceId], references: [id])

  @@index([deviceId, timestamp])
}

model ControlCommand {
  id            String   @id @default(cuid())
  deviceId      String
  command       String   // "OPEN" | "CLOSE" | "AUTO"
  requestedByUserId String?
  createdAt     DateTime @default(now())

  device Device? @relation(fields: [deviceId], references: [id])
  requestedBy   User?   @relation(fields: [requestedByUserId], references: [id])
}

model AlertLog {
  id       String   @id @default(cuid())
  deviceId String
```

```
  type      String   @default("TELEGRAM")
  message   String
  severity  String   @default("INFO")  // "INFO" | "WARN" | "CRITICAL"
  createdAt DateTime @default(now())

  device Device @relation(fields: [deviceId], references: [id])
}
```

3.3. API Endpoints

- **Cho ESP32** (yêu cầu Device API Key):
  - POST /api/status: ESP32 gửi status updates.
  - GET /api/control/latest?deviceId=esp32-001: ESP32 polling lệnh điều khiển.
- **Cho Web Dashboard** (yêu cầu NextAuth session):
  - GET /api/status/latest?deviceId=esp32-001: Lấy trạng thái mới nhất.
  - GET /api/logs?deviceId=esp32-001&limit=100: Lấy lịch sử.
  - POST /api/control: Gửi lệnh điều khiển (chỉ ADMIN).

3.4. Real-time updates

- **Cơ chế**: Polling (không dùng WebSocket để đơn giản hóa):
  - Dashboard polling /api/status/latest mỗi 5 giây.
  - ESP32 polling /api/control/latest mỗi 5-10 giây.
  - Sử dụng useEffect và setInterval trong React.
- **Tối ưu**:
  - Chỉ fetch khi component mounted.
  - Cleanup interval khi unmount.
  - Hiển thị loading state khi đang fetch.

VI. Kết quả

1. Về phần cứng

1.1. Mô hình Prototype

- **Hoàn thiện mô hình cửa chống ngập mini:**
  - Khung cơ khí với cửa chắn có thể nâng/hạ.
  - Bể nước mô phỏng để test.
  - Đầy đủ cảm biến: HC-SR04, DHT22.
  - Servo motor (SG90) điều khiển cửa.
  - LED cảnh báo và công tắc gạt.
- **Mạch hoạt động ổn định:**
  - ESP32 kết nối WiFi thành công.
  - Cảm biến đọc giá trị chính xác.
  - Servo quay mượt mà (nhờ smoothing algorithm).
  - Gửi dữ liệu về server đều đặn.

1.2. Mô phỏng trên Wokwi

- **File src/diagram.json**: Cấu hình mô phỏng đầy đủ.

- **Components:**
  - ESP32 DevKit v1.
  - DHT22 (GPIO 4).
  - HC-SR04 (TRIG: GPIO 5, ECHO: GPIO 18).
  - Servo (GPIO 25).
  - LED (GPIO 2) với điện trở 220Ω.
  - 2 Slide Switch (GPIO 26, 27).
- **Kết quả**: Mô phỏng chạy ổn định, logic hoạt động đúng như thiết kế.

2. Về Machine Learning

2.1. Model Training

- **Dataset**: 181,960 bản ghi thời tiết Việt Nam.
- **Features**: TempMean, Humidity.
- **Model**: Logistic Regression với class_weight="balanced".
- **Performance:**
  - **Test Accuracy**: 69.4%.
  - **Test AUC-ROC**: 0.771.
  - **Recall (Class 1)**: 71% → Phát hiện tốt mưa lớn.

2.2. Model Deployment trên ESP32

- **Nhúng thành công**: Implement hàm predictHeavyRainNext24h() bằng C++.
- **Tính toán nhanh**: Inference chỉ mất vài microsecond.
- **Độ chính xác**: Kết quả khớp với Python (sai số < 0.001 do floating point).
- **Code thực tế:**

```
float predictHeavyRainNext24h(float temp_c, float humidity_pct) {
  const float BIAS   = -18.49602386139671f;
  const float W_TEMP = 0.235201f;
  const float W_HUM  = 0.151022f;

  float z = BIAS + W_TEMP * temp_c + W_HUM * humidity_pct;
  float p = 1.0f / (1.0f + expf(-z));
  return p;
}
```

2.3. Tích hợp vào Logic Điều khiển

- **Điều chỉnh ngưỡng động**: Hệ thống tự động thay đổi levelHigh và levelLow dựa trên pHeavyRainNext24h.
- **Cảnh báo sớm**: LED bật khi xác suất ≥ 60%.
- **Phòng ngừa**: Mở cửa sớm hơn khi xác suất mưa cao.

3. Về ứng dụng Web

3.1. Web dashboard

- **Framework**: Next.js 14 (App Router).
- **UI**: shadcn/ui + Tailwind CSS.
- **Charts**: Recharts.
- **Tính năng đã triển khai:**
  - Dashboard realtime với 4 summary cards.
  - 2 biểu đồ line chart (mực nước, xác suất mưa).
  - Bảng điều khiển (OPEN/CLOSE/AUTO) cho ADMIN.
  - Trang logs hiển thị lịch sử đầy đủ.
  - Hệ thống xác thực (NextAuth) với phân quyền ADMIN/VIEWER.
  - Phát hiện thiết bị offline.
  - Responsive design.

3.2. API Backend

- **RESTful API** đầy đủ:
  - POST /api/status: Nhận dữ liệu từ ESP32.
  - GET /api/status/latest: Lấy trạng thái mới nhất.
  - GET /api/logs: Lấy lịch sử.
  - POST /api/control: Gửi lệnh điều khiển.
  - GET /api/control/latest: ESP32 polling lệnh.
- **Xác thực 2 lớp:**
  - User authentication (NextAuth).
  - Device authentication (API Key).
- **Database**: SQLite với Prisma ORM.

3.3. Hệ thống cảnh báo Telegram

- **Tích hợp thành công**: Gửi cảnh báo qua Telegram Bot
- **Điều kiện kích hoạt:**
  - pHeavyRainNext24h >= 0.8 (80%).
  - waterLevelCm <= 20 (mực nước nguy hiểm).
- **Nội dung**: Đầy đủ thông tin (device ID, mực nước, xác suất mưa, trạng thái, nhiệt độ, độ ẩm, timestamp).

4. Kịch bản kiểm thử

4.1. Test case 1: Điều chỉnh ngưỡng động

- **Mô tả**: Thay đổi thông số nhiệt độ/độ ẩm giả lập để kiểm tra logic điều chỉnh ngưỡng.
- **Kịch bản:**
  1. Nhiệt độ: 28°C, Độ ẩm: 85% → pHeavyRainNext24h = 0.65 (65%).
  2. Hệ thống điều chỉnh: levelHigh = 25cm (tăng 5cm), levelLow = 50cm (tăng 10cm).
  3. Khi mực nước = 25cm → Cửa mở (thay vì đợi đến 20cm).
- **Kết quả**: Hoạt động đúng logic thiết kế.

4.2. Test case 2: Backup plan

- **Mô tả**: Giả lập cảm biến siêu âm lỗi (distance > 404cm).
- **Kịch bản:**
  1. Cảm biến siêu âm trả về 9999cm (lỗi).
  2. Hệ thống chuyển sang Backup Plan.
  3. Độ ẩm tăng từ 80% → 95% trong 5 phút.
  4. Nhiệt độ giảm từ 30°C → 27°C.
  5. Hệ thống phát hiện điều kiện bất thường → Mở cửa.
- **Kết quả**: Backup Plan hoạt động, cửa mở khi phát hiện nguy cơ.

4.3. Test case 3: Manual Override

- **Mô tả**: Kiểm tra điều khiển thủ công bằng công tắc.
- **Kịch bản:**
  1. Hệ thống đang ở AUTO mode, cửa đóng (mực nước = 50cm).
  2. Gạt công tắc OPEN → manualOverride = true.
  3. Cửa mở ngay lập tức (không phụ thuộc mực nước).
  4. Gạt công tắc về OFF → Trở về AUTO mode.
- **Kết quả**: Manual override hoạt động đúng.

4.4. Test Case 4: Web Dashboard

- **Mô tả**: Kiểm tra tính năng Dashboard.
- **Kịch bản:**
  1. Đăng nhập với tài khoản ADMIN.
  2. Xem Dashboard → Hiển thị đầy đủ thông tin.
  3. Gửi lệnh OPEN từ Web → ESP32 nhận và thực thi.
  4. Xem trang Logs → Hiển thị lịch sử đầy đủ.
- **Kết quả**: Dashboard hoạt động tốt, realtime updates chính xác.

5. Đánh giá tổng thể

- **Điểm mạnh:**
  - **Tích hợp ML thành công**: Mô hình chạy trực tiếp trên ESP32, không cần server.
  - **Logic thông minh**: Điều chỉnh ngưỡng động dựa trên dự báo, không chỉ phản ứng.
  - **Backup Plan**: Xử lý được trường hợp cảm biến lỗi.
  - **Web Dashboard hoàn chỉnh**: Đầy đủ tính năng giám sát và điều khiển.
  - **Bảo mật**: Xác thực 2 lớp (user + device).
  - **Chi phí thấp**: Sử dụng linh kiện phổ biến, giá rẻ.
- **Hạn chế:**
  - **Dataset**: Dữ liệu thời tiết ở cấp tỉnh/thành phố, chưa phải hyper-local.
  - **Model đơn giản**: Chỉ 2 features, có thể cải thiện với thêm features.
  - **Precision thấp**: Nhiều false positive (cảnh báo nhầm).
  - **Web App MVP**: Chưa có một số tính năng nâng cao (export data, filter logs, v.v.).

VII. Kết luận

1. Kết luận

- Dự án **Hệ thống cửa chống ngập thông minh** đã đạt được các mục tiêu đề ra:
  - Hệ thống tự động đo mực nước và điều khiển cửa mà không cần can thiệp con người.
  - Mô hình Logistic Regression được nhúng thành công vào ESP32, dự đoán xác suất mưa lớn trong 24h tới.
  - Web Dashboard Next.js cho phép theo dõi và điều khiển hệ thống realtime.
  - **Chi phí thấp**: Sử dụng các linh kiện phổ biến, tổng chi phí < 500,000 VNĐ.
- **Điểm sáng tạo chính:**
  - **Điều chỉnh ngưỡng động**: Hệ thống không chỉ "phản ứng" với nước, mà còn "phòng ngừa" dựa trên dự báo ML.
  - **Backup Plan**: Xử lý được trường hợp cảm biến lỗi bằng cách phân tích nhiệt độ/độ ẩm.
  - **Edge AI**: ML model chạy trực tiếp trên ESP32, không cần kết nối server liên tục.
- **Ứng dụng thực tế:**
  - Hầm gửi xe chung cư.
  - Nhà phố thấp trũng.
  - Khu vực thường xuyên ngập lụt.

2. Hạn chế và thách thức

2.1. Về Dữ liệu

- **Dataset**: Dữ liệu thời tiết ở cấp tỉnh/thành phố, không phải hyper-local tại điểm đặt thiết bị.
- **Thời gian**: Dữ liệu từ 2009-2019, có thể không phản ánh điều kiện hiện tại.
- **Độ chính xác**: Model có precision thấp (15%), nhiều false positive.

2.2. Về Mô hình ML

- **Đơn giản**: Chỉ 2 features (TempMean, Humidity), có thể cải thiện với thêm features.
- **Thuật toán**: Logistic Regression là mô hình tuyến tính, không nắm bắt được các tương tác phức tạp.
- **Performance**: AUC-ROC = 0.771 là tốt nhưng chưa xuất sắc.

2.3. Về Hệ thống

- **Web App**: Mới ở mức MVP, thiếu một số tính năng nâng cao.
- **Bảo mật**: Chưa có HTTPS trong development, cần cải thiện cho production.
- **Scalability**: SQLite không phù hợp cho nhiều thiết bị đồng thời.

3. Hướng mở rộng (Future Work)

3.1. Cải thiện Machine Learning

3.1.1. Thu thập dữ liệu thực tế

- Lắp đặt thiết bị tại điểm thực tế.
- Thu thập dữ liệu nhiệt độ/độ ẩm và lượng mưa thực tế.
- Retrain model với dữ liệu mới (transfer learning.

3.1.2. Bổ sung features

- **Cảm biến mưa (Rain Sensor)**: Phát hiện mưa trực tiếp.
- **Cảm biến áp suất khí quyển (Barometric Pressure Sensor)**: Dự đoán thời tiết chính xác hơn.
- **Cảm biến gió**: Tốc độ và hướng gió.
- **Lịch sử mực nước**: Sử dụng dữ liệu quá khứ để dự đoán.

3.1.3. Nâng cấp mô hình

- **XGBoost**: Mô hình ensemble mạnh hơn, có thể chạy trên server.
- **Deep Learning**: LSTM/GRU để dự đoán chuỗi thời gian.
- **Hybrid Model**: Kết hợp nhiều mô hình (ensemble).

3.1.4. Deployment strategy

- **Cloud Inference**: Chạy mô hình phức tạp trên server, ESP32 chỉ gửi dữ liệu.
- **Federated Learning**: Huấn luyện mô hình từ nhiều thiết bị.

3.2. Cải thiện Phần cứng

3.2.1. Cảm biến bổ sung

- Rain Sensor (GPIO).
- Barometric Pressure Sensor (BMP280/BME280) qua I2C.
- Cảm biến mực nước dự phòng (Water Level Sensor).

3.2.2. Tăng độ tin cậy

- **Battery Backup**: Pin dự phòng khi mất điện.
- **Solar Panel**: Sạc pin bằng năng lượng mặt trời.
- **LoRa/4G Module**: Kết nối dự phòng khi WiFi mất.

3.2.3. Cơ cấu chấp hành

- **Stepper Motor**: Thay servo để điều khiển chính xác hơn.
- **Linear Actuator**: Điều khiển cửa lớn hơn.

3.3. Cải thiện Web Application

3.3.1. Tính năng mới

- **Export Data**: Xuất dữ liệu ra CSV/Excel.
- **Filter & Search**: Lọc logs theo thời gian, điều kiện.
- **Notifications**: Push notifications thay vì chỉ Telegram.
- **Multi-device Support**: Quản lý nhiều thiết bị.
- **User Management**: Admin có thể thêm/xóa user.

3.3.2. Tối ưu Performance

- **WebSocket**: Thay polling bằng WebSocket cho realtime tốt hơn.
- **Caching**: Cache dữ liệu để giảm load database.
- **Pagination**: Phân trang cho logs lớn.

3.3.3. Bảo mật

- **HTTPS**: Sử dụng SSL/TLS trong production.
- **Rate Limiting**: Giới hạn số request để chống DdoS.
- **API Versioning**: Quản lý version API.

3.4. Triển khai Thực tế

3.4.1. Pilot Project

- Lắp đặt tại 1-2 hầm gửi xe thực tế.
- Thu thập dữ liệu trong 3-6 tháng.
- Đánh giá hiệu quả và cải thiện.

3.4.2. Production Deployment

- **Cloud Hosting**: Deploy Web App lên VPS/Cloud (AWS, Azure, GCP).
- **Database**: Chuyển sang PostgreSQL/MySQL.
- **Monitoring**: Sử dụng tools như Grafana, Prometheus.
- **Logging**: Centralized logging (ELK Stack).

3.4.3. Maintenance

- **OTA Updates**: Cập nhật firmware qua không dây.
- **Remote Diagnostics**: Chẩn đoán lỗi từ xa.
- **Automated Alerts**: Cảnh báo khi thiết bị offline.

4. Kết luận cuối cùng

- Dự án **Hệ thống cửa chống ngập thông minh** đã chứng minh tính khả thi của việc tích hợp IoT và Machine Learning vào một giải pháp thực tế. Với chi phí thấp, hệ thống có thể tự động hóa việc chống ngập và cảnh báo sớm, giúp giảm thiểu thiệt hại về tài sản.
- **Đóng góp chính:**
  - Tích hợp ML vào edge device (ESP32).
  - Logic điều chỉnh ngưỡng động dựa trên dự báo.
  - Backup plan khi cảm biến lỗi.
  - Web Dashboard hoàn chỉnh với realtime updates.
- **Hướng phát triển**: Với việc cải thiện dữ liệu, mô hình ML, và triển khai thực tế, hệ thống có tiềm năng trở thành một giải pháp thương mại khả thi cho thị trường Việt Nam.