BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA ĐIỆN - ĐIỆN TỬ

BỘ MÔN: CƠ SỞ VÀ ỨNG DỤNG IOT

BÁO CÁO GIỮA KỲ

HỆ THỐNG GIÁM SÁT HỒ CÁ SỬ DỤNG CẤU TRÚC 2

Mã môn học và mã lớp: ITFA336064_03CLC, HK1, NH: 2024-2025

Giảng viên hướng dẫn: Vũ Chí Cường

Nhóm thực hiện: Nhóm 01, Thứ 7, tiết 10-12

Danh sách thành viên nhóm:

Họ tên	Mã số sinh viên
Đặng Hưng Thịnh	22119137
Trần Tuấn Kiệt	22119095
Lê Khải Hưng	22119086
Lê Hoàng Gia Khang	22119087
Trần Hoàng Hải	22119066

Thành phố Hồ Chí Minh - Tháng 10, năm 2024

PHÂN CÔNG NHIỆM VỤ

1. **Tên đề tài**: Hệ thống giám sát hồ cá sử dụng cấu trúc 2.
2. **Bản phân công nhiệm vụ:**

STT	Công việc	Mô tả chi tiết	Sinh viên thực hiện
1	Thu thập thông tin về nhiệt độ	Sử dụng DHT22 thu thập nhiệt độ, sau đó dùng MQTT public cho user subscribe bởi ESP32	Lê Hoàng Gia Khang
2	Thu thập độ đục của nước	Thu thập khoảng tín hiệu tương tự (cảm biến MJKDZ) từ đó tính toán phần trăm, thông qua ESP32 gửi trạng thái đục/trong của nước public cho user subscribe	Lê Khải Hưng
3	Xác định sự thay đổi mực nước bể cá	Sử dụng cảm biến HY-SRF05 để thu thập thông tin về mực nước của bể, sau đó public bằng ESP32	Đặng Hưng Thịnh
4	Nhận thông tin từ thiết bị biên, xử lí hiển thị web server	Sử dụng Raspberry Pi nhận từ	Trần Hoàng Hải

		broker server và xử lý và hiển thị	
5	Xác định vấn đề , nhận diện yêu cầu người dùng và chỉnh sửa báo cáo	Tìm hiểu rõ về vấn đề cần giải quyết và giới thiệu những tiện ích mà hệ thống mang lại cho người sử dụng	Trần Tuấn Kiệt

NHẬN XÉT CỦA GIẢNG VIÊN

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Chữ kí của giảng viên

(Kí, ghi rõ họ tên)

Vũ Chí Cường

MỤC LỤC

PHẦN A: XÁC ĐỊNH VẤN ĐỀ

Trong những năm gần đây, nuôi cá cảnh trong nhà đã trở thành một sở thích phổ biến, không chỉ vì tính thẩm mỹ mà còn vì khả năng tạo ra không gian thư giãn cho gia đình.Nhưng đối với những người mới nuôi cá, việc duy trì một môi trường lý tưởng cho cá sinh sống không hề đơn giản.

Các yếu tố như nhiệt độ, độ đục và mực nước trong bể cá đòi hỏi phải được theo dõi và điều chỉnh liên tục, nếu không sẽ gây ảnh hưởng xấu đến sức khỏe của cá. Người mới bắt đầu thường thiếu kinh nghiệm trong việc kiểm soát các yếu tố này, dễ dẫn đến sai lầm trong quá trình chăm sóc. Chính vì vậy, việc phát triển một hệ thống giám sát hồ cá tự động là cần thiết, với hai chức năng chính: đo nhiệt độ nước, độ đục của nước và theo dõi mực nước hao hụt trong bể cá qua thời gian.

Hệ thống sẽ cập nhật và truyền tải các thông số này lên một website, cho phép người dùng giám sát bể cá từ xa qua điện thoại hoặc máy tính. Điều này đặc biệt hữu ích cho những người bận rộn hoặc thường xuyên vắng nhà, giúp họ tiết kiệm thời gian và công sức mà vẫn đảm bảo môi trường sống ổn định cho cá. Ngoài ra, hệ thống này cũng hỗ trợ người mới nuôi cá dễ dàng hơn trong việc theo dõi và điều chỉnh các thông số quan trọng, tránh được các sai lầm có thể gây hại cho cá.

Bằng cách tích hợp công nghệ IoT, hệ thống không chỉ giúp việc nuôi cá trở nên thuận tiện và hiệu quả hơn mà còn tăng cường sức khỏe và tuổi thọ của cá bằng cách duy trì một môi trường ổn định. Việc chọn đề tài này có ý nghĩa lớn trong việc áp dụng công nghệ vào cuộc sống, đồng thời giải quyết những khó khăn thực tế mà người nuôi cá cảnh, đặc biệt là người mới, đang gặp phải.

1. Yêu cầu người dùng

- Hệ thống giám sát nhiệt độ, mực nước, độ đục - trong của nước, hiển thị các thông này lên website. Người dùng có thể truy cập thông tin qua điện thoại, máy tính,…

- Giao diện đơn giản, hiển thị thông tin rõ ràng, trực quan để người dùng dễ theo dõi, cập nhật tình hình.

- Hệ thống cập nhật thông tin về nhiệt độ, lượng nước, mức độ sạch của nước.

- Tốc độ phản hồi nhanh chóng , hiển thị các thông số nhanh và chính xác. Giúp người dùng có những điều chỉnh kịp thời.

2. Đặc tả kỹ thuật

2.1. Chức năng của hệ thống

- Cảm biến nhiệt độ sẽ đo nhiệt độ nước trong bể cá.

- Cảm biến siêu âm dùng để đo mực nước trong bể cá.

- Cảm biến độ đục đo mức độ sạch của bể cá.

- Hệ thống sẽ cập nhật và truyền tải các thông số này lên một website, cho phép người dùng giám sát bể cá từ xa qua điện thoại.

2.2. Thông số kỹ thuật

- Nguồn điện: +5V

- Dòng: 1 (2A)

- Ngưỡng nhận biết độ sạch của nước (0-100%).

- Ngưỡng nhiệt độ cho phép cảm biến đo được (0^oC - 50^oC)

- Chiều cao của bể cá: h = 50 cm.

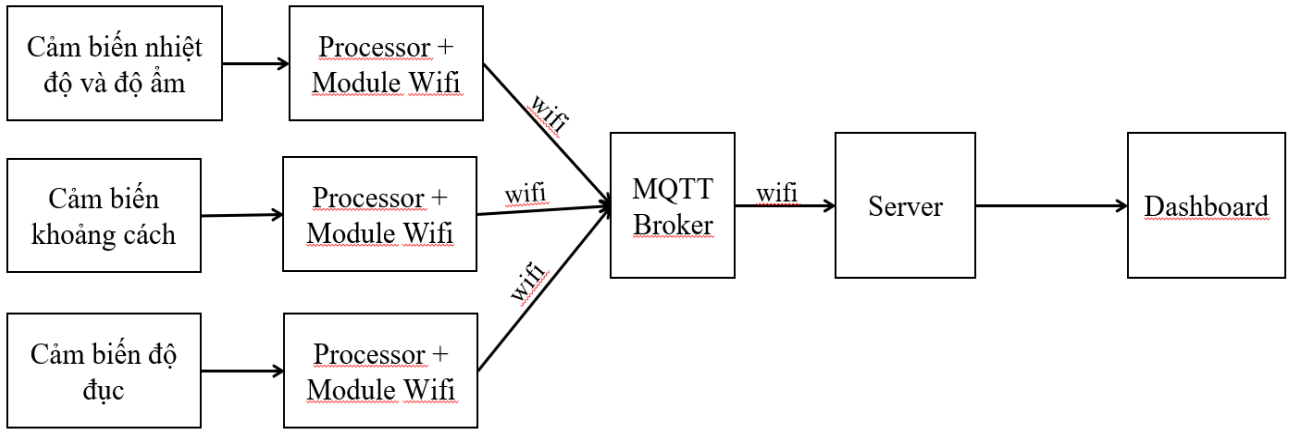
- Khoảng cách từ cảm biến siêu âm đến mực nước: n

- Mực nước trong bể cá: h - n = (50 - n) cm

PHẦN B: THIẾT KẾ HỆ THỐNG

CHƯƠNG 1: KIẾN TRÚC HỆ THỐNG

1.1 Sơ đồ khối tổng quát



Hình 1. Kiến trúc tổng quát.

1.2 Mô tả các thành phần chính

- Cảm biến nhiệt độ và độ ẩm, cảm biến khoảng cách: Nhận tín hiệu từ môi trường. Sau đó được giao tiếp để đẩy dữ liệu lên.

- Processor + Module Wifi: Nhận tín hiệu từ các cảm biến. Kết nối wifi và gửi lên MQTT Broker.

- MQTT Broker: Nơi trung nhận các tin nhắn và lưu trữ tạm thời.

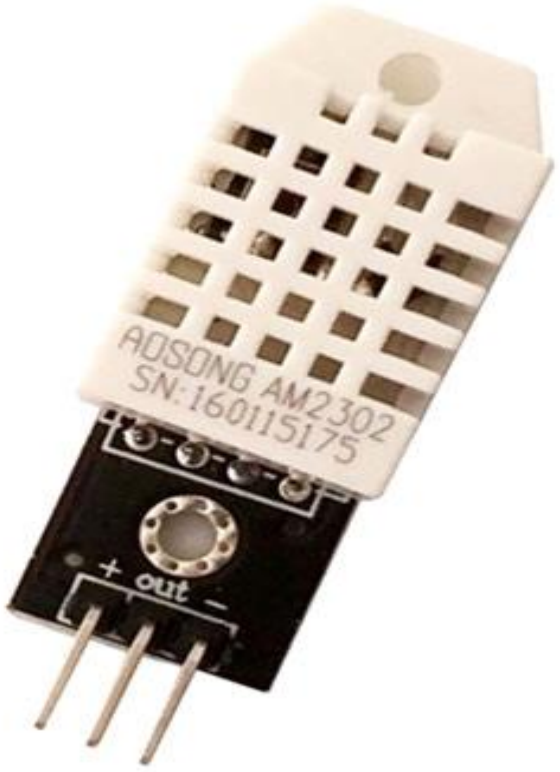
- Server: Lấy các tin nhắn từ MQTT Broker. Xử lý dữ liệu thô, sau đó hiện lên Dashboard.

- Dashboard: Hiển thị các thông số theo dạng gauge, đồ thị.

CHƯƠNG 2: THIẾT KẾ PHẦN CỨNG

2.1 Lựa chọn và đặc tả kỹ thuật của các thành phần phần cứng

2.1.1. Cảm biến nhiệt độ và độ ẩm (DHT22 Module)



Hình 2. Module DHT22.

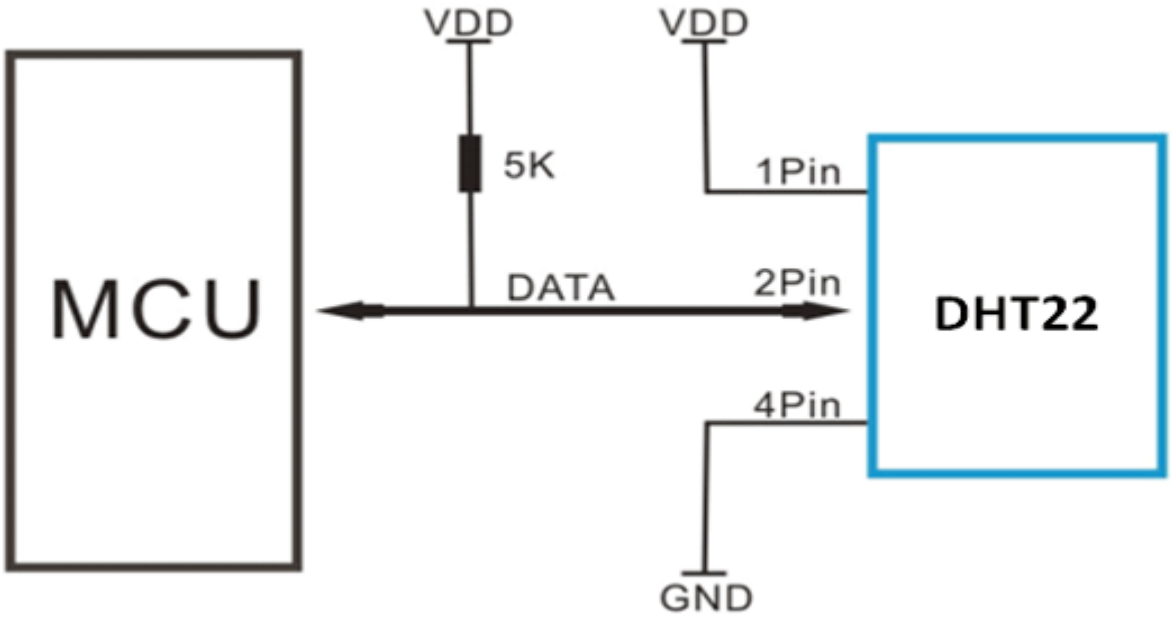
Thông số kỹ thuật:

- Điện áp hoạt động: 3,3V-5,5V.
- Dòng điện tiêu thụ: 0,3mA (khi đo), 60 µA (chế độ chờ).
- Phạm vi nhiệt độ: -40°C đến 80°C.
- Phạm vi độ ẩm: 0% đến 100%.
- Độ phân giải: Nhiệt độ và Độ ẩm đều là 16-bit.
- Độ chính xác: ±0,5°C và ±2% (max ±5%) .
- Kích thước: 15.1mm x 25mm x 7.7mm.
- Tín hiệu ngõ ra: Tín hiệu số thông qua 1 dây bus.

Sơ đồ chân: 3 Chân

- Vcc: Nguồn điện 3.5V đến 5.5V.
- Data: Xuất ra cả Nhiệt độ và Độ ẩm thông qua Dữ liệu nối tiếp.
- GND: Kết nối với mặt đất của mạch.

Sơ đồ kết nối của cảm biến:



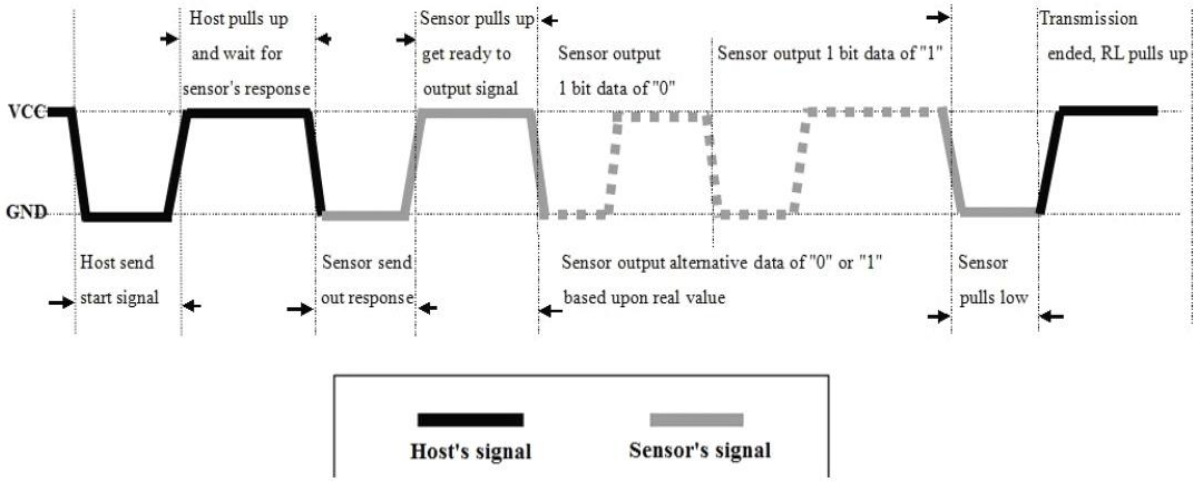
Hình 3. Sơ đồ kết nối MCU với DHT22.

Nguyên lý hoạt động:

- Dữ liệu (40 bit): 16 bits dữ liệu của độ ẩm, 16 bits dữ liệu của nhiệt độ, 8 bits checksum.
- Trong đó nhiệt độ và độ ẩm sẽ có 8 bits đầu là phần nguyên 8 bits sau là thập phân.
- 16 bits dữ liệu sẽ được chuyển từ nhị phân sang thập phân để đưa ra kết quả, khi MCU nhận sẽ tính tổng 8 bits so sánh với 8 bits check-sum.

  Khi vi điều khiển (MCU) gửi tín hiệu khởi động, DHT22 sẽ chuyển từ trạng thái chờ sang trạng thái hoạt động. Khi MCU hoàn tất việc gửi tín hiệu khởi động, DHT22 sẽ gửi tín hiệu phản hồi bao gồm 40 bit dữ liệu phản ánh độ ẩm tương đối và nhiệt độ đến MCU. Nếu không có tín hiệu khởi động từ MCU, DHT22 sẽ không gửi tín hiệu phản hồi. Một tín hiệu khởi động từ MCU sẽ nhận được một phản hồi dữ liệu từ DHT22, phản ánh độ ẩm tương đối và nhiệt độ. DHT22 sẽ chuyển về trạng thái chờ khi quá trình thu thập dữ liệu hoàn tất, nếu không nhận được tín hiệu khởi động từ MCU nữa.

  Xem hình dưới đây để biết quá trình giao tiếp tổng thể, khoảng thời gian của toàn bộ quá trình phải lớn hơn 2 giây.



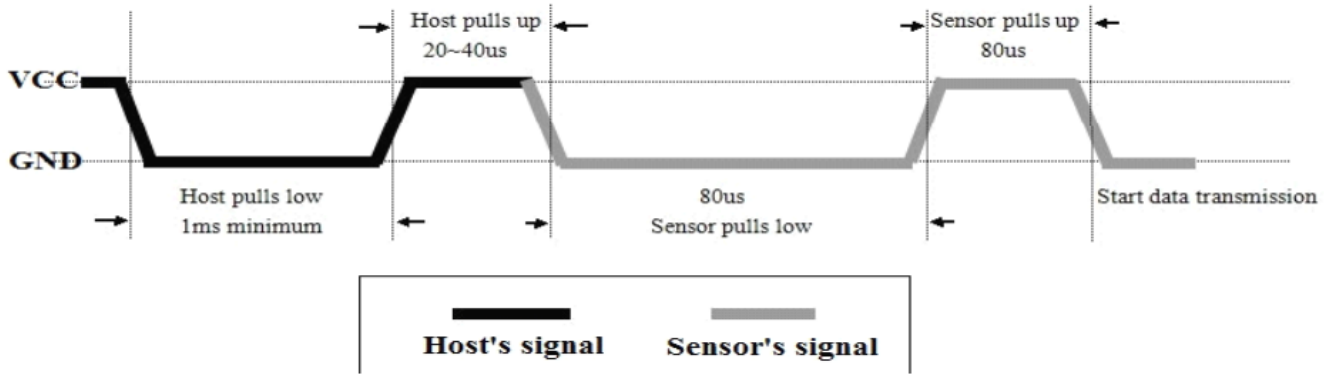
Hình 4. Sơ đồ nguyên lí hoạt động của DHT22

Bước 1: MCU gửi tín hiệu khởi động đến DHT22 và DHT22 sẽ gửi tín hiệu phản hồi đến MCU.

- MCU kéo mức điện áp trên đường truyền dữ liệu xuống thấp trong khoảng 1-10ms để báo hiệu bắt đầu giao tiếp.

- Sau đó, MCU kéo mức điện áp lên cao và chờ khoảng 20-40µs để nhận tín hiệu phản hồi.

- Khi DHT22 nhận được tín hiệu khởi động, nó sẽ kéo mức điện áp xuống thấp trong 80µs như một tín hiệu phản hồi, rồi kéo mức điện áp lên cao trong 80µs để chuẩn bị gửi dữ liệu nhiệt độ và độ ẩm.



Hình 5. Nguyên lí truyền tín hiệu khởi đầu.

Bước 2: DHT22 gửi dữ liệu đến MCU

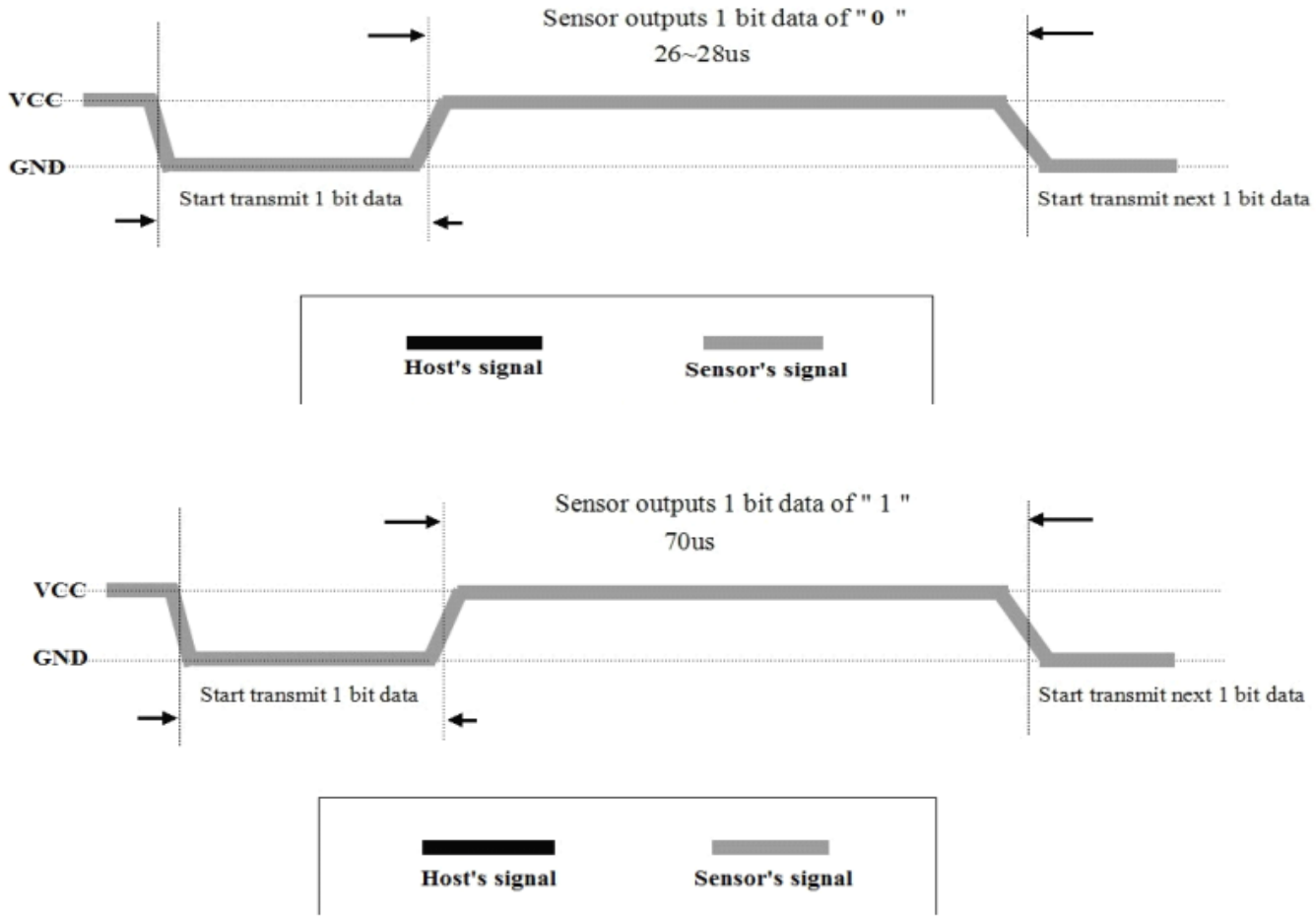
Khi DHT22 gửi dữ liệu đến MCU, quá trình truyền mỗi bit bắt đầu bằng mức điện áp thấp kéo dài 50µs. Tiếp theo, độ dài của tín hiệu ở mức điện áp cao sẽ quyết định bit

đó là "1" hay "0".

- Nếu tín hiệu mức cao kéo dài khoảng 26-28µs, bit sẽ là "0".

- Nếu tín hiệu mức cao kéo dài khoảng 70µs, bit sẽ là "1".

DHT22 sẽ lặp lại quá trình này cho mỗi bit trong chuỗi 40 bit dữ liệu, gồm độ ẩm và nhiệt độ.



Hình 6. Cách truyền dữ liệu và phân biệt được bit 1 và bit 0.

2.1.2 Cảm biến siêu âm dùng để đo mực nước (HY-SRF05)

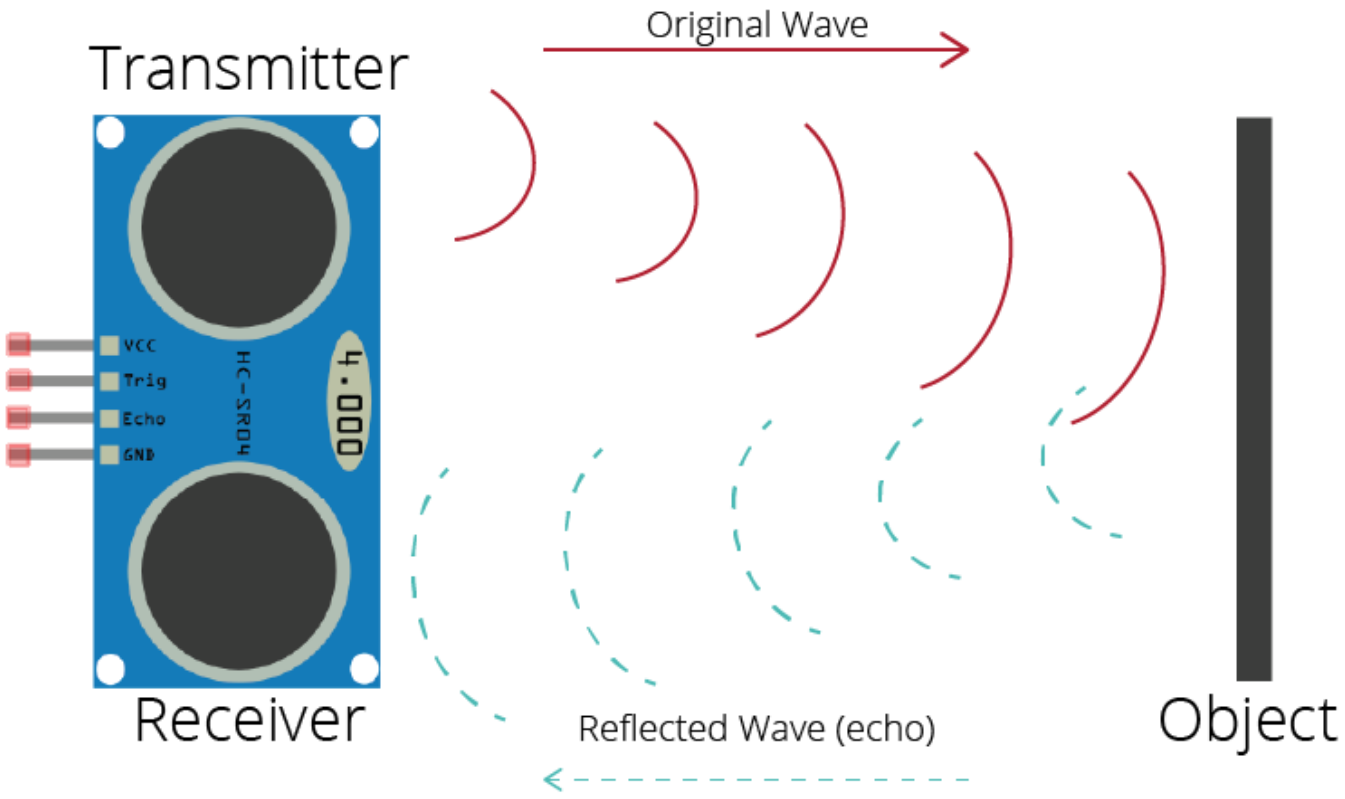
Hình 7. Cảm biến sóng âm HY-SRF05

2.1.2.1 Thông số kỹ thuật

- Điện áp vào: 5V
- Dòng tiêu thụ : <2mA
- Tín hiệu đầu ra: xung HIGH (5V) và LOW (0V)
- Khoảng cách đo : 2cm – 450cm
- Độ chính xác : 0.5cm
- Kích thước: 20mm*45mm*15mm
- Góc cảm biến : <15^{o}

2.1.2.2 Sơ đồ chân: 5 chân

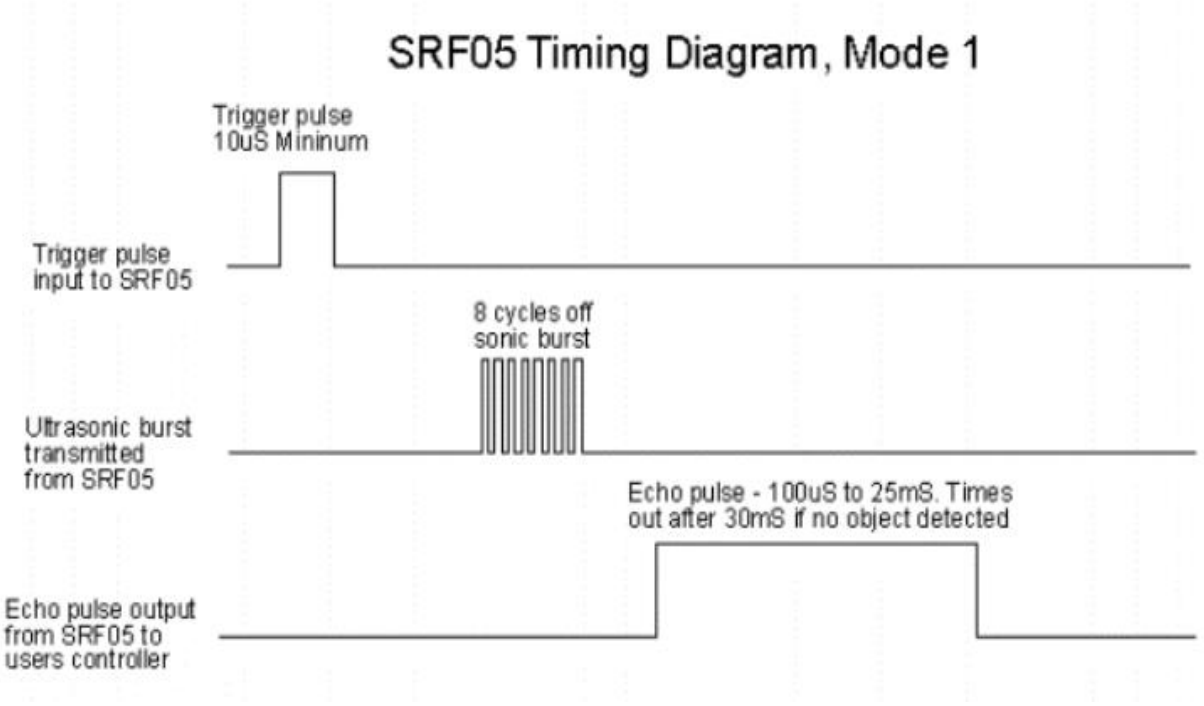
- VCC: 5V (3.3V).
- Trig(T): digital input, khởi động quá trình phát sóng siêu âm
- Echo ®: digital output, chân này nhận tín hiệu phản xạ sau khi sóng siêu âm dội lại vật cản
- OUT.
- GND.



Hình 8: Ảnh mô phỏng quá trình phát sóng (Trig) và phản xạ lại (Echo)

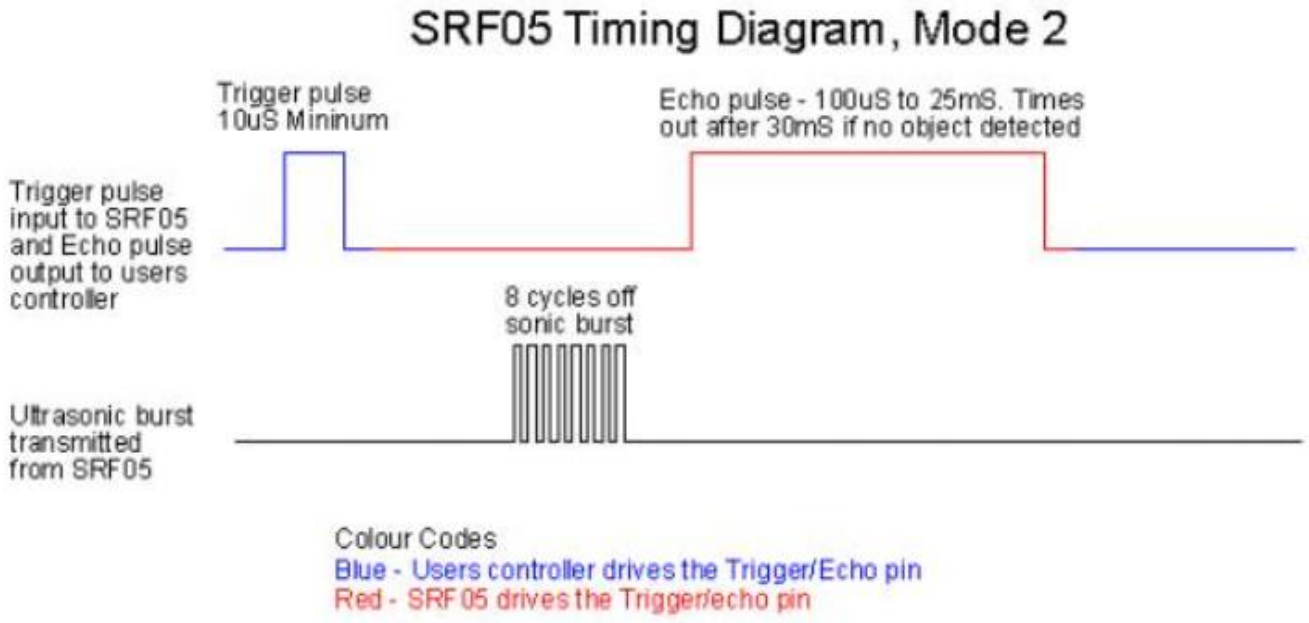
2.1.2.3 Nguyên lý hoạt động

- Để đo khoảng cách, ta phát 1 xung rất ngắn (5µs) từ chân TRIG. Sau đó cảm biến sẽ tạo ra 1 xung HIGH ở chân ECHO cho đến khi nhận được xung phản xạ ở chân này. Chiều rộng của xung sẽ bằng với thời gian sóng siêu âm được phát từ cảm biến quay trở lại. Tốc độ của âm thanh trong không khí là 340 m/s



Hình 9. Ở chế độ 1: tách biệt, kích hoạt và phản hồi

- Ta sử dụng chân OUT để nó vừa phát ra xung rồi nhận xung phản xạ về,chân chế độ thì nối đất. Tín hiệu hồi tiếp sẽ xuất hiện trên cùng 1 chân với tín hiệu kích hoạt. SR05 sẽ không tăng dòng phản hồi cho đến 700µs sau khi kết thúc các tín hiệu kích hoạt và ta đã có thời gian để kích hoạt pin xoay quanh và làm cho nó trở thành 1 đầu vào.



Hình 10. Ở chế độ 2: Dùng 1 chân cho cả kích hoạt và phản hồi.

2.1.3 Cảm biến đo độ đục của nước (MJKDZ)

- Độ đục là một trong những tiêu chí dùng để kiểm tra chất lượng nước, nó thể hiện bằng lượng hạt tồn tại trong nước, lượng hạt càng tăng thì mức độ đục của nước càng tăng

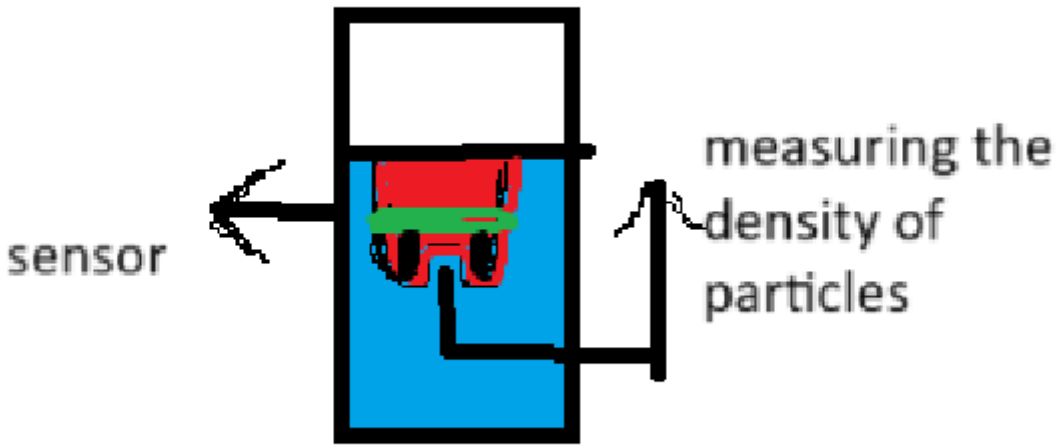
2.1.3.1. Thông số kỹ thuật

- Điện áp hoạt động: 5V
- Dòng điện làm việc: 40mA (Max)
- Thời gian đáp ứng: < 500ms
- Điện trở cách điện: 100M (Min)
- Đầu ra Analog 0 ~ 4.5V
- Đầu ra Digital: High/Low (Có thể điều chỉnh giá trị ngưỡng bằng biến trở)
- Nhiệt độ hoạt động: 5^{o}C - 90^{o}C
- Nhiệt độ dự trữ: -10^{o}C - 90^{o}C
- Kích thước: 38mm*28mm*10mm
- Khối lượng: 30g

2.1.3.2. Sơ đồ chân

- VCC: 5V (3.3V).
- D: Chân thu - nhận tín hiệu từ cảm biến
- GND.

2.1.3.3. Nguyên lý hoạt động



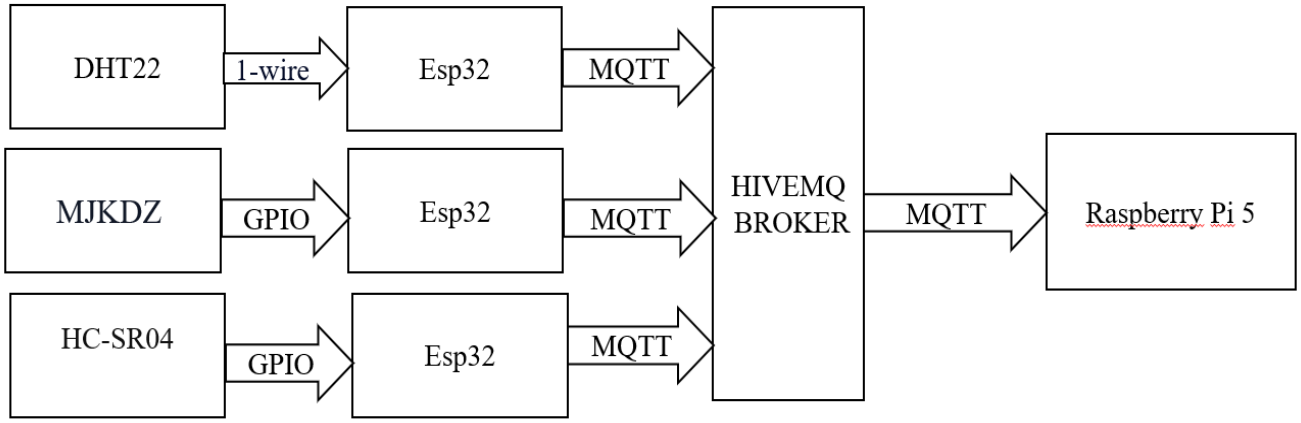
Hình 11. Nguyên lí hoạt động của cảm biến MJKDZ

- Dựa vào mật độ ở 2 đầu cảm biến mà thu được giá trị tương tự (tùy thuộc vào lượng hạt trong dung dịch chất lỏng mà đưa ra được giá trị).

2.1.4 Server: Raspberry Pi 5 4gb

- 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU, 512KB per-core L2 caches và 2MB shared L3 cache
- VideoCore VII GPU @800MHz, hỗ trợ OpenGL ES 3.1, Vulkan 1.2
- 2.4GHz/5GHz dual-band 802.11ac Wi-Fi®
- Nguồn DC 5V/5A qua USB-C, có hỗ trợ Power Delivery

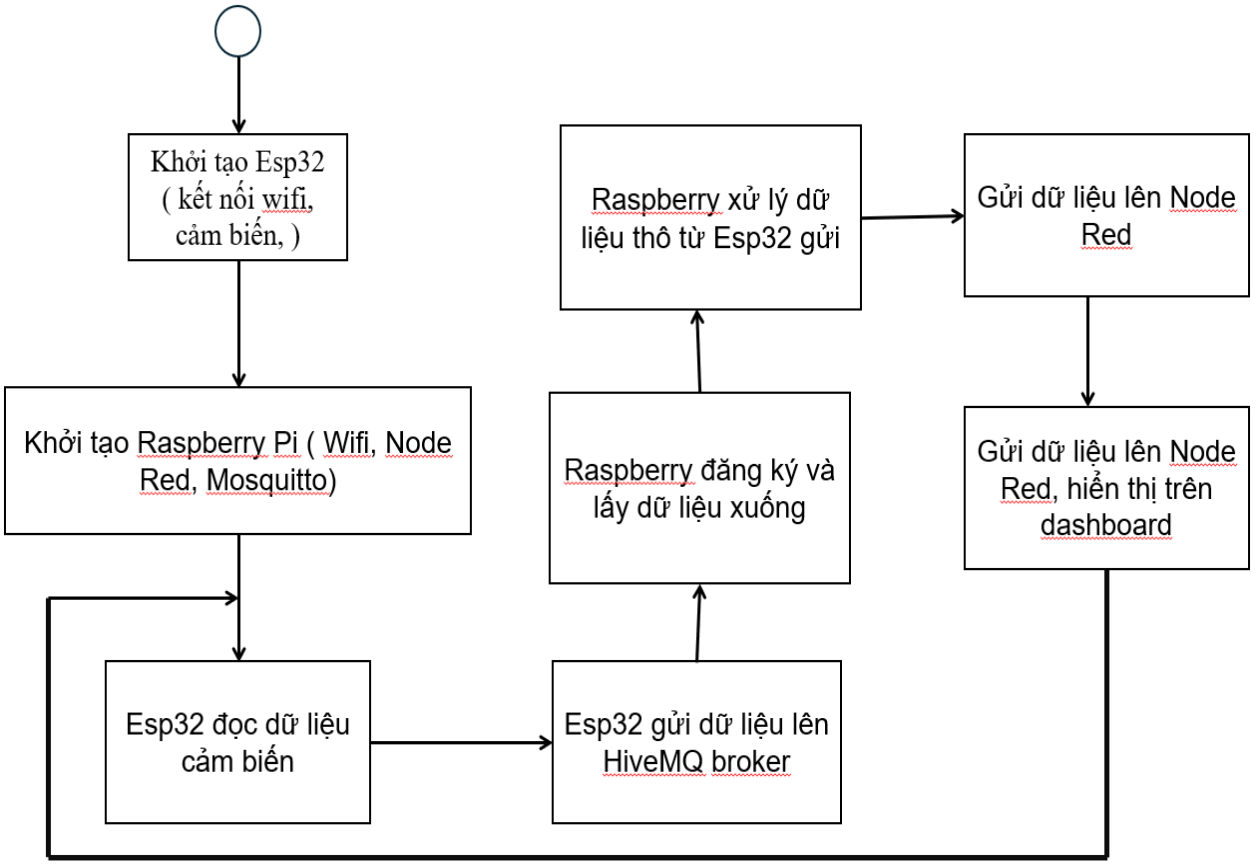
2.2 Sơ đồ kết nối phần cứng



Hình 12. Sơ đồ kết nối.

CHƯƠNG 3: THIẾT KẾ PHẦN MỀM

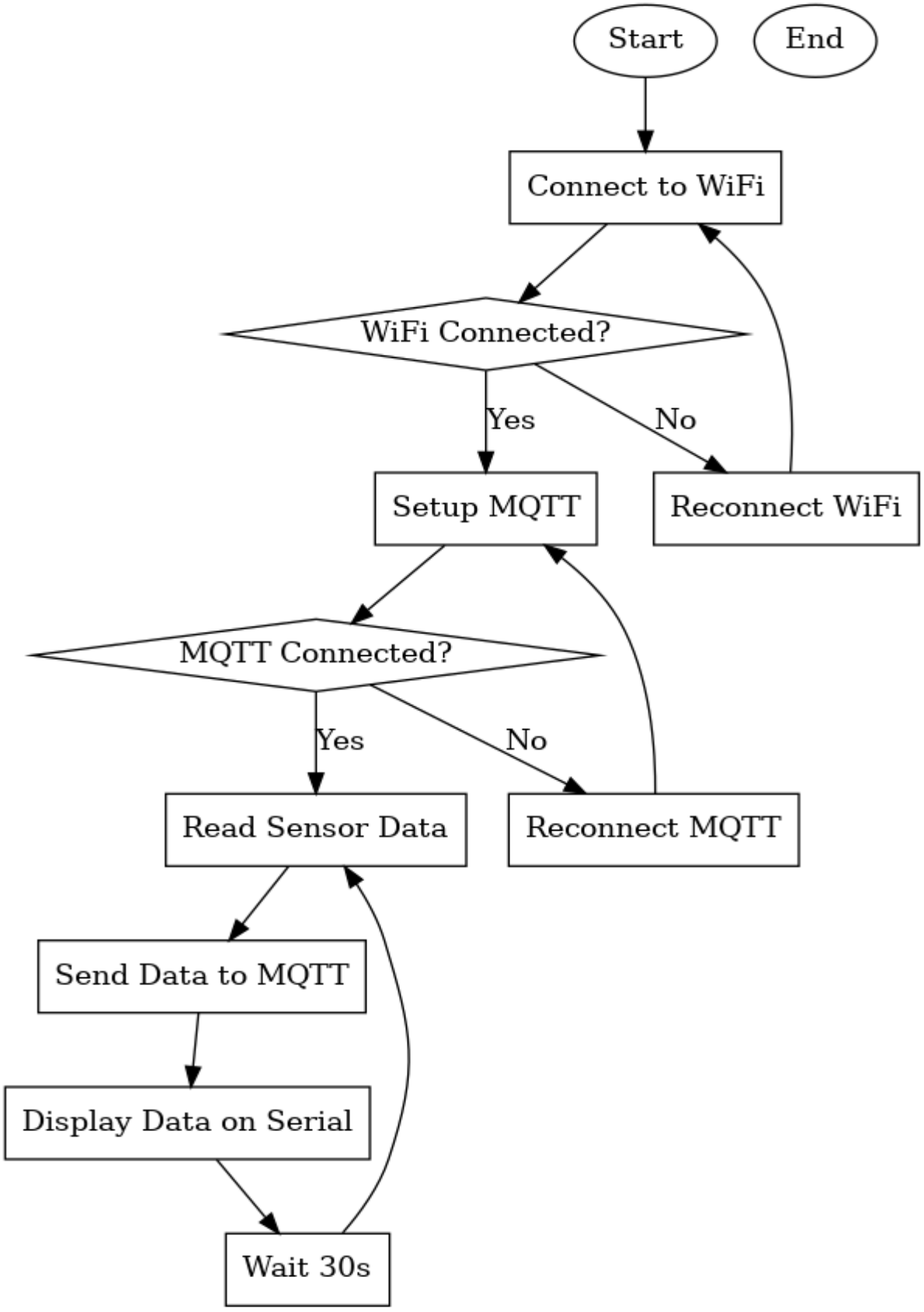
3.1. Thuật toán chính

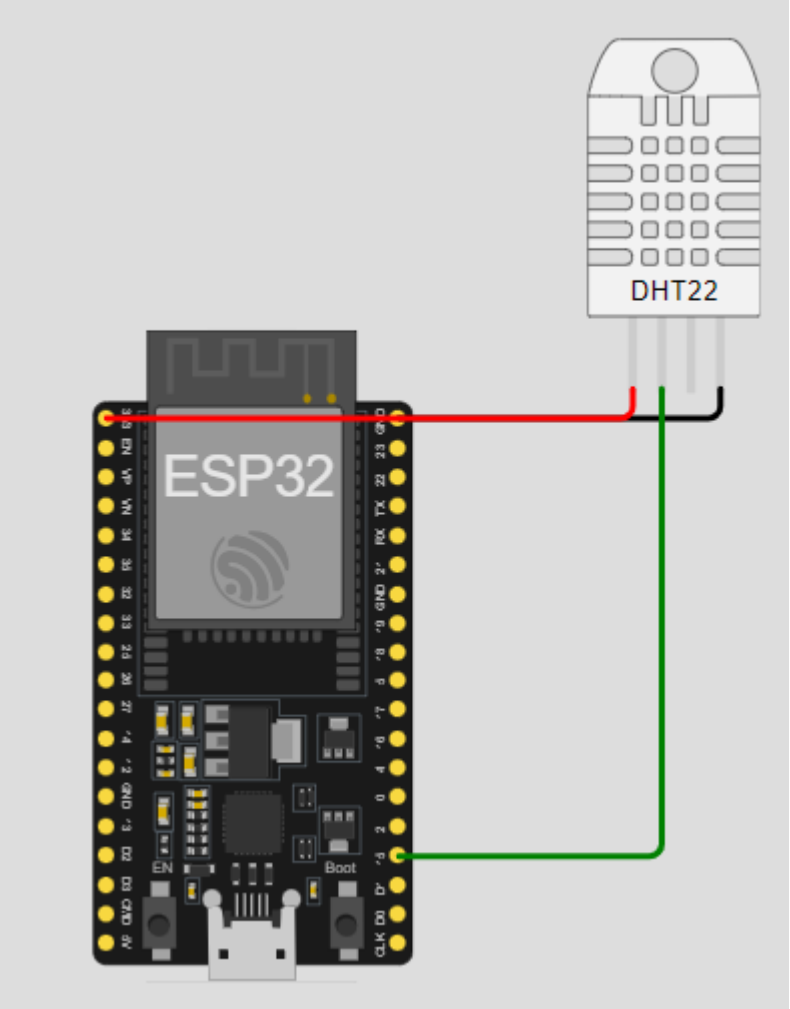


Hình 13: Giải thuật xử lý của hệ thống

3.2. Thiết lập Esp32 thứ nhất

3.2.1. Sơ đồ thuật toán và sơ đồ kết nối





Hình 14. Sơ đồ thuật toán và sơ đồ kết nối DHT22 với ESP32.

3.2.2 Source code

3.2.2.1 Khai báo thư viện, port, biến:

- **Khai báo thư viện**

```
//Thư viện sử dụng
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHTesp.h>
```

- **Khai báo chân DHT và thông tin wifi, MQTT**

```
//Khai báo chân DHT và thông tin Wifi, MQTT
const int DHT_PIN = 15;
DHTesp dht;
const char* ssid = "Le Thang";
const char* password = "lethang1976";
const char* mqtt_server = "test.mosquitto.org";
```

- **Khởi tạo đối tượng cần kết nối**

```
//khởi tạo đối tượng cần kết nối
WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
float temp = 0;
```

3.2.2.2. Kết nối wifi

```
//Kết nối Wifi
void setup_wifi() {
 delay(10);
 Serial.println();
 Serial.print("Connecting to ");
 Serial.println(ssid);
 WiFi.mode(WIFI_STA);
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
 }
 randomSeed(micros());
 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
}
```

3.2.2.3. Xử lí dữ liệu được gửi từ MQTT đến ESP32

```
//Xử lí dữ liệu từ MQTT broker gửi về ESP32
void callback(char* topic, byte* payload, unsigned int length) {
 Serial.print("Message arrived [");
 Serial.print(topic);
 Serial.print("] ");
 for (int i = 0; i < length; i++) {
  Serial.print((char)payload[i]);
 }}
```

3.2.2.4. Hàm kết nối MQTT broker

```
void reconnect() {
 while (!client.connected()) {
  Serial.print("Attempting MQTT connection...");
  String clientId = "ESP32Client-";
  clientId += String(random(0xffff), HEX);
  if (client.connect(clientId.c_str())) {
   Serial.println("Connected");
   client.publish("topic/Publish", "Welcome");
   client.subscribe("topic/Subscribe");
  } else {
   Serial.print("failed, rc=");
   Serial.print(client.state());
   Serial.println(" try again in 5 seconds");
   delay(5000);
```

```
  }}

}
```

3.2.2.5. Thiết lập phần cứng và giao tiếp cơ bản

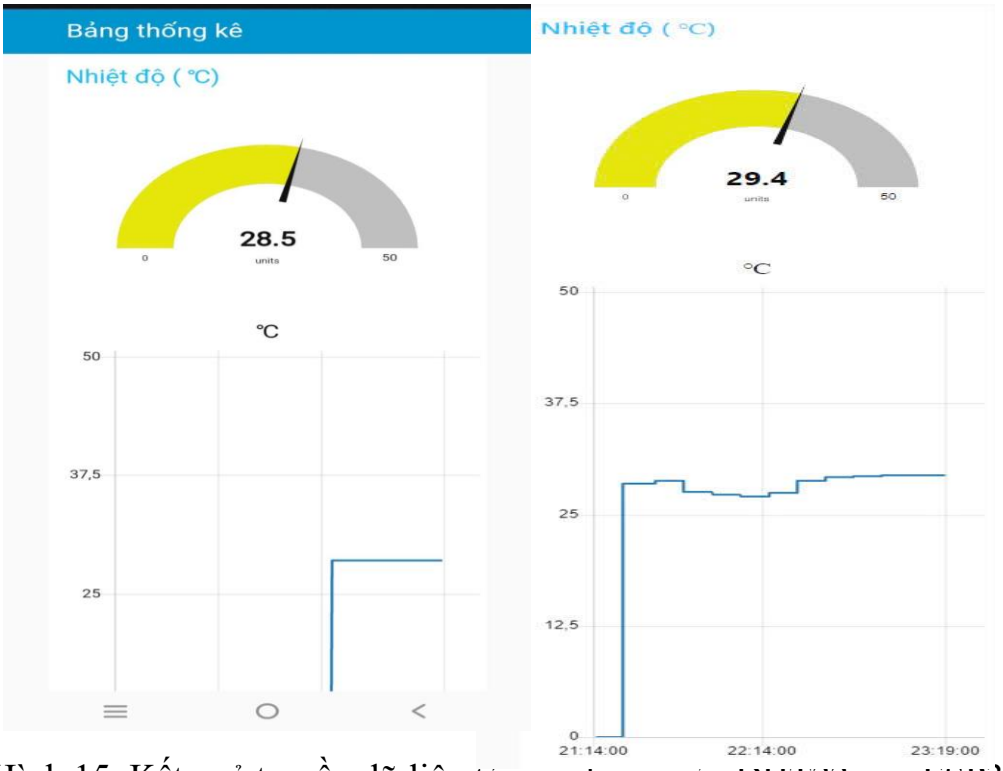
```
//Thiết lập phần cứng và giao tiếp cơ bản
void setup() {
 pinMode(2, OUTPUT);
 Serial.begin(115200);
 setup_wifi();
 client.setServer(mqtt_server, 1883);
 client.setCallback(callback);
 dht.setup(DHT_PIN, DHTesp::DHT22);
}
```

3.2.2.6. Đọc và gửi dữ liệu của cảm biến DHT22 từ ESP32 lên MQTT broker.

```
void loop() {
 if (!client.connected()) {
   reconnect();
 }
 client.loop();
 unsigned long now = millis();
 if (now - lastMsg > 30000) {
   lastMsg = now;
   TempAndHumidity  data = dht.getTempAndHumidity();
   String temp = String(data.temperature, 2);
   client.publish("topic/sen1", temp.c_str());
   Serial.print("Temperature: ");
   Serial.println(temp);
 }
}
```

3.2.2.7. Kết quả và đánh giá

- Kết quả truyền dữ liệu thành công, gửi giá trị nhiệt đồ của dht22 tới raspberry mà không bị ảnh hưởng bởi nhiễu và giá trị chính xác. Chu kì gửi là 30 giây/lần.
- Hình ảnh kết quả:



Hình 15. Kết quả truyền dữ liệu tới raspberry của DHT22 qua ESP32.

- Kết quả là 2 hình bắt đầu lúc 9h30 tối kết thúc vào 11h30 tối. Trong quá trình truyền không có 1 vấn đề trục trặc gì xảy ra. Nhiệt độ hiển thị trong quá trình truyền thay đổi liên tục và không có sự chênh lệch quá lớn.

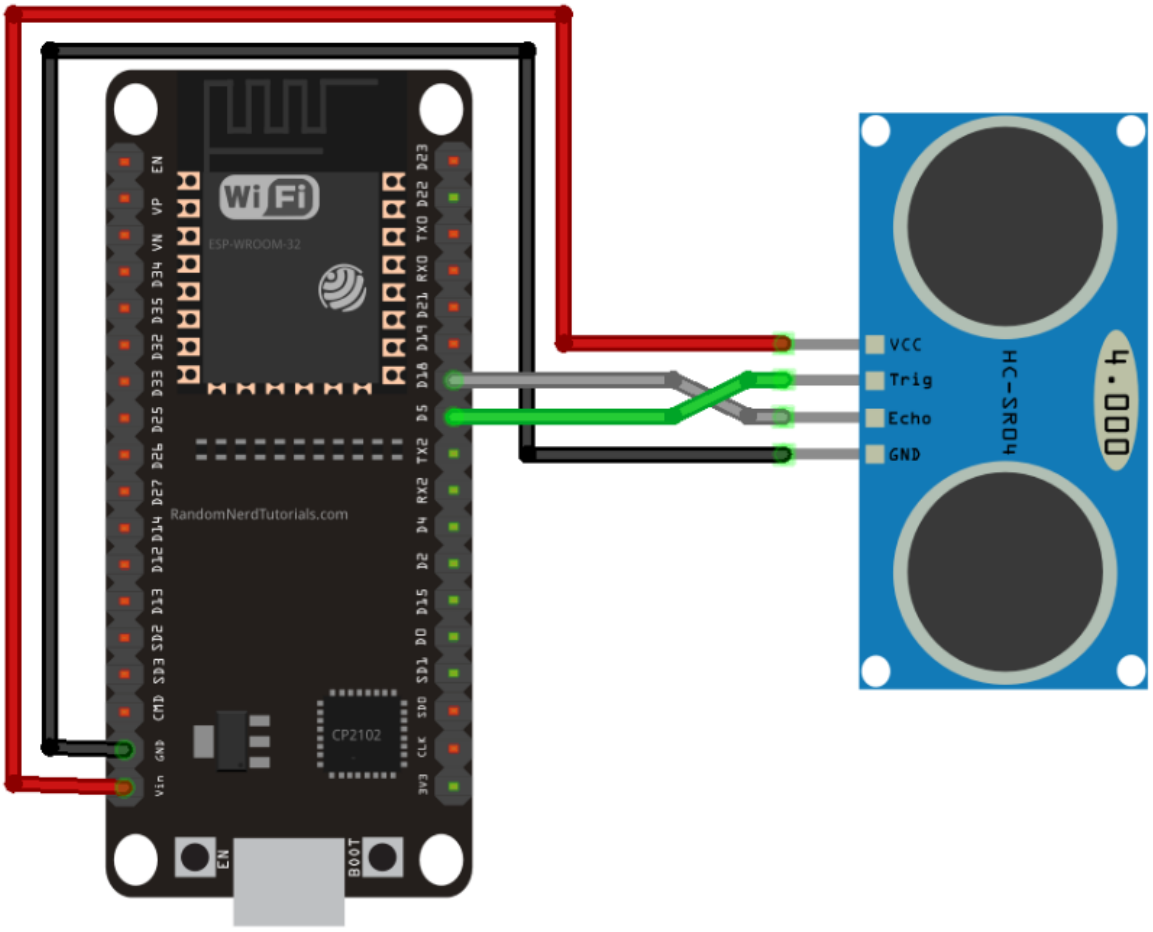
3.3. Thiết lập trên Esp32 thứ hai

Hình 16. Lưu đồ giải thuật

3.3.1. Khai báo thư viện

```
#include <WiFi.h>
#include <PubSubClient.h>
```

3.3.2. Khởi tạo port chân out/in, kết nối wifi và MQTT broker



Hình 17: Sơ đồ kết nối ESP32 với HC-SR04 (05)

```
const int trigPin=5;
const int echoPin=18;
const char* ssid = "P813";     // Your WiFi SSID
const char* password = "dddddddd"; // Your WiFi Password
const char* mqtt_server = "test.mosquitto.org";  // MQTT broker server
```

Khởi tạo các thông tin về chân cấu hình, Wifi, broker

```
void setup() {

  Serial.begin(115200);
  setup_wifi();  // Connect to WiFi
  clientmqtt.setServer(mqtt_server, 1883);  // Set the MQTT server and port
```

Chọn port 1883

```
unsigned long now = millis();
if (now - lastMsg > 10000) {  // Publish every 2 seconds
  lastMsg = now;
```

Setup gửi dữ liệu (Publish 10s/lần)

```
clientmqtt.publish("topic/sen3", data.c_str());
```

Gửi lên Topic/sen3 của Raspberry Pi

3.3.3. Thuật toán để thu được giá trị từ cảm biến

- Tốc độ âm thanh ở 20°C (68°F): 343 m/s.

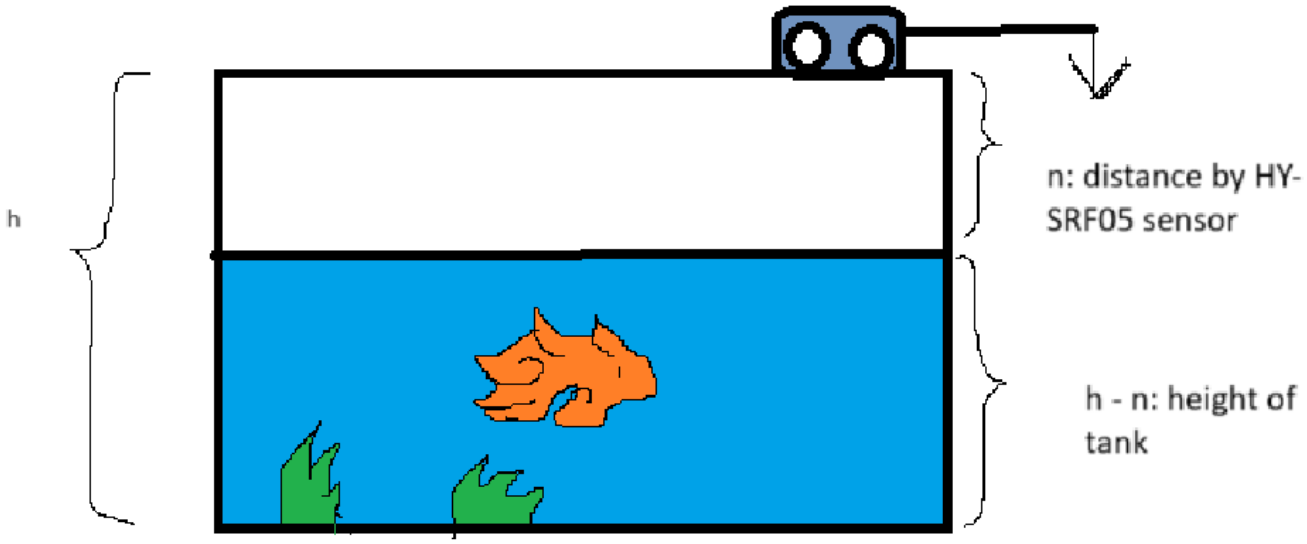
```
#define SOUND_SPEED 0.034
```

3.3.3.1. Công thức

$$Khoảng\ cách = \frac{Thời\ gian\ phản\ hồi\ (micro\ giây) * tốc\ độ\ âm\ thanh\ (343\ m/s)}{2}$$

**Chú thích: chia 2 vì có sóng đi và sóng về.*

3.3.3.2. Triển khai chương trình



```
// Sets the trigPin on HIGH state for 10 microseconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH, 23500);  // Timeout to avoid lockups

// Nếu duration là 0, nghĩa là không nhận được tín hiệu Echo
if (duration > 0) {
  // Calculate the distance in cm
  distanceCm = duration * SOUND_SPEED / 2;
  h_of_lake=50-distanceCm;

  // Print the water level to the Serial Monitor
  Serial.print("heigh of lake (cm): ");
  Serial.println(h_of_lake);
```

- Giả sử chiều cao là 50 cm

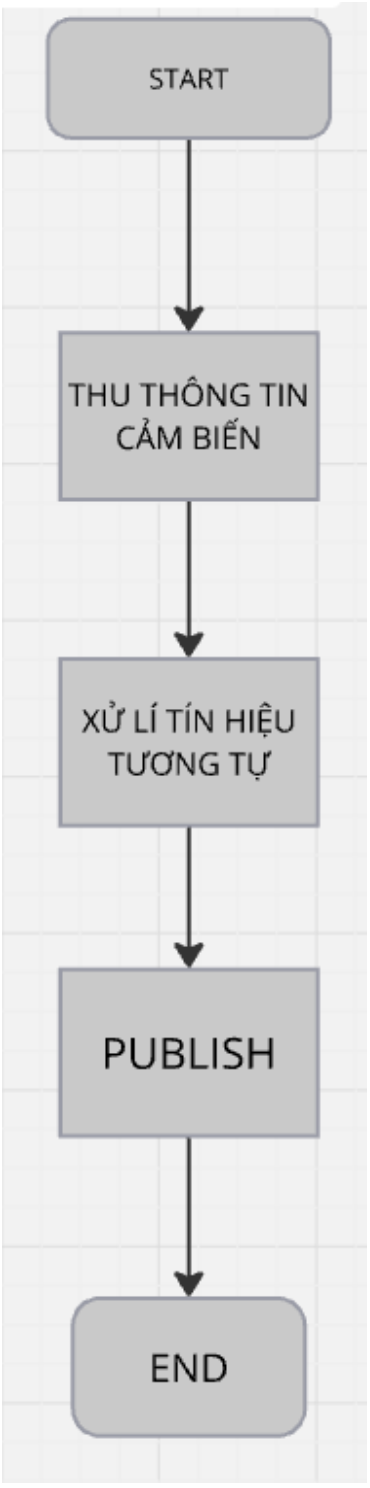
3.3.4. Đánh giá kết quả

Hình 18: Khoảng cách từ cảm biến đến bề mặt chất lỏng: 5cm

```
heigh of lake (cm): 4.95
heigh of lake (cm): 4.95
heigh of lake (cm): 5.01
heigh of lake (cm): 4.95
heigh of lake (cm): 4.90
```

Monitor serial hiển thị

3.4. Thuật toán trên Esp32 thứ ba



Hình 19. Lưu đồ giải thuật

3.4.1. Khai báo thư viện, kết nối wifi và broker MQTT

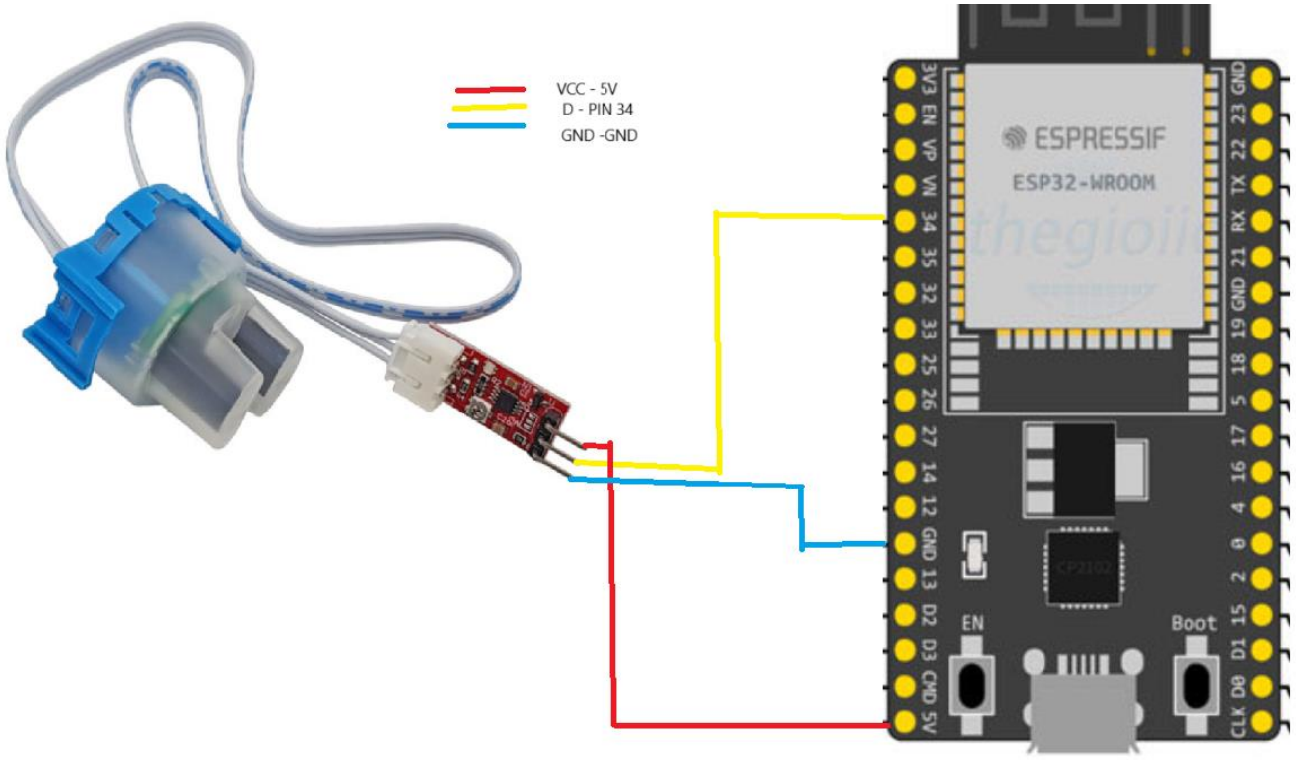
```
#include <WiFi.h>
#include <PubSubClient.h>


const int sensorPin =34;
const char* ssid = "P417";      // Your WiFi SSID
const char* password = "00006666"; // Your WiFi Password
const char* mqtt_server = "test.mosquitto.org";  // MQTT broker server
```

```
setup_wifi();  // Connect to WiFi
clientmqtt.setServer(mqtt_server, 1883);  // Set the MQTT server and port
```

```
  String data = String(sensorValue);
  clientmqtt.publish("topic/sen4", data.c_str());
```

Gửi lên Topic/sen4 của Raspberry Pi

3.4.2. Khởi tạo port chân out/in



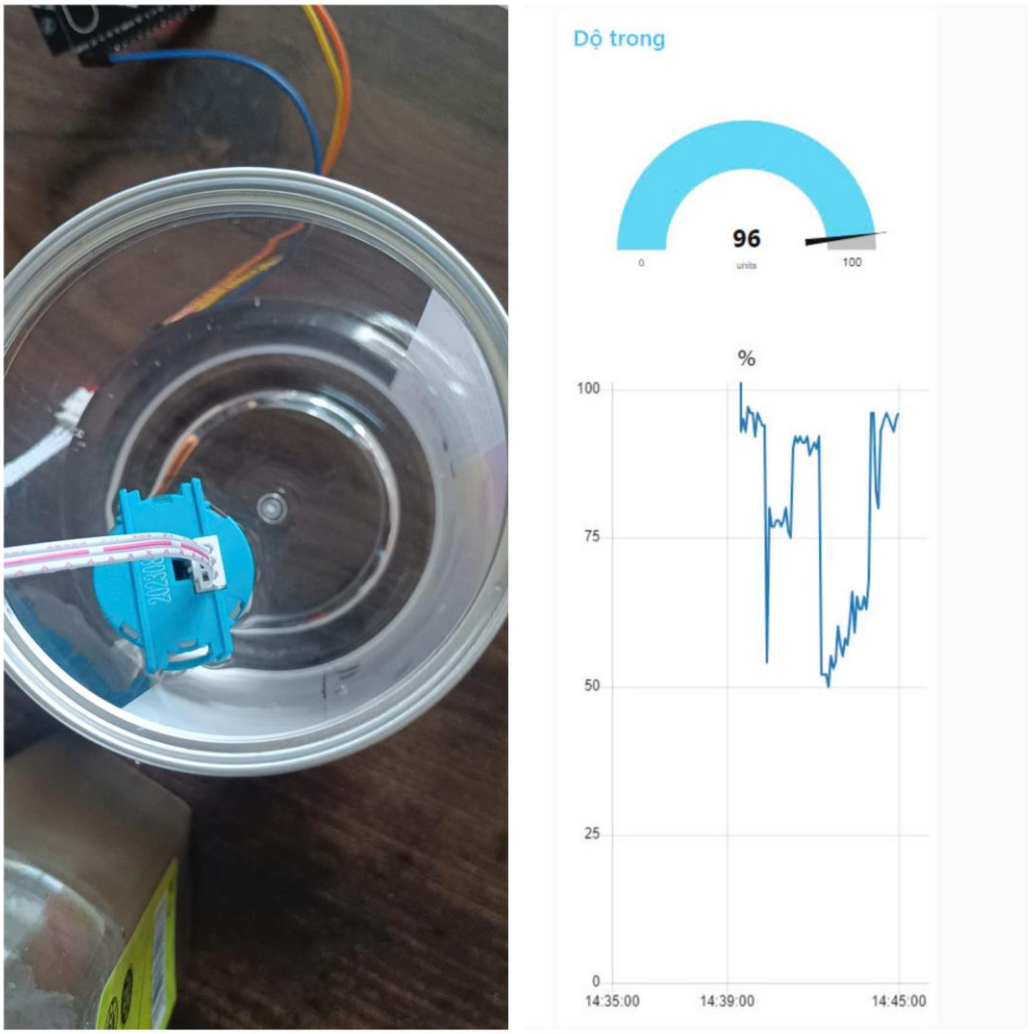
Hình 20. Sơ đồ kết nối phần cứng

3.4.3. Thuật toán để thu được giá trị từ cảm biến

```
int sensorValue = analogRead(sensorPin);  // Đọc giá trị từ cảm biến (0 - 4095)

// Chuyển đổi giá trị ADC sang tỷ lệ 0-100% để dễ hiểu
float turbidityPercent = sensorValue*100/2200; // 100% là nước trong, 0% là rất đục, 2200 là ngưỡng giá trị cao nhất
```

3.4.4. Đánh giá kết quả trả về từ cảm biến

- Độ trong của nước ở mức sạch (>90%)

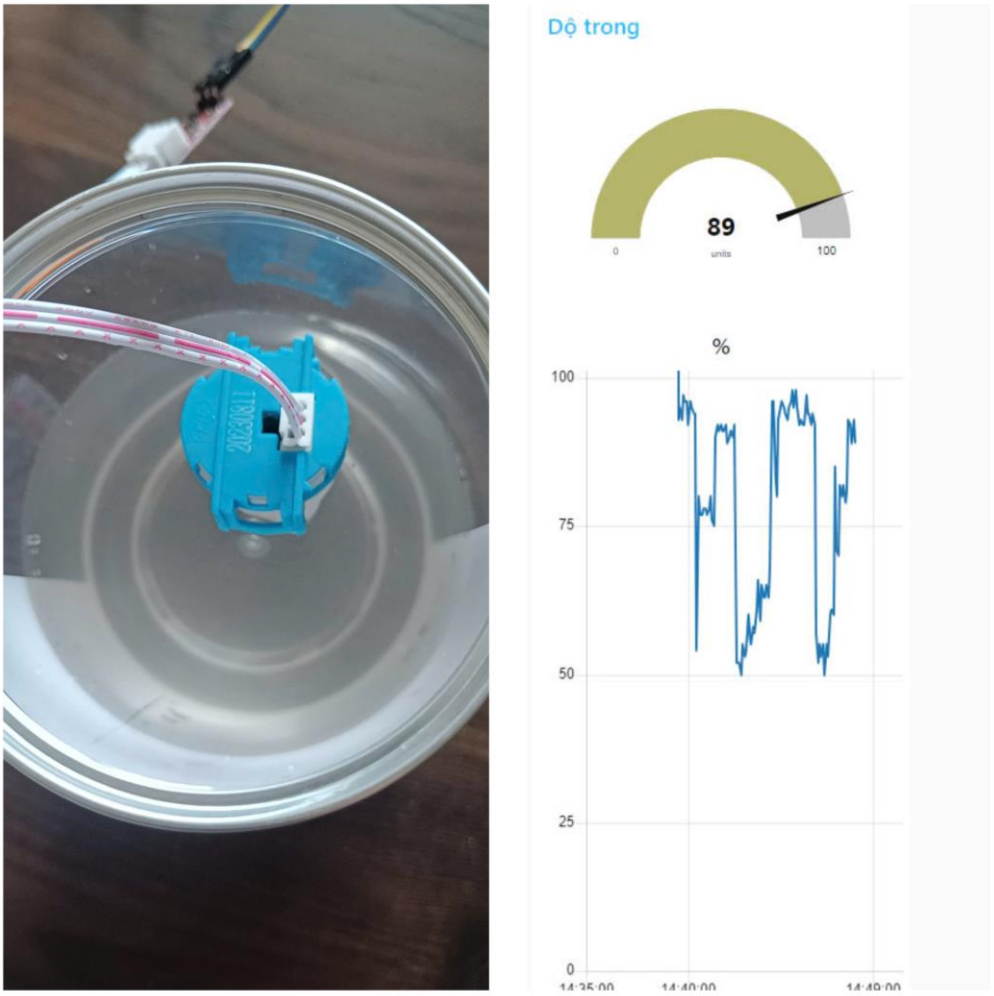


Hình 21. Khảo sát độ trong của nước ở mức sạch

Kết quả trả về: 96%

Kết luận: hoạt động đúng

- Độ trong của nước ở mức trung bình (>75%)

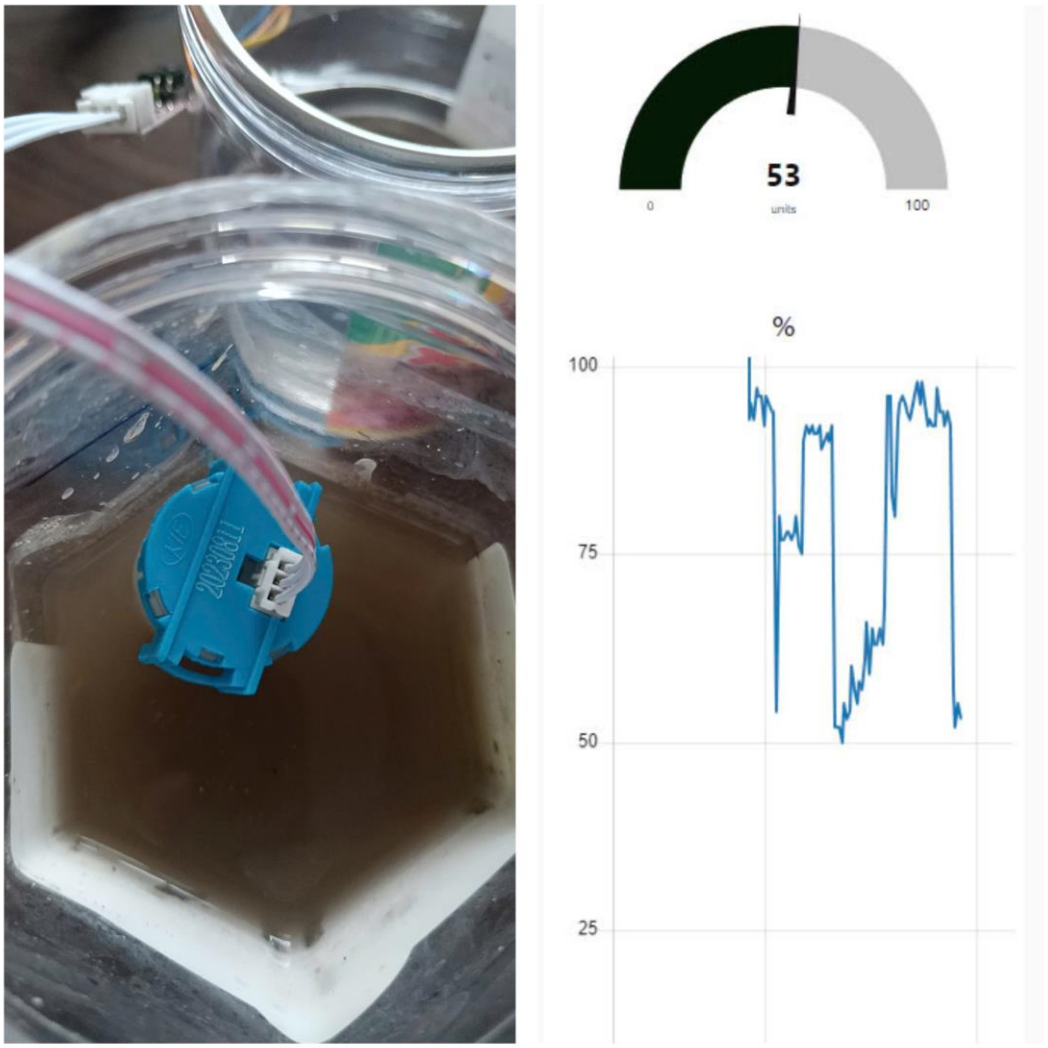


Hình 22. Khảo sát độ trong của nước ở mức trung bình.

Kết quả trả về: 89%

Kết luận: hoạt động đúng

- Độ trong của nước ở mức thấp (<75%)



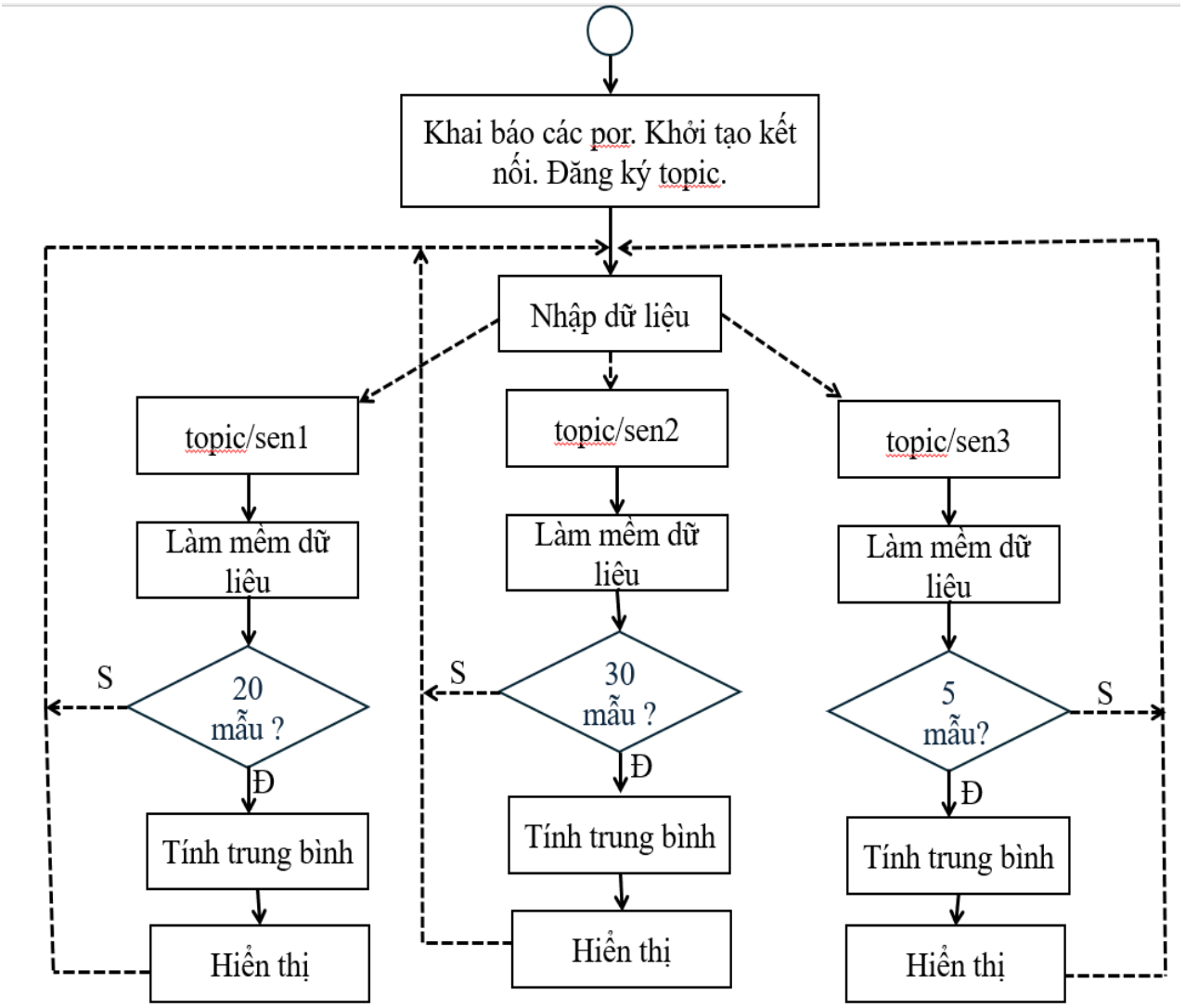
Hình 23. Khảo sát độ trong của nước ở mức thấp

Kết quả trả về: 53%

Kết luận: hoạt động đúng

3.5. Thiết lập Raspberry Pi

3.5.1 Lưu đồ giải thuật



Hình 24: Thuật toán xử lý trên Raspberry Pi

3.5.2 Xử lý dữ liệu thô từ các thiết bị biên Esp32

3.5.2.1 Phân tích cách làm

Đầu tiên, làm mịn giá trị đầu vào với bộ lọc Gaussian.

Sau đó, tính giá trị trung bình cộng (lọc trung bình) từ các dữ liệu đã làm mịn.

3.5.2.2 Bộ lọc Gaussian (Gaussian Filter)

Bộ lọc Gaussian là một loại bộ lọc thông thấp (low-pass filter), có tác dụng loại bỏ các thành phần tần số cao trong tín hiệu (những thay đổi đột ngột hoặc nhiễu), giúp làm mịn dữ liệu. Nó được dựa trên hàm Gaussian, một hàm phân bố xác suất nổi tiếng trong xác suất và thống kê.

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

Trong đó:

- G(x) là giá trị của hàm tại điểm x.
- σ(sigma) là độ lệch chuẩn, kiểm soát độ rộng của phân bố Gaussian (hay mức độ làm mịn).
- e là số mũ Euler, và π là hằng số Pi.

Đặc điểm của bộ lọc Gaussian:

- Làm mịn dữ liệu: Bộ lọc Gaussian giúp giảm thiểu các biến đổi đột ngột.
- Kiểm soát độ mịn bằng sigma: Tham số σ quyết định mức độ làm mịn. Khi σ lớn, độ mịn

3.5.2.3 Lý do việc áp dụng 2 bộ lọc

Thứ nhất, giảm nhiễu tốt hơn:

- Bộ lọc Gaussian: Giúp làm mịn tín hiệu bằng cách giảm ảnh hưởng của các giá trị sai lệch, làm cho tín hiệu trở nên mượt mà hơn.
- Khi kết hợp với bộ lọc trung bình, tận dụng tối đa công dụng.

Thứ hai, cải thiện độ chính xác với số mẫu ít:

Việc kết hợp hai phương pháp lọc có thể cải thiện độ chính xác của giá trị trung bình tính toán. Bộ lọc Gaussian giúp làm mịn dữ liệu trước khi tính toán trung bình, làm giảm tác động của nhiễu.

Thứ ba, giảm số lượng mẫu phải lấy:

Bộ lọc Gaussian làm mịn dữ liệu trước khi qua lọc trung bình giúp việc lấy mẫu giảm đi, không cần phải lấy quá nhiều mẫu để xử lý.

Thứ tư, tối ưu hóa quá Trình xử lý dữ liệu:

Việc áp dụng hai bộ lọc sẽ tối ưu hóa quá trình xử lý dữ liệu, giúp nâng cao hiệu suất của các thuật toán sau này. Giảm thời gian phản hồi.

3.5.2.3 Lựa chọn số mẫu cần lấy

- Nhiệt độ: 20 mẫu. Bởi vì nhiệt độ trong môi trường thay đổi không quá nhanh. Esp32 truyền tham số này sau mỗi 15s. Việc lấy 20 là vừa đủ để đưa ra phản hồi. Việc tính toán như vậy mất 5 phút để đưa kết quả cuối cùng.
- Độ đục của nước: 40 mẫu. Bởi vì độ đục trong của bể cá thay đổi rất chậm nhưng rất dễ bị ảnh hưởng bởi các yếu tố ngẫu nhiên. Esp32 truyền tham số này sau

mỗi 15s. Việc lấy 40 là đủ để loại bỏ phần nào sự cản trở của các yếu tố ấy. Đồng thời việc cập nhật là đủ nhanh để kịp thời xử lý nguồn nước nuôi cá. Việc phản hồi sẽ mất 10 phút mỗi lần.

- Khoảng cách: 5 mẫu. Mục đích của cảm biến là để đo mực nước trong bể cá. Lỡ có 1 trường hợp bất thường xảy ra (bể vỡ, tràn,..) thì cần phản hồi nhanh. Esp32 sẽ gửi sau mỗi 2s. Số mẫu lấy là 5. Thời gian phản hồi là 10 giây. Tốc độ đủ để xử lý các ngoại lệ bất thường.

3.5.3 Mã được sử dụng trên Raspberry Pi và phân tích:

- Khởi tạo các ip, port kết nối, tên topic; biến giá trị:

```
hive = "test.mosquitto.org"

node_red = "localhost"

port = 1883

topic1 = "topic/sen1" //chủ đề nhiệt độ

topic2 = "topic/sen2" //chủ đề mực nước

topic3 = "topic/sen3" //chủ đề độ đục trong nước

values1 = []

values2 = []

values3 = []

v1 = 0

v2 = 0

v3 = 0
```

- Khởi tạo client, kết nối, đăng ký topic:

```
hive_client = mqtt.Client()

hive_client.on_message = on_mes_hive

node_red_client = mqtt.Client()

hive_client.connect(hive, port, 60)

node_red_client.connect(node_red, port, 60)

hive_client.subscribe(topic1)
```

```
hive_client.subscribe(topic2)
hive_client.subscribe(topic3)
hive_client.loop_start() //Thiết lập vòng lặp nhận tin nhắn
```

- Hàm callback khi nhận được tin nhắn từ Esp32:

```
def on_mes_hive (client, userdata, message):
global values1, values2, values3
global v1, v2, v3
if message.topic == topic1:
value = float(message.payload.decode())
values1.append(value)
print(f"Nhan tu {topic1}: {value}")
if len(values1) >= 20:
 value1_chuan = gaussian_filter1d(values1, sigma=1)
 v1 = sum(value1_chuan) / len(value1_chuan)
 v1 = round(v1, 1)
 print(f"Trung binh {topic1}: {v1}")
 values1.clear()
elif message.topic == topic2:
value = float(message.payload.decode())
values2.append(value)
print(f"Nhan tu {topic2}: {value}")
if len(values2) >= 5:
 value2_chuan = gaussian_filter1d(values2, sigma=1)
 v2 = sum(value2_chuan) / len(value2_chuan)
 v2 = round(v2, 1)
```

```
print(f"Trung binh {topic2}: {v2}")
values2.clear()
if len(values3) >= 40:
value3_chuan = gaussian_filter1d(values3, sigma=1)
v3 = sum(value3_chuan) / len(value3_chuan)
v3=(v3/2200)*1000
v3 = round ( v3,1)
print(f"Trung binh {topic3}: {v3}")
values3.clear()
```

- Gửi tin nhắn đến NodeRed để hiển thị, ngắt khi lỗi, kết thúc:

```
try:
while True:
node_red_client.publish(topic1, str(v1))
node_red_client.publish(topic2, str(v2))
node_red_client.publish(topic3, str(v3))
sleep(5)  //Vòng lặp gửi tin để tránh timeout trên NodeRed
except KeyboardInterrupt:
print("Stopping...")
finally:
hive_client.loop_stop()
```

3.5.4 Giao diện NodeRed chạy trên Raspberry

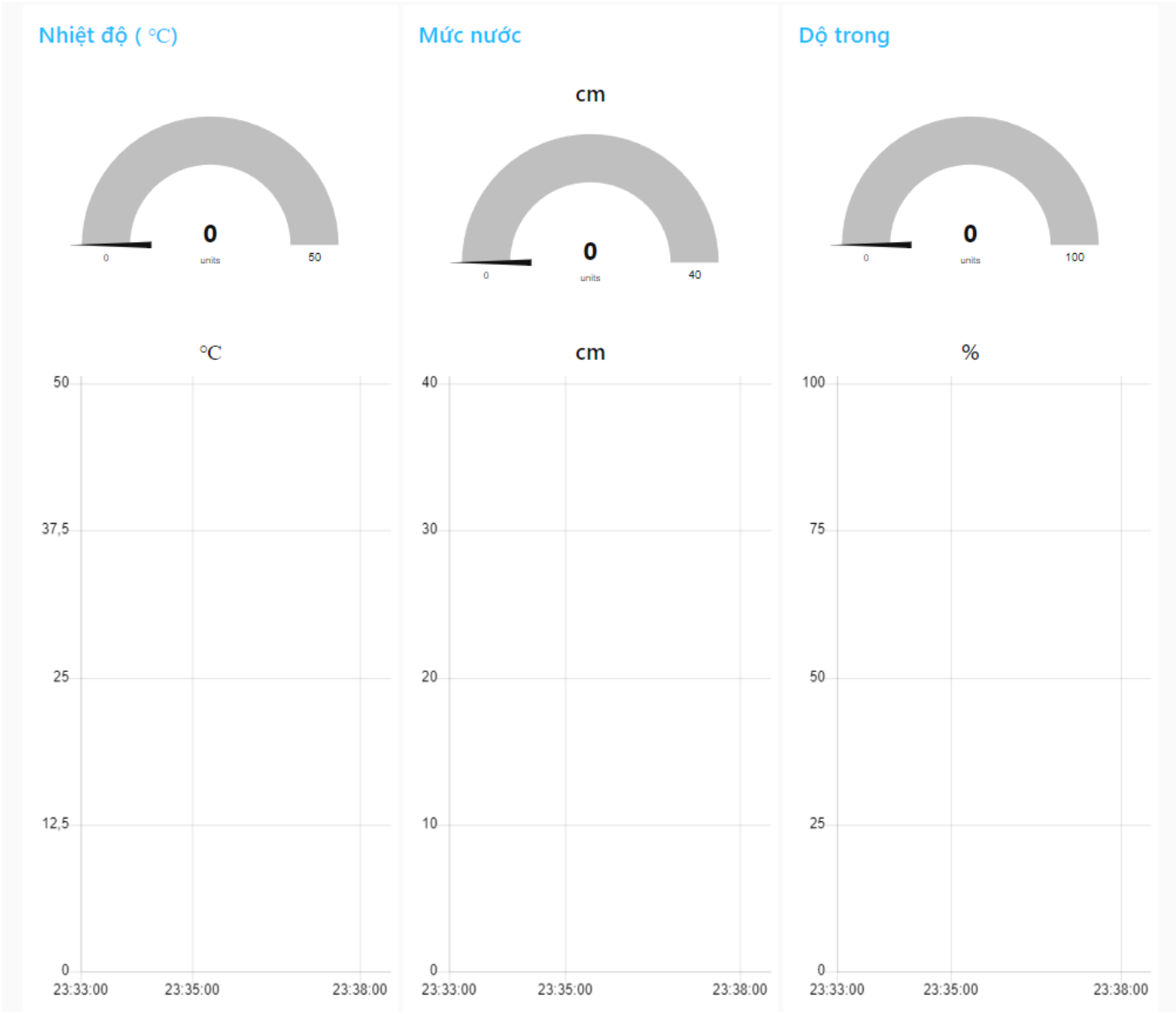
Từ trái qua phải lần lượt hiển thị: nhiệt độ (°C), mức nước (cm), độ trong (%)

Bên trên là biểu đồ dạng gauge: biểu thị mức độ ở thời điểm hiện tại

Bên dưới là đồ thị Oxy: biểu thị giá trị mức độ theo thời gian.

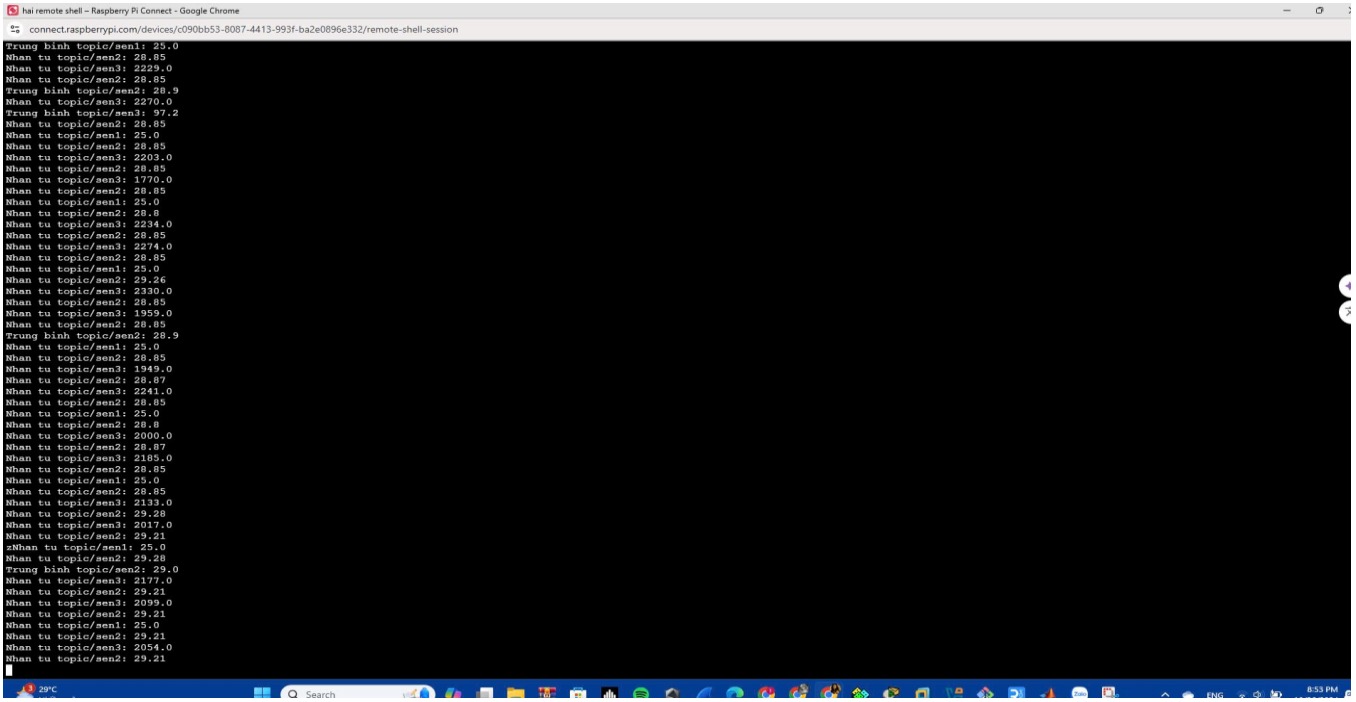
Trục Ox: biên độ.

Trục Oy: thời gian (trong 1 ngày)

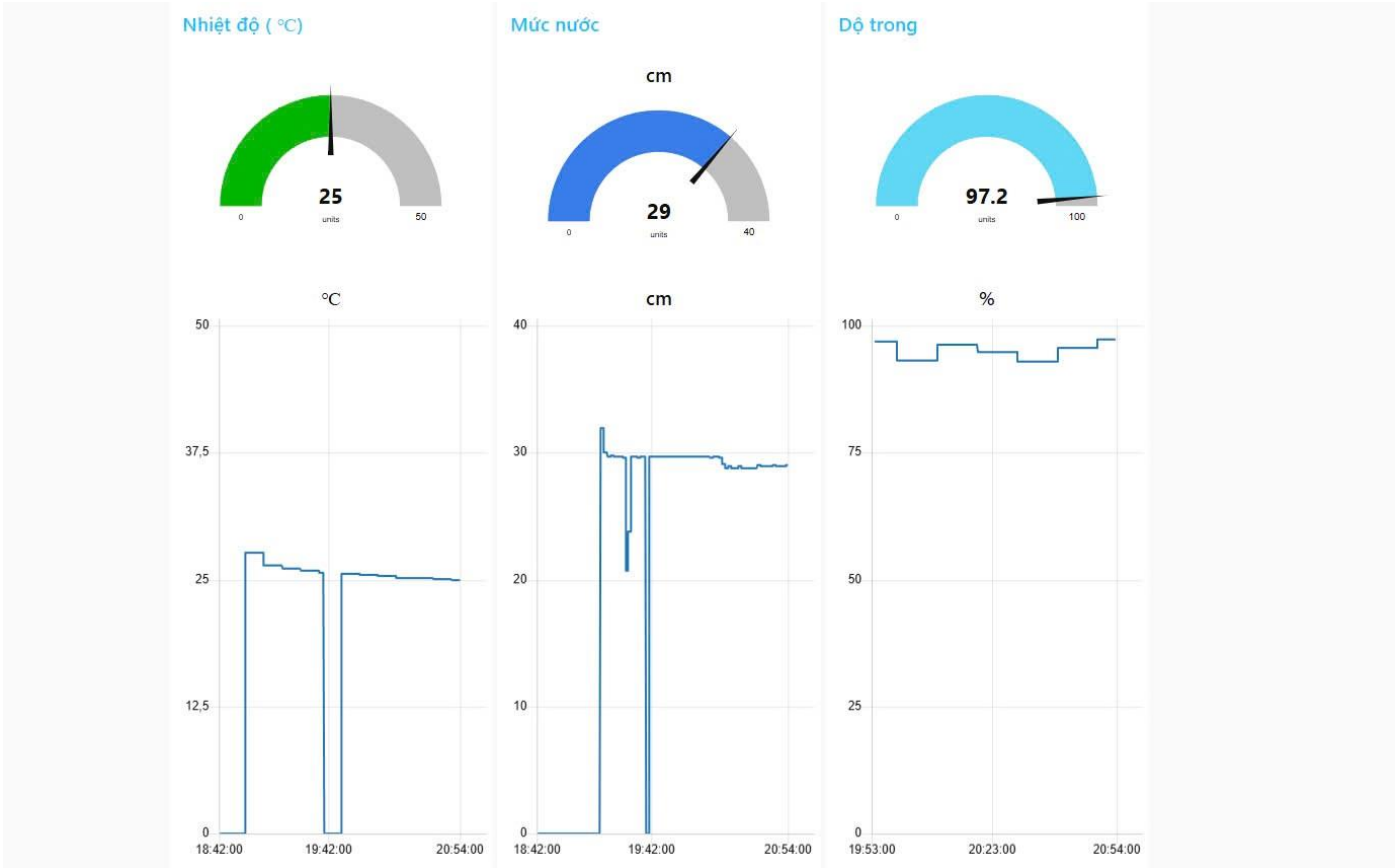


Hình 25. Website hiển thị thông số nhiệt độ, mức nước, độ trong

PHẦN C: KẾT QUẢ HỆ THỐNG



Hình 26. Kết quả hiển thị Terminal để so sánh



Hình 27. Kết quả hiển thị trên website

1. Nhận xét

- Quá trình nhận dữ liệu từ cảm biến hoạt động ổn định.
- Kết quả từ 2 lần kiểm nghiệm không sai lệch quá nhiều.
- Xét kết quả nhiệt độ (biểu đồ, đồ thị ngoài cùng bên trái), kết quả trả về từ bộ lọc, xử lý dữ liệu tương đối chính xác.
- Xét kết quả mực nước (biểu đồ, đồ thị ở giữa), dữ liệu trả về từ bộ lọc, xử lý đưa ra dữ liệu gần như chính xác. Có 1 vết lõm ở giữa đồ thị là kết quả của việc thử nghiệm tình huống, thời gian phản hồi đủ nhanh như đã tính toán.
- Xét kết quả độ trong của nước (biểu đồ, đồ thị ngoài cùng bên phải), kết quả có hơi chút sai số nhưng vẫn thể hiện được việc nước đang trong khớp với điều kiện nước đang thí nghiệm.

2. Kết luận

- Hệ thống hoạt động ổn định, không xuất hiện hiện tượng bất đồng bộ, mất mát dữ liệu. Đáp ứng được những yêu cầu đặt ra.
- Hiệu quả bộ lọc là tương đối cao, nhưng chưa quá chuẩn so với dự kiến.

TÀI LIỆU THAM KHẢO

[1] Nshop, https://nshopvn.com/product/cam-bien-do-do-duc-cua-nuoc/ , 21/10/2024.

[2] ThegioiIC, https://www.thegioiic.com/mjkdz-cam-bien-do-do-duc-cua-nuoc-0-1000ntu-3-3-5vdc , 21/10/2024.

[3] Mlab embedded solutions, https://mlab.vn/2292196-cam-bien-do-do-duc-chat-long.html , 21/10/2024.

[4] RANDOM NERD TUTORIALS, ESP32 with HC-SR04 Ultrasonic Sensor with Arduino IDE | Random Nerd Tutorials, 20/10/2024.

[5] Mlab embedded solution, https://mlab.vn/20736-huong-dan-su-dung-module-cam-bien-sieu-am-hy-srf05.html, 20/10/2024.

[6] Phân tích làm mịn dữ liệu bằng Gaussian

https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm, 20/10/2024.

[7] Tìm hiểu về DHT22, https://components101.com/sensors/dht22-pinout-specs-datasheet,

21/10/2024.

[8] Tìm hiểu về DHT22 truyền lên MQTT, https://www.iotzone.vn/esp32/mqtt-arduino-esp32/, 18/10/2024.