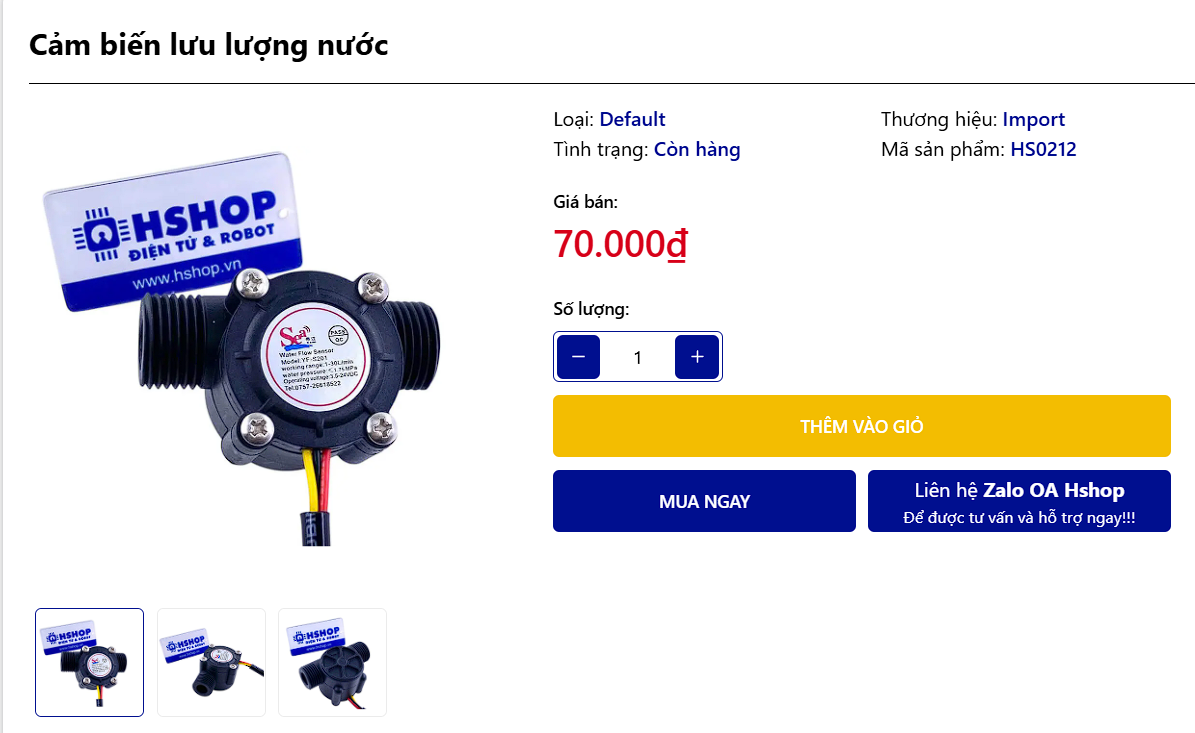
**BÁO CÁO GIỮA KÌ**

1. **Các linh kiện và cảm biến sử dụng trong dự án:**
2. **ESP32 bộ xử lý trung tâm**: Thu thập dữ liệu từ các cảm biến, giao tiếp với hệ thống máy chủ bằng Mqtt thông qua WiFi và nhận lệnh điều khiển theo chiều ngược lại.

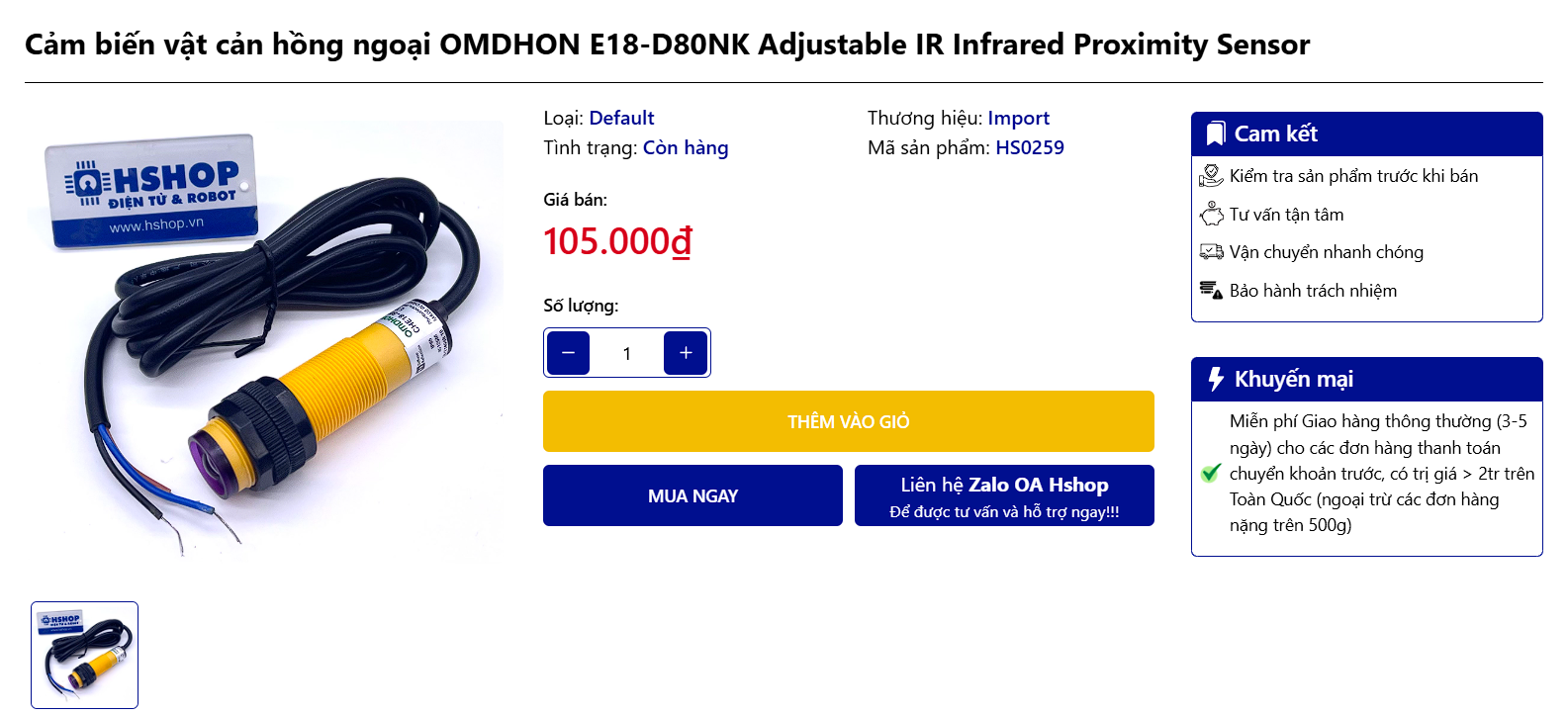


1. **Cảm biến lưu lượng nước YF-S401 và YF-s402:** Đo lưu lượng nước chảy qua cảm biến để tính được lượng nước tiêu thụ





1. **Cảm biến tiệm cận E18-D80NK:** Dùng để phát hiện vật thể hoặc có người tiếp cận để bật máy bơm.



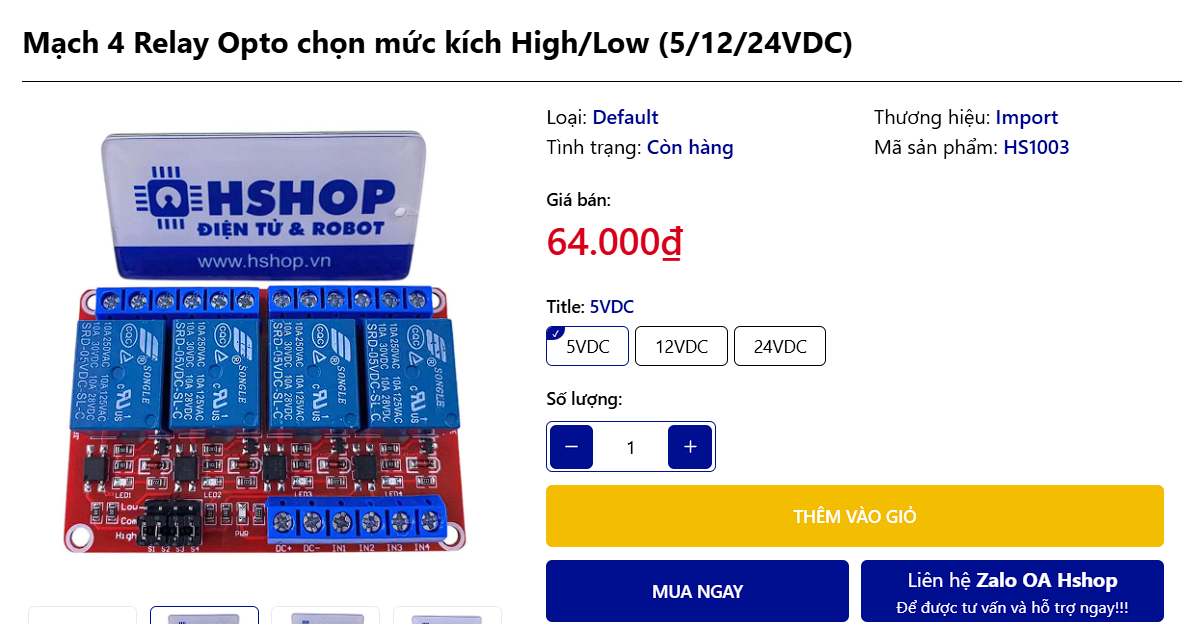
1. **Cảm biến dòng điện ACS712:** Đo dòng điện tiêu thụ của máy bơm nước



1. **Màn hình LCD TFT LIL9431:** Hiển thị các thông số cần thiết như lượng nước tiêu thụ, dòng điện tiêu thụ và dự báo ngày thay lọc trên màn hình



1. **Module Relay 5V:** Dùng để bật tắt máy bơm theo lệnh điều khiển từ hệ thống máy chủ hoặc theo hiện diện của người dùng



1. **Mạch hạ áp 12V-5V:** Giảm điện áp đầu vào 12V thành 5V để cấp cho hệ thống



1. **Máy bơm DC 12V:** Bơm nước theo lệnh điều khiển hoặc hiện diện của người dùng



1. **Các phần mềm nền tảng sử dụng trong dự án**
2. **Ubuntu (Virtual Box):**

Công nghệ ảo hóa nhẹ giúp đóng gói và triển khai các ứng dụng một cách độc lập.

Dùng để triển khai các thành phần phần mềm như: MQTT broker, Home Assistant, Node-RED, InfluxDB, Grafana một cách tách biệt, dễ quản lý và nâng cấp.

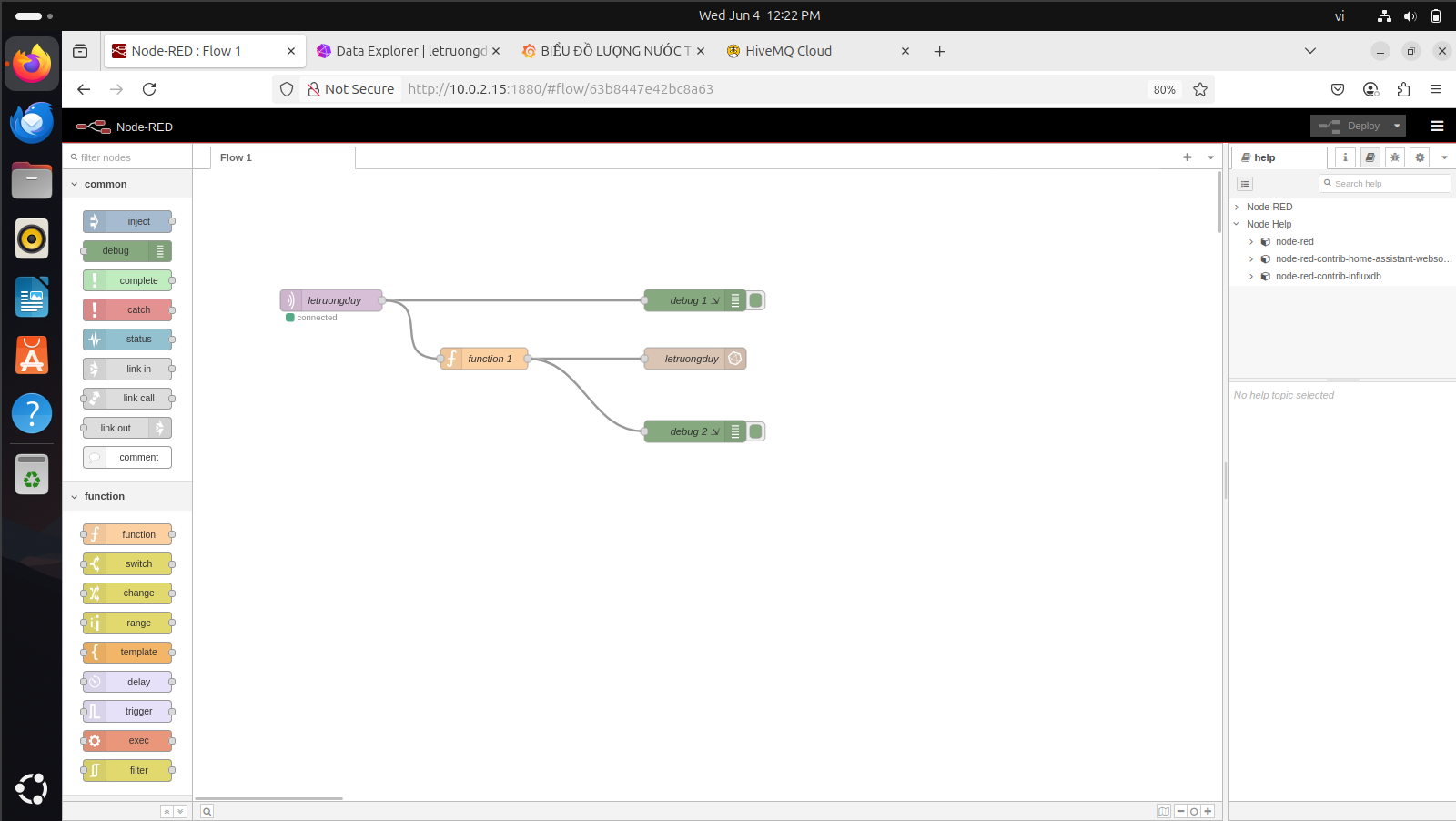
Đảm bảo hệ thống hoạt động ổn định, bảo mật và có thể sao lưu/phục hồi nhanh chóng nếu có sự cố.

Hỗ trợ triển khai linh hoạt: trên máy tính cá nhân, server mini tại nhà hoặc server trên cloud.

1. **Node-Red**

Công cụ lập trình trực quan bằng sơ đồ khối để thiết kế luồng dữ liệu và logic điều khiển.

Kết nối và phối hợp với Home Assistant, InfluxDB và MQTT broker để tạo thành hệ thống hoàn chỉnh.



1. **InfluxDB và Grafana**

InfluxDB: Cơ sở dữ liệu thời gian thực, chuyên dùng lưu dữ liệu cảm biến.

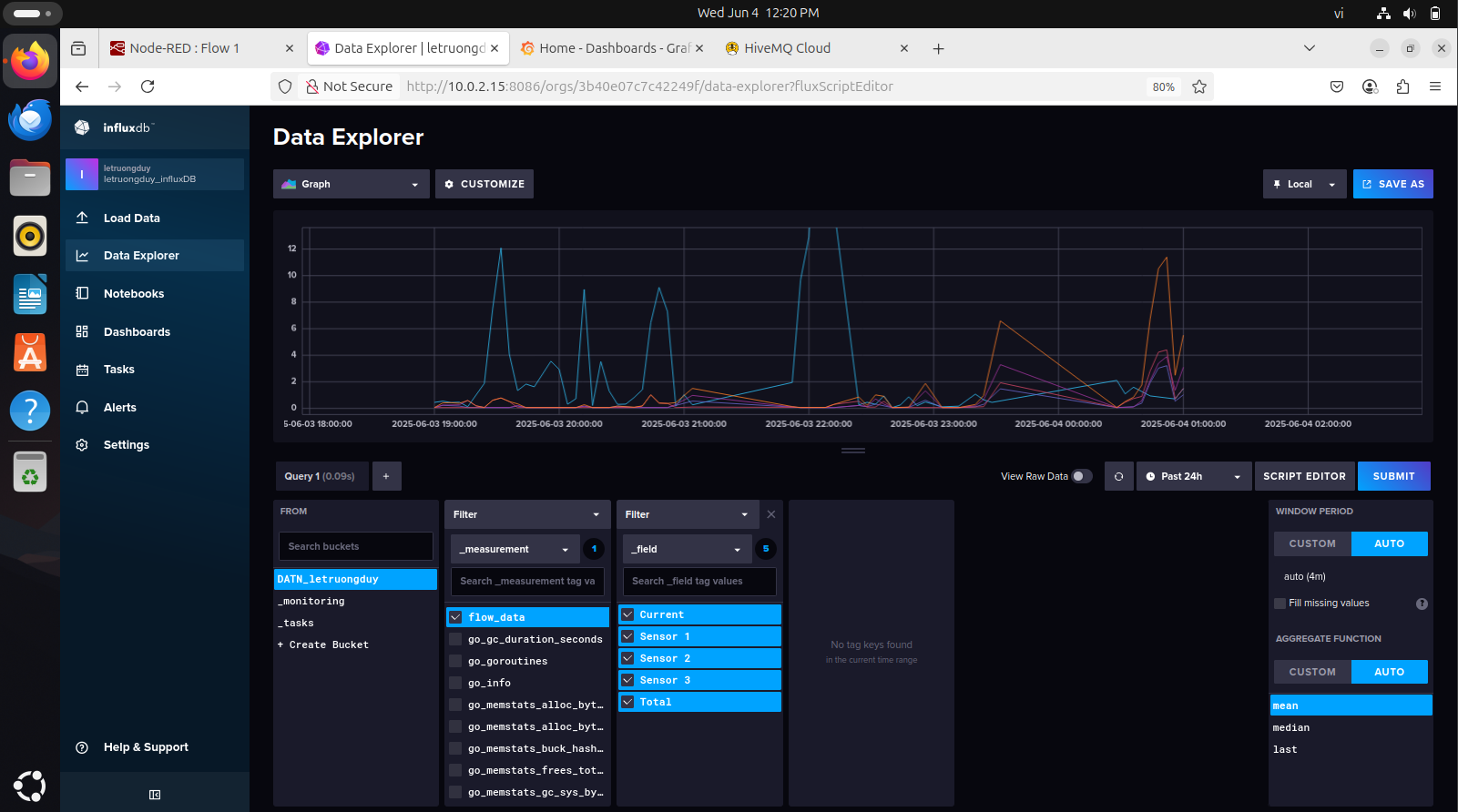
Grafana: Công cụ hiển thị dữ liệu từ InfluxDB dưới dạng biểu đồ.

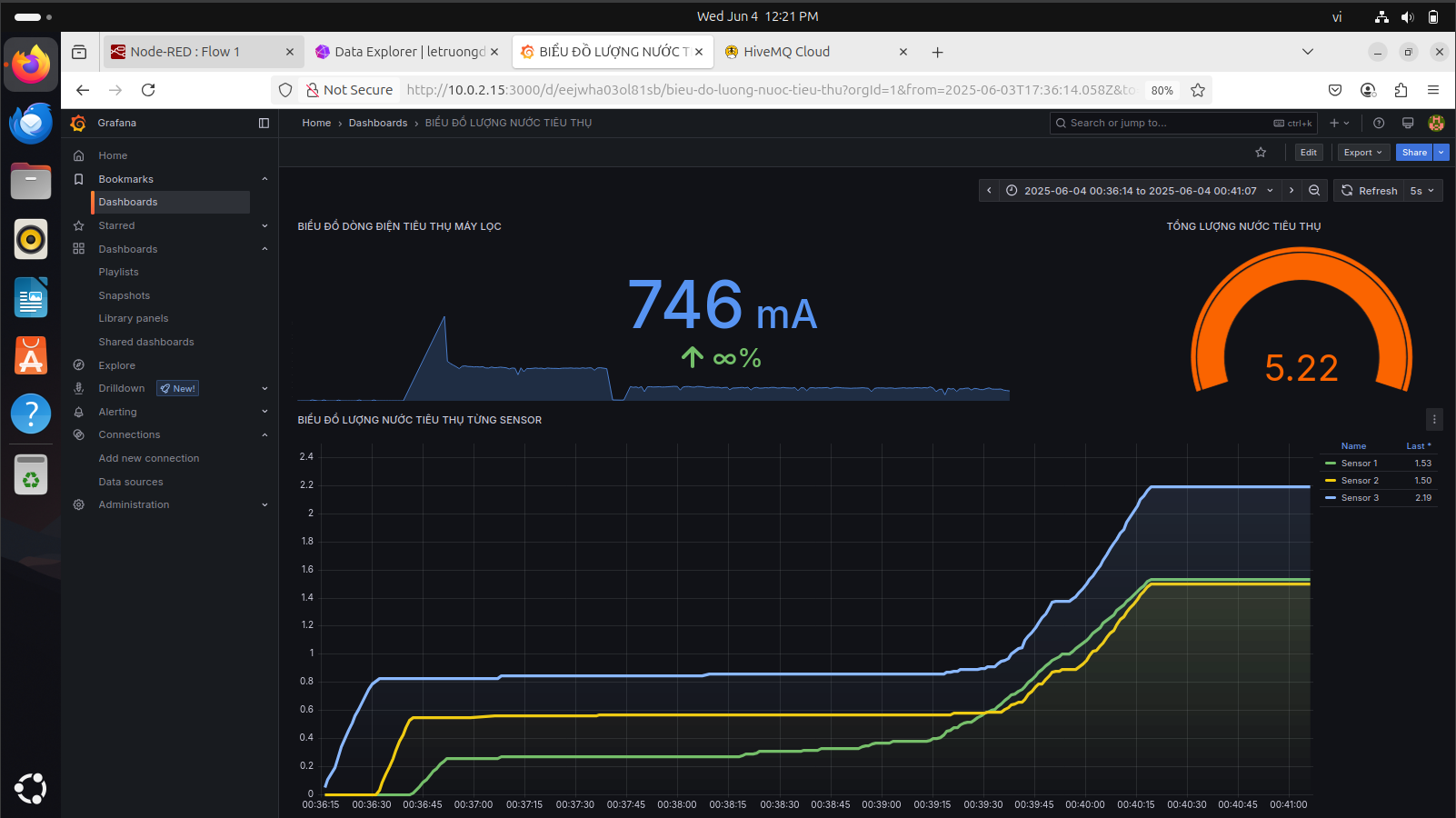
Lưu trữ dữ liệu thời gian thực về lưu lượng nước, điện năng tiêu thụ, trạng thái bơm, van.

Phân tích và đánh giá xu hướng tiêu thụ nước trong ngày/tuần/tháng.

Grafana giúp người dùng dễ dàng hình dung thông tin thông qua các biểu đồ động, trực quan, tích hợp trên Home Assistant hoặc truy cập độc lập.

Hỗ trợ AI học từ dữ liệu lịch sử để đưa ra các đề xuất tối ưu.





1. **Home Assistant (chưa hoàn thành)**

Nền tảng quản lý và tự động hóa các thiết bị IoT trong nhà.

Là trung tâm điều khiển giao diện người dùng, cho phép giám sát hệ thống nước thông minh qua trình duyệt hoặc ứng dụng điện thoại.

Hiển thị các thông tin: lượng nước tiêu thụ, điện năng tiêu thụ, trạng thái máy bơm, cảnh báo rò rỉ nước, lịch thay lõi lọc.

Cho phép người dùng tùy chỉnh ngưỡng cảnh báo, lập lịch hoạt động thiết bị (bơm, van điện từ).

Kết hợp với các nền tảng khác như Node-RED để xử lý logic, gửi thông báo và tự động hóa.

1. **MQTT Broker (HiveMQ)**

Kết nối ESP32 với hệ thống máy chủ Ubuntu (Node-Red, Home Assistant, InfluxDB và Grafana)

ESP32 sử dụng MQTT để gửi dữ liệu cảm biến như:

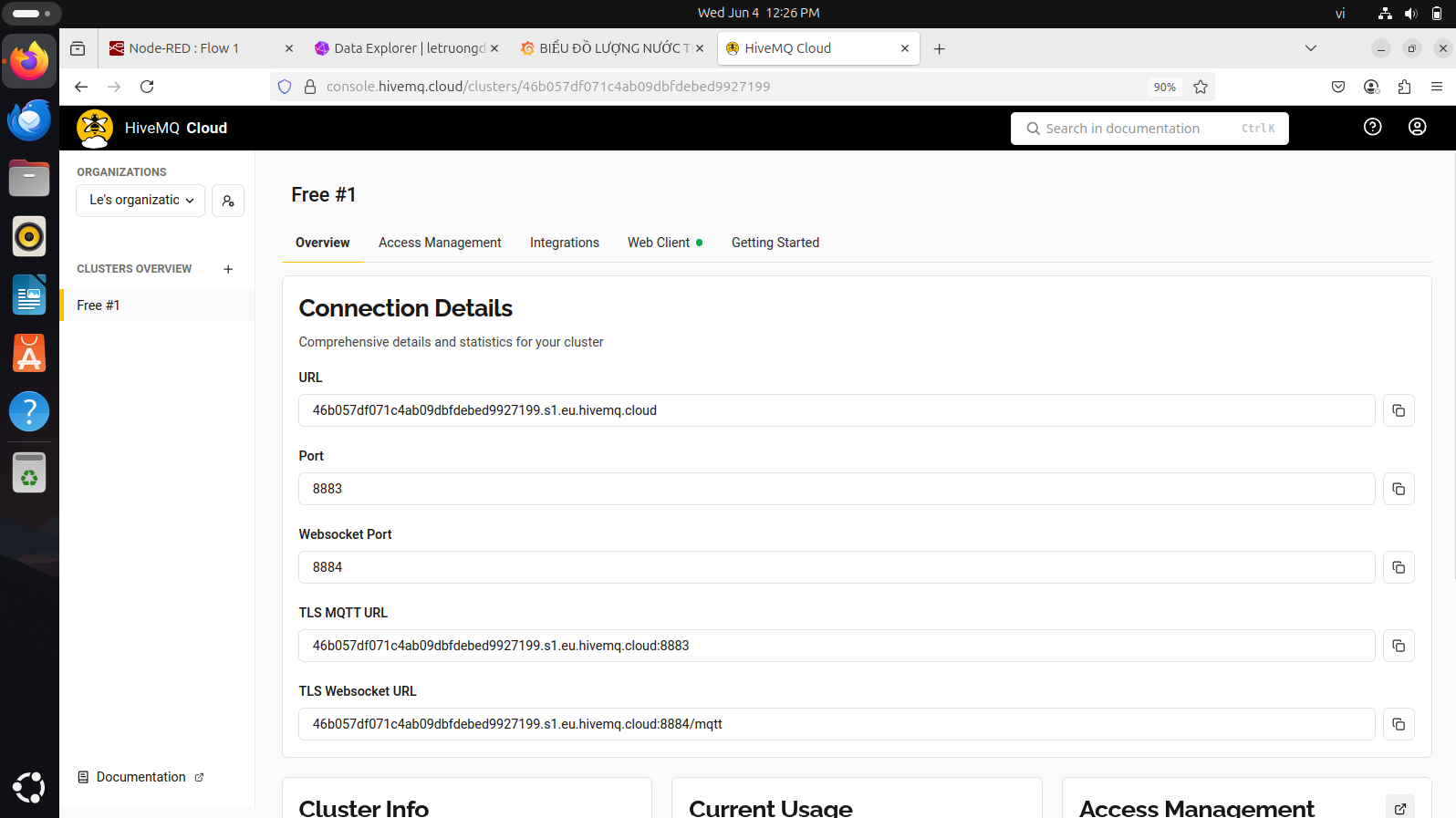
Lưu lượng nước (từ cảm biến nước)

Dòng điện tiêu thụ (từ cảm biến dòng)

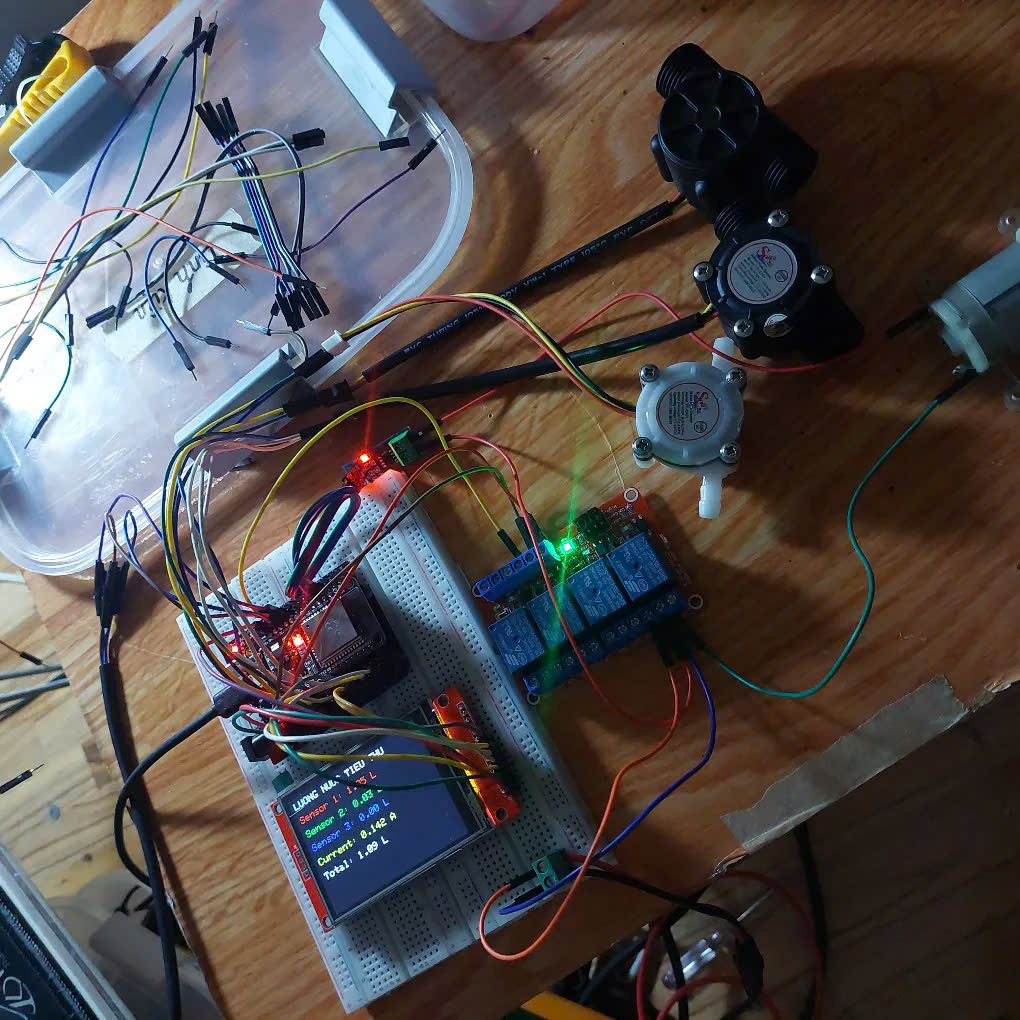
Trạng thái bơm, van điện từ

Đồng thời ESP32 cũng nhận lệnh điều khiển từ Home Assistant hoặc Node-RED thông qua MQTT, ví dụ:

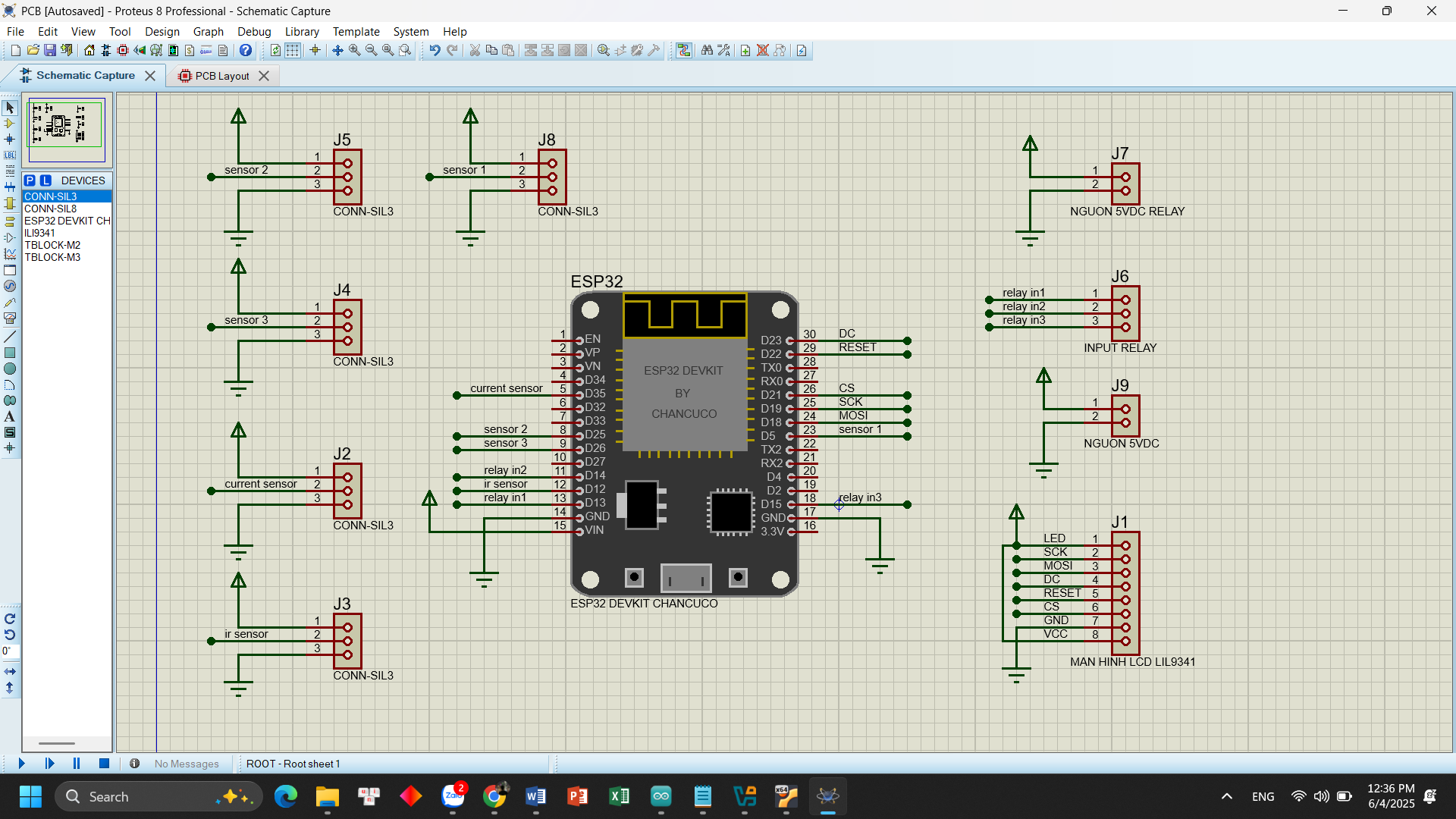
Bật/tắt bơm hay Mở/đóng van điện từ

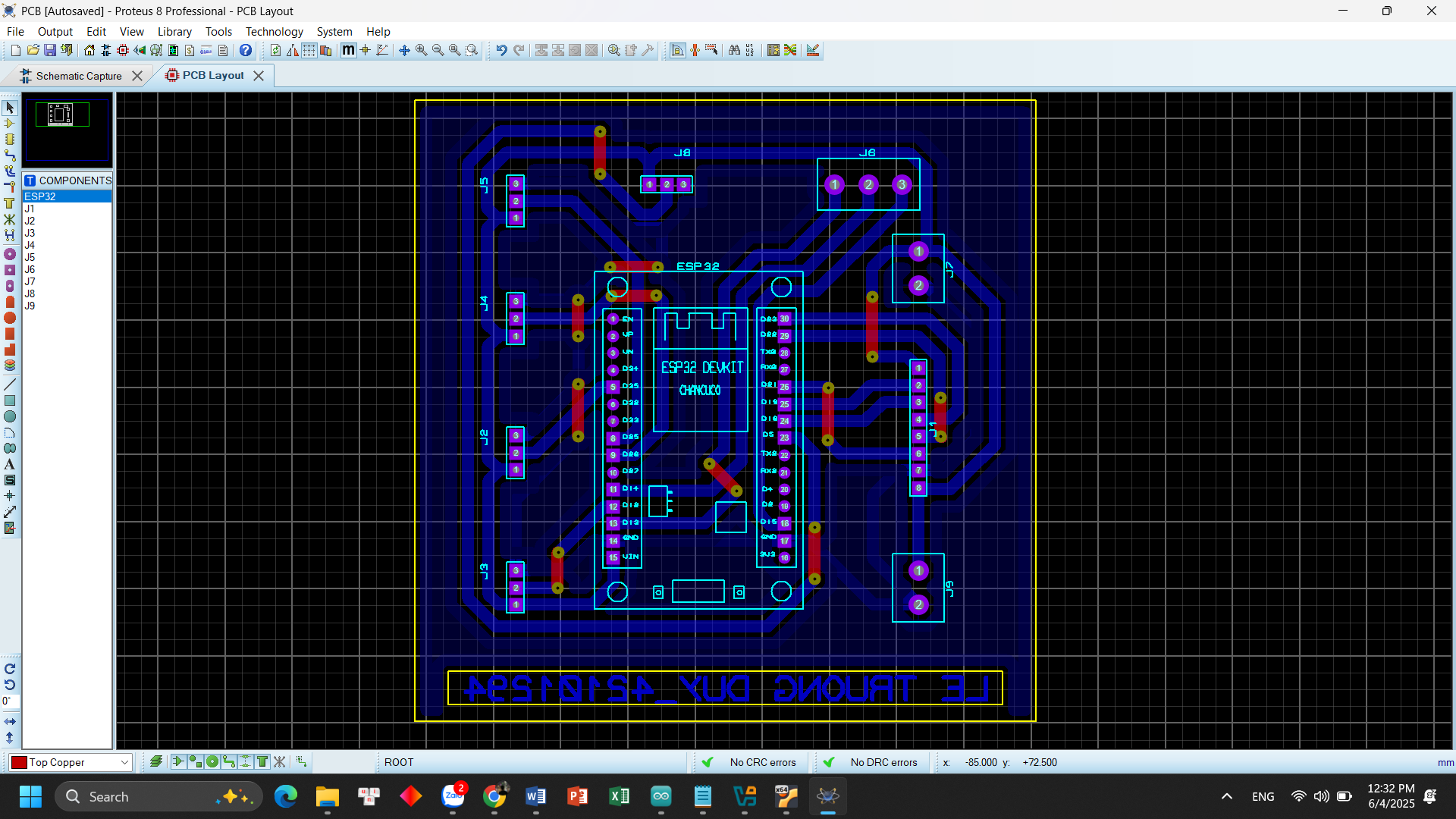


1. **Thi công mạch thực tế**
2. **Thử nghiệm bằng Testboard**

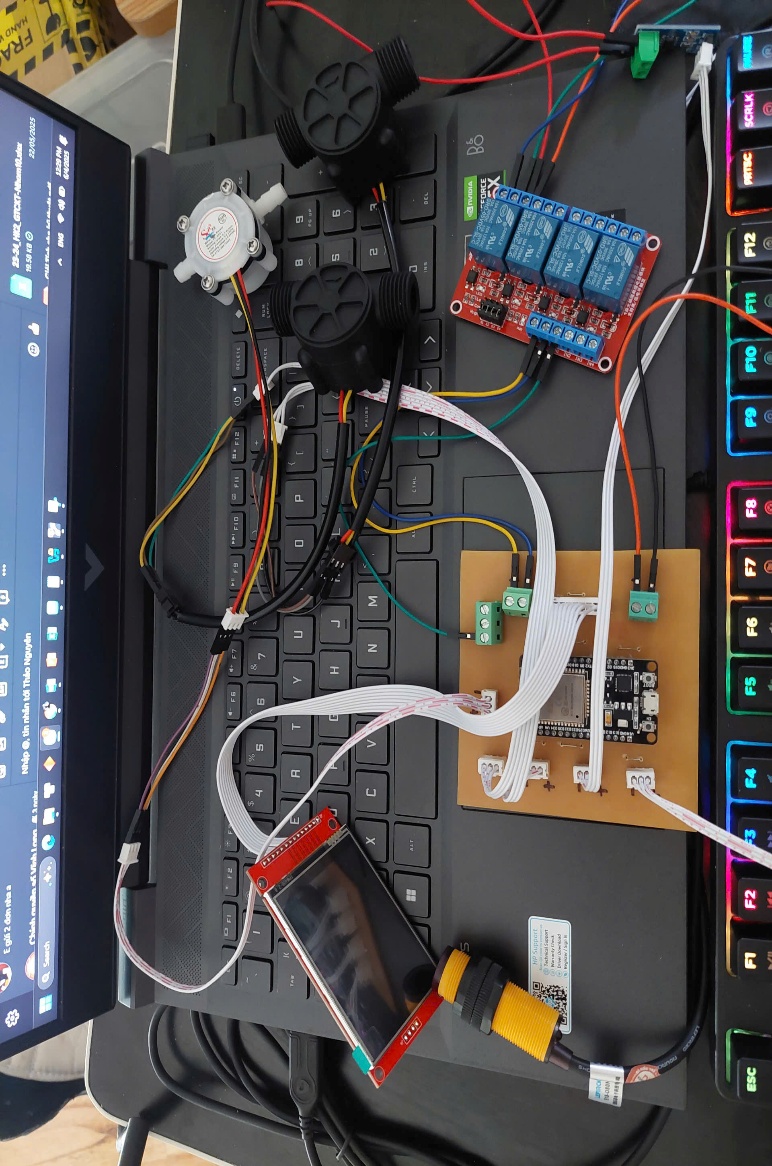


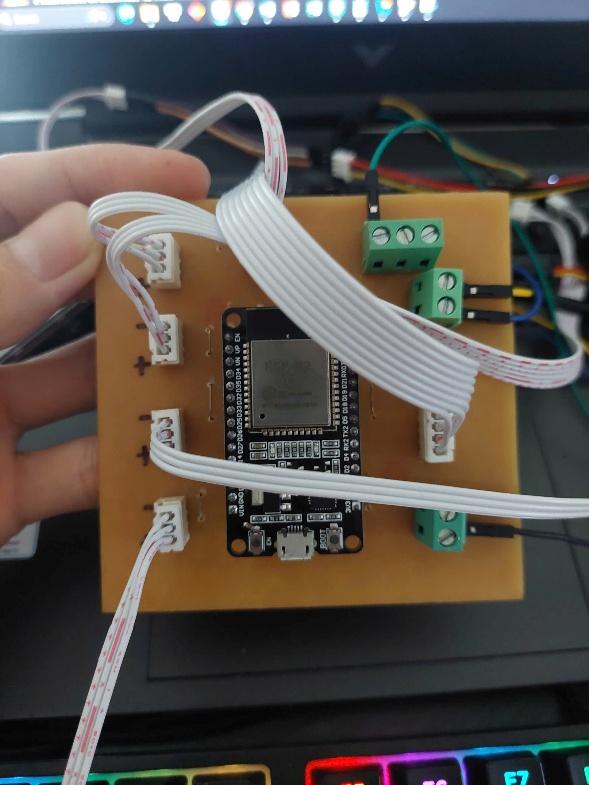
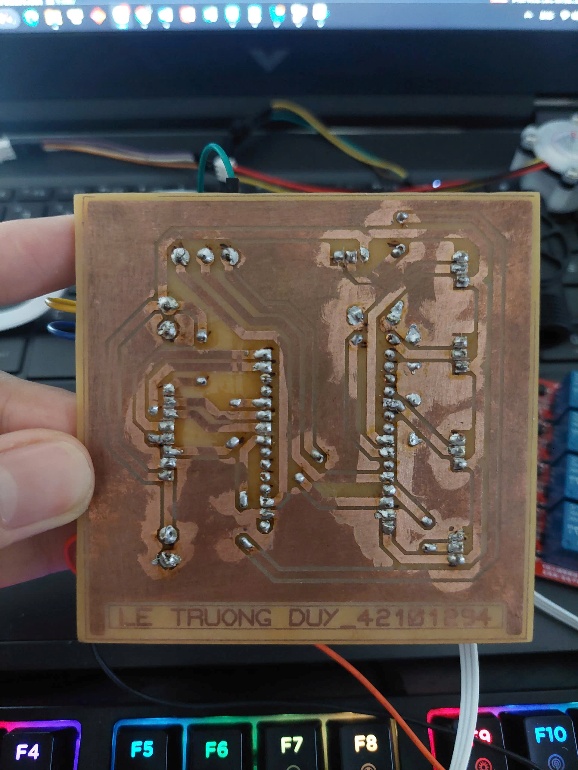
1. **Thiết kế mạch bằng Proteus**





1. **Mạch thực tế đạt được**



1. **Phụ lục Code**
2. **Code Node-Red chuyển dữ liệu dạng String nhận từ Esp sang Json hoặc các dạng dữ liệu phù hợp để chuyển vào InfluxDB:**

// Nhận dữ liệu từ msg.payload

let inputData = msg.payload;

// Regex cho các cảm biến Sensor

let regex = /Sensor (\d+): ([0-9]+\.[0-9]+) L/g;

// Regex cho tổng lượng nước

let totalRegex = /Total: ([0-9]+\.[0-9]+) L/g;

// Regex cho dòng điện

let currentRegex = /Current: ([0-9]+\.[0-9]+) A/g;

let match;

let jsonData = {};

// Tách dữ liệu các Sensor

while ((match = regex.exec(inputData)) !== null) {

let sensorName = `Sensor ${match[1]}`;

let value = parseFloat(match[2]);

jsonData[sensorName] = value;

}

// Tách giá trị dòng điện (Current)

let currentMatch = currentRegex.exec(inputData);

if (currentMatch) {

jsonData['Current'] = parseFloat(currentMatch[1]);

}

// Tách giá trị tổng (Total)

let totalMatch = totalRegex.exec(inputData);

if (totalMatch) {

jsonData['Total'] = parseFloat(totalMatch[1]);

}

// Gán payload cho msg dưới dạng đối tượng JSON

msg.payload = jsonData;

// Trả về msg đã chỉnh sửa

return msg;

1. **Code Querry Script của InfluxDB để vẽ biểu đồ thông qua Grafana**

from(bucket: "DATN\_letruongduy")

|> range(start: v.timeRangeStart, stop: v.timeRangeStop)

|> filter(fn: (r) => r["\_measurement"] == "flow\_data")

|> filter(fn: (r) => r["\_field"] == "Current" or r["\_field"] == "Sensor 1" or r["\_field"] == "Sensor 2" or r["\_field"] == "Sensor 3" or r["\_field"] == "Total")

|> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)

|> yield(name: "mean")

1. **Code Esp thu dữ liệu từ các cảm biến IR, Current, Flow hiển thị lên màn hình LCD TFT và gửi tới hệ thống máy chủ thông qua Mqtt**

#include <WiFi.h>

#include <WiFiClientSecure.h> // Thư viện cho kết nối SSL

#include <PubSubClient.h>

#include <TFT\_eSPI.h> // Thư viện màn hình TFT

// Thông tin kết nối Wi-Fi

const char\* ssid = "wifi của trường duy"; // Thay thế bằng SSID Wi-Fi của bạn

const char\* password = "letruongduy"; // Thay thế bằng mật khẩu Wi-Fi của bạn

// Thông tin kết nối MQTT

const char\* mqttServer = "46b057df071c4ab09dbfdebed9927199.s1.eu.hivemq.cloud";

const int mqttPort = 8883;

const char\* mqttUser = "letruongduy";

const char\* mqttPassword = "Letruongduy972003";

// Tạo đối tượng WiFiSecure và MQTT

WiFiClientSecure espClient;

PubSubClient client(espClient);

// Tạo đối tượng TFT

TFT\_eSPI tft = TFT\_eSPI(); // Khởi tạo đối tượng TFT

// Định nghĩa chân kết nối cho cảm biến hồng ngoại và relay

const int irSensorPin = 12; // Chân D12 kết nối với cảm biến hồng ngoại

const int relayPin = 13; // Chân D13 kết nối với relay

// Định nghĩa chân cảm biến lưu lượng

const byte sensorPin1 = 25; // D25

const byte sensorPin2 = 26; // D26

const byte sensorPin3 = 5; // D27

// Hệ số hiệu chỉnh cho mỗi cảm biến

float calibrationFactor1 = 9.5; // Cảm biến 1

float calibrationFactor2 = 4.2; // Cảm biến 2

float calibrationFactor3 = 4.2; // Cảm biến 3

// Biến lưu lượng cảm biến lưu lượng

volatile byte pulseCount1 = 0;

volatile byte pulseCount2 = 0;

volatile byte pulseCount3 = 0;

// Các biến lưu lượng cảm biến

float flowRate1, flowRate2, flowRate3; // L/min

unsigned int flowMilliLitres1, flowMilliLitres2, flowMilliLitres3; // ml/s

unsigned long totalMilliLitres1, totalMilliLitres2, totalMilliLitres3; // Tổng ml

unsigned long totalAllSensors; // Tổng ml

// Thời gian

unsigned long oldTime;

// Biến kiểm tra debounce

volatile unsigned long lastPulseTime1 = 0;

volatile unsigned long lastPulseTime2 = 0;

volatile unsigned long lastPulseTime3 = 0;

const unsigned long debounceDelay = 50;

const unsigned long minPulseInterval = 10; // 10ms

// Các biến cho cảm biến dòng (Current)

const int currentPin = 35; // GPIO35

const float sensitivity = 185.0; // mV/A cho ACS712-5A

const float ADC\_resolution = 4095.0;

const float ref\_voltage = 3.3; // Điện áp tham chiếu

float zeroCurrentVoltage = 0.0;

float currentMeasurement = 0.0; // Biến lưu dòng

// Hàm xử lý tin nhắn nhận từ MQTT (ẩn đi)

void callback(char\* topic, byte\* payload, unsigned int length) {

    Serial.print("Message arrived [");

    Serial.print(topic);

    Serial.print("]: ");

    String message;

    for (int i = 0; i < length; i++) {

        message += (char)payload[i];

    }

    Serial.println(message);

    // // Có thể xử lý tiếp nếu cần

}

// Hàm kết nối lại MQTT

void reconnect() {

    while (!client.connected()) {

        Serial.print("Attempting MQTT connection...");

        if (client.connect("ESP32Client", mqttUser, mqttPassword)) {

            Serial.println("connected");

            // // Đăng ký topic sau khi kết nối thành công

            // client.subscribe("DATN\_letruongduy");

        } else {

            Serial.print("failed, rc=");

            Serial.print(client.state());

            Serial.println(" try again in 5 seconds");

            delay(5000);

        }

    }

}

// Ngắt cảm biến lưu lượng

void IRAM\_ATTR pulseCounter1() {

    unsigned long currentTime = millis();

    if (currentTime - lastPulseTime1 >= debounceDelay && currentTime - lastPulseTime1 >= minPulseInterval) {

        pulseCount1++;

        lastPulseTime1 = currentTime;

    }

}

void IRAM\_ATTR pulseCounter2() {

    unsigned long currentTime = millis();

    if (currentTime - lastPulseTime2 >= debounceDelay && currentTime - lastPulseTime2 >= minPulseInterval) {

        pulseCount2++;

        lastPulseTime2 = currentTime;

    }

}

void IRAM\_ATTR pulseCounter3() {

    unsigned long currentTime = millis();

    if (currentTime - lastPulseTime3 >= debounceDelay && currentTime - lastPulseTime3 >= minPulseInterval) {

        pulseCount3++;

        lastPulseTime3 = currentTime;

    }

}

// Hàm đọc dòng (Current) từ cảm biến ACS712

void readCurrentSensor() {

    long total = 0;

    for (int i = 0; i < 100; i++) {

        total += analogRead(currentPin);

        delay(2);

    }

    float averageADC = total / 100.0;

    float sensorVoltage = (averageADC / ADC\_resolution) \* ref\_voltage \* 1000.0; // mV

    float voltageDiff = sensorVoltage - zeroCurrentVoltage;

    float current = voltageDiff / sensitivity; // A

    // Nhân kết quả đo dòng với hệ số 2 để chính xác hơn

    current \*= 2;

    if (abs(current) < 0.05) current = 0; // lọc nhiễu nhỏ

    currentMeasurement = current;

}

// Hàm cập nhật hiển thị

void updateDisplay() {

    tft.fillScreen(TFT\_BLACK);

    tft.setTextColor(TFT\_WHITE);

    tft.setTextSize(2);

    tft.setCursor(10, 10);

    tft.print("LUONG NUOC TIEU THU");

    // Sensor 1

    tft.setTextColor(TFT\_RED);

    tft.setCursor(10, 40);

    tft.print("Sensor 1: ");

    tft.print(totalMilliLitres1 / 1000.0);

    tft.print(" L");

    // Sensor 2

    tft.setTextColor(TFT\_GREEN);

    tft.setCursor(10, 70);

    tft.print("Sensor 2: ");

    tft.print(totalMilliLitres2 / 1000.0);

    tft.print(" L");

    // Sensor 3

    tft.setTextColor(TFT\_BLUE);

    tft.setCursor(10, 100);

    tft.print("Sensor 3: ");

    tft.print(totalMilliLitres3 / 1000.0);

    tft.print(" L");

    // Current sensor

    tft.setTextColor(TFT\_YELLOW);

    tft.setCursor(10, 130);

    tft.print("Current: ");

    tft.print(currentMeasurement, 3);

    tft.print(" A");

    // Tổng

    tft.setTextColor(TFT\_WHITE);

    tft.setCursor(10, 160);

    tft.print("Total: ");

    tft.print(totalAllSensors / 1000.0);

    tft.print(" L");

}

void setup() {

    Serial.begin(9600);

    delay(1000);

    // Kết nối Wi-Fi

    WiFi.begin(ssid, password);

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

        Serial.print(".");

        delay(1000);

    }

    Serial.println("WiFi connected");

    // Thiết lập máy chủ MQTT

    client.setServer(mqttServer, mqttPort);

    client.setCallback(callback);

    espClient.setInsecure();

    // Khởi tạo màn hình TFT

    tft.init();

    tft.setRotation(1);

    tft.fillScreen(TFT\_BLACK);

    // Chân cảm biến lưu lượng

    pinMode(sensorPin1, INPUT\_PULLUP);

    pinMode(sensorPin2, INPUT\_PULLUP);

    pinMode(sensorPin3, INPUT\_PULLUP);

    // Chân hồng ngoại và relay

    pinMode(irSensorPin, INPUT);

    pinMode(relayPin, OUTPUT);

    digitalWrite(relayPin, HIGH);

    // Ngắt cảm biến lưu lượng

    attachInterrupt(digitalPinToInterrupt(sensorPin1), pulseCounter1, FALLING);

    attachInterrupt(digitalPinToInterrupt(sensorPin2), pulseCounter2, FALLING);

    attachInterrupt(digitalPinToInterrupt(sensorPin3), pulseCounter3, FALLING);

    // Đọc offset dòng

    long total = 0;

    for (int i = 0; i < 1000; i++) {

        total += analogRead(currentPin);

        delay(1);

    }

    zeroCurrentVoltage = (total / 1000.0) / ADC\_resolution \* ref\_voltage \* 1000.0; // mV

    // Khởi tạo dữ liệu ban đầu

    pulseCount1 = pulseCount2 = pulseCount3 = 0;

    flowRate1 = flowRate2 = flowRate3 = 0.0;

    flowMilliLitres1 = flowMilliLitres2 = flowMilliLitres3 = 0;

    totalMilliLitres1 = totalMilliLitres2 = totalMilliLitres3 = 0;

    totalAllSensors = 0;

    oldTime = millis();

}

void loop() {

    // Đọc trạng thái cảm biến IR

    int irState = digitalRead(irSensorPin);

    if (irState == HIGH) {

        digitalWrite(relayPin, HIGH);

    } else {

        digitalWrite(relayPin, LOW);

    }

    // Kết nối MQTT

    if (!client.connected()) {

        reconnect();

    }

    client.loop();

    unsigned long currentTime = millis();

    // Cập nhật dữ liệu mỗi giây

    if (currentTime - oldTime > 1000) {

        // Tạm detach ngắt để tính toán an toàn

        detachInterrupt(digitalPinToInterrupt(sensorPin1));

        detachInterrupt(digitalPinToInterrupt(sensorPin2));

        detachInterrupt(digitalPinToInterrupt(sensorPin3));

        // Tính lưu lượng cho từng cảm biến

        flowRate1 = ((1000.0 / (currentTime - oldTime)) \* pulseCount1) / calibrationFactor1;

        flowMilliLitres1 = (flowRate1 / 60.0) \* 1000.0;

        totalMilliLitres1 += flowMilliLitres1;

        flowRate2 = ((1000.0 / (currentTime - oldTime)) \* pulseCount2) / calibrationFactor2;

        flowMilliLitres2 = (flowRate2 / 60.0) \* 1000.0;

        totalMilliLitres2 += flowMilliLitres2;

        flowRate3 = ((1000.0 / (currentTime - oldTime)) \* pulseCount3) / calibrationFactor3;

        flowMilliLitres3 = (flowRate3 / 60.0) \* 1000.0;

        totalMilliLitres3 += flowMilliLitres3;

        totalAllSensors = totalMilliLitres1 + totalMilliLitres2 + totalMilliLitres3;

        // Đọc dòng (Current)

        readCurrentSensor();

        // Gửi dữ liệu qua MQTT

        if (pulseCount1 > 0 || pulseCount2 > 0 || pulseCount3 > 0 || abs(currentMeasurement) > 0.05) {

            String message = String("Sensor 1: ") + totalMilliLitres1 / 1000.0 + " L, "

                             + "Sensor 2: " + totalMilliLitres2 / 1000.0 + " L, "

                             + "Sensor 3: " + totalMilliLitres3 / 1000.0 + " L, "

                             + "Current: " + String(currentMeasurement, 3) + " A, "

                             + "Total: " + totalAllSensors / 1000.0 + " L";

            client.publish("DATN\_letruongduy", message.c\_str());

            Serial.println(message);

        }

        // Cập nhật màn hình

        updateDisplay();

        // Reset cảm biến đếm

        pulseCount1 = pulseCount2 = pulseCount3 = 0;

        oldTime = currentTime;

        // Gắn lại ngắt

        attachInterrupt(digitalPinToInterrupt(sensorPin1), pulseCounter1, FALLING);

        attachInterrupt(digitalPinToInterrupt(sensorPin2), pulseCounter2, FALLING);

        attachInterrupt(digitalPinToInterrupt(sensorPin3), pulseCounter3, FALLING);

    }

}