ĐẠI HỌC HUẾ
TRƯỜNG ĐẠI HỌC KHOA HỌC
KHOA CÔNG NGHỆ THÔNG TIN

HỆ THỐNG ĐO MỨC NƯỚC THÔNG MINH BẰNG ESP32

TÊN LỚP HỌC PHẦN: PHÁT TRIỂN ỨNG DỤNG IOT
MÃ HỌC PHẦN: TIN4024.005

GIẢNG VIÊN HƯỚNG DẪN: VÕ VIỆT DŨNG

HUẾ, THÁNG 4 NĂM 2025

DANH MỤC CÁC TỪ VIẾT TẮT

Tiếng Anh:

SMS: Short Message Service

GPIO: General Purpose Input/Output

MQTT: Message Queuing Telemetry Transport

HTTP: Hypertext Transfer Protocol

IP: Internet Protocol

MỤC LỤC

PHẦN MỞ ĐẦU

Trong bối cảnh đô thị hóa và công nghiệp hóa ngày càng phát triển, nhu cầu quản lý nguồn nước một cách hiệu quả trở thành một vấn đề cấp thiết. Tình trạng lãng phí nước, rò rỉ nước hoặc thiếu nước trong các bể chứa có thể gây ra nhiều hậu quả nghiêm trọng, từ ảnh hưởng đến sinh hoạt cá nhân đến việc suy giảm hiệu suất trong sản xuất và nông nghiệp. Việc ứng dụng công nghệ IoT để giám sát và điều khiển mức nước trong bể chứa một cách tự động là một giải pháp tối ưu giúp khắc phục những vấn đề trên.

Cùng với sự phát triển mạnh mẽ của các vi điều khiển hỗ trợ kết nối không dây như ESP32 và các cảm biến đo khoảng cách như HC-SR04, việc xây dựng một hệ thống đo mức nước thông minh trở nên khả thi hơn bao giờ hết. Hệ thống này không chỉ giúp người dùng theo dõi mực nước theo thời gian thực mà còn có thể đưa ra cảnh báo khi mực nước đạt đến ngưỡng nguy hiểm và tự động kích hoạt bơm nước khi cần thiết.

Hệ thống đo mức nước thông minh bằng ESP32 không chỉ mang ý nghĩa khoa học trong việc nghiên cứu và ứng dụng công nghệ IoT vào đời sống mà còn có tính thực tiễn cao khi được triển khai trong nhiều lĩnh vực.

Về mặt khoa học: Hệ thống này ứng dụng nguyên lý đo khoảng cách bằng sóng siêu âm, xử lý tín hiệu thời gian thực và truyền dữ liệu lên đám mây để giám sát từ xa. Việc nghiên cứu, triển khai và tối ưu hệ thống giúp nâng cao hiểu biết về công nghệ vi điều khiển, lập trình nhúng và xử lý tín hiệu cảm biến.

Về mặt thực tiễn : Hệ thống giúp tiết kiệm nước, giảm công sức giám sát thủ công và ngăn chặn tình trạng cạn kiệt hoặc tràn nước trong bể chứa. Nó có thể áp dụng trong nhiều lĩnh vực như sinh hoạt hộ gia đình, quản lý bể chứa nước trong nông nghiệp, công nghiệp hay hệ thống phòng cháy chữa cháy.

PHẦN NỘI DUNG
HỆ THỐNG ĐO MỨC NƯỚC THÔNG MINH BẰNG ESP32

I. Mô tả hệ thống.

1. Giới thiệu về hệ thống đo mức nước thông minh.

Trong bối cảnh toàn cầu ngày càng chú trọng đến quản lý tài nguyên một cách hiệu quả, đặc biệt là tài nguyên nước, việc giám sát và kiểm soát mức nước trong các bể chứa, hồ chứa, sông ngòi và các hệ thống thủy lợi trở nên vô cùng quan trọng. Các phương pháp đo mức nước truyền thống thường dựa trên quan sát thủ công hoặc sử dụng các thiết bị cơ học đơn giản, dẫn đến nhiều hạn chế như tốn thời gian, công sức, độ chính xác không cao và khó khăn trong việc theo dõi dữ liệu theo thời gian thực cũng như cảnh báo sớm các sự cố.

Sự phát triển mạnh mẽ của Internet of Things (IoT) đã mở ra một hướng đi mới, mang đến các giải pháp thông minh và tự động hóa trong nhiều lĩnh vực, bao gồm cả việc đo lường và quản lý mức nước. Hệ thống đo mức nước thông minh là một hệ thống tích hợp các cảm biến, vi điều khiển, mạng truyền thông và nền tảng phần mềm để tự động thu thập, xử lý, truyền tải và hiển thị dữ liệu về mức nước. Hệ thống này không chỉ cung cấp thông tin chính xác và kịp thời mà còn có khả năng cảnh báo khi mức nước vượt quá hoặc xuống dưới các ngưỡng an toàn, thậm chí có thể tự động thực hiện các hành động điều khiển như bơm hoặc xả nước.

-Các đặc điểm chính của hệ thống đo mức nước thông minh bao gồm:

- **Đo lường tự động và liên tục:** Sử dụng các cảm biến để đo mức nước một cách tự động và liên tục, loại bỏ sự cần thiết của việc đo đạc thủ công.
- **Dữ liệu thời gian thực:** Cung cấp dữ liệu mức nước theo thời gian thực, cho phép người dùng theo dõi và giám sát tình hình một cách trực quan.
- **Khả năng kết nối mạng:** Tích hợp khả năng kết nối mạng (thường là Wi-Fi, đôi khi là các công nghệ khác như LoRaWAN, NB-IoT) để truyền dữ liệu đến các hệ thống trung tâm hoặc nền tảng đám mây.
- **Giám sát và điều khiển từ xa:** Cho phép người dùng giám sát mức nước và có thể điều khiển các thiết bị liên quan (ví dụ: bơm, van) từ bất kỳ đâu có kết nối internet.

- **Cảnh báo thông minh:** Thiết lập các ngưỡng cảnh báo (mức nước quá cao hoặc quá thấp) và tự động thông báo cho người dùng thông qua các kênh khác nhau (ví dụ: ứng dụng di động, email, tin nhắn SMS).
- **Lưu trữ và phân tích dữ liệu:** Dữ liệu mức nước được lưu trữ trên nền tảng đám mây, cho phép người dùng xem lại lịch sử, phân tích xu hướng và đưa ra các quyết định quản lý hiệu quả hơn.
- **Khả năng tích hợp:** Có khả năng tích hợp với các hệ thống quản lý khác, tạo thành một giải pháp toàn diện hơn.

-Các ứng dụng phổ biến của hệ thống đo mức nước thông minh trong đời sống và công nghiệp:

- **Quản lý nước sinh hoạt:** Giám sát mức nước trong các bể chứa nước tại hộ gia đình, khu dân cư, tòa nhà, đảm bảo nguồn cung cấp nước ổn định và phát hiện sớm các sự cố tràn hoặc cạn nước.
- **Nông nghiệp:** Theo dõi mức nước trong các kênh tưới tiêu, ao hồ, giúp quản lý việc tưới tiêu hiệu quả, tiết kiệm nước và tối ưu hóa năng suất cây trồng.
- **Công nghiệp:** Giám sát mức chất lỏng trong các bồn chứa hóa chất, nhiên liệu, nước thải, đảm bảo an toàn vận hành và quản lý quy trình sản xuất.
- **Quản lý tài nguyên nước:** Theo dõi mực nước sông, hồ, đập, phục vụ công tác dự báo lũ lụt, quản lý nguồn nước và điều tiết dòng chảy.
- **Hệ thống xử lý nước thải:** Giám sát mức nước trong các bể chứa và giai đoạn xử lý, đảm bảo hệ thống hoạt động ổn định và hiệu quả.
- **Các ứng dụng đặc biệt:** Giám sát mực nước biển, cảnh báo sóng thần, theo dõi mực nước trong các khu vực bị ngập lụt, ...v.v.

2. Nền tảng đám mây cho IOT.

-Giới thiệu: Nền tảng đám mây cho IoT cung cấp một cơ sở hạ tầng mạnh mẽ và linh hoạt để kết nối, quản lý, lưu trữ và phân tích dữ liệu từ các thiết bị IoT. Chúng cung cấp các dịch vụ và công cụ giúp đơn giản hóa quá trình phát triển và triển khai các ứng dụng IoT.

-Các nền tảng đám mây phổ biến:

Có rất nhiều nền tảng đám mây IoT khác nhau, mỗi nền tảng có những ưu và nhược điểm riêng.

Một số nền tảng phổ biến bao gồm:

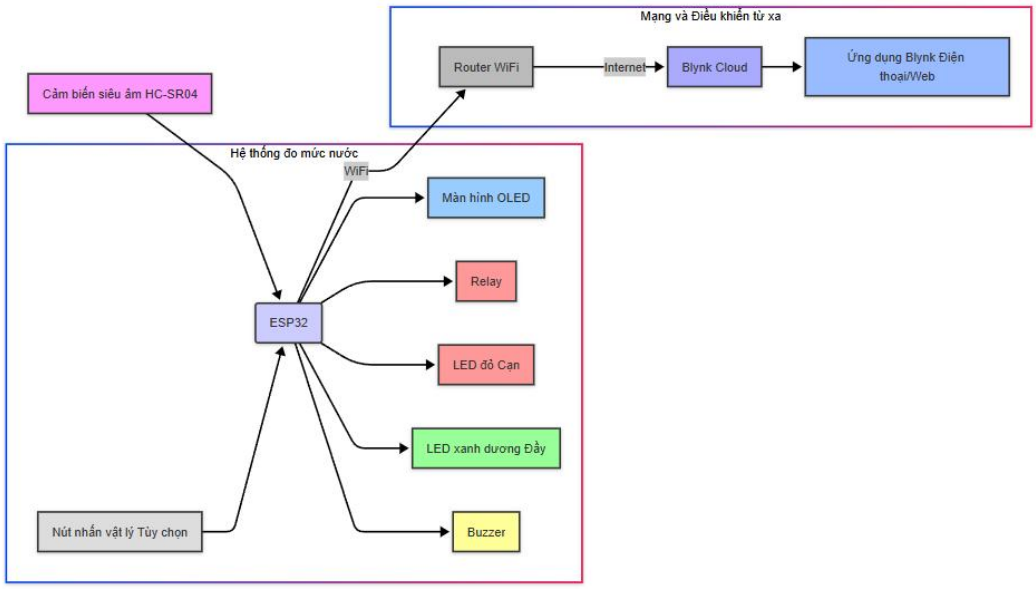
- **Blynk:** Một nền tảng dễ sử dụng, tập trung vào việc tạo giao diện người dùng trực quan để điều khiển và giám sát các thiết bị IoT thông qua ứng dụng di động.
- **Firebase (Google):** Một nền tảng phát triển ứng dụng toàn diện, cung cấp các dịch vụ như cơ sở dữ liệu thời gian thực, lưu trữ đám mây, xác thực người dùng và hosting, có thể được sử dụng để xây dựng ứng dụng giám sát IoT.
- **Thingspeak (MathWorks):** Một nền tảng đám mây mã nguồn mở, miễn phí cho mục đích phi thương mại, được thiết kế đặc biệt để thu thập, trực quan hóa và phân tích dữ liệu cảm biến.
- **AWS IoT Core (Amazon Web Services):** Một dịch vụ đám mây mạnh mẽ và có khả năng mở rộng cao, cung cấp đầy đủ các tính năng để quản lý và bảo mật các thiết bị IoT.
- **Google Cloud IoT Platform (Google Cloud Platform):** Tương tự như AWS IoT Core, cung cấp một bộ công cụ toàn diện cho việc xây dựng và quản lý các giải pháp IoT trên nền tảng Google Cloud.
- **Azure IoT Hub (Microsoft Azure):** Nền tảng IoT của Microsoft Azure, cung cấp các dịch vụ kết nối, quản lý và bảo mật thiết bị.

-Các chức năng chính của nền tảng đám mây trong ứng dụng IoT:

- **Kết nối và quản lý thiết bị:** Cung cấp các cơ chế để thiết bị IoT kết nối an toàn với đám mây và cho phép người dùng quản lý số lượng lớn thiết bị.
- **Thu thập và lưu trữ dữ liệu:** Nhận dữ liệu từ các thiết bị IoT và lưu trữ chúng trong cơ sở dữ liệu đám mây có khả năng mở rộng.
- **Trực quan hóa dữ liệu:** Cung cấp các công cụ và dịch vụ để tạo biểu đồ, đồ thị và dashboard trực quan, giúp người dùng dễ dàng theo dõi và hiểu dữ liệu.
- **Phân tích dữ liệu:** Cung cấp các dịch vụ phân tích dữ liệu, bao gồm cả phân tích thời gian thực và phân tích lịch sử, giúp người dùng trích xuất thông tin giá trị từ dữ liệu IoT.
- **Điều khiển từ xa:** Cho phép người dùng gửi lệnh từ đám mây đến các thiết bị IoT để điều khiển hoạt động của chúng.
- **Cảnh báo và thông báo:** Thiết lập các quy tắc cảnh báo dựa trên dữ liệu cảm biến và tự động gửi thông báo cho người dùng khi có sự kiện bất thường xảy ra.

- **Bảo mật:** Cung cấp các cơ chế bảo mật để đảm bảo tính toàn vẹn và bảo mật của dữ liệu và giao tiếp giữa thiết bị và đám mây.

II. Thiết kế và triển khai hệ thống.



Sơ đồ khối hệ thống đo mực nước thông minh bằng ESP32

1. Thành phần phần cứng.

-Hệ thống đo mức nước thông minh được xây dựng dựa trên các thành phần phần cứng chính sau:

1.1. Vi điều khiển ESP32.

Mô tả: ESP32 là bộ xử lý trung tâm của hệ thống, chịu trách nhiệm điều khiển toàn bộ hoạt động, bao gồm giao tiếp với cảm biến, xử lý dữ liệu, điều khiển các thiết bị ngoại vi và kết nối Wi-Fi.

Thông số kỹ thuật chính:

-Vi xử lý lõi kép Tensilica LX6 240 MHz.

-Bộ nhớ Flash 4 MB.

-Bộ nhớ SRAM 520 KB.

-Tích hợp Wi-Fi 802.11 b/g/n HT40.

-Tích hợp Bluetooth v4.2 BR/EDR và BLE.

-Số lượng chân GPIO: 32.

-Điện áp hoạt động: 3.3V.

Lý do lựa chọn: ESP32 được lựa chọn vì khả năng tích hợp Wi-Fi và Bluetooth, hiệu năng xử lý mạnh mẽ, số lượng chân GPIO dồi dào, chi phí hợp lý và sự hỗ trợ

rộng rãi từ cộng đồng phát triển, tạo điều kiện thuận lợi cho việc xây dựng và mở rộng dự án IoT.

1.2. Cảm biến siêu âm HC-SR04.

Hình ảnh cảm biến siêu âm HC-SR04

Mô tả: HC-SR04 là cảm biến khoảng cách không tiếp xúc, sử dụng sóng siêu âm để đo khoảng cách từ cảm biến đến bề mặt nước.

Thông số kỹ thuật chính:

-Điện áp hoạt động: 5V DC (trong mô phỏng sử dụng 3.3V).

-Dòng điện hoạt động: < 15mA.

-Tần số siêu âm: 40 kHz.

-Tầm đo: 2 cm - 400 cm.

-Độ phân giải: 0.3 cm.

-Góc quét hiệu quả: < 15 độ.

Lý do lựa chọn: HC-SR04 có chi phí thấp, dễ dàng giao tiếp với vi điều khiển và cung cấp khả năng đo khoảng cách phù hợp cho việc giám sát mức nước trong nhiều loại bể chứa.

1.3. Nguồn điện.

-Mô tả: Hệ thống sử dụng nguồn điện 5V DC để cấp nguồn cho ESP32 (thông qua cổng USB hoặc chân VIN) và các thành phần 5V khác như relay và buzzer. ESP32 tự điều chỉnh xuống 3.3V cho các hoạt động nội bộ và cấp nguồn cho các thiết bị 3.3V như OLED (trong mô phỏng).

-Loại nguồn: Trong quá trình mô phỏng trên Wokwi, nguồn điện được cung cấp ảo. Trong thực tế, có thể sử dụng adapter 5V/1A hoặc nguồn pin phù hợp.

1.4. Mô-đun Wi-Fi.

-Mô tả: ESP32 tích hợp sẵn chip Wi-Fi, hỗ trợ chuẩn 802.11 b/g/n, cho phép kết nối với mạng Wi-Fi 2.4 GHz.

-Cấu hình: ESP32 sẽ được lập trình để kết nối với một mạng Wi-Fi cụ thể bằng cách cung cấp SSID và mật khẩu trong code.

1.5. Module Relay.

-Mô tả: Module relay được sử dụng như một công tắc điện tử để điều khiển việc bật/tắt máy bơm nước (tùy chọn) dựa trên tín hiệu điều khiển từ ESP32.

-Loại relay: (Cần xác định loại relay cụ thể nếu có thông tin).

-Thông số kỹ thuật: (Cần xác định thông số kỹ thuật của relay nếu có thông tin, ví dụ: điện áp và dòng điện chịu tải).

-Kết nối: Chân điều khiển (IN) của relay kết nối với GPIO4 của ESP32. Các chân COM, NO (thường mở) và NC (thường đóng) sẽ được sử dụng để kết nối với mạch điện của máy bơm nước.

1.6. Màn hình hiển thị OLED (oled1 - SSD1306 0.96 inch I2C).

-Mô tả: Màn hình OLED được sử dụng để hiển thị trực tiếp các thông tin về mức nước, bao gồm khoảng cách đo được và mức nước theo phần trăm hoặc centimet.

-Giao thức giao tiếp: I2C.

-Địa chỉ I2C: 0x3C.

-Thư viện sử dụng: Thường sử dụng thư viện Adafruit_SSD1306 và Adafruit_GFX.

1.7. Các thiết bị cảnh báo và điều khiển.

-LED đỏ (led1): Cảnh báo mức nước cao, kết nối với GPIO15 qua điện trở hạn dòng R1 (220kΩ - cần điều chỉnh giá trị).

-LED xanh dương (led2): Cảnh báo mức nước thấp, kết nối với GPIO2 qua điện trở hạn dòng R2 (220kΩ - cần điều chỉnh giá trị).

-Buzzer (bz1): Phát âm thanh cảnh báo khi mức nước quá thấp, chân dương (+) kết nối với 5V, chân âm (-) kết nối với GPIO GND.3 (mức thấp kích hoạt).

-Nút nhấn (btn1): Có thể được sử dụng để chuyển đổi chế độ hoạt động hoặc kích hoạt bơm thủ công, một chân kết nối với GND.3, chân còn lại kết nối với GPIO0.

2. Nguyên lý hoạt động của mô hình.

-Hệ thống hoạt động dựa trên sự phối hợp giữa các thành phần phần cứng và phần mềm để thực hiện việc đo lường, giám sát và cảnh báo mức nước.

-Đo khoảng cách bằng cảm biến HC-SR04: ESP32 định kỳ gửi một xung kích hoạt (10 micro giây) đến chân TRIG của HC-SR04. Cảm biến phát ra 8 xung siêu âm 40 kHz. Thời gian để sóng siêu âm phản xạ trở lại và được thu nhận bởi chân ECHO được đo. Khoảng cách đến bề mặt nước được tính toán dựa trên thời gian này và vận tốc âm thanh trong không khí.

-Xử lý dữ liệu và tính toán mức nước bằng ESP32: ESP32 nhận thời gian phản xạ từ chân ECHO, tính toán khoảng cách theo công thức:

$$d = (t * v) / 2$$

Trong đó:

d: Khoảng cách (cm): Là khoảng cách từ bề mặt của cảm biến siêu âm đến vật cản (trong trường hợp này là bề mặt nước), được tính bằng centimet.

t: Thời gian ECHO (micro giây): Là khoảng thời gian mà chân ECHO của cảm biến ở trạng thái HIGH, được đo bằng micro giây (μs). Thời gian này đại diện cho khoảng thời gian sóng siêu âm đi từ cảm biến đến vật cản và phản xạ trở lại cảm biến.

v: Vận tốc âm thanh (cm/micro giây): Là tốc độ lan truyền của sóng âm trong không khí, thường được xấp xỉ là 0.0343 centimet trên micro giây (tương đương 343 mét trên giây ở điều kiện nhiệt độ và áp suất tiêu chuẩn).

2: Hệ số chia 2 xuất hiện vì thời gian ECHO đo được là tổng thời gian sóng siêu âm đi từ cảm biến đến vật cản *và* phản xạ ngược lại. Để có được khoảng cách một chiều, chúng ta cần chia đôi thời gian này.

-Sau đó, dựa trên khoảng cách tối đa có thể đo được của bể (cần được xác định và cấu hình trong phần mềm), ESP32 sẽ tính toán mức nước theo phần trăm hoặc centimet.

-Hiển thị thông tin trên màn hình OLED: ESP32 sử dụng thư viện Adafruit_SSD1306 để hiển thị các thông tin như khoảng cách đo được và mức nước (theo % hoặc cm) lên màn hình OLED kết nối qua giao thức I2C.

-Kết nối Wi-Fi và gửi dữ liệu lên Blynk (giả định): ESP32 sẽ cố gắng kết nối với mạng Wi-Fi đã được cấu hình. Sau khi kết nối thành công, nó sẽ sử dụng thư viện Blynk (hoặc thư viện phù hợp với nền tảng đám mây bạn chọn) để gửi dữ liệu mức nước lên ứng dụng Blynk thông qua giao thức MQTT hoặc HTTP.

-Cảnh báo mức nước: ESP32 sẽ so sánh mức nước hiện tại với các ngưỡng đã được định nghĩa trước (ví dụ: mức thấp, mức cao).

- Nếu mức nước thấp hơn ngưỡng dưới, chân GPIO kết nối với LED xanh dương sẽ được kích hoạt (mức HIGH hoặc LOW tùy thuộc vào cách kết nối) và buzzer sẽ phát âm thanh cảnh báo.
- Nếu mức nước cao hơn ngưỡng trên, chân GPIO kết nối với LED đỏ sẽ được kích hoạt.

-Điều khiển Relay (tùy chọn): Dựa trên mức nước và logic điều khiển được lập trình, ESP32 có thể điều khiển chân GPIO4 để bật hoặc tắt relay, từ đó điều khiển máy bơm nước. Ví dụ, nếu mức nước xuống quá thấp, relay sẽ được bật để cấp nguồn cho máy bơm, và khi mức nước đạt đến một ngưỡng nhất định, relay sẽ tắt.

-Xử lý nút nhấn (tùy chọn): Chân GPIO0 được kết nối với nút nhấn. ESP32 có thể được lập trình để phát hiện trạng thái nhấn của nút và thực hiện các hành động tương ứng, chẳng hạn như chuyển đổi chế độ tự động/thủ công hoặc kích hoạt bơm nước thủ công.

3. Thiết kế phần cứng chi tiết.

Thành phần	**Chân**	**ESP32 Chân**	**Mục đích**	**Ghi chú**
HC-SR04	VCC	3V3	Cấp nguồn cho cảm biến	Kết nối với chân 3V3 của ESP32
HC-SR04	GND	GND.1	Nối đất cho cảm biến	Kết nối với chân GND.1 của ESP32
HC-SR04	TRIG	GPIO5	Chân kích hoạt phát sóng siêu âm	
HC-SR04	ECHO	GPIO18	Chân nhận tín hiệu phản xạ	Kết nối trực tiếp với GPIO18 của ESP32
OLED (oled1)	VCC	3V3	Cấp nguồn cho màn hình	

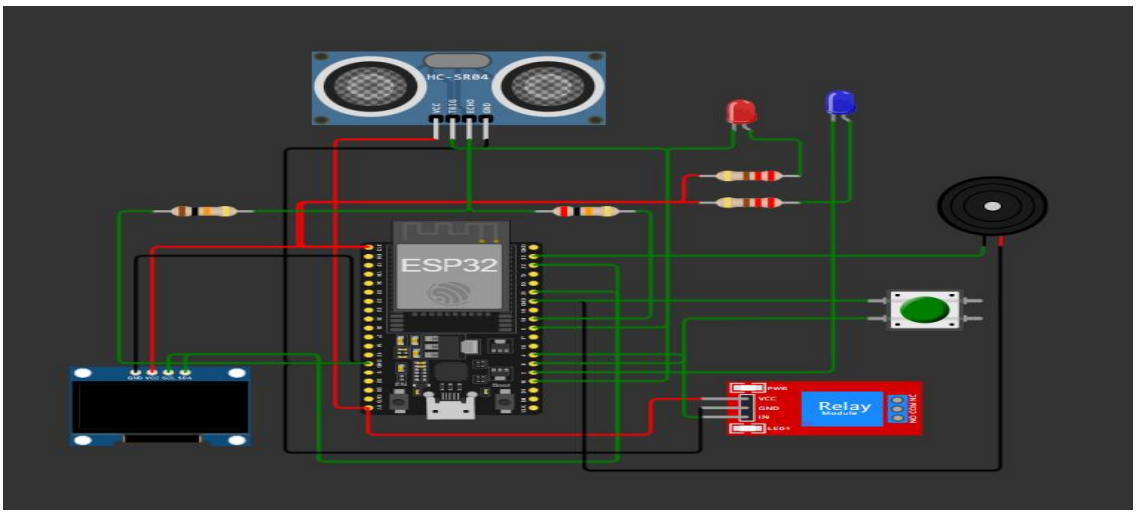
OLED (oled1)	GND	GND.1	Nối đất cho màn hình	
OLED (oled1)	SDA	GPIO21	Dữ liệu I2C	
OLED (oled1)	SCL	GPIO22	Xung Clock I2C	
Relay (relay1)	VCC	5V	Cấp nguồn cho relay	Kết nối với chân 5V của ESP32
Relay (relay1)	GND	GND.1	Nối đất cho relay	
Relay (relay1)	IN	GPIO4	Chân điều khiển relay	
LED đỏ (led1)	Anode (+)	Qua R1 (220kΩ)	Nguồn (chưa rõ ràng, có thể là 3V3 hoặc 5V)	Giá trị điện trở rất cao
LED đỏ (led1)	Cathode (-)	GPIO15	Điều khiển bật/tắt LED đỏ	
LED xanh dương (led2)	Anode (+)	Qua R2 (220kΩ)	Nguồn (chưa rõ ràng, có thể là 3V3 hoặc 5V)	Giá trị điện trở rất cao
LED xanh dương (led2)	Cathode (-)	GPIO2	Điều khiển bật/tắt LED xanh dương	
Buzzer (bz1)	+ (Chân 1)	5V	Cấp nguồn cho buzzer	
Buzzer (bz1)	- (Chân 2)	GND.3	Điều khiển bật/tắt buzzer (mức thấp kích hoạt)	
Nút nhấn	1.1	GND.3	Nối đất cho nút	

(btn1)			nhấn	
Nút nhấn (btn1)	2.1	GPIO0	Đọc trạng thái nút nhấn	Sử dụng pull-up/pull-down nội bộ hoặc bên ngoài

-Phân tích các kết nối:

- **Nguồn điện:** Cảm biến siêu âm và OLED được cấp nguồn 3.3V từ ESP32, trong khi relay và buzzer được cấp nguồn 5V. ESP32 có cả chân 3.3V và 5V, điều này phù hợp.
- **Cảm biến siêu âm:** Chân TRIG được điều khiển bởi GPIO5, chân ECHO được đọc bởi GPIO18.
- **Màn hình OLED:** Giao tiếp I2C được thiết lập thông qua các chân SDA (GPIO21) và SCL (GPIO22). Địa chỉ I2C là 0x3C.
- **Relay:** Chân điều khiển IN được kết nối với GPIO4.
- **LED:** LED đỏ và xanh dương có anode kết nối qua điện trở (cần điều chỉnh giá trị) đến nguồn (cần xác định rõ là 3.3V hay 5V) và cathode kết nối với các chân GPIO15 và GPIO2 để điều khiển.
- **Buzzer:** Buzzer được cấp nguồn 5V và chân điều khiển (âm) được kết nối với GPIO GND.3. Điều này có nghĩa buzzer sẽ kêu khi chân GPIO GND.3 được kéo xuống mức thấp.
- **Nút nhấn:** Một chân nối đất, chân còn lại nối với GPIO0. GPIO0 thường được sử dụng cho boot mode trên ESP32, nên cần cẩn thận khi sử dụng nó làm chân đầu vào thông thường.

Hình ảnh sơ đồ mạch điện:



Hình ảnh sơ đồ hệ thống đo mức nước thông minh bằng ESP32 kết hợp cảm biến siêu âm HC-SR04 để đo mực nước

4. Phát triển phần mềm

-Quá trình phát triển phần mềm cho hệ thống đo mức nước thông minh bằng ESP32 được thực hiện nhằm mục đích thu thập dữ liệu từ cảm biến siêu âm, xử lý dữ liệu để tính toán mức nước, hiển thị thông tin trên màn hình OLED, kết nối với mạng Wi-Fi và gửi dữ liệu lên nền tảng đám mây (giả định sử dụng Blynk), đồng thời điều khiển các thiết bị cảnh báo và relay dựa trên các ngưỡng mức nước.

4.1. Môi trường phát triển và thư viện sử dụng

-Môi trường phát triển: Arduino IDE được sử dụng làm môi trường chính để viết, biên dịch và tải code lên ESP32. Arduino IDE cung cấp một giao diện thân thiện và nhiều thư viện hỗ trợ cho các tác vụ khác nhau.

-Các thư viện chính đã sử dụng:

WiFi.h: Cung cấp các hàm cần thiết để ESP32 kết nối với mạng Wi-Fi.

BlynkSimpleEsp32.h: Thư viện Blynk dành riêng cho ESP32, giúp thiết lập kết nối với server Blynk và sử dụng các hàm virtualWrite và BLYNK_WRITE.

Adafruit_GFX.h: Thư viện đồ họa cơ bản của Adafruit, cần thiết cho việc điều khiển màn hình OLED.

Adafruit_SSD1306.h: Thư viện đặc biệt cho màn hình OLED dựa trên chip điều khiển SSD1306, cung cấp các hàm để hiển thị văn bản và hình ảnh.

4.2. Cấu trúc code chính

-Code chương trình cho ESP32 được tổ chức thành hai hàm chính: setup() và loop().

-setup(): Hàm này được thực thi một lần duy nhất khi ESP32 khởi động. Các tác vụ khởi tạo ban đầu được thực hiện trong hàm này bao gồm:

Khởi động giao tiếp Serial: Thiết lập tốc độ truyền dữ liệu nối tiếp ở mức 115200 baud bằng lệnh Serial.begin(115200);. Điều này cho phép ESP32 gửi thông tin gỡ lỗi và trạng thái hoạt động đến máy tính qua cổng USB.

Thiết lập kết nối Wi-Fi: Bắt đầu quá trình kết nối với mạng Wi-Fi được chỉ định bởi ssid và pass bằng hàm WiFi.begin(ssid, pass);. Chương trình sẽ tạm dừng và chờ cho đến khi kết nối Wi-Fi được thiết lập thành công, đồng thời in dấu chấm trên Serial Monitor để báo hiệu quá trình kết nối. Sau khi kết nối thành công, thông báo "WiFi Connected" và địa chỉ IP của ESP32 sẽ được in ra.

Khởi tạo kết nối Blynk: Thiết lập kết nối với máy chủ Blynk bằng cách sử dụng mã xác thực BLYNK_AUTH_TOKEN, cùng với thông tin đăng nhập mạng Wi-Fi (ssid, pass) thông qua hàm Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);.

Cấu hình chân GPIO: Các chân GPIO được sử dụng cho các thành phần phần cứng được cấu hình chế độ hoạt động bằng hàm pinMode():

Chân TRIG_PIN (GPIO5) được đặt là OUTPUT để gửi tín hiệu kích hoạt cảm biến siêu âm.

Chân ECHO_PIN (GPIO18) được đặt là INPUT để nhận tín hiệu phản xạ từ cảm biến siêu âm.

Chân RELAY_PIN (GPIO4) được đặt là OUTPUT để điều khiển relay.

Chân BUZZER_PIN (GPIO23) được đặt là OUTPUT để phát âm thanh cảnh báo.

Chân LED_RED (GPIO15) và LED_BLUE (GPIO2) được đặt là OUTPUT để điều khiển đèn LED cảnh báo.

Chân BUTTON_PIN (GPIO0) được đặt là INPUT_PULLUP. Cấu hình này kích hoạt điện trở kéo lên nội bộ, giúp chân này ở mức HIGH khi nút nhấn không được nhấn và chuyển xuống mức LOW khi nút nhấn được nhấn.

Khởi tạo màn hình OLED:

Hàm display.begin(SSD1306_SWITCHCAPVCC, 0x3C) được gọi để khởi tạo màn hình OLED SSD1306 sử dụng nguồn điện tích hợp (SSD1306_SWITCHCAPVCC) và địa chỉ I2C là 0x3C.

Một câu lệnh if (!display.begin(...)) kiểm tra xem quá trình khởi tạo có thành công hay không. Nếu không thành công, một thông báo lỗi "OLED không khởi động được"

sẽ được in ra Serial Monitor, và chương trình sẽ bị khóa trong một vòng lặp vô hạn (while (true);).

display.clearDisplay(); được gọi để xóa bất kỳ nội dung nào có thể hiển thị trên màn hình OLED sau khi khởi tạo thành công.

-loop(): Hàm này được thực thi lặp đi lặp lại sau khi hàm setup() hoàn thành, tạo thành vòng lặp chính của chương trình:

Duy trì kết nối Blynk: Blynk.run(); cho phép thư viện Blynk xử lý giao tiếp với máy chủ Blynk, quản lý kết nối và xử lý dữ liệu gửi và nhận.

Kiểm tra và kết nối lại Wi-Fi: if (WiFi.status() != WL_CONNECTED) kiểm tra trạng thái kết nối Wi-Fi. Nếu kết nối bị gián đoạn, thông báo "WiFi mất kết nối, đang kết nối lại..." sẽ được in ra, và WiFi.reconnect(); sẽ được gọi để cố gắng thiết lập lại kết nối.

Đo khoảng cách: Hàm getDistanceCm() được gọi để đo khoảng cách từ cảm biến siêu âm đến bề mặt nước, và giá trị đo được được lưu trữ trong biến distanceCm.

Tính toán mức nước:

waterLevelPercentage = calculateWaterLevelPercentage(distanceCm);: Hàm này chuyển đổi khoảng cách đo được thành phần trăm mức nước dựa trên khoảng cách tối đa (MAX_DISTANCE_CM).

waterLevelCm = calculateWaterLevelCm(distanceCm);: Hàm này tính toán mức nước theo đơn vị centimet dựa trên khoảng cách đo được và khoảng cách tối đa.

In thông tin ra Serial Monitor: Các giá trị của khoảng cách (distanceCm) và mức nước (waterLevelPercentage, waterLevelCm) được in ra Serial Monitor để theo dõi và gỡ lỗi.

Hiển thị trên OLED: Hàm displayOled() được gọi để hiển thị các thông tin về mức nước lên màn hình OLED.

Cập nhật Blynk: Hàm updateBlynk() được gọi để gửi các giá trị mức nước (phần trăm và centimet) và trạng thái cảnh báo lên ứng dụng Blynk thông qua các Virtual Pins đã được cấu hình.

Điều khiển LED cảnh báo: Dựa trên giá trị waterLevelPercentage và các ngưỡng LOW_WATER_THRESHOLD và HIGH_WATER_THRESHOLD, trạng thái của LED đỏ và LED xanh dương được điều khiển như sau:

Nếu waterLevelPercentage **lớn hơn** HIGH_WATER_THRESHOLD**:** LED đỏ sẽ được bật (digitalWrite(LED_RED, HIGH);) và LED xanh dương sẽ được tắt (digitalWrite(LED_BLUE, LOW);).

Ngược lại, nếu waterLevelPercentage **nhỏ hơn** LOW_WATER_THRESHOLD**:** LED đỏ sẽ được tắt (digitalWrite(LED_RED, LOW);) và LED xanh dương sẽ được bật (digitalWrite(LED_BLUE, HIGH);).

Trong trường hợp còn lại (mức nước nằm giữa LOW_WATER_THRESHOLD **và** HIGH_WATER_THRESHOLD**):** Cả LED đỏ và LED xanh dương sẽ được bật (digitalWrite(LED_RED, HIGH); và digitalWrite(LED_BLUE, HIGH);).

Điều khiển Buzzer cảnh báo: Nếu waterLevelPercentage nhỏ hơn tone(BUZZER_PIN,1000); ngược lại, buzzer sẽ tắt bằng hàm noTone(BUZZER_PIN);.

Đọc nút nhấn vật lý: Trạng thái của nút nhấn kết nối với BUTTON_PIN được đọc. Nếu nút được nhấn (mức LOW do cấu hình pull-up), biến autoMode sẽ được đảo ngược, và trạng thái mới của autoMode sẽ được gửi đến Blynk thông qua Virtual Pin V4. Một khoảng trễ 500ms được thêm vào để chống rung nút nhấn.

Điều khiển Relay (chế độ tự động): Nếu biến autoMode là true, relay sẽ được bật (digitalWrite(RELAY_PIN, HIGH);) khi waterLevelPercentage nhỏ hơn 30% và tắt (digitalWrite(RELAY_PIN, LOW);) khi mức nước lớn hơn hoặc bằng 30%.

Thời gian trễ: Chương trình tạm dừng trong 1000 mili giây (1 giây) bằng hàm delay(1000); trước khi thực hiện vòng lặp tiếp theo.

5. Mô phỏng hệ thống.

Quá trình mô phỏng hệ thống đo mức nước thông minh trên nền tảng Wokwi được thực hiện theo các bước sau:

5.1. Thiết lập project.

-Để bắt đầu, một project mới được tạo trên Wokwi. Sau đó, các linh kiện cần thiết cho hệ thống được thêm vào bằng cách tìm kiếm trong thư viện linh kiện và kéo thả vào canvas làm việc.

-Các linh kiện đã được thêm bao gồm:

- Một board ESP32 DevKitC V4 (id: esp).
- Một cảm biến siêu âm HC-SR04 (id: ultrasonic1).
- Một màn hình OLED SSD1306 0.96 inch với địa chỉ I2C 0x3C (id: oled1).
- Một module relay (id: relay1).

- Một nút nhấn (id: btn1) màu xanh lá cây.
- Một LED đỏ (id: led1).
- Một buzzer (id: bz1).
- Một LED xanh dương (id: led2).
- Hai điện trở (id: r1 và r2) giá trị ban đầu là 220kΩ (sau này sẽ được điều chỉnh).

5.2. Kết nối ảo.

-Các chân của các linh kiện được kết nối với ESP32 theo sơ đồ mạch điện đã thiết kế. Việc kết nối được thực hiện bằng cách kéo từ chân của linh kiện này đến chân của linh kiện khác trên giao diện Wokwi.

-Các kết nối quan trọng bao gồm:

- Nguồn (3.3V và 5V) và GND từ ESP32 đến các linh kiện tương ứng.
- Các chân tín hiệu của HC-SR04 (TRIG đến GPIO5, ECHO đến GPIO18).
- Các chân I2C của OLED (SDA đến GPIO21, SCL đến GPIO22).
- Chân điều khiển relay (IN đến GPIO4).
- Các LED được kết nối qua điện trở (ban đầu 220kΩ) đến nguồn và chân điều khiển GPIO15 (đỏ) và GPIO2 (xanh dương).

Lưu ý: Trong quá trình mô phỏng, giá trị của R1 và R2 đã được điều chỉnh xuống các giá trị hợp lý hơn (ví dụ: 220Ω) để LED có thể sáng.

- Buzzer được kết nối giữa 5V và GPIO GND.3.
- Nút nhấn được kết nối giữa GND.3 và GPIO0.

5.3. Tải code lên Wokwi.

-Code chương trình Arduino được sao chép và dán vào trình soạn thảo code tích hợp của Wokwi. Các thông tin cấu hình Blynk (TEMPLATE_ID, TEMPLATE_NAME, BLYNK_AUTH_TOKEN) và cấu hình Wi-Fi (ssid: "Wokwi-GUEST", pass: "") được giữ nguyên cho phù hợp với môi trường mô phỏng. Giá trị MAX_DISTANCE_CM (ví dụ: 25.0 cm) và các ngưỡng cảnh báo (LOW_WATER_THRESHOLD, HIGH_WATER_THRESHOLD, SAFE_WATER_THRESHOLD) được thiết lập trong code.

5.4. Quá trình mô phỏng và kiểm tra.

Sau khi code được tải lên và không có lỗi biên dịch, nút "Start Simulator" được nhấn để bắt đầu quá trình mô phỏng.

Các bước kiểm tra sau đó được thực hiện:

-Kiểm tra hiển thị OLED: Quan sát màn hình OLED ảo trên Wokwi. Ban đầu, nó hiển thị các thông tin khởi động. Sau khi hệ thống bắt đầu đo, màn hình hiển thị khoảng cách đo được (thay đổi bằng cách điều chỉnh "Distance" của cảm biến HC-SR04 trong cửa sổ Inspector của Wokwi) và mức nước tương ứng theo phần trăm và centimet. Sự thay đổi này được theo dõi để đảm bảo các phép tính và hiển thị là chính xác.

-Kiểm tra kết nối Blynk (giả lập): Mặc dù Wokwi không mô phỏng kết nối internet thực tế, thư viện Blynk vẫn hoạt động và hiển thị các giá trị được gửi từ ESP32 trong cửa sổ "Blynk" của Wokwi. Các giá trị waterLevelPercentage (gửi đến V0), waterLevelCm (gửi đến V2), trạng thái báo động thấp (V1), trạng thái báo động cao (V3) và trạng thái Auto/Manual (V4) được theo dõi để đảm bảo dữ liệu được gửi đi đúng. Một công tắc ảo được thêm vào giao diện Blynk ảo (widget V4) để kiểm tra chức năng chuyển đổi chế độ Auto/Manual.

Lưu ý: Ý nghĩa của V1 và V3 trên Blynk sẽ tương ứng với logic cảnh báo mới (V1: Báo động thấp, V3: Báo động cao).

-Kiểm tra ngưỡng cảnh báo (LED và Buzzer): Giá trị "Distance" của cảm biến HC-SR04 được điều chỉnh để mô phỏng các mức nước khác nhau:

-Khi mức nước vượt quá HIGH_WATER_THRESHOLD (80%): LED đỏ trên Wokwi sáng lên và LED xanh dương tắt. Giá trị báo động cao (V3) trên Blynk chuyển sang 1. Buzzer vẫn im lặng.

-Khi mức nước xuống dưới LOW_WATER_THRESHOLD (30%): LED xanh dương trên Wokwi sáng lên và LED đỏ tắt. Buzzer phát ra âm thanh (có thể nghe được trong mô phỏng). Giá trị báo động thấp (V1) trên Blynk chuyển sang 1.

-Khi mức nước nằm giữa hai ngưỡng (30% - 80%): Cả LED đỏ và LED xanh dương trên Wokwi đều sáng, và buzzer ngừng kêu.

-Kiểm tra hoạt động của Relay: Chế độ tự động được kích hoạt (thông qua công tắc ảo trên Blynk hoặc nút nhấn vật lý). Khi mức nước giảm xuống dưới 30%, trạng thái của chân GPIO4 (kết nối với IN của relay) trên Wokwi được theo dõi để xem nó chuyển sang mức HIGH, mô phỏng việc bật relay. Khi mức nước tăng lên trên 30%, chân GPIO4 chuyển về mức LOW, mô phỏng việc tắt relay.

-Kiểm tra chức năng của nút nhấn: Nút nhấn ảo trên Wokwi được "nhấn" (bằng cách click vào nó). Quan sát giá trị của biến autoMode trên Serial Monitor của Wokwi (và giá trị gửi đến V4 trên Blynk) để xác nhận rằng trạng thái đã được chuyển đổi giữa true và false.

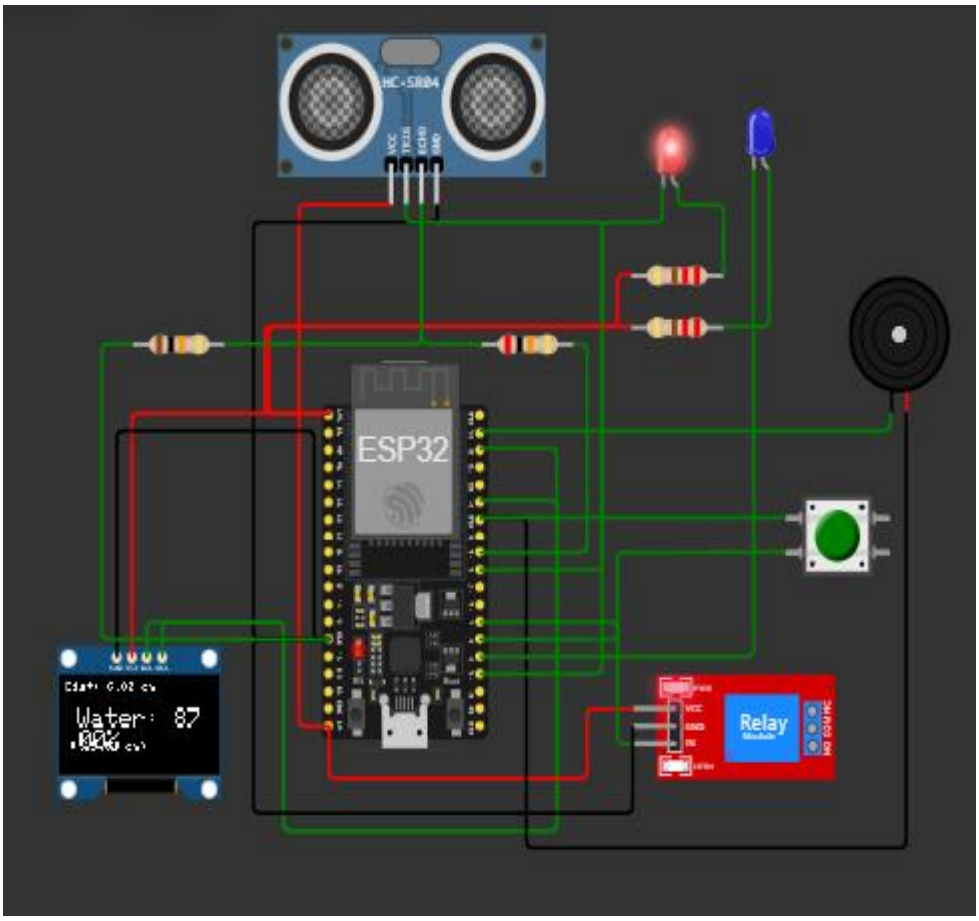
5.5. Kết quả mô phỏng.

Quá trình mô phỏng cho thấy hệ thống hoạt động tương đối đúng theo thiết kế mong muốn:

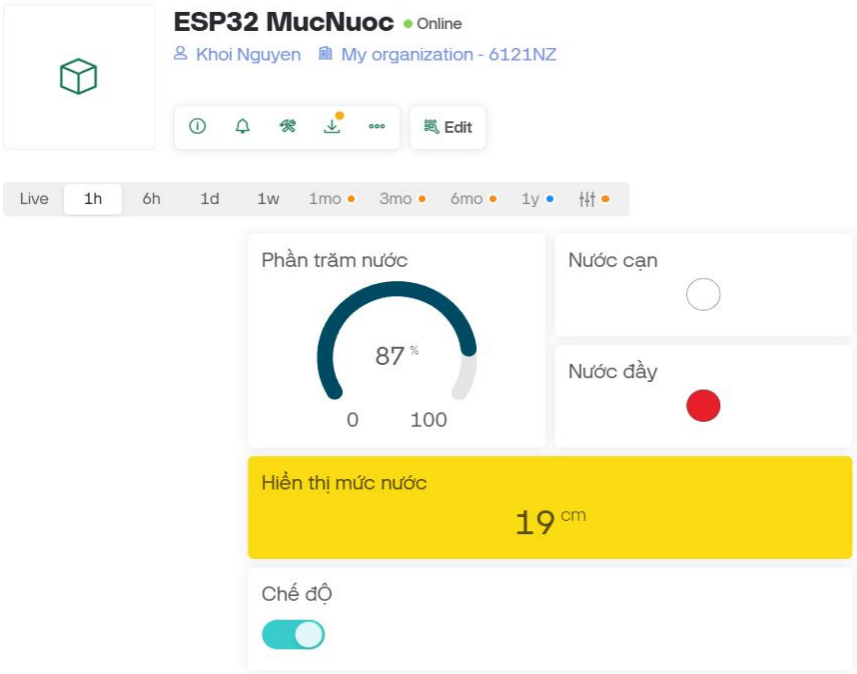
-Mức nước được đo và tính toán dựa trên khoảng cách từ cảm biến, và giá trị này được hiển thị chính xác trên màn hình OLED ảo.

· **Mức nước cao:** Khi giá trị "Distance" của cảm biến HC-SR04 được điều chỉnh để mô phỏng mức nước trên 80% (HIGH_WATER_THRESHOLD), **LED đỏ trên Wokwi sáng lên**, và LED xanh dương tắt. Giá trị báo đầy (V3) trên Blynk cũng chuyển sang 1 (theo logic cũ, bạn có thể cần điều chỉnh ý nghĩa của V1 và V3 trên Blynk nếu muốn). Buzzer vẫn im lặng do mức nước không thấp.

-Hình ảnh kết quả mô phỏng ở Wokwi khi mức nước trên 80%

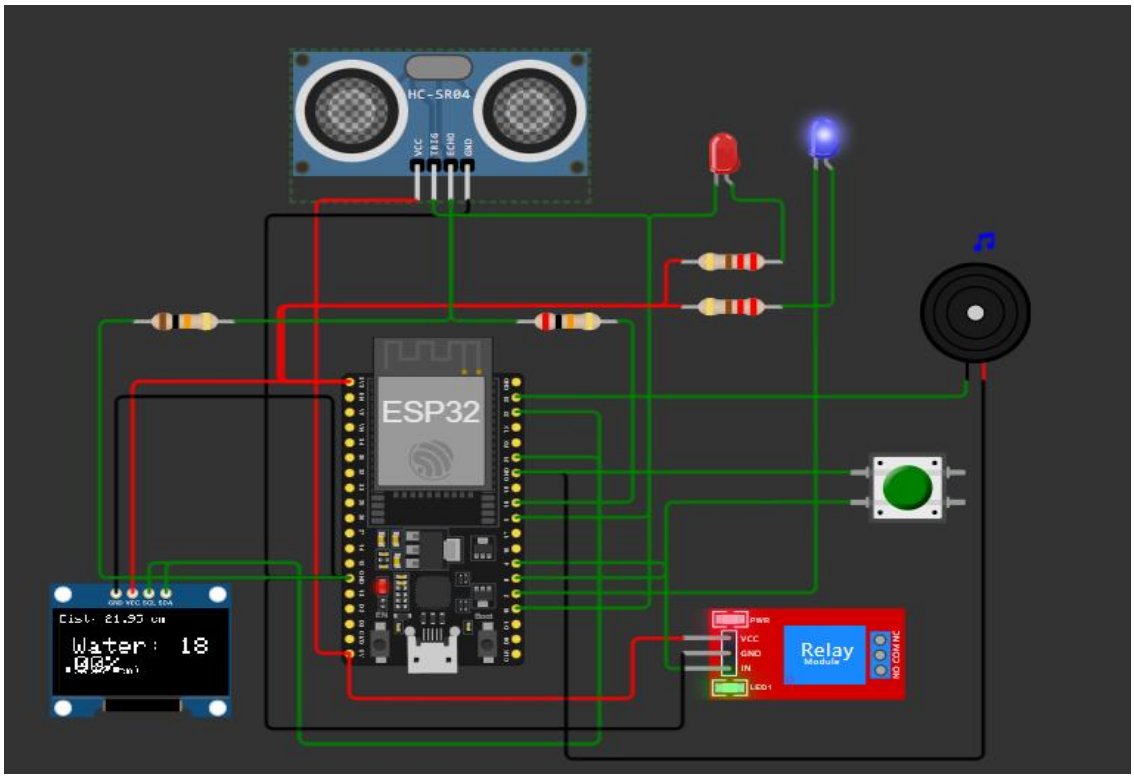


-Hình ảnh kết quả mô phỏng ở Blynk khi mức nước trên 80%

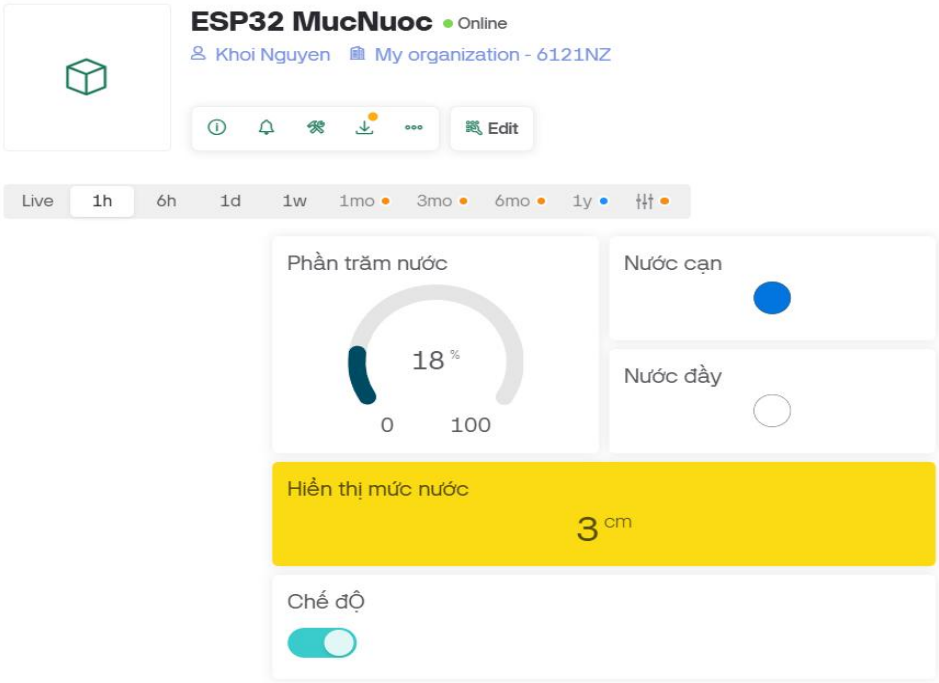


· **Mức nước thấp:** Khi mức nước mô phỏng xuống dưới 30% (LOW_WATER_THRESHOLD), **LED xanh dương trên Wokwi sáng lên**, và LED đỏ tắt. **Buzzer phát ra âm thanh** (có thể nghe được trong mô phỏng). Giá trị báo cạn (V1) trên Blynk cũng chuyển sang 1. Relay (ở chế độ Auto) được kích hoạt (chân GPIO4 chuyển sang HIGH).

-Hình ảnh kết quả mô phỏng ở Wokwi khi mức nước dưới 30%

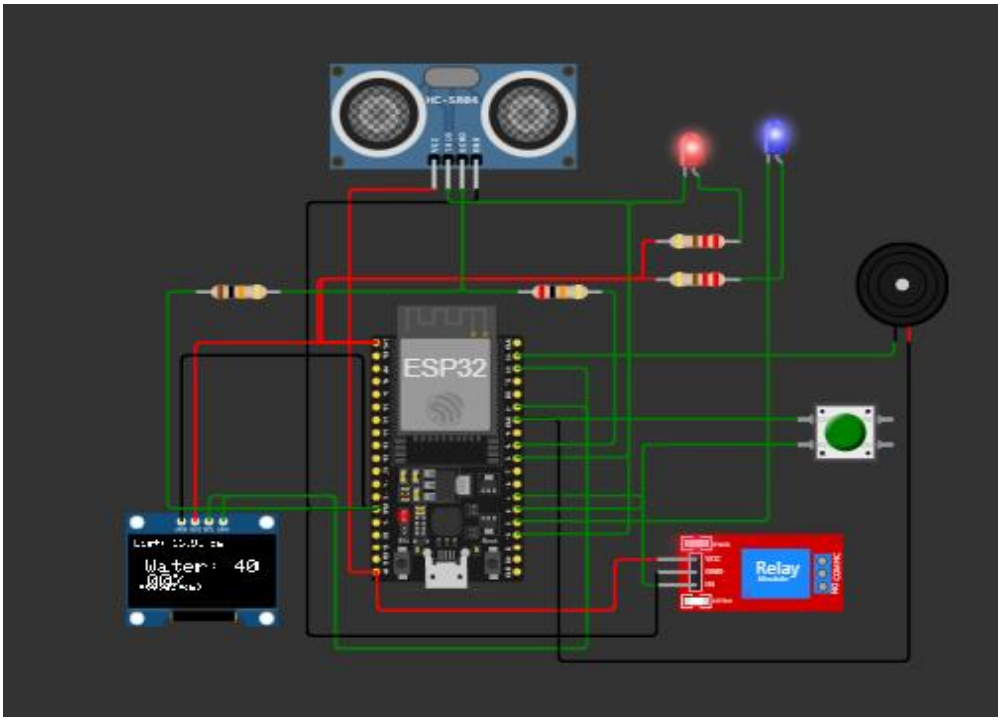


-Hình ảnh kết quả mô phỏng ở Blynk khi mức nước dưới 30%



· **Mức nước trung bình:** Khi mức nước mô phỏng nằm trong khoảng từ 30% đến 80%, **cả LED đỏ và LED xanh dương trên Wokwi đều sáng**, và buzzer ngừng kêu.

-Hình ảnh kết quả mô phỏng khi mực nước nằm trong khoảng từ 30% đến 80%



-Dữ liệu mức nước và trạng thái cảnh báo được gửi thành công đến giao diện Blynk ảo.

-Các LED cảnh báo và buzzer hoạt động đúng theo các ngưỡng mức nước đã được thiết lập.

-Relay được điều khiển tự động dựa trên mức nước ở chế độ Auto.

-Nút nhấn vật lý có khả năng chuyển đổi chế độ Auto/Manual.

5.6. Các vấn đề và cách khắc phục.

Trong quá trình mô phỏng, một số vấn đề đã được ghi nhận và khắc phục:

Giá trị điện trở cho LED: Giá trị điện trở ban đầu (220kΩ) khiến LED không sáng rõ. Giá trị này đã được điều chỉnh giảm xuống (ví dụ: 220Ω) để LED hiển thị đúng trạng thái.

Đọc giá trị ECHO: Ban đầu, có thể có sự không ổn định trong việc đọc độ rộng xung ECHO. Điều này có thể do nhiễu ảo trong môi trường mô phỏng. Việc sử dụng hàm pulseIn với timeout có thể giúp cải thiện độ ổn định.

Hiển thị Blynk: Mặc dù Blynk ảo hoạt động, việc tương tác thực tế với các widget (ví dụ: thay đổi giá trị gửi về ESP32) có thể có độ trễ hoặc không hoàn toàn giống như khi chạy trên phần cứng thực tế kết nối với internet. Tuy nhiên, việc gửi dữ liệu từ ESP32 lên Blynk ảo đã được xác nhận.

Nhìn chung, quá trình mô phỏng trên Wokwi đã cung cấp một môi trường hữu ích để kiểm tra logic hoạt động của hệ thống và phát hiện sớm các vấn đề tiềm ẩn trước khi triển khai trên phần cứng thực tế.

PHẦN KẾT LUẬN

Tiểu luận này đã trình bày quá trình xây dựng và thử nghiệm hệ thống đo mức nước thông minh sử dụng vi điều khiển ESP32 và cảm biến siêu âm HC-SR04. Việc tích hợp hai thành phần này đã tạo ra một giải pháp hiệu quả, chi phí hợp lý cho việc giám sát mực nước từ xa.

Hệ thống đã chứng minh khả năng thu thập dữ liệu mực nước chính xác thông qua cảm biến HC-SR04 và truyền tải thông tin này lên đám mây một cách ổn định nhờ khả năng kết nối Wi-Fi của ESP32. Dữ liệu được lưu trữ và có thể truy cập từ xa, mang lại sự tiện lợi trong việc theo dõi và quản lý mực nước, đặc biệt trong các ứng dụng như giám sát bể chứa nước sinh hoạt, nông nghiệp hoặc công nghiệp.

Mặc dù đã đạt được những kết quả ban đầu khả quan, hệ thống vẫn còn tiềm năng để phát triển và hoàn thiện hơn nữa. Các hướng nghiên cứu và cải tiến có thể bao gồm:

-Nâng cao độ tin cậy và độ chính xác của phép đo: Nghiên cứu và áp dụng các thuật toán lọc nhiễu, hiệu chuẩn cảm biến để giảm thiểu sai số do môi trường hoặc các yếu tố bên ngoài.

-Tối ưu hóa hiệu suất năng lượng: Triển khai các chế độ ngủ sâu cho ESP32 để kéo dài thời gian hoạt động của hệ thống khi sử dụng nguồn pin.

-Phát triển giao diện người dùng trực quan: Xây dựng các ứng dụng web hoặc di động thân thiện, cung cấp khả năng hiển thị dữ liệu trực quan, cảnh báo ngưỡng mực nước và quản lý hệ thống dễ dàng hơn.

-Tích hợp các tính năng thông minh hơn: Áp dụng các kỹ thuật học máy để dự đoán xu hướng mực nước, phát hiện các sự cố bất thường hoặc tự động điều khiển các thiết bị khác dựa trên trạng thái mực nước.

Nhìn chung, hệ thống đo mức nước thông minh sử dụng ESP32 và HC-SR04 là một bước tiến quan trọng trong việc ứng dụng công nghệ IoT vào giải quyết các bài toán thực tế. Với những tiềm năng phát triển, hệ thống này hứa hẹn sẽ đóng góp vào việc quản lý tài nguyên nước hiệu quả hơn trong tương lai.

TÀI LIỆU THAM KHẢO

[1] https://www.iotzone.vn/esp32/esp32-iot/do-an-he-thong-tuoi-nuoc-thong-minh-voi-esp32-blynk-app/

[2] http://arduino.vn/bai-viet/233-su-dung-cam-bien-khoang-cach-hc-sr04

[3] https://dientutuonglai.com/tim-hieu-hc-sr04.html

[4] https://mecsu.vn/ho-tro-ky-thuat/giam-sat-muc-nuoc-khong-tiep-xuc-dua-tren-iot-voi-esp32-va-hcsr04.yLD

[5] https://dientutuonglai.com/do-muc-nuoc-bang-cam-bien-sieu-am.html