

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH**



BÁO CÁO TỔNG KẾT

**ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN
NĂM 2025**

**NGHIÊN CỨU THIẾT KẾ THIẾT BỊ XÁC ĐỊNH CHỈ SỐ
NƯỚC DỰA TRÊN CẢM BIẾN DÒ CHUYỂN ĐỘNG
QUAY**

Sinh viên thực hiện

Ngô Huỳnh Quốc Huy

Mssv: 6251020057

Võ Văn Tuấn

Mssv: 6251020094

Lớp: Điện tử tin học công nghiệp K62

Người hướng dẫn: TS. Phạm Việt Thành

TPHCM, 2025

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH**



BÁO CÁO TỔNG KẾT

**ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN
NĂM 2025**

**NGHIÊN CỨU THIẾT KẾ THIẾT BỊ XÁC ĐỊNH CHỈ SỐ
NƯỚC DỰA TRÊN CẢM BIẾN DÒ CHUYỂN ĐỘNG
QUAY**

Sinh viên thực hiện

Ngô Huỳnh Quốc Huy

Mssv: 6251020057

Võ Văn Tuấn

Mssv: 6251020094

Lớp: Điện tử tin học công nghiệp K62

Người hướng dẫn: TS. Phạm Việt Thành

TPHCM, 2025

MỤC LỤC

MỤC LỤC	3
DANH MỤC HÌNH ẢNH	5
DANH MỤC BẢNG	6
DANH MỤC VIẾT TẮT	7
CHƯƠNG 1 . GIỚI THIỆU ĐỀ TÀI	8
1.1. Lý do chọn đề tài	8
1.2. Mục tiêu đề tài	9
1.3 Phương pháp nghiên cứu	9
1.3.1. Tìm hiểu lý thuyết và lựa chọn cảm biến	9
1.3.2. Thu thập dữ liệu và huấn luyện mô hình	9
1.3.3. Thiết kế và tối ưu thuật toán	9
1.3.4. Kiểm thử và đánh giá hệ thống	10
1.4. Các nội dung trọng tâm cần thực hiện	10
1.5 Giới hạn của đề tài	11
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	12
2.1. Khái niệm về hệ thống nhúng	12
2.2. Chức năng cơ bản của hệ thống nhúng	13
2.3. Tín hiệu cảm biến và xử lý tín hiệu	14
2.4. Tổng quan về hệ thống nhúng trong đo lường	15
CHƯƠNG 3: CÁC LOẠI CẢM BIẾN ĐO SỐ VÒNG QUAY	17
3.1. Tổng quan các loại cảm biến	17
3.2. Cảm biến Hall	Error! Bookmark not defined.
3.2.1. Khái niệm	Error! Bookmark not defined.
3.2.2. Cấu tạo	Error! Bookmark not defined.
3.2.3. Nguyên lý hoạt động	Error! Bookmark not defined.
3.3. Cảm biến quang	Error! Bookmark not defined.
3.3.1. Khái niệm	Error! Bookmark not defined.

3.3.2. Cấu tạo	Error! Bookmark not defined.
3.3.3. Nguyên lý hoạt động	Error! Bookmark not defined.
3.4. Cảm biến LC	18
3.4.1. Khái niệm	18
3.4.2. Cấu tạo	18
3.4.3. Nguyên lý hoạt động	20
3.5. Lựa chọn giải pháp	20
CHƯƠNG 4: THIẾT KẾ HỆ THỐNG ĐO SỐ VÒNG QUAY ĐỒNG HỒ NƯỚC..	23
4.1. Kiến trúc tổng thể của hệ thống	23
4.2. Phần cứng hệ thống	24
4.2.1.Sơ đồ mạch điện	24
4.2.2. Giới thiệu ESP8266	24
4.2.3. Giới thiệu mạch cảm biến LC	26
4.3. Phần mềm hệ thống	27
CHƯƠNG 5: KẾT QUẢ THỰC NGHIỆM.....	28
5.1: Mô hình phần cứng và thực tế	28
5.2. Quá trình thiết lập cho mạch	29
5.3 Đọc và xử lý dữ liệu tần số từ cảm biến LC	30
5.4. Kết quả thực nghiệm	Error! Bookmark not defined.
5.4.1 Mục tiêu	Error! Bookmark not defined.
5.4.2 Quy trình đo thực tế và kết quả đo	Error! Bookmark not defined.
5.4.3 Phân tích kết quả	Error! Bookmark not defined.
5.4.4 Điện năng tiêu thụ của nguồn	35
CHƯƠNG 6: KẾT LUẬN VÀ KIẾN NGHỊ	37
6.1. Kết luận	37
6.2. Kiến nghị	37
6.3.Hướng phát triển tiếp theo	37
TÀI LIỆU THAM KHẢO	39
PHỤ LỤC	40

DANH MỤC HÌNH ẢNH

Hình 1.1: Hình ảnh đồng hồ nước	11
Hình 2.1: Hệ thống nhúng	13
Hình 2.2: Mô hình cảm biến	15
Hình 3.1: Tổng quang về 3 cảm biến	17
Hình 3.2: Ảnh về cảm biến hall	Error! Bookmark not defined.
Hình 3.3: Cấu tạo của cảm biến Hall	Error! Bookmark not defined.
Hình 3.4 : Cảm biến quang	Error! Bookmark not defined.
Hình 3.5: Cảm biến LC	18
Hình 3.6: Ảnh về cuộn cảm	19
Hình 3.7: Ảnh về tụ điện	19
Hình 4.1: Sơ đồ mạch	24
Hình 4.2: Hình ảnh về ESP8266	25
Hình 5.1: Mô hình phần cứng	28
Hình 5.2: Mô hình phần cứng	29
Hình 5.3: Giao diện chương trình	30
Hình 5.4: So sánh tần số thay đổi	Error! Bookmark not defined.
Hình 5.5: Sau khi đưa dữ liệu lên FireBase	32
Hình 5.6: Cấu hình Firebase cho app	32
Hình 5.7: Giao diện ứng dụng	33
Hình 5.8: Hiển thị lượng nước tiêu thụ trên phần mềm	Error! Bookmark not defined.
Hình 5.9: Hiển thị giá tiền tiêu thụ trên phần mềm	Error! Bookmark not defined.

DANH MỤC BẢNG

<i>Bảng 3.1 Thống kê về các tiêu chuẩn cảm biến</i>	<i>22</i>
<i>Bảng 4.1 lưu đồ thuật toán</i>	<i>23</i>
<i>Bảng 5.1 Thống kê về lượng nước tiêu thụ theo dạng bảng</i>	Error! Bookmark not defined.
<i>Bảng 5.2 Thống kê về lượng nước tiêu thụ theo dạng cột</i>	Error! Bookmark not defined.
<i>Bảng 5.3 Thống kê về điện năng tiêu thụ trong 7 ngày</i>	<i>36</i>

DANH MỤC VIẾT TẮT

Viết tắt	Đầy đủ
ADC	Analog to Digital Converter – Bộ chuyển đổi tín hiệu tương tự sang số
ESP8266	Wi-Fi System-on-Chip do Espressif phát triển – vi điều khiển tích hợp Wi-Fi dùng trong IoT
GPIO	General Purpose Input/Output – Chân vào/ra có thể lập trình
I2C	Serial Peripheral Interface – Giao tiếp nối tiếp tốc độ cao
JSON	JavaScript Object Notation – Định dạng dữ liệu nhẹ, phổ biến trong truyền dữ liệu
LC	bộ Inductor – Capacitor – Mạch dao động cộng hưởng dùng cảm và tụ
LED	Light Emitting Diode – Đi-ốt phát quang
LCD	Liquid Crystal Display – Màn hình tinh thể lỏng
LoRa	Long Range – Công nghệ truyền không dây tầm xa tiết kiệm năng lượng
OLED	Organic Light Emitting Diode – Màn hình phát sáng hữu cơ
PWM	Pulse Width Modulation – Điều chế độ rộng xung
PID	Proportional – Integral – Derivative – Thuật toán điều khiển phản hồi
SPI	Serial Peripheral Interface – Giao tiếp nối tiếp tốc độ cao
SD	Secure Digital – Thẻ nhớ dùng để lưu dữ liệu
UART	Universal Asynchronous Receiver Transmitter – Giao tiếp nối tiếp không đồng
Wi-Fi	Wireless Fidelity – Mạng không dây

CHƯƠNG 1 . GIỚI THIỆU ĐỀ TÀI

1.1. Lý do chọn đề tài

Nhu cầu thị trường : Hiện nay, việc thống kê lượng nước tiêu thụ hàng tháng tại các hộ gia đình và doanh nghiệp chủ yếu vẫn được thực hiện bằng phương pháp thủ công, thông qua việc nhân viên đọc trực tiếp chỉ số trên đồng hồ cơ tại vị trí lắp đặt. Cách làm này dễ xảy ra sai số do ảnh hưởng của yếu tố khách quan (ánh sáng, môi trường, vị trí lắp đặt) và chủ quan (lỗi con người, sai sót khi ghi nhận số liệu). Điều này không chỉ ảnh hưởng đến độ chính xác trong tính toán lượng nước tiêu thụ mà còn làm giảm hiệu quả trong công tác quản lý cấp nước.

Tiềm năng ứng dụng rộng rãi: Giải pháp đo lường dựa trên cảm biến chuyển động quay có thể triển khai linh hoạt cho nhiều mô hình như: hộ gia đình, khu dân cư, chung cư, nhà máy, hoặc các cơ sở công nghiệp. Việc tự động hóa quá trình ghi nhận chỉ số tiêu thụ giúp nhà cung cấp dịch vụ cấp nước quản lý hiệu quả hơn, giảm tổn thất và hỗ trợ lập hóa đơn chính xác theo thời gian thực.

Ứng dụng công nghệ tiên tiến: Việc ứng dụng các loại cảm biến cơ bản kết hợp với hệ thống xử lý tín hiệu và truyền dữ liệu không dây, giải pháp này có khả năng cung cấp thông tin tiêu thụ nước theo thời gian thực mà không cần thay thế hoàn toàn đồng hồ hiện có, tiết kiệm chi phí và thuận tiện triển khai.

Thách thức kỹ thuật mang tính nghiên cứu: Việc tích hợp cảm biến chuyển động quay với hệ thống đồng hồ cơ đòi hỏi giải quyết đồng thời nhiều bài toán kỹ thuật, từ hiệu chuẩn cảm biến, khuếch đại và xử lý tín hiệu, đến truyền nhận dữ liệu ổn định, chính xác trong điều kiện dân dụng. Đây là một cơ hội để nghiên cứu và phát triển các giải pháp đo lường thông minh, có khả năng mở rộng và ứng dụng trong thực tế.

Chi phí triển khai hợp lý: Trên thị trường hiện nay, đồng hồ nước thông minh nhập khẩu có giá thành khá cao (từ 900.000 đến 4.000.000 đồng/thiết bị), trong khi đồng hồ nước cơ truyền thống chỉ dao động từ 100.000 đến 700.000 đồng. Việc thay thế toàn bộ hệ thống là một bài toán kinh tế lớn. Đề tài này hướng đến giải pháp nâng cấp đồng hồ cơ hiện có thành đồng hồ thông minh với chi phí thấp hơn nhiều, góp phần tiết kiệm chi phí đầu tư và phù hợp với khả năng triển khai đại trà tại các khu dân cư.

1.2. Mục tiêu đề tài

Xây dựng hệ thống nâng cấp đồng hồ nước cơ thành đồng hồ thông minh với chi phí thấp, không cần thay thế đồng hồ hiện có.

Đọc chính xác số vòng quay của kim đồng hồ nước bằng cảm biến nhằm thu thập dữ liệu tiêu thụ theo thời gian thực.

Tăng độ chính xác và khả năng tự động hóa trong ghi nhận chỉ số nước, góp phần giảm sai số và nâng cao hiệu quả quản lý.

Tiết kiệm năng lượng và đảm bảo hệ thống hoạt động bền bỉ trong thời gian dài. (sử dụng pin khoảng 2-3 năm).

1.3 Phương pháp nghiên cứu

1.3.1. Tìm hiểu lý thuyết và lựa chọn cảm biến

Nhóm nghiên cứu tiến hành khảo sát và đánh giá các loại cảm biến có tiềm năng ứng dụng trong việc xác định vòng quay của kim đồng hồ nước cơ

1.3.2. Thu thập dữ liệu và huấn luyện mô hình

Ghi nhận tín hiệu cảm biến tương ứng với số vòng quay của kim đồng hồ nước ở các mức tiêu thụ khác nhau.

Xây dựng tập dữ liệu đầu vào phục vụ cho quá trình phân tích và nhận diện chuyển động.

Phát triển thuật toán xử lý tín hiệu nhằm nhận biết chính xác từng đơn vị nước tiêu thụ thông qua đặc trưng của tín hiệu.

1.3.3. Thiết kế và tối ưu thuật toán

Phân tích đặc điểm tín hiệu từ từng loại cảm biến để xác định quy luật chuyển động của kim đồng hồ.

Xây dựng thuật toán nhận diện vòng quay dựa trên ngưỡng, tần suất hoặc mẫu hình dao động.

Tối ưu hóa thuật toán nhằm giảm nhiễu, bù sai số và nâng cao độ chính xác trong các điều kiện thực tế.

1.3.4. Kiểm thử và đánh giá hệ thống

Đánh giá độ chính xác của hệ thống thông qua so sánh kết quả đo được với chỉ số thực tế từ đồng hồ nước.

Tiến hành thử nghiệm trong nhiều điều kiện môi trường khác nhau để kiểm tra độ ổn định và độ bền của thiết bị.

Xác định sai số trung bình và đề xuất các hướng cải thiện về mặt thuật toán và phần cứng.

1.4. Các nội dung trọng tâm cần thực hiện

Nâng cao độ chính xác: Áp dụng các kỹ thuật lọc nhiễu, hiệu chỉnh sai số và tối ưu quá trình đếm xung nhằm đảm bảo kết quả đo ổn định, tin cậy.

Tăng cường khả năng dự đoán: Phân tích tín hiệu cảm biến theo thời gian thực để nhận diện chính xác số vòng quay, đồng thời tối ưu hóa quá trình lưu trữ và xử lý dữ liệu.

Tiết kiệm năng lượng: Lựa chọn cảm biến tiêu thụ điện năng thấp kết hợp với thuật toán điều khiển tiết kiệm điện để kéo dài thời gian sử dụng pin trong các hệ thống hoạt động liên tục.

Thử nghiệm thực tế: Đánh giá hiệu năng thiết bị thông qua các bài kiểm tra thực địa nhằm xác nhận tính khả thi, độ chính xác và hiệu quả sử dụng năng lượng của giải pháp đề xuất.



Hình 1.1: Hình ảnh đồng hồ nước

1.5 Giới hạn của đề tài

Đòi hỏi vi điều khiển hoạt động liên tục: Hệ thống cần duy trì trạng thái hoạt động liên tục để theo dõi và ghi nhận chỉ số nước tiêu thụ theo thời gian thực.

Ổn định lâu dài, xuyên suốt: Cảm biến và vi điều khiển phải đảm bảo độ bền và hoạt động ổn định trong thời gian dài mà không cần can thiệp thường xuyên.

Nguồn năng lượng cung cấp: Hệ thống có thể hoạt động bằng nguồn điện hoặc pin, tuy nhiên, nếu dùng pin thì cần tối ưu mức tiêu thụ năng lượng để đảm bảo thời gian sử dụng dài.

Mạng truyền thông :Hệ thống có thể để truyền dữ liệu từ cảm biến đến trung tâm xử lý và trung tâm trên hệ thống , đảm bảo kết nối ổn định và phạm vi truyền xa.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

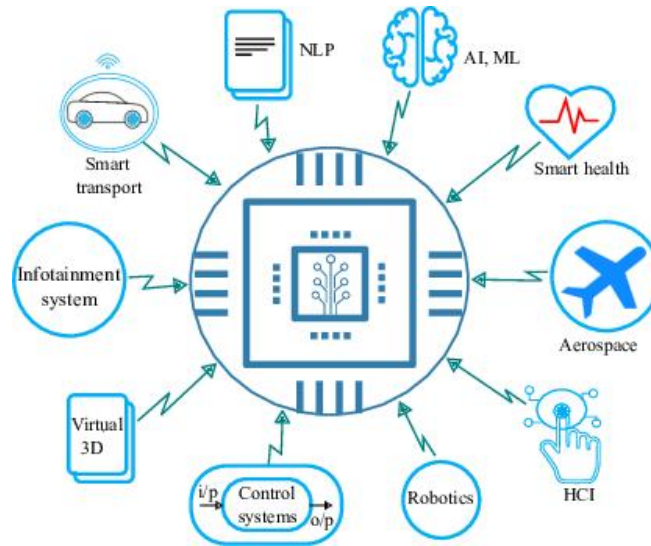
2.1. Khái niệm về hệ thống nhúng

Hệ thống nhúng (embedded system) là một thuật ngữ để chỉ một hệ thống có khả năng tự trị được nhúng vào trong một môi trường hay hệ thống mẹ. Đó là các hệ thống tích hợp cả phần cứng và phần mềm phục vụ các bài toán chuyên dụng trong nhiều lĩnh vực công nghiệp, tự động hoá điều khiển, quan trắc và truyền tin. Đặc điểm của các hệ thống nhúng là hoạt động ổn định và có tính năng tự động hoá cao.

Hệ thống nhúng thường được thiết kế để thực hiện một chức năng chuyên biệt nào đó. Khác với các máy tính đa chức năng, chẳng hạn như máy tính cá nhân, một hệ thống nhúng chỉ thực hiện một hoặc một vài chức năng nhất định, thường đi kèm với những yêu cầu cụ thể và bao gồm một số thiết bị máy móc và phần cứng chuyên dụng mà ta không tìm thấy trong một máy tính đa năng nói chung. Vì hệ thống chỉ được xây dựng cho một số nhiệm vụ nhất định nên các nhà thiết kế có thể tối ưu hóa nó nhằm giảm thiểu kích thước và chi phí sản xuất. Các hệ thống nhúng thường được sản xuất hàng loạt với số lượng lớn. Hệ thống nhúng rất đa dạng, phong phú về chủng loại. Đó có thể là những thiết bị cầm tay nhỏ gọn như đồng hồ kỹ thuật số và máy chơi nhạc MP3, hoặc những sản phẩm lớn như đèn giao thông, bộ kiểm soát trong nhà máy hoặc hệ thống kiểm soát các máy năng lượng hạt nhân. Xét về độ phức tạp, hệ thống nhúng có thể rất đơn giản với một vi điều khiển hoặc rất phức tạp với nhiều đơn vị, các thiết bị ngoại vi và mạng lưới được nằm gọn trong một lớp vỏ máy lớn.

Các thiết bị PDA hoặc máy tính cầm tay cũng có một số đặc điểm tương tự với hệ thống nhúng như các hệ điều hành hoặc vi xử lý điều khiển chúng nhưng các thiết bị này không phải là hệ thống nhúng thật sự bởi chúng là các thiết bị đa năng, cho phép sử dụng nhiều ứng dụng và kết nối đến nhiều thiết bị ngoại vi.

Bên cạnh đó, xu hướng hiện nay đang hướng đến việc phát triển các hệ thống nhúng thông minh, có khả năng kết nối Internet (IoT – Internet of Things) và xử lý dữ liệu ngay tại thiết bị mà không cần phụ thuộc vào máy chủ trung tâm. Việc tích hợp trí tuệ nhân tạo (AI) vào hệ thống nhúng cũng đang được nghiên cứu và ứng dụng mạnh mẽ, mở ra nhiều tiềm năng mới trong lĩnh vực tự động hóa và điều khiển thông minh.



Hình 2.1: Hệ thống nhúng

2.2. Chức năng cơ bản của hệ thống nhúng

Một hệ thống nhúng là tập hợp phần cứng và phần mềm chuyên dụng được thiết kế để thực hiện một chức năng xác định trong một hệ thống lớn hơn. Trong các ứng dụng điều khiển – giám sát, hệ thống nhúng đảm nhiệm vai trò xử lý tín hiệu đầu vào, ra quyết định, và truyền thông tin đến người dùng hoặc hệ thống quản lý. Dưới đây là các chức năng cốt lõi của một hệ thống nhúng:

-Tiếp nhận tín hiệu (Input Acquisition): Là khả năng giao tiếp với cảm biến để thu nhận dữ liệu từ môi trường thực. Tùy theo yêu cầu, hệ thống có thể tiếp nhận tín hiệu analog hoặc digital, và chuẩn hóa chúng trước khi xử lý.

-Xử lý tín hiệu (Signal Processing): Dữ liệu đầu vào cần được xử lý qua các thuật toán nhằm loại bỏ nhiễu, phát hiện sự kiện hoặc trích xuất thông tin. Đây là chức năng then chốt giúp hệ thống phản ứng đúng với thay đổi của môi trường.

-Ra quyết định và điều khiển (Decision & Control): Dựa trên kết quả xử lý, hệ thống đưa ra phản hồi như bật tắt thiết bị, đếm sự kiện, hoặc kích hoạt cảnh báo. Cấu trúc điều khiển có thể đơn giản (if-else) hoặc phức tạp (logic mờ, mạng neuron).

-Truyền thông (Communication): Hệ thống nhúng có thể truyền dữ liệu đến thiết bị khác qua giao thức UART, SPI, I2C, hoặc gửi qua mạng không dây như WiFi, LoRa. Đây là chức năng giúp hệ thống mở rộng kết nối và tích hợp vào nền tảng giám sát tập trung.

-Hiển thị hoặc phản hồi (Output Feedback): Ngoài điều khiển tự động, hệ thống có thể cung cấp thông tin cho người dùng qua màn hình, đèn LED hoặc truyền tới ứng dụng giám sát. Phản hồi trực quan là một phần quan trọng giúp kiểm tra và hiệu chỉnh hệ thống trong quá trình vận hành.

2.3. Tín hiệu cảm biến và xử lý tín hiệu

Trong một hệ thống đo lường, cảm biến là thành phần đầu vào có nhiệm vụ chuyển đổi các đại lượng vật lý cần đo (chẳng hạn như chuyển động, áp suất, nhiệt độ, ánh sáng, từ trường...) thành tín hiệu điện. Đây là bước quan trọng đầu tiên để biến thông tin thực tế thành dữ liệu có thể phân tích và xử lý trong môi trường số.

Tín hiệu do cảm biến tạo ra thường thuộc một trong hai dạng chính:

Tín hiệu tương tự (analog): Là tín hiệu biến thiên liên tục theo thời gian, biểu hiện qua điện áp hoặc dòng điện thay đổi theo giá trị đo được. Loại tín hiệu này yêu cầu sử dụng bộ chuyển đổi tương tự – số (ADC) để đưa vào xử lý trên hệ thống vi điều khiển.

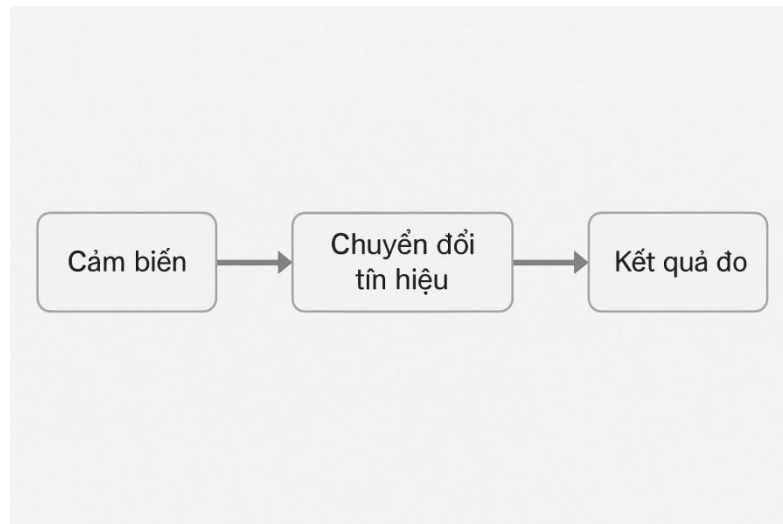
Tín hiệu số (digital): Là tín hiệu có dạng rời rạc, thường biểu diễn bằng hai mức logic (cao/thấp, 1/0). Loại tín hiệu này có thể được đọc trực tiếp bởi các chân vào số của vi điều khiển, và được xử lý dễ dàng hơn so với tín hiệu tương tự.

Việc xử lý tín hiệu cảm biến trong hệ thống đo lường thường bao gồm các bước:

Xử lý: Bao gồm khuếch đại tín hiệu yếu, lọc nhiễu điện, ổn định mức tín hiệu đầu vào.

Chuyển đổi tín hiệu: Nếu là tín hiệu analog thì cần chuyển đổi sang dạng số; nếu là tín hiệu số thì có thể đọc và xử lý trực tiếp.

Trích xuất thông tin: Là bước phân tích dữ liệu nhận được để rút ra giá trị đo có ý nghĩa, như tốc độ, vị trí...



Hình 2.2: Mô hình cảm biến

2.4. Tổng quan về hệ thống nhúng trong đo lường

Một hệ thống đo lường sử dụng nền tảng hệ nhúng thường hoạt động theo một chu trình khép kín và liên tục, đảm bảo thu thập và xử lý dữ liệu một cách hiệu quả và đáng tin cậy. Quy trình cơ bản có thể mô tả qua các bước sau:

Giai đoạn thu nhận dữ liệu đầu vào: Các cảm biến chuyên dụng sẽ ghi nhận các đại lượng vật lý như chuyển động, áp suất, ánh sáng, nhiệt độ, độ ẩm, tốc độ dòng chảy,... tùy theo yêu cầu của ứng dụng. Mỗi loại cảm biến thường cho ra tín hiệu điện áp, dòng điện hoặc xung số tương ứng với giá trị của đại lượng đo được.

Giai đoạn chuyển đổi và đưa vào hệ thống: Tín hiệu đầu ra của cảm biến sẽ được đưa vào hệ thống nhúng thông qua các chân analog hoặc digital trên vi điều khiển. Trường hợp tín hiệu là analog, hệ thống sẽ sử dụng bộ chuyển đổi ADC (Analog to Digital Converter) để biến đổi thành tín hiệu số phục vụ cho xử lý.

Xử lý tín hiệu và tính toán: Vi điều khiển trong hệ thống nhúng sẽ nhận dữ liệu và thực hiện các thuật toán xử lý: lọc nhiễu, tính trung bình, phân tích ngưỡng, hoặc các thuật toán điều khiển như PID. Đây là khâu quan trọng giúp đảm bảo kết quả đo chính xác, ổn định và phù hợp với yêu cầu thực tế.

Giai đoạn xuất dữ liệu: Sau khi xử lý, kết quả có thể được hiển thị trực tiếp trên màn hình LCD/OLED, hoặc được gửi đi thông qua giao tiếp UART, I2C, SPI hoặc Wi-Fi/Bluetooth. Ngoài ra, hệ thống cũng có thể ghi log vào bộ nhớ trong, thẻ SD hoặc gửi lên server/cloud/Firebase để lưu trữ và phân tích từ xa.

Tổng quát, mô hình hoạt động có thể biểu diễn như sau:

Cảm biến → Vi điều khiển → Xử lý tín hiệu → Hiển thị / Giao tiếp / Lưu trữ

Trong các ứng dụng hiện đại, hệ thống nhúng còn có thể mở rộng thêm các chức năng như gửi cảnh báo khi vượt ngưỡng, kích hoạt relay điều khiển thiết bị, hoặc đồng bộ dữ liệu thời gian thực với hệ thống giám sát trung tâm. Điều này giúp hệ nhúng không chỉ là công cụ đo lường đơn thuần mà còn đóng vai trò như một phần tử quyết định trong hệ thống điều khiển tự động và thông minh hóa quy trình.

CHƯƠNG 3: CẢM BIẾN ĐO SỐ VÒNG QUAY

3.1. Tổng quan các loại cảm biến

Trong các hệ thống khi đo lường tự động, việc xác định chính xác số vòng quay của một bộ phận chuyển động là yếu tố quan trọng nhằm suy ra các đại lượng vật lý như vận tốc, lưu lượng, công suất hoặc lượng tiêu thụ. Để thực hiện điều này một cách chính xác, ổn định và liên tục, hệ thống cần sử dụng các loại cảm biến ghi nhận chuyển động quay.

Hiện nay, có nhiều loại cảm biến được ứng dụng để đo vòng quay, tùy thuộc vào nguyên lý hoạt động, môi trường ứng dụng, yêu cầu kỹ thuật và chi phí triển khai. Một số cảm biến hoạt động theo nguyên lý từ tính, một số sử dụng ánh sáng để phát hiện chuyển động, trong khi những loại khác dựa trên đặc tính cộng hưởng hoặc biến thiên điện từ của vật thể quay.

Các loại cảm biến phổ biến thường được sử dụng trong đo chuyển động quay bao gồm:

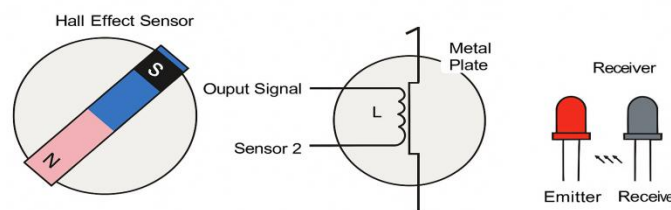
Cảm biến từ (Hall): Phát hiện từ trường thay đổi do nam châm gắn trên kim.

Cảm biến quang: Dựa vào ánh sáng phản xạ trên kim quay

Cảm biến LC: Phát hiện sự thay đổi trong điện cảm hoặc tần số dao động khi có vật liệu kim loại chuyển động gần cuộn dây.

Mỗi loại cảm biến đều có những ưu điểm và hạn chế riêng, và việc lựa chọn loại cảm biến phù hợp phụ thuộc vào đặc tính chuyển động cần đo, môi trường làm việc, cũng như mức độ chính xác, tốc độ phản hồi và độ ổn định mà hệ thống yêu cầu.

Cảm biến Hall – LC – Quang

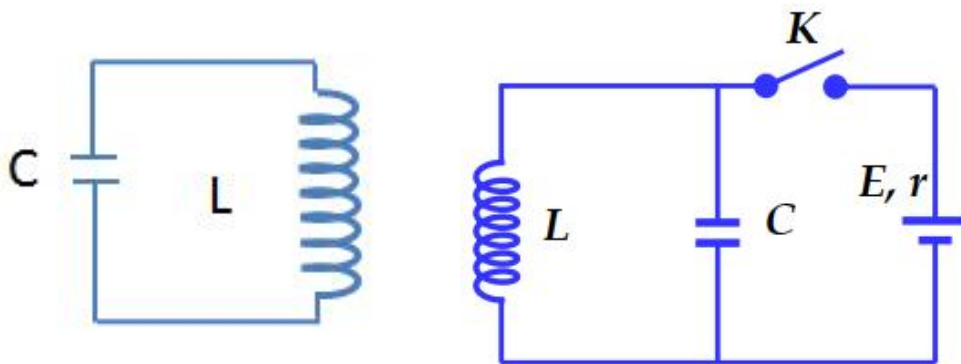


Hình 3.1: Tổng quan về 3 cảm biến

3.2. Cảm biến LC

3.2.1. Khái niệm

Cảm biến LC là loại cảm biến hoạt động dựa trên nguyên lý cộng hưởng điện từ của mạch cuộn cảm (L – inductance) và tụ điện (C – capacitance). Khi hai linh kiện này được kết nối thành một mạch dao động LC, hệ thống sẽ tạo ra một tần số cộng hưởng đặc trưng. Sự thay đổi trong môi trường xung quanh – đặc biệt là sự xuất hiện của vật kim loại gần cuộn dây – sẽ làm thay đổi độ tự cảm của cuộn dây hoặc điện dung tổng thể, dẫn đến sự thay đổi tần số dao động.



Hình 3.5: Cảm biến LC

3.2.2. Cấu tạo

Cuộn cảm (L)

Cuộn cảm là một cuộn dây dẫn (thường làm bằng đồng), được quấn thành nhiều vòng. Khi dòng điện chạy qua cuộn dây, một từ trường được tạo ra bao quanh các vòng dây. Đặc điểm chính của cuộn cảm là kháng lại sự thay đổi dòng điện, thông qua hiện tượng cảm ứng điện từ. Độ tự cảm L của cuộn dây phụ thuộc vào số vòng, kích thước, hình dạng và lõi vật liệu. Khi có vật kim loại (đặc biệt là kim loại dẫn điện) tiếp cận cuộn dây, từ trường bị ảnh hưởng – hiện tượng cảm ứng dòng điện Foucault xuất hiện làm giảm từ thông tổng quát – từ đó làm thay đổi giá trị L của cuộn.



Hình 3.6: Ảnh về cuộn cảm

Tụ Điện (C)

Tụ điện là linh kiện gồm hai bản dẫn điện song song, ngăn cách bởi một lớp điện môi (cách điện). Khi đặt hiệu điện thế giữa hai bản, tụ điện sẽ tích tụ điện tích và tạo ra điện trường. Tụ điện có đặc tính kháng lại sự thay đổi điện áp, và là phần tử lưu trữ năng lượng điện trong mạch LC. Giá trị điện dung C phụ thuộc vào diện tích bản cực, khoảng cách giữa chúng và hằng số điện môi của vật liệu cách điện.

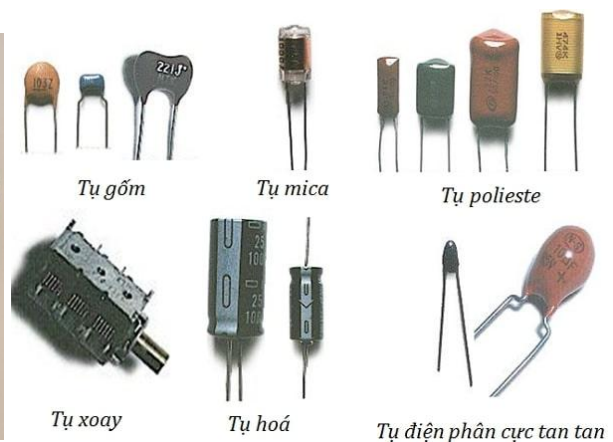
Điện áp hoạt động: 3.3V ~ 5V

Tần số dao động: 100 kHz ~ 5 MHz (phụ thuộc vào giá trị L và C)

Dòng tiêu thụ: < 1 mA (tùy vào mạch dao động và tần số)

Độ nhạy tần số: ~ vài kHz thay đổi khi có kim loại tiếp cận (với thiết kế phù hợp)

Nhiệt độ hoạt động: -20°C ~ 85°C



Hình 3.7: Ảnh về tụ điện

3.2.3. Nguyên lý hoạt động

Cảm biến LC hoạt động dựa trên nguyên lý dao động cộng hưởng điện từ trong mạch LC, gồm một cuộn cảm (L) và một tụ điện (C). Khi được kích thích, mạch LC sẽ tạo ra một dao động điện có tần số xác định gọi là tần số cộng hưởng, được tính bằng công thức:

$$f = \frac{1}{2\pi\sqrt{LC}}$$

Trong đó:

f là tần số cộng hưởng (Hz)

L là độ tự cảm của cuộn dây (Henry – H)

C là điện dung của tụ điện (Farad – F)

Ở trạng thái bình thường, tần số dao động của mạch LC sẽ ổn định theo giá trị ban đầu của L và C. Tuy nhiên, khi có vật thể dẫn điện hoặc kim loại tiến gần đến cuộn dây, trường điện từ của cuộn cảm bị ảnh hưởng, làm thay đổi độ tự cảm L. Cụ thể, sự xuất hiện của dòng điện xoáy (Foucault) trong vật kim loại sẽ tạo ra từ trường đối kháng, dẫn đến giảm giá trị L và theo đó làm tăng tần số dao động f.

Ngược lại, trong một số cấu hình khác, có thể xảy ra hiện tượng tăng cảm ứng điện từ hoặc thay đổi điện dung tổng thể (nếu vật thể ảnh hưởng đến tụ điện hoặc môi trường điện môi), làm giảm tần số cộng hưởng. Dù theo chiều hướng nào, sự thay đổi tần số là tín hiệu để phát hiện sự xuất hiện hoặc chuyển động của vật thể.

Trong thực tế, vi điều khiển hoặc mạch đo sẽ theo dõi sự thay đổi của tần số f trong thời gian thực. Khi tần số vượt qua một ngưỡng định trước, hệ thống sẽ xác định có vật thể xuất hiện hoặc có chuyển động gần khu vực cảm biến.

3.3. Lựa chọn giải pháp

Trong các hệ thống đo vòng quay hoặc phát hiện vật thể, ba loại cảm biến được sử dụng phổ biến hiện nay là: cảm biến Hall, cảm biến quang và cảm biến điện từ (LC). Mỗi loại cảm biến có nguyên lý hoạt động, ưu điểm và hạn chế riêng, phù hợp với từng điều kiện sử dụng cụ thể. Việc lựa chọn cảm biến phù hợp không chỉ dựa trên độ chính xác mà còn phụ thuộc vào chi phí triển khai và mức tiêu thụ điện năng của hệ thống.

Cảm biến Hall là một lựa chọn phổ biến nhờ thiết kế đơn giản, kích thước nhỏ gọn và khả năng phát hiện từ trường nhanh nhạy. Loại cảm biến này tiêu thụ điện năng ở mức vừa phải (khoảng 4–9 mA) và xuất ra tín hiệu số rõ ràng, dễ dàng tích hợp vào các hệ thống xử lý tín hiệu số. Do đó, cảm biến Hall được ứng dụng rộng rãi trong các thiết bị điện tử tiêu dùng, đồng hồ điện hoặc hệ thống đo tốc độ quay. Tuy nhiên, cảm biến Hall có thể bị nhiễu khi ở gần các nguồn phát từ trường mạnh như motor, cuộn dây, biến áp..., do đó yêu cầu bố trí lắp đặt hợp lý để đảm bảo độ chính xác. Với chi phí rẻ, thiết kế nhỏ gọn và điện năng tiêu thụ vừa phải, cảm biến Hall là lựa chọn kinh tế và hiệu quả cho các ứng dụng đo đếm đơn giản, ổn định.

Cảm biến quang học có ưu điểm nổi bật về tốc độ phản hồi nhanh và độ chính xác cao, đặc biệt trong các môi trường sạch như phòng thí nghiệm hoặc dây chuyền sản xuất tự động. Cảm biến này có thể phát hiện những thay đổi rất nhỏ thông qua phản xạ ánh sáng, phù hợp để đếm vật thể tốc độ cao hoặc phát hiện vạch đánh dấu trên vật quay. Tuy nhiên, cảm biến quang thường tiêu thụ điện năng cao hơn (thường 20–30 mA hoặc hơn) so với Hall và dễ bị ảnh hưởng bởi ánh sáng môi trường, bụi bẩn hoặc sai lệch góc chiếu. Vì vậy, chúng yêu cầu môi trường hoạt động ổn định và thiết kế quang học chính xác. Giá thành của cảm biến quang cũng cao hơn so với hai loại còn lại, nên thường chỉ phù hợp cho các hệ thống yêu cầu độ chính xác cao và được đầu tư đầy đủ về điều kiện vận hành.

Cảm biến điện từ (LC) hoạt động dựa trên nguyên lý cộng hưởng của mạch LC, gồm cuộn cảm và tụ điện. Khi vật thể kim loại đến gần cuộn cảm, tần số dao động thay đổi và được xử lý để xác định sự hiện diện của vật. Loại cảm biến này có ưu điểm không tiếp xúc cơ học, độ bền cao, ít bị ảnh hưởng bởi môi trường bụi bẩn hay ánh sáng, đồng thời tiêu thụ điện năng cực thấp (chỉ khoảng 0.005 mA nếu thiết kế đơn giản). Ngoài ra, cảm biến LC có thể dễ dàng tự chế tạo bằng linh kiện phổ thông như cuộn cảm và tụ điện, giúp tiết kiệm chi phí. Tuy vậy, việc xử lý tín hiệu đòi hỏi cao, đồng thời cần tính toán đúng thông số L và C để đảm bảo tần số hoạt động phù hợp. Độ nhạy của cảm biến còn phụ thuộc vào chất lượng cuộn cảm, khoảng cách đến vật thể và khả năng lọc nhiễu của toàn mạch.

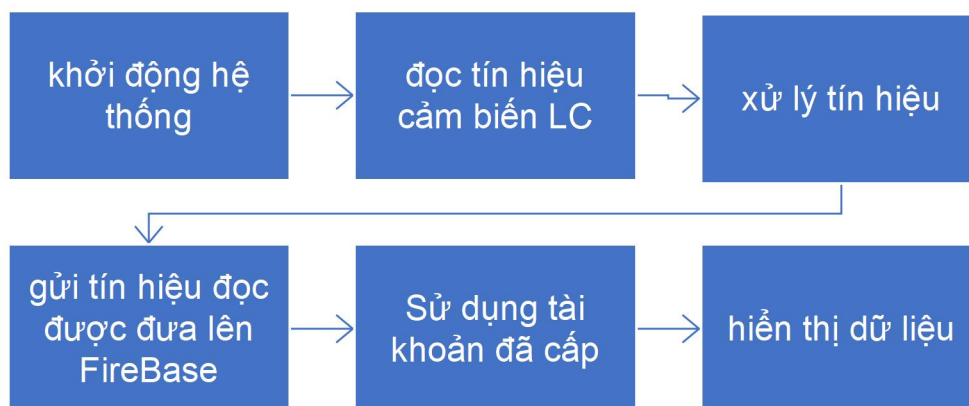
Để hệ thống có thể hoạt động bền bỉ, tiêu thụ ít năng lượng, dễ chế tạo và phù hợp với điều kiện lắp đặt thực tế trên đồng hồ nước cơ, nhóm nghiên cứu đã lựa chọn sử dụng cảm biến điện từ (LC) làm giải pháp chính. Việc lựa chọn này không phủ nhận giá trị và tính hiệu quả của hai loại cảm biến còn lại, mà dựa trên sự phù hợp giữa đặc tính kỹ thuật và yêu cầu cụ thể của đề tài.

Tiêu Chí	Cảm biến Hall	Cảm biến quang	Cảm biến LC (điện từ)
Nguyên lý hoạt động	Phát hiện từ trường từ nam châm	Phát hiện phản xạ hoặc chắn ánh sáng	Phát hiện thay đổi tần số cộng hưởng LC
Ảnh hưởng môi trường	Nhạy với từ trường xung quanh	Dễ nhiễu bởi ánh sáng, bụi bẩn	Ít bị ảnh hưởng, hoạt động ổn định
Tín hiệu đầu ra	Analog hoặc digital	Analog hoặc digital	Dao động/tần số
Mức tiêu thụ điện	Thấp (4–9 mA)	Trung bình cao (20-60mA)	Rất thấp (<1 mA)
Chi phí linh kiện	Rẻ (5.000 – 10.000 VNĐ)	Tương đối (5.000-60.000 VNĐ)	Rất rẻ (tự chế cuộn cảm + tụ điện)

Bảng 3.1 Thống kê về các tiêu chuẩn cảm biến

CHƯƠNG 4: THIẾT KẾ HỆ THỐNG ĐO SỐ VÒNG QUAY ĐỒNG HỒ NƯỚC

4.1. Kiến trúc tổng thể của hệ thống



Bảng 4.1 lưu đồ thuật toán

Đầu tiên, hệ thống bắt đầu với khối “khởi động hệ thống”, nơi ESP8266 được cấp nguồn và thực hiện các bước khởi tạo cần thiết như kết nối mạng WiFi, cấu hình Firebase và thiết lập thông số cho các chân đọc tín hiệu.

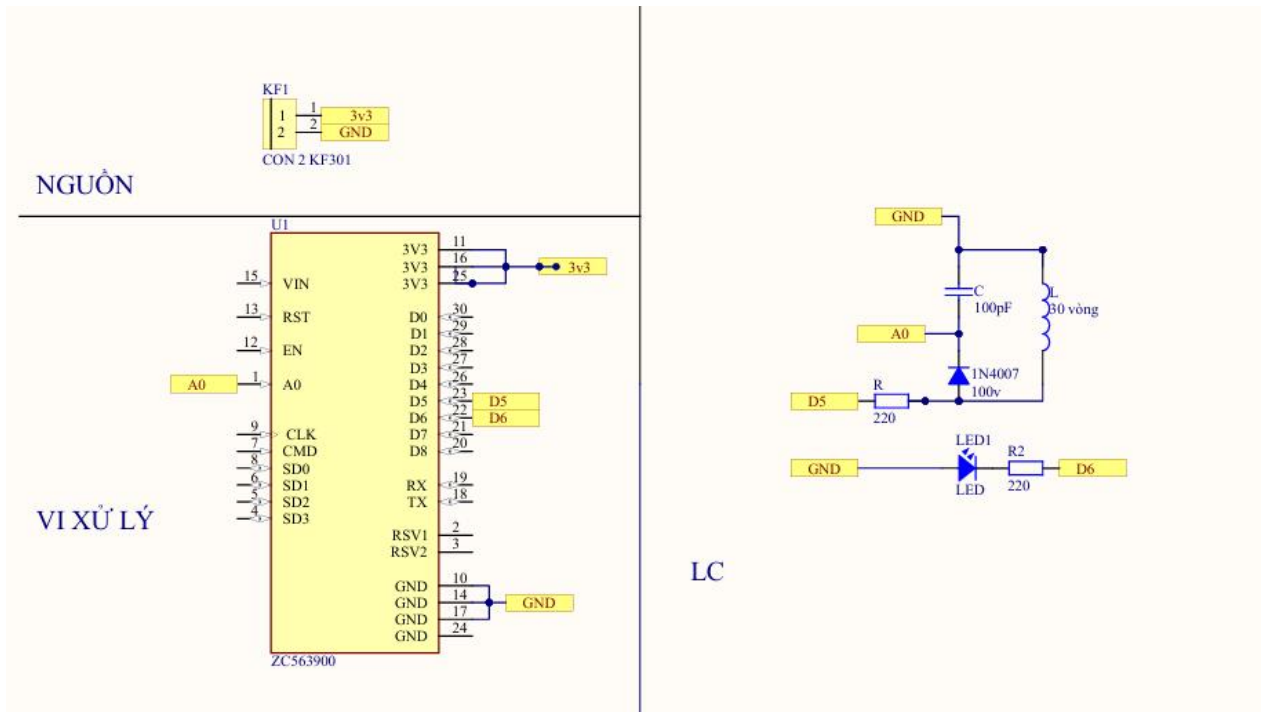
Tiếp theo, khối “đọc tín hiệu cảm biến LC” đảm nhận nhiệm vụ nhận dữ liệu từ mạch cộng hưởng LC. Tín hiệu này có thể là tín hiệu analog (biên độ dao động) hoặc digital (xung tần số), tùy thuộc vào cách thiết kế mạch.

Tín hiệu thu được sẽ được đưa vào khối “xử lý tín hiệu”, nơi vi điều khiển ESP8266 phân tích sự thay đổi về biên độ hoặc tần số để phát hiện chuyển động quay. Khi phát hiện một vòng quay hợp lệ, hệ thống sẽ tăng biến đếm và tính toán thể tích nước tiêu thụ tương ứng.

Sau khi xử lý xong, dữ liệu được chuyển đến khối “gửi tín hiệu đọc được đưa lên Firebase”. Tại đây, ESP8266 đóng gói thông tin (số vòng quay, thể tích nước, thời gian...) và truyền lên cơ sở dữ liệu Firebase thông qua WiFi.

4.2. Phần cứng hệ thống

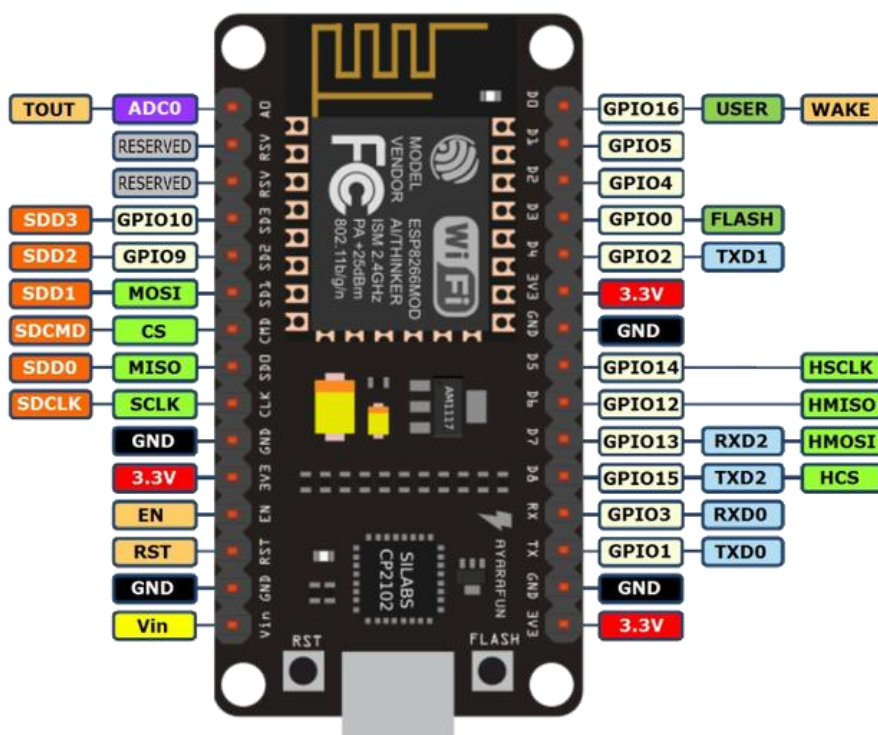
4.2.1. Sơ đồ mạch điện



Hình 4.1: Sơ đồ mạch

4.2.2. Giới thiệu ESP8266

Trong thiết kế hệ thống đo vòng quay của đồng hồ nước cơ, vi điều khiển đóng vai trò trung tâm trong việc thu thập, xử lý và truyền dữ liệu từ cảm biến ra bên ngoài. Để đáp ứng yêu cầu về hiệu năng xử lý, tính linh hoạt và khả năng kết nối không dây, nhóm nghiên cứu lựa chọn sử dụng ESP8266 – một dòng vi điều khiển phổ biến được phát triển bởi Espressif Systems, được biết đến rộng rãi trong các ứng dụng Internet of Things (IoT) và hệ thống nhúng. ESP8266 là dòng SoC (System on Chip) tích hợp nhiều chức năng hữu ích trong một gói nhỏ gọn. Nó nổi bật nhờ khả năng xử lý tác vụ, hỗ trợ các giao tiếp ngoại vi cơ bản và tích hợp WiFi, đồng thời có mức tiêu thụ năng lượng thấp. Việc tích hợp các thành phần phần cứng quan trọng giúp giảm thiểu số lượng mạch phụ trợ và đơn giản hóa thiết kế mạch tổng thể. Điều này đặc biệt phù hợp với các hệ thống đo lường cần hoạt động ổn định, chi phí hợp lý và dễ triển khai thực tế.



Hình 4.2: Hình ảnh về ESP8266

Cấu trúc và tính năng chính của ESP8266:.

ESP8266 là dòng SoC (System on Chip) tích hợp nhiều chức năng hữu ích trong một gói nhỏ gọn. Nó nổi bật nhờ khả năng xử lý tác vụ, hỗ trợ các giao tiếp ngoại vi cơ bản và tích hợp WiFi, đồng thời có mức tiêu thụ năng lượng thấp. Việc tích hợp các thành phần phần cứng quan trọng giúp giảm thiểu số lượng mạch phụ trợ và đơn giản hóa thiết kế mạch tổng thể. Điều này đặc biệt phù hợp với các hệ thống đo lường cần hoạt động ổn định, chi phí hợp lý và dễ triển khai thực tế.

ESP8266 sử dụng bộ xử lý Tensilica Xtensa LX106, hoạt động ở tốc độ xung nhịp lên đến 80 MHz, đáp ứng các tác vụ xử lý cơ bản và giao tiếp mạng. Bộ nhớ trên chip bao gồm một lượng SRAM hạn chế cho dữ liệu và chương trình, cùng với Flash bên ngoài để lưu trữ chương trình chính và cấu hình hệ thống.

Một trong những điểm mạnh đáng chú ý của ESP8266 là khả năng giao tiếp linh hoạt. Con chip hỗ trợ chuẩn giao tiếp UART (Serial) để truyền dữ liệu với các thiết bị khác hoặc máy tính, SPI và I2C để giao tiếp với cảm biến và module mở rộng (ví dụ: màn hình OLED). ESP8266 cũng có một kênh ADC (bộ chuyển đổi tín hiệu analog-số) với độ phân giải 10-bit, cho phép đọc tín hiệu analog từ cảm biến. Ngoài ra, nó hỗ trợ PWM để điều khiển thiết bị ngoại vi như LED. Khả năng kết nối WiFi cho phép truyền dữ liệu về các nền tảng đám mây.

ESP8266 hỗ trợ các chế độ tiết kiệm năng lượng như Sleep Mode, giúp tối ưu thời gian hoạt động khi hệ thống chạy bằng pin, phù hợp cho các ứng dụng đo lường dài hạn như trong đồng hồ nước thông minh.

Ưu điểm nổi bật khi dùng ESP8266 trong hệ đo vòng quay:

- Tích hợp WiFi: cho phép truyền dữ liệu lên cloud mà không cần module rời.
- Số lượng chân GPIO và ADC đủ dùng: thuận tiện cho kết nối cảm biến và các mạch ngoại vi cơ bản.
- Khả năng xử lý đủ cho các tác vụ đo lường và truyền dữ liệu.

Thông số kỹ thuật tóm tắt của ESP8266 ESP-01 (một trong những module phổ biến):

- Chip điều khiển: ESP8266
- Số lượng chân: 8 pin
- Kết nối không dây: Wi-Fi (802.11 b/g/n)
- Điện áp hoạt động: 3.3V
- Bộ nhớ: RAM tùy thuộc vào phiên bản, Flash bên ngoài (thường 1MB hoặc lớn hơn)
- GPIO: Một vài chân GPIO có thể lập trình
- ADC: 1 kênh ADC 10-bit
- PWM: Hỗ trợ PWM thông qua phần mềm
- Giao tiếp: SPI, I2C (thông qua phần mềm), UART

Vi điều khiển có thể đọc tín hiệu từ mạch LC qua chân ADC (trong trường hợp sử dụng tín hiệu analog) hoặc chân digital nếu mạch LC được xử lý qua một bộ so sánh (comparator) để tạo xung.

4.2.3. Giới thiệu mạch cảm biến LC.

Trong thiết kế thực nghiệm, nhóm lựa chọn sử dụng cuộn cảm tự cuốn gồm 30 vòng dây đồng đường kính 0,3 mm, quấn quanh lõi không khí có đường kính 2 cm, kết hợp với tụ điện 100pF (ký hiệu 104) để tạo thành mạch cộng hưởng LC.

Lý do lựa chọn cấu hình này xuất phát từ mục tiêu thiết kế mạch hoạt động ở tần số cộng hưởng cao (khoảng vài MHz), phù hợp với khả năng phát hiện sự thay đổi tần số do kim loại ảnh hưởng đến trường điện từ của cuộn dây.

Về cuộn cảm: Với 30 vòng quấn đều trên ống 2 cm, cuộn dây đạt cảm kháng vào khoảng 1–2 μH , nằm trong dải phù hợp để kết hợp với tụ 100pF cho tần số cộng hưởng cao mà ESP8266 vẫn có thể đo được gián tiếp thông qua mạch điện.

Về tụ điện: Tụ 100pF là loại phổ biến, giá rẻ, có sai số thấp và thường được sử dụng trong các mạch dao động tần số cao. Giá trị nhỏ giúp tăng tần số cộng hưởng, đồng thời giảm thiểu ảnh hưởng của nhiễu nền trong môi trường thực nghiệm.

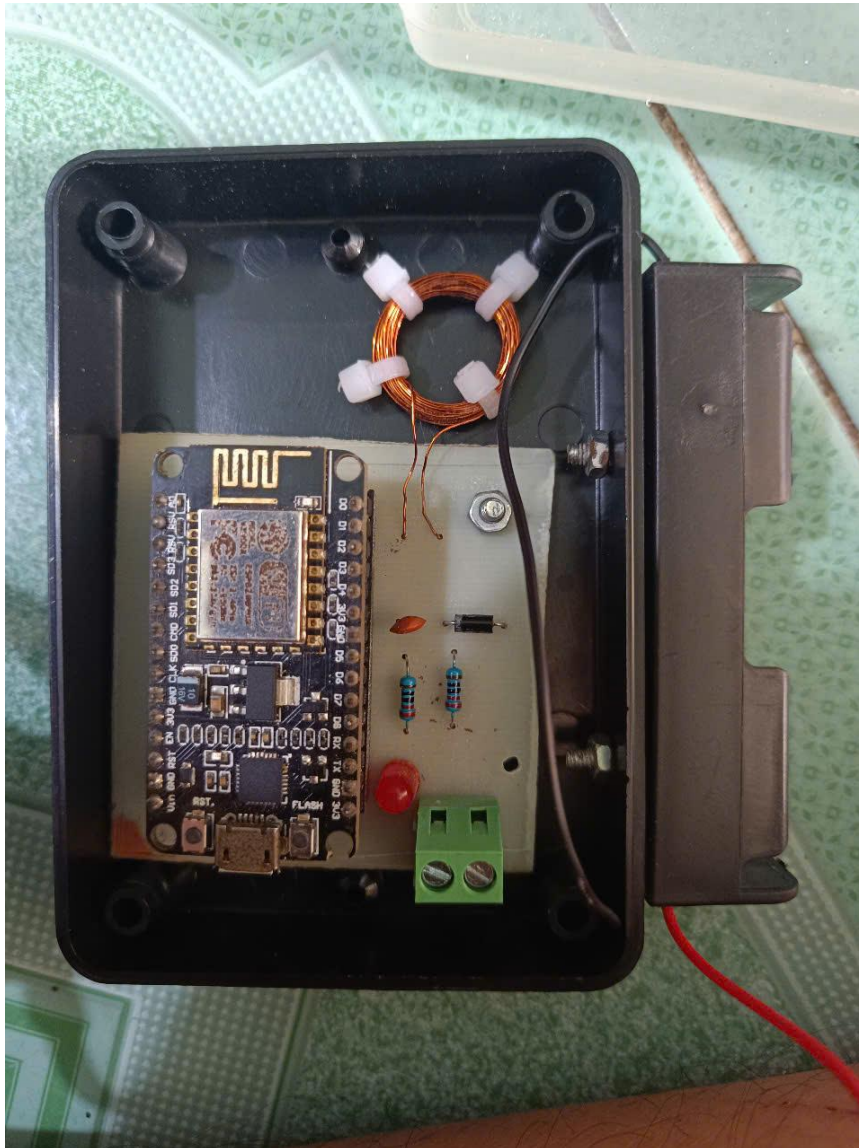
Việc lựa chọn tổ hợp linh kiện này giúp hệ thống vừa dễ thi công, chi phí thấp, lại đáp ứng yêu cầu về tần số, đảm bảo độ nhạy khi phát hiện vật kim loại tiến gần cuộn dây. Đây cũng là cấu hình cân đối giữa độ ổn định, khả năng xử lý tín hiệu và điều kiện thực tế tại phòng thí nghiệm.

4.3. Phần mềm hệ thống

Hệ thống phần mềm trong đề tài được triển khai nhằm đảm nhiệm toàn bộ quy trình từ thu thập dữ liệu đến hiển thị thông tin cho người dùng. Vi điều khiển ESP8266 đóng vai trò trung tâm, thực hiện việc đọc tín hiệu từ cảm biến LC, xử lý số vòng quay tương ứng với lượng nước tiêu thụ và đóng gói dữ liệu dưới dạng JSON gồm thời gian, số vòng quay và thể tích nước. Dữ liệu sau đó được gửi lên Firebase Realtime Database thông qua kết nối Wi-Fi. Firebase đóng vai trò là nền tảng lưu trữ trung tâm, hỗ trợ đồng bộ dữ liệu theo thời gian thực giữa các thiết bị. Mọi thay đổi từ vi điều khiển sẽ được cập nhật ngay lập tức đến tất cả ứng dụng theo dõi mà không cần tải lại thủ công. Bảo mật dữ liệu được đảm bảo bằng hệ thống Firebase Rules, giới hạn quyền truy cập dựa trên UID người dùng, nghĩa là một cảm biến chỉ được dùng cho một tài khoản mới có thể đọc hoặc ghi dữ liệu của chính họ. Ứng dụng theo dõi được phát triển bằng React Native với ngôn ngữ JavaScript, sử dụng Firebase SDK để kết nối cơ sở dữ liệu và hiển thị thông tin theo thời gian thực, bao gồm số vòng quay đo được, tổng lượng nước tiêu thụ và phân quy đổi ra chi phí sử dụng theo đơn giá năm 2025. Giao diện ứng dụng được tối ưu cho thiết bị di động, giúp người dùng dễ dàng theo dõi mức tiêu thụ nước một cách trực quan và tiện lợi.

CHƯƠNG 5: KẾT QUẢ THỰC NGHIỆM

5.1: Mô hình phần cứng và thực tế



Hình 5.1: Mô hình phần cứng

Bo mạch được thiết kế gọn gàng trong một hộp nhựa kín, giúp bảo vệ linh kiện khỏi bụi và độ ẩm, phù hợp cho các ứng dụng ngoài trời hoặc môi trường công nghiệp. Module WiFi ESP8266 (NodeMCU) được lắp cố định bên trái, đảm nhiệm chức năng xử lý và truyền dữ liệu không dây.

Cảm biến cuộn dây phát hiện kim loại được gắn chắc chắn ở phía trên hộp bằng các kẹp nhựa định vị, đảm bảo vị trí ổn định và tín hiệu đầu vào không bị nhiễu. Phần

mạch cảm biến bao gồm các điện trở, diode và đèn LED báo trạng thái, được bố trí thẳng hàng trên PCB giúp dễ quan sát và kiểm tra khi cần bảo trì.

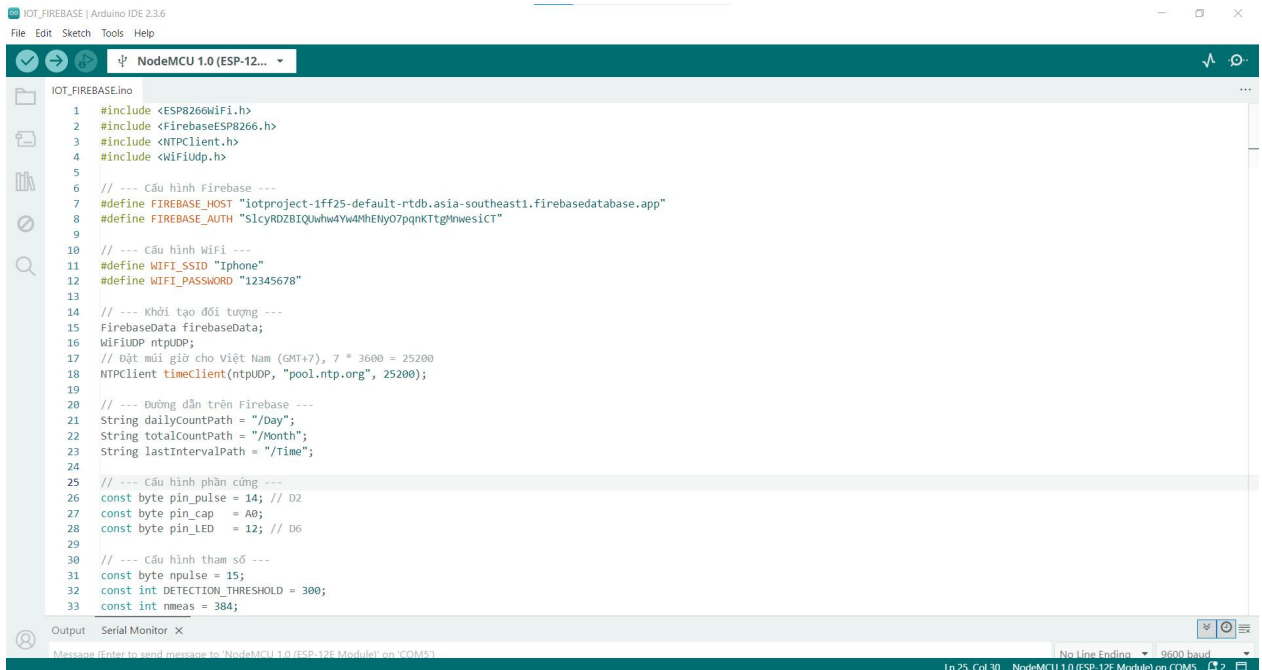
Dây tín hiệu và dây cấp nguồn được đưa vào hộp qua các đầu nối chắc chắn, hạn chế lỏng hoặc đứt trong quá trình sử dụng. Tổng thể bố cục mạch được tối ưu hóa cả về kích thước và kết nối, đảm bảo thuận tiện trong lắp đặt vào các thiết bị hoặc hệ thống đo đạc thực tế.



Hình 5.2: Mô hình phân cứng

5.2. Quá trình thiết lập cho mạch

Sau khi được nạp chương trình điều khiển, vi điều khiển ESP8266 thực hiện chức năng đọc tín hiệu từ cảm biến LC thông qua chân ADC hoặc digital. Tín hiệu dao động thu được phản ánh sự thay đổi tần số hoặc biên độ khi có chuyển động quay gần cuộn cảm. Chương trình sẽ liên tục lấy mẫu tín hiệu, tính toán giá trị biên độ hoặc đếm số xung trong một khoảng thời gian nhất định để xác định xem có vòng quay xảy ra hay không.



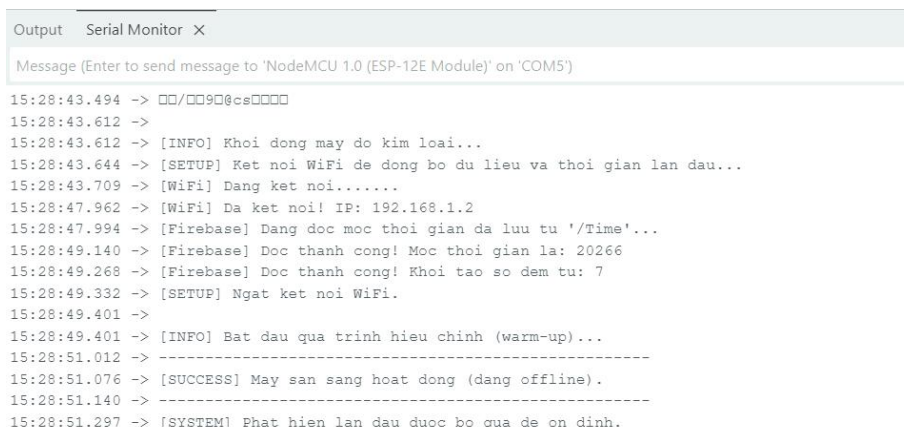
Hình 5.3: Giao diện chương trình

5.3. Hệ thống cảm biến và gửi dữ liệu lên Firebase

5.3.1. Chế độ cảm biến

Sau khi được nạp chương trình và khởi động hệ thống thì hệ thống bắt đầu kết nối với Wifi đang cài đặt khi nạp chương trình để đọc dữ liệu từ Firebase để hệ thống hoạt động luôn đồng bộ với nhau , khi đồng bộ thành công thì hệ thống sẽ ngắt kết nối với wifi và Firebase để tối ưu nguồn tài nguyên năng lượng và dữ liệu đồng thời việc ngắt dữ liệu Wifi sẽ hạn chế nhiễu từ sóng điện từ vì hệ thống sử dụng cảm biến LC.

Sau đó hệ thống sẽ khởi động mạch cảm biến LC để phát hiện sự xuất hiện của vật kim loại dựa trên nguyên lý thay đổi tần số dao động điện từ.



Trong quá trình đo, cảm biến liên tục ghi lại các giá trị dao động và tính toán tần số trung bình từ các mẫu gần nhất.

5.3.1. Chế độ cập nhật dữ liệu lên Firebase

Khi không có kim loại tiến gần, hệ thống giữ trạng thái ổn định với các giá trị dao động xung quanh mức trung bình xác định. Đây là vùng dao động nền, thể hiện điều kiện bình thường của hệ thống.

```
15:33:28.473 -> >>> Da phat hien kim loai! Khoi dong cap nhat len Firebase...
15:33:28.537 ->
15:33:28.537 -> [SYSTEM] Bat dau qua trinh cap nhat Firebase...
15:33:28.610 -> [WiFi] Dang ket noi.....
15:33:32.810 -> [WiFi] Da ket noi! IP: 192.168.1.2
15:33:32.841 -> [Firebase] Dang doc va cap nhat tong so (Month)...
15:33:34.208 -> [Firebase] Cap nhat 'Month' thanh cong: 10
15:33:34.475 -> [Firebase] Dang doc gia tri 'Day' hien tai...
15:33:34.599 -> [Firebase] Doc 'Day' thanh cong: 9
15:33:34.663 -> [Firebase] Dang cap nhat 'Day' len gia tri moi: 10
15:33:34.830 -> [Firebase] Cap nhat 'Day' thanh cong: 10
15:33:34.861 -> [WiFi] Ngat ket noi WiFi de tiet kiem nang luong.
15:33:34.925 -> [SYSTEM] Da hoan tat. Quay lai che do cam bien.
15:33:34.990 -> -----
```

Khi kim loại tiến gần đến cuộn cảm, tần số dao động thay đổi đột ngột khiến hệ thống phát hiện sự bất thường. Nếu biên độ sai lệch vượt ngưỡng cho phép, đèn LED sẽ được kích hoạt và hệ thống bắt đầu quá trình cập nhật lên Firebase. Hệ hống sẽ kết nối Wifi và Firebase sau đó hệ thống sẽ đọc dữ liệu từ FireBase, căn cứ vào dữ liệu vừa đọc từ dữ liệu “Day”, “Month” và “Time” trên Firebase hệ thống xử lý.

Hệ thống so sánh giá trị “Time” được lưu trên Firebase và giá trị “Time” hiện tại hệ thống vừa phát hiện để biết lần phát hiện này có trong ngày hay không. Nếu không trong ngày thì hệ thống sẽ reset giá trị “Day” về 0 và cộng dồn giá trị “Month” để biểu thị cho tổng số lần phát hiện. Nếu trong ngày thì hệ thống sẽ tăng giá trị “Day” và “Month” để biểu thị cho tổng số lần phát hiện trong ngày và tổng số lần phát hiện.

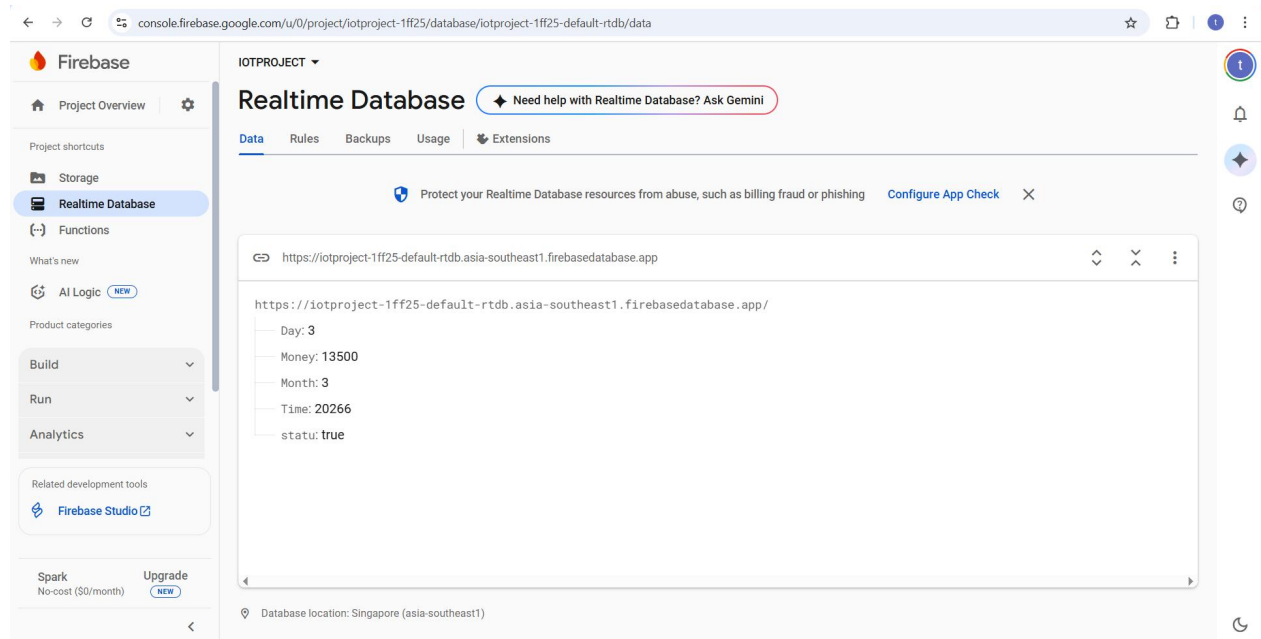
Hệ thống sẽ tự động ghi nhận và đẩy dữ liệu tần số dao động mỗi lần có sự thay đổi vượt ngưỡng lên Firebase, phục vụ cho việc giám sát từ xa.

5.4. Lưu trữ dữ liệu Firebase

Khi hệ thống hoạt động thì sẽ gửi dữ liệu lên Firebase, dữ liệu được lưu ở “Realtime Database” của Firebase với dữ liệu Day biểu diễn tổng số nước (m³) sử dụng

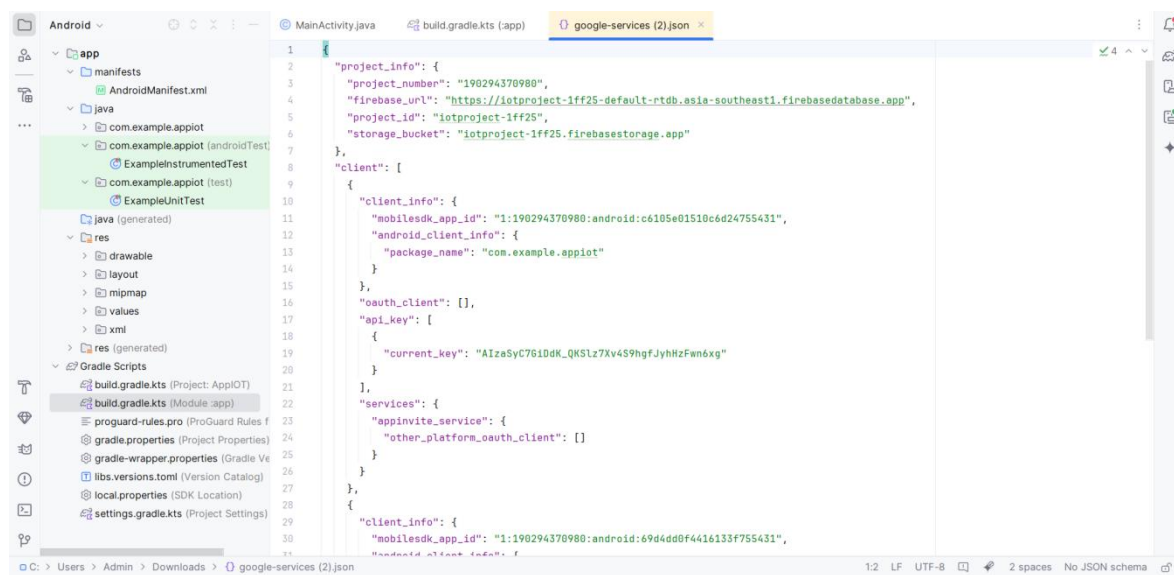
trong ngày, dữ liệu Month biểu diễn tổng số nước (m^3) sử dụng trong tháng, dữ liệu Time là giá trị để so sánh để biết được các lần đo có cùng trong 24h không.

Cuối cùng là giá trị Money là giá tiền/ (m^3) nước và statu là trạng thái của hệ thống cấp nước có hoạt động bình thường không. 2 dữ liệu này sẽ được nhà cung cấp thay đổi trên Firebase để cập nhập từ xa cho người dùng thông qua ứng dụng điện thoại .



Hình 5.5: Sau khi đưa dữ liệu lên FireBase

5.4. Ứng dụng theo dõi mức sử dụng nước từ xa

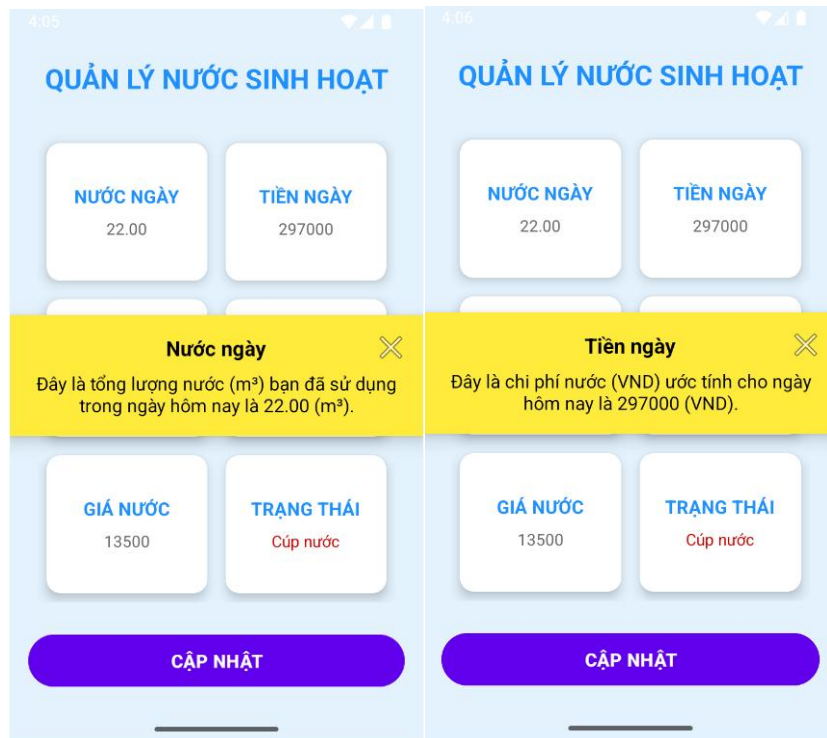


Hình 5.6: Cấu hình google-service.json của Firebase cho app



Hình 5.7: Giao diện chính của ứng dụng

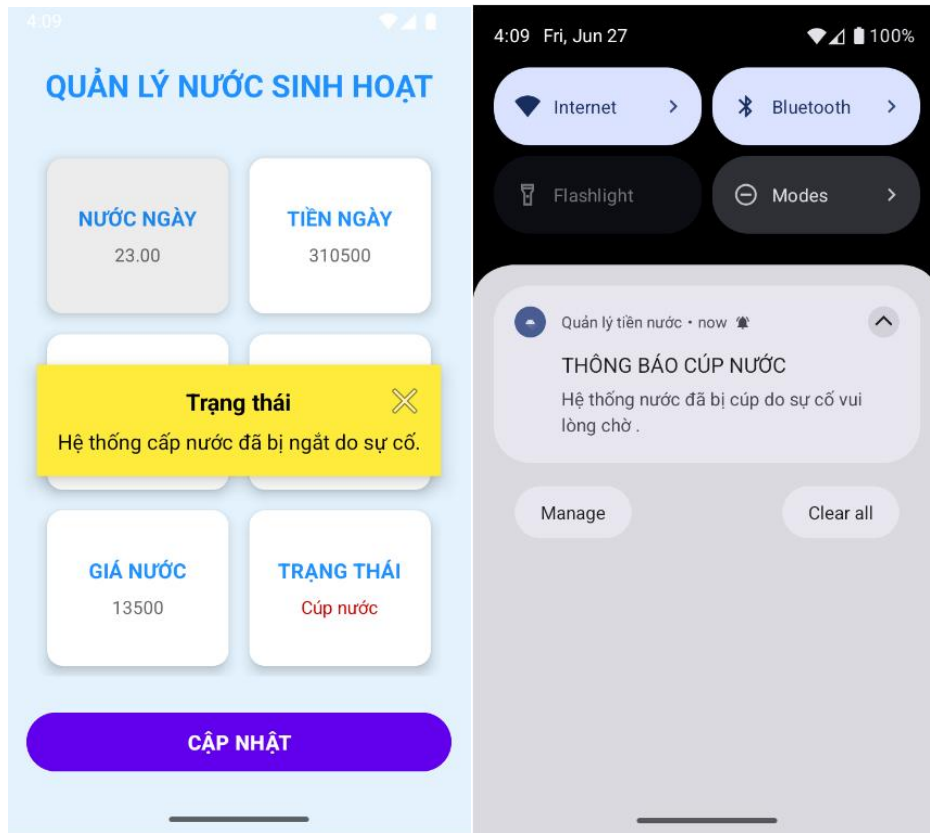
Ứng dụng sẽ giúp người dùng theo dõi lượng nước sử dụng sinh hoạt từ xa không cần đọc đồng hồ số nước. Hệ thống sẽ tự cập nhật số nước sử dụng lên Firebase theo ngày và tổng lượng nước. Ứng dụng quản lý sẽ đọc các dữ liệu từ Firebase và ứng dụng sẽ đọc hiển thị cho người dùng một cách chi tiết nhất về các thông số cần biết như tổng số nước sử dụng ngày hôm nay và từ trước đến nay, hay tổng tiền nước sử dụng nước ngày hôm nay và từ trước đến nay.



Hình 5.7: Giao của ứng dụng khi chọn mục muốn xem chi tiết

Hơn nữa ứng dụng sẽ cập nhập cho chúng ta biết đơn giá trên mỗi (m^3) nước và trạng thái hoạt động của hệ thống cấp nước theo thời gian thực để nhà cup cấp có thể thông báo cho người dùng.

Khi hệ thống cấp nước bị sự cố thì nhà cung cấp sẽ cập nhập trên Firebase và người dùng sẽ nhận thông báo nổi lên điện thoại biết được hệ thống đang bị sự cố.



Hình 5.7: Giao của ứng dụng khi hệ thống cấp nước bị lỗi

5.4.4 Điện năng tiêu thụ của nguồn

Hệ thống sử dụng vi điều khiển ESP8266 để xử lý dữ liệu và truyền tải thông tin qua Wi-Fi, kết hợp với cảm biến LC để đo lưu lượng nước. Thiết bị được cấp nguồn bằng pin 18650 dung lượng 3.7V – 2200 mAh, cho phép hoạt động độc lập mà không cần điện lưới.

Trong quá trình thực hiện đề tài, nhóm đã tối ưu phần mềm và sử dụng chế độ Deep Sleep cho ESP8266, giúp giảm đáng kể điện năng tiêu thụ. Cụ thể:

- ESP8266 chỉ bật trong vài giây để đo và gửi dữ liệu.
- Thời gian còn lại, chip chuyển sang Deep Sleep, tiêu thụ cực thấp.
- Cảm biến LC tiêu thụ liên tục ở mức rất nhỏ: khoảng 0.005 mA.

Mức tiêu thụ trung bình mỗi ngày:

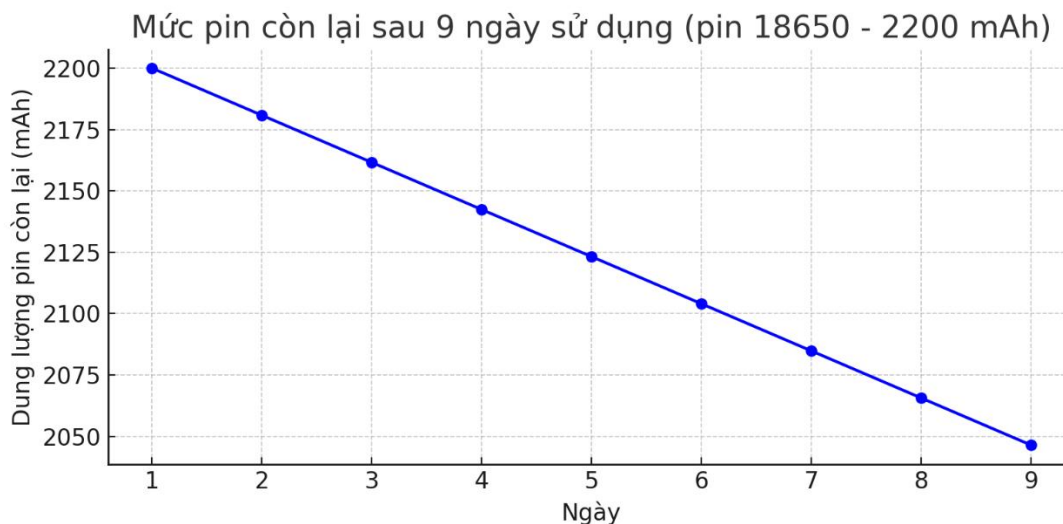
- Dòng tiêu thụ tổng cộng ước tính: 0.8 mAh (ESP khi bật) + 0.005 mAh (LC liên tục) \approx 20 mAh/ngày (sau khi tính tổng chu kỳ hoạt động/ngủ)

Công thức tính thời gian hoạt động:

Thời gian sử dụng (ngày) = $\frac{\text{Dung lượng pin}}{\text{Tiêu thụ trung bình mỗi ngày}} = \frac{2200}{20} = 110 \text{ ngày}$
 $\approx 3.5 \text{ tháng}$

Ưu điểm:

- Không cần nguồn điện lưới, phù hợp với các vị trí lắp đặt xa nguồn hoặc ngoài trời.
- Hoạt động bền bỉ dài ngày, giảm công bảo trì và sạc pin thường xuyên.
- Khả năng nâng cấp: Có thể kết hợp thêm tấm pin năng lượng mặt trời để sạc tự động, hướng tới giải pháp IoT tự chủ hoàn toàn



Bảng 5.3 Thống kê về điện năng tiêu thụ trong 7 ngày

CHƯƠNG 6: KẾT LUẬN VÀ KIẾN NGHỊ

6.1. Kết luận

Trong quá trình thực hiện đề tài, nhóm đã cố gắng thiết kế và chế tạo hệ thống đo số vòng quay của đồng hồ nước cơ bằng cảm biến LC, sử dụng vi điều khiển ESP8266 và truyền dữ liệu qua Firebase. Mạch cảm biến đã được thiết kế mô hình riêng, tuy nhiên vẫn còn xảy ra hiện tượng nhiễu trong một số trường hợp, đặc biệt khi môi trường có nhiễu thiết bị điện. Thiết kế nguồn phải ổn định, kim quá bé thì rất khó để mà xác định được

Phần mềm theo dõi được nhóm phát triển bằng React Native, sử dụng Firebase SDK để kết nối đến cơ sở dữ liệu và hiển thị các thông tin về số vòng quay và lượng nước tiêu thụ. Ứng dụng hoạt động tốt trong môi trường có kết nối Wifi, nên khi mất mạng thì dữ liệu sẽ bị reset

Dù hệ thống chưa hoàn chỉnh 100% so với dự kiến ban đầu, nhưng nhóm vẫn đánh giá đây là một bước đi khả quan. Việc kết hợp cả phần cứng và phần mềm đã giúp nhóm hiểu rõ hơn về nguyên lý hoạt động của cảm biến LC, cách xử lý tín hiệu và kết nối dữ liệu thời gian thực.

6.2. Kiến nghị

Trong tương lai, tiếp tục phát triển và hoàn thiện ở cả hai khía cạnh phần cứng và phần mềm để nâng cao độ chính xác, độ ổn định và khả năng ứng dụng thực tế. Về phần cứng, cần tập trung cải tiến mạch cảm biến LC theo hướng ổn định hơn, giảm thiểu tối đa nhiễu do môi trường xung quanh gây ra, đặc biệt là trong các khu vực có nhiều thiết bị điện tử hoạt động. Việc áp dụng các module xử lý tín hiệu chuyên dụng hoặc bộ lọc nhiễu có thể giúp hệ thống nhận diện dao động rõ ràng và chính xác hơn, từ đó tăng độ tin cậy của dữ liệu đo được. Nhóm cũng mong muốn có thêm thời gian để cải thiện phần mềm ứng dụng, tăng cường bảo mật, tăng thêm dung lượng lưu trữ và xử lý dữ liệu và tăng thêm lưu lượng truy cập đồng thời trên hệ thống.

6.3. Hướng phát triển tiếp theo

Hệ thống dò kim đồng hồ nước cơ bằng mạch cộng hưởng LC kết hợp với ESP8266 có tiềm năng trở thành một giải pháp giám sát lượng nước tiêu thụ đơn giản, chi phí thấp và dễ triển khai rộng rãi.

Trong giai đoạn tiếp theo, nhóm định hướng phát triển hệ thống theo hướng ứng dụng thực tế, phục vụ nhiều đối tượng sử dụng như hộ gia đình, nhà trọ, khu dân cư hoặc các cơ sở kinh doanh nhỏ. Thiết bị có thể được đóng gói thành module gọn nhẹ, dễ lắp đặt trên đồng hồ nước hiện có mà không cần can thiệp vào kết cấu đường ống.

Giao diện theo dõi có thể được tích hợp vào ứng dụng điện thoại hoặc nền tảng web, giúp người dùng dễ dàng kiểm tra lượng nước tiêu thụ hàng ngày và phát hiện bất thường như rò rỉ hoặc sử dụng quá mức. Đặc biệt, với ưu điểm giá thành thấp và sử dụng linh kiện phổ biến, hệ thống có thể nhân rộng dễ dàng trong cộng đồng, góp phần thúc đẩy ý thức tiết kiệm nước và quản lý tài nguyên hiệu quả hơn.

TÀI LIỆU THAM KHẢO

- [1] **Sở Khoa học & Công nghệ TP.HCM.** (2023). *Chế tạo thành công thiết bị đo lường nước thông minh trên nền tảng đồng hồ cơ truyền thống, tích hợp quản lý tập trung IoT*. Truy cập từ: <https://dost.hochiminhcity.gov.vn>
- [2] **Nguyen, T. T., Pham, Q. C., et al.** (2021). *A smart water metering system using FireBase and edge computing for water resource management*. **HardwareX**, **9**. Truy cập từ: <https://www.sciencedirect.com/science/article/pii/S2590005621000047>
- [3] **Phong, N. H., Phuc, N. V., Huy, N. M., Dung, P. C., & Phuong, L. M.** (2022). *Development and Implementation of Smart Water Metering System based on LoRa Technology*. **Science & Technology Development Journal – Engineering and Technology**, **5(1)**, 1342–1370. <https://doi.org/10.32508/stdjet.v5i1.955>
- [4] **Google.** (2024). *Get started with Firebase for Android*. Truy cập từ: <https://firebase.google.com/docs/android/setup>
- [5] **Random Nerd Tutorials.** (2023). *ESP8266: Send Data to Firebase Realtime Database*. Truy cập từ: <https://randomnerdtutorials.com/esp8266-firebase-realtime-database/>
- [6] **Espressif Systems.** (2023). *ESP8266 Technical Reference Manual*. Truy cập từ: https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf
- [7] **Martin, T.** (2022). *Practical ESP8266 IoT Projects with Arduino IDE and Firebase*. **Packt Publishing**.
- [8] **Nguyen, D. H., & Le, T. P.** (2020). *Ứng dụng công nghệ IoT trong đo lường và giám sát tiêu thụ nước tại hộ gia đình*. **Tạp chí Khoa học & Công nghệ Đại học Cần Thơ**.

PHỤ LỤC

Code cuộn cảm biến LC:

```
#include <ESP8266WiFi.h>

#include <FirebaseESP8266.h>

#include <NTPClient.h>

#include <WiFiUdp.h>


#define FIREBASE_HOST "iotproject-1ff25-default-rtdb.asia-southeast1.firebaseio.com"

#define FIREBASE_AUTH "SlcyRDZBIQUwhw4Yw4MhENyO7pqnKTtgMnwesiCT"

#define WIFI_SSID "Iphone"

#define WIFI_PASSWORD "12345678"


FirebaseData firebaseData;

WiFiUDP ntpUDP;

NTPClient timeClient(ntpUDP, "pool.ntp.org", 25200); // Đặt múi giờ cho Việt Nam (GMT+7), 7 * 3600 = 25200

String dailyCountPath = "/Day";

String totalCountPath = "/Month";

String lastIntervalPath = "/Time";


const byte pin_pulse = 14; // D2
const byte pin_cap = A0;
const byte pin_LED = 12; // D6


// --- Cấu hình tham số ---
const byte npulse = 15;
const int DETECTION_THRESHOLD = 300;
```

```

const int nmeas = 384;

const unsigned long DETECTION_COOLDOWN = 800; // Thời gian chờ giữa các lần
phát hiện (ms)

long int metalDetectedCount = 0;
long lastResetInterval = -1;
bool metalPreviouslyDetected = false;
bool boQuaLanDau = true;
long int sumsum = 0;
long int diff = 0;
unsigned long lastDetectionTime = 0;
long int skip = 0;

void handleFirebaseUpdate() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  int timeout = 0;
  while (WiFi.status() != WL_CONNECTED && timeout < 20000) {
    delay(500);
    timeout += 500;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

    // --- BƯỚC 1: CẬP NHẬT "MONTH" (TỔNG SỐ) ---
    long currentTotalCount = 0;
    if (Firebase.getInt(firebaseData, totalCountPath)) {
      currentTotalCount = firebaseData.intData();
    }
  }
}

```

```

    }
    currentTotalCount++;
    Firebase.setInt(firebaseData, totalCountPath, currentTotalCount);

    // --- BƯỚC 2: KIỂM TRA THỜI GIAN VÀ XỬ LÝ "DAY" (SỐ TRONG
    NGÀY/PHIÊN) ---
    timeClient.update();
    long currentInterval = timeClient.getEpochTime() / 86400;

    long lastResetIntervalFromFB = -1;
    if(Firebase.getInt(firebaseData, lastIntervalPath)){
        lastResetIntervalFromFB = firebaseData.intData();
    }

    bool isNewInterval = (currentInterval != lastResetIntervalFromFB);

    if (isNewInterval) {
        // Nếu là khoảng thời gian mới, reset "Day" về 1
        metalDetectedCount = 1;
        Firebase.setInt(firebaseData, dailyCountPath, 1);
        Firebase.setInt(firebaseData, lastIntervalPath, currentInterval);
        lastResetInterval = currentInterval;
    } else {
        // Nếu vẫn trong khoảng thời gian cũ, đọc giá trị 'Day' hiện tại, cộng 1, rồi cập nhật
        long currentDailyCount = 0;
        if (Firebase.getInt(firebaseData, dailyCountPath)) {
            currentDailyCount = firebaseData.intData();
        }
        currentDailyCount++;
    }

```

```

    if (Firebase.setInt(firebaseData, dailyCountPath, currentDailyCount)) {
        metalDetectedCount = currentDailyCount; // Đồng bộ lại biến cục bộ
    }
}
}
WiFi.disconnect(true);
WiFi.mode(WIFI_OFF);
}

```

```

// Hàm đo lường (giữ nguyên)
long int measureAndProcess() {
    int minval = 2000;
    int maxval = 0;
    unsigned long sum = 0;
    for (int imeas = 0; imeas < nmeas + 2; imeas++) {
        pinMode(pin_cap, OUTPUT);
        digitalWrite(pin_cap, LOW);
        delayMicroseconds(40);
        pinMode(pin_cap, INPUT);
        for (int ipulse = 0; ipulse < npulse; ipulse++) {
            digitalWrite(pin_pulse, HIGH); delayMicroseconds(3);
            digitalWrite(pin_pulse, LOW); delayMicroseconds(3);
        }
        int val = analogRead(pin_cap);
        minval = min(val, minval);
        maxval = max(val, maxval);
        sum += val;
    }
    sum -= minval;
}

```

```

    sum -= maxval;
    return sum;
}

void setup() {
    pinMode(pin_pulse, OUTPUT);
    digitalWrite(pin_pulse, LOW);
    pinMode(pin_LED, OUTPUT);
    digitalWrite(pin_LED, LOW);
    WiFi.mode(WIFI_STA);
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    int timeout = 0;
    while (WiFi.status() != WL_CONNECTED && timeout < 20000) {
        delay(500);
        timeout += 500;
    }
    if (WiFi.status() == WL_CONNECTED) {
        Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
        timeClient.begin();
        if (Firebase.getInt(firebaseData, lastIntervalPath)) {
            lastResetInterval = firebaseData.intData();
        } else {
            timeClient.update();
            lastResetInterval = timeClient.getEpochTime() / 86400;
            Firebase.setInt(firebaseData, lastIntervalPath, lastResetInterval);
        }
        if (Firebase.getInt(firebaseData, dailyCountPath)) {
            metalDetectedCount = firebaseData.intData();
        } else {

```

```

    metalDetectedCount = 0;
}
}

WiFi.disconnect(true);
WiFi.mode(WIFI_OFF);
for (int i = 0; i < 10; i++) {
    measureAndProcess();
    delay(50);
}
sumsum = 0;
}

void loop() {
    long int currentSum = measureAndProcess();
    if (sumsum == 0) sumsum = currentSum << 6;
    long int avgsum = (sumsum + 32) >> 6;
    diff = currentSum - avgsum;
    if (abs(diff) < (avgsum >> 10)) {
        sumsum = sumsum + currentSum - avgsum;
        skip = 0;
    } else {
        skip++;
    }
    if (skip > 64) {
        sumsum = currentSum << 6;
        skip = 0;
    }

    unsigned long currentTime = millis();

```

```

if (abs(diff) > DETECTION_THRESHOLD) {
    digitalWrite(pin_LED, HIGH);
    if (!metalPreviouslyDetected && (currentTime - lastDetectionTime >
DETECTION_COOLDOWN)) {
        if (boQuaLanDau) {
            boQuaLanDau = false;
        } else {
            // Chỉ gọi hàm `handleFirebaseUpdate` để nó tự xử lý việc đọc-cộng-ghi.
            handleFirebaseUpdate();
        }
        metalPreviouslyDetected = true;
        lastDetectionTime = currentTime;
    }
} else {
    digitalWrite(pin_LED, LOW);
    metalPreviouslyDetected = false;
}
}

```

Code ứng dụng JavaScript bằng Androi studio

```

package com.example.apptot;

import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;

```

```

import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;
import androidx.core.app.NotificationCompat;
import androidx.core.content.ContextCompat;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.HashMap;
import java.util.Map;

public class MainActivity extends AppCompatActivity {
    private TextView tvDailyWater, tvDailyCost, tvTotalWater, tvTotalDebt, tvWaterPrice,
    tvStatus;
    private Button btnRefresh;
    private CardView cardDailyWater, cardDailyCost, cardTotalWater, cardTotalDebt,
    cardWaterPrice, cardStatus;

    private ViewGroup popupInfoContainer;
    private TextView tvPopupTitle;
    private TextView tvPopupMessage;
    private ImageButton btnClosePopup;

    private DatabaseReference databaseReference;
    private static final String CHANNEL_ID = "water_notification_channel";
    private boolean isWaterCutNotified = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tvDailyWater = findViewById(R.id.tvDailyWater);
        tvDailyCost = findViewById(R.id.tvDailyCost);
        tvTotalWater = findViewById(R.id.tvTotalWater);
        tvTotalDebt = findViewById(R.id.tvTotalDebt);
        tvWaterPrice = findViewById(R.id.tvWaterPrice);
        tvStatus = findViewById(R.id.tvStatus);
        btnRefresh = findViewById(R.id.btnRefresh);

```

```

cardDailyWater = findViewById(R.id.cardDailyWater);
cardDailyCost = findViewById(R.id.cardDailyCost);
cardTotalWater = findViewById(R.id.cardTotalWater);
cardTotalDebt = findViewById(R.id.cardTotalDebt);
cardWaterPrice = findViewById(R.id.cardWaterPrice);
cardStatus = findViewById(R.id.cardStatus);

popupInfoContainer = findViewById(R.id.popupInfoContainer);
tvPopupTitle = popupInfoContainer.findViewById(R.id.tvToastTitle);
tvPopupMessage = popupInfoContainer.findViewById(R.id.tvToastMessage);
btnClosePopup = popupInfoContainer.findViewById(R.id.btnCloseToast);

btnClosePopup.setOnClickListener(v ->
popupInfoContainer.setVisibility(View.GONE));

// Khởi tạo Firebase Database
databaseReference = FirebaseDatabase.getInstance().getReference();

// Thiết lập các chức năng
setupNotificationChannel();
setupFirebaseListeners();
setupRefreshButton();
setupCardClickListeners();
}

private double getDoubleValueFromSnapshot(DataSnapshot dataSnapshot) {
    if (dataSnapshot.exists()) {
        Object valueObj = dataSnapshot.getValue();
        if (valueObj instanceof Long) {
            return ((Long) valueObj).doubleValue();
        } else if (valueObj instanceof Double) {
            return (Double) valueObj;
        }
    }
    return 0.0;
}

private void setupFirebaseListeners() {
    databaseReference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            double dailyWater = getDoubleValueFromSnapshot(snapshot.child("Day"));
            double totalWater = getDoubleValueFromSnapshot(snapshot.child("Month"));

```

```

double waterPrice = getDoubleValueFromSnapshot(snapshot.child("Money"));

double dailyCost = dailyWater * waterPrice;
double totalDebt = totalWater * waterPrice;

tvDailyWater.setText(String.format("%.2f", dailyWater));
tvTotalWater.setText(String.format("%.2f", totalWater));
tvWaterPrice.setText(String.format("%.0f", waterPrice));

tvDailyCost.setText(String.format("%.0f", dailyCost));
tvTotalDebt.setText(String.format("%.0f", totalDebt));

Boolean status = snapshot.child("statu").getValue(Boolean.class);

if (status != null && status) {
    tvStatus.setText("Bình thường");
    tvStatus.setTextColor(ContextCompat.getColor(MainActivity.this,
android.R.color.holo_green_dark));
    isWaterCutNotified = false;
} else {
    tvStatus.setText("Cúp nước");
    tvStatus.setTextColor(ContextCompat.getColor(MainActivity.this,
android.R.color.holo_red_dark));
    showWaterCutNotification();
}
}

@Override
public void onCancelled(@NonNull DatabaseError error) {
    showErrorToast("Lỗi tải dữ liệu từ Firebase");
}
});
}

private void setupCardClickListeners() {
    cardDailyWater.setOnClickListener(v -> {
        String currentValue = tvDailyWater.getText().toString();
        String message = String.format("Đây là tổng lượng nước (m³) bạn đã sử dụng
trong ngày hôm nay là %s (m³).", currentValue);
        showInfoPopup("Nước ngày", message);
    });

    cardDailyCost.setOnClickListener(v -> {
        String currentValue = tvDailyCost.getText().toString();

```

```

        String message = String.format("Đây là chi phí nước (VND) ước tính cho ngày  
hôm nay là %s (VND).", currentValue);
        showInfoPopup("Tiền ngày", message);
    });

    cardTotalWater.setOnClickListener(v -> {
        String currentValue = tvTotalWater.getText().toString();
        String message = String.format("Đây là tổng lượng nước (m³) bạn đã sử dụng tính  
đến kỳ này là %s (m³).", currentValue);
        showInfoPopup("Tổng nước", message);
    });

    cardTotalDebt.setOnClickListener(v -> {
        String currentValue = tvTotalDebt.getText().toString();
        String message = String.format("Đây là tổng số tiền nước (VND) bạn cần thanh  
toán cho kỳ này là %s (VND).", currentValue);
        showInfoPopup("Tổng tiền", message);
    });

    cardWaterPrice.setOnClickListener(v -> {
        String currentValue = tvWaterPrice.getText().toString();
        String message = String.format("Đây là đơn giá hiện tại cho mỗi mét khối (m³)  
nước là %s (VND).", currentValue);
        showInfoPopup("Giá nước", message);
    });

    cardStatus.setOnClickListener(v -> {
        String statusMessage = tvStatus.getText().toString().equals("Bình thường")
            ? "Hệ thống cấp nước đang hoạt động ổn định."
            : "Hệ thống cấp nước đã bị ngắt do sự cố.";
        showInfoPopup("Trạng thái", statusMessage);
    });
}

private void showInfoPopup(String title, String message) {
    tvPopupTitle.setText(title);
    tvPopupMessage.setText(message);
    popupInfoContainer.setVisibility(View.VISIBLE);
}

// --- ĐÂY LÀ HÀM ĐÃ ĐƯỢC SỬA ĐỔI ---
private void setupRefreshButton() {
    btnRefresh.setOnClickListener(v -> {
        // Hiển thị thông báo đang xử lý
    });
}

```

```

        Toast.makeText(this, "Đang reset dữ liệu...", Toast.LENGTH_SHORT).show();

        // Tạo một Map để chứa các giá trị cần cập nhật
        Map<String, Object> updates = new HashMap<>();
        updates.put("/Day", 0L); // Dùng 0L để đảm bảo kiểu Long, tương thích với
        Firebase
        updates.put("/Month", 0L);

        // Gửi lệnh cập nhật nhiều trường cùng lúc tới Firebase
        databaseReference.updateChildren(updates)
            .addOnSuccessListener(aVoid -> {
                // Khi thành công, listener thời gian thực sẽ tự động cập nhật lại giao diện.
                // Chúng ta chỉ cần hiển thị một thông báo xác nhận.
                Toast.makeText(MainActivity.this, "Reset dữ liệu thành công!",
                    Toast.LENGTH_SHORT).show();
            })
            .addOnFailureListener(e -> {
                // Nếu có lỗi, thông báo cho người dùng
                Toast.makeText(MainActivity.this, "Reset thất bại: " + e.getMessage(),
                    Toast.LENGTH_LONG).show();
            });
    });
}

private void showWaterCutNotification() {
    if (!isWaterCutNotified) {
        NotificationManager notificationManager = (NotificationManager)
        getSystemService(NOTIFICATION_SERVICE);
        Intent intent = new Intent(this, MainActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
        Intent.FLAG_ACTIVITY_CLEAR_TASK);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent,
        PendingIntent.FLAG_IMMUTABLE);
        NotificationCompat.Builder builder = new NotificationCompat.Builder(this,
        CHANNEL_ID)
            .setSmallIcon(R.drawable.ic_launcher_foreground)
            .setContentTitle("THÔNG BÁO CÚP NƯỚC")
            .setContentText("Hệ thống nước đã bị cúp do sự cố vui lòng chờ .")
            .setPriority(NotificationCompat.PRIORITY_HIGH)
            .setContentIntent(pendingIntent)
            .setAutoCancel(true);
        if (notificationManager != null) {
            notificationManager.notify(1, builder.build());
        }
    }
}

```

```

        isWaterCutNotified = true;
    }
}

private void setupNotificationChannel() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence name = "Thông báo nước";
        String description = "Thông báo trạng thái hệ thống nước";
        int importance = NotificationManager.IMPORTANCE_HIGH;
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name,
importance);
        channel.setDescription(description);
        NotificationManager notificationManager =
getSystemService(NotificationManager.class);
        if (notificationManager != null) {
            notificationManager.createNotificationChannel(channel);
        }
    }
}

private void showErrorToast(String message) {
    Toast.makeText(MainActivity.this, message, Toast.LENGTH_SHORT).show();
}
}

```