

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM

KHOA ĐIỆN – ĐIỆN TỬ

BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG



**HCMUTE**

ĐỒ ÁN 1

**THIẾT KẾ VÀ THI CÔNG ĐỒNG HỒ ĐỂ BÀN  
ĐA NĂNG**

**Ngành Công Nghệ Kỹ Thuật Điện Tử - Viễn Thông (CLC)**

**Sinh viên: NGUYỄN PHẠM HUY HOÀNG**

MSSV: 22161125

**TRẦN NGUYỄN GIA HUY**

MSSV: 22161129

**GVHD: PGS.TS Phan Văn Ca**

**TP.HỒ CHÍ MINH, tháng 06 năm 2025**

# NHIỆM VỤ ĐỒ ÁN 1

## 1. Thông tin sinh viên

Họ và tên: Nguyễn Phạm Huy Hoàng MSSV: 22161125

Tel: 0392806131 Email: huyhoang123456789f@gmail.com

Họ và tên: Trần Nguyễn Gia Huy MSSV: 22161129

Tel: 0365089063 Email: huygia271204@gmail.com

## 2. Thông tin đề tài

Tên của đề tài: Thiết kế và thi công đồng hồ để bàn đa năng.

Mục đích của đề tài: Thiết kế và chế tạo một đồ hòng để bàn có nhiều chức năng hỗ trợ trong đời sống hàng ngày.

Đồ án 1 được thực hiện tại: Bộ môn Điện Tử Viễn Thông, Khoa Điện – Điện Tử, Trường Đại Học Sư Phạm Kỹ Thuật Thành Phố Hồ Chí Minh.

Thời gian thực hiện: Từ ngày 27/02/2025 đến 09/06/2025

## 3. Các nhiệm vụ cụ thể của đề tài

- Phân tích đề tài đồ án.
- Lựa chọn linh kiện, thiết bị phù hợp.
- Thiết kế phần cứng và lập trình hệ thống.
- Thi công, chạy thử và cải thiện hệ thống.
- Báo cáo.

## 4. Lời cam đoan của sinh viên

Chúng tôi – Nguyễn Phạm Huy Hoàng và Trần Nguyễn Gia Huy cam đoan Đồ Án 1 là công trình nghiên cứu của bản thân chúng tôi dưới sự hướng dẫn của PGS.TS Phan Văn Ca.

Các kết quả công bố trong Đồ Án 1 là trung thực và không sao chép từ bất kỳ công trình nào khác

Tp.HCM, ngày tháng 06 năm 2025

SV thực hiện đồ án

Nguyễn Phạm Huy Hoàng

Tp.HCM, ngày tháng 06 năm 2025

SV thực hiện đồ án

Trần Nguyễn Gia Huy

# PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Họ và tên Sinh viên: Nguyễn Phạm Huy Hoàng      MSSV: 22161125

Ngành: Công Nghệ Kỹ Thuật Điện tử - Viễn thông

Họ và tên Sinh viên: Trần Nguyễn Gia Huy      MSSV: 22161129

Ngành: Công Nghệ Kỹ Thuật Điện tử - Viễn thông

Tên đề tài: Thiết kế và thi công đồng hồ để bàn đa năng

Họ và tên Giáo viên hướng dẫn: PGS.TS Phan Văn Ca

## NHẬN XÉT

1. Về nội dung đề tài & khối lượng thực hiện:

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

2. Ưu điểm:

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

3. Khuyết điểm:

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

4. Đề nghị cho bảo vệ hay không?

.....

5. Đánh giá loại:

.....

6. Điểm: ..... Bằng chữ: .....

TP. Hồ Chí Minh, ngày..... tháng..... năm 2025

Giáo viên hướng dẫn

PGS.TS Phan Văn Ca

## LỜI CẢM ƠN

Để hoàn thành báo cáo Đồ Án 1, chúng em xin chân thành cảm ơn quý Thầy/Cô khoa Điện – Điện tử, Trường Đại học Sư phạm Kỹ thuật TP. Hồ Chí Minh đã tận tnhg giảng dạy và tạo điều kiện thuận lợi trong suốt quá trình học tập và thực hiện đồ án.

Đặc biệt, chúng em xin cảm ơn thầy Phan Văn Ca đã tận tình hướng dẫn, giúp đỡ và tạo điều kiện cho chúng em trong suốt quá trình thực hiện đồ án. Chúng em xin gửi đến thầy lời cảm ơn chân thành và sâu sắc nhất.

Mặc dù đã cố gắng hết sức nhưng không trách các thiếu sót trong quá trình thực hiện đồ án. Do vậy, chúng em rất mong nhận được sự góp ý của Thầy/Cô để có thể hoàn thiện và tốt hơn nữa cũng như tích lũy kinh nghiệm để hoàn thành tốt báo cáo đồ án 1 và cải thiện đồ án 2, đồ án tốt nghiệp sau này.

Cuối cùng, em kính chúc quý thầy cô thật dồi dào sức khỏe, luôn tràn đầy nhiệt huyết và gặt hái được những thành tựu rực rỡ trong sự nghiệp.

Chúng em xin chân thành cảm ơn!

## TÓM TẮT

Trong bối cảnh hiện tại, nhiều thiết bị trong cuộc sống ngày càng được tích hợp IoTs. Thế nên các thiết bị có nhiều chức năng và được tích hợp IoTs có vị trí cao hơn trong đời sống hàng ngày. Đề tài “Thiết kế và thi công đồng hồ để bàn đa năng” được thực hiện để tạo ra một thiết bị IoTs không chỉ có chức năng hiển thị thời gian mà còn có thể đo nhiệt độ, độ ẩm và các dữ liệu khác được hiển thị trên dashboard và điều khiển đặt báo thức qua dashboard. Hệ thống sử dụng vi điều khiển esp 8266 làm khôi điều khiển trung tâm, kết hợp module thời gian thực ds1307 và module cảm biến DHT22 với một màn hình LCD và linh kiện phụ trợ khác. Với thiết kế có kích thước nhỏ gọn, dễ sử dụng, phù hợp để đặt tại bàn làm việc hoặc trong nhà.

Việc thực hiện đề tài này sẽ giúp sinh viên củng cố kiến thức lý thuyết, đồng thời nâng cao kỹ năng giải quyết các vấn đề thường gặp khi thiết kế một thiết bị IoTs đơn giản, cũng là hành trang cho những đồ án sau này cũng như các dự án tương lai.

# MỤC LỤC

DANH MỤC HÌNH ẢNH .....	vì
DANH MỤC BẢNG .....	ix
DANH MỤC TỪ VIẾT TẮT .....	x
CHƯƠNG 1 TỔNG QUAN ĐỀ TÀI.....	1
1.1    GIỚI THIỆU .....	1
1.2    TÍNH CẤP THIẾT .....	1
1.3    MỤC TIÊU NGHIÊN CỨU .....	1
1.4    NHIỆM VỤ NGHIÊN CỨU.....	2
1.5    ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU .....	2
1.6    PHƯƠNG PHÁP NGHIÊN CỨU.....	2
1.7    BÓ CỤC ĐỀ TÀI .....	3
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT.....	4
2.1    TỔNG QUAN VỀ CÁC PHẦN CỨNG TRONG HỆ THỐNG.....	4
2.1.1    Nguồn Pin 9V .....	4
2.1.2    LM7805 IC Ôn Áp 5V/1.5A.....	4
2.1.3    LCD 20x04 kèm I2C Driver .....	6
2.1.4    Module RTC DS1307 .....	7
2.1.5    Module DHT22 .....	8
2.1.6    Buzzer .....	10
2.1.7    Module nút nhấn .....	10
2.1.8    NodeMCU ESP8266 .....	12
2.2    GIAO THỨC KẾT NỐI .....	14
2.2.1    UART.....	14
2.2.2    I2C .....	15
2.3    TỔNG QUAN VỀ FIREBASE .....	16
2.4    TỔNG QUAN VỀ WEB .....	17
2.5    TỔNG QUAN VỀ CÔNG CỤ PHẦN MỀM.....	17
2.5.1    Arduino IDE .....	17
2.5.2    Visual Studio Code.....	18
2.5.3    Altium Designer.....	19
2.6    CÁC THU VIỆN LẬP TRÌNH.....	20

2.6.1	Thư viện ESP8266WiFi.h .....	20
2.6.2	Thư viện FirebaseESP8266.h .....	20
2.6.3	Thư viện Wire.h.....	21
2.6.4	Thư viện LiquidCrystal_PCF8574.h .....	21
2.6.5	Thư viện RTClib.h .....	21
2.6.6	Thư viện DHTesp.h .....	22
<b>CHƯƠNG 3: THIẾT KẾ VÀ THI CÔNG HỆ THỐNG</b>		<b>23</b>
<b>3.1</b>	<b>YÊU CẦU VÀ SƠ ĐỒ KHỐI HỆ THỐNG</b> .....	<b>23</b>
3.1.1	Yêu cầu của hệ thống .....	23
3.1.2	Sơ đồ khối.....	23
3.1.3	Chức năng và hoạt động từng khối .....	24
<b>3.2</b>	<b>NGUYÊN LÝ HOẠT ĐỘNG CỦA HỆ THỐNG</b> .....	<b>26</b>
<b>3.3</b>	<b>THIẾT KẾ HỆ THỐNG</b> .....	<b>26</b>
3.3.1	Khối nguồn.....	26
3.3.2	Khối xử lí trung tâm .....	27
<b>3.4</b>	<b>SƠ ĐỒ NGUYÊN LÝ HỆ THỐNG</b> .....	<b>28</b>
<b>3.5</b>	<b>TÍNH TOÁN THÔNG SỐ ĐẶC TẢ HỆ THỐNG</b> .....	<b>29</b>
<b>3.6</b>	<b>LUU ĐỒ GIẢI THUẬT</b> .....	<b>30</b>
3.6.1	Chương trình chính.....	30
3.6.2	Chương trình con doc_capnhatDHT22() .....	31
3.6.3	Chương trình con xuli_nutlen() .....	33
3.6.4	Chương trình con xuli_nutxuong() .....	40
3.6.5	Chương trình con xuli_nutmenu().....	46
<b>3.7</b>	<b>VẼ MẠCH PCB</b> .....	<b>53</b>
<b>3.8</b>	<b>THI CÔNG HỆ THỐNG</b> .....	<b>54</b>
<b>3.9</b>	<b>THIẾT KẾ DASHBOARD</b> .....	<b>56</b>
<b>CHƯƠNG 4 THỰC NGHIỆM VÀ ĐÁNH GIÁ</b>		<b>57</b>
<b>4.1</b>	<b>THỰC NGHIỆM</b> .....	<b>57</b>
4.1.1	Kết quả mô phỏng .....	57
4.1.2	Kết quả thực tế .....	60
<b>4.2</b>	<b>ĐÁNH GIÁ</b> .....	<b>66</b>
4.2.1	Ưu điểm .....	66
4.2.2	Nhược điểm .....	66

<b>CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....</b>	<b>67</b>
<b>    5.1    KẾT LUẬN.....</b>	<b>67</b>
<b>    5.2    HƯỚNG PHÁT TRIỂN .....</b>	<b>67</b>
<b>TÀI LIỆU THAM THẢO .....</b>	<b>68</b>

## **DANH MỤC HÌNH ẢNH**

Hình 2.1: Nguồn pin 9V .....	4
Hình 2.2: LM7805CV IC Ôn Áp 5V/1.5A.....	5
Hình 2.3: LCD 20x04 kèm I2C Driver.....	7
Hình 2.4: Module RTC DS1307 .....	8
Hình 2.5: Module DHT22 .....	9
Hình 2.6: Buzzer.....	10
Hình 2.7: Module nút nhấn.....	11
Hình 2.8: NodeMCU ESP8266 .....	12
Hình 2.9: Truyền dữ liệu UART .....	15
Hình 2.10: Khung dữ liệu UART .....	15
Hình 2.11 Giao tiếp I2C giữa các thiết bị.....	16
Hình 2.12: Realtime Database của Firebase.....	17
Hình 2.13: Hình ảnh về Web .....	17
Hình 2.14 Giao diện phần mềm Arduino IDE.....	18
Hình 2.15: Giao diện Visual Studio Code .....	18
Hình 2.16: Giao diện ban đầu phần mềm Altium Designer .....	19
Hình 2.17: Giao diện thiết kế schematic của Altium Designer .....	19
Hình 2.18: Giao diện vẽ PCB của Altium Designer.....	20
Hình 3.1: Sơ đồ khối hệ thống.....	24
Hình 3.2: Sơ đồ nguyên lý mạch ổn áp dùng LM7805 .....	27
Hình 3.3 Sơ đồ nối chân ESP8266 .....	28
Hình 3.4 Sơ đồ nguyên lý hệ thống .....	29
Hình 3.5: Lưu đồ giải thuật của chương trình chính .....	31
Hình 3.6: Lưu đồ giải thuật của chương trình đọc và cập nhật DHT22 .....	32
Hình 3.7: Lưu đồ chính của chương trình xuli_nutlen() .....	34
Hình 3.8: Lưu đồ của trường hợp 1 xuli_nutlen() .....	35
Hình 3.9: Lưu đồ của trường hợp 2 xuli_nutlen() .....	36
Hình 3.10: Lưu đồ trường hợp 3 của xuli_nutlen() .....	37
Hình 3.11: Lưu đồ của trường hợp 4 xuli_nutlen() .....	38
Hình 3.12: Lưu đồ của trường hợp 5 xuli_nutlen() .....	39
Hình 3.13: Lưu đồ chính của chương trình xuli_nutxuong() .....	40

Hình 3.14: Lưu đồ của trường hợp 1 xuli_nutxuong()	41
Hình 3.15: Lưu đồ của trường hợp 2 xuli_nutxuong()	42
Hình 3.16: Lưu đồ trường hợp 3 của xuli_nutxuong()	43
Hình 3.17: Lưu đồ của trường hợp 4 xuli_nutxuong()	44
Hình 3.19: Lưu đồ của trường hợp 5 xuli_nutxuong()	45
Hình 3.20: Lưu đồ chính của chương trình xuli_nutmenu()	46
Hình 3.21: Lưu đồ của trường hợp 2 xuli_nutmenu()	47
Hình 3.22: Lưu đồ của trường hợp 3 xuli_nutmenu()	48
Hình 3.23: Lưu đồ trường hợp 4 của xuli_nutmenu()	50
Hình 3.24: Lưu đồ của trường hợp 5 xuli_nutmenu()	51
Hình 3.25: Lưu đồ của trường hợp 6 xuli_nutmenu()	52
Hình 3.26: Mạch PCB của hệ thống trên Altium Designer	53
Hình 3.27: Mạch PCD của hệ thống ở chế độ 3D	53
Hình 3.28: Bảng mạch in	54
Hình 3.29: Mạch in hoàn thiện của hệ thống	54
Hình 3.30: Góc nhìn từ trên xuống mô hình	55
Hình 3.31: Góc nhìn trực diện mô hình	55
Hình 3.32: Giao diện Dashboard	56
Hình 4.1 Giao diện màn hình chính của hệ thống	57
Hình 4.2: Giao diện menu chính của hệ thống	57
Hình 4.3: Giao diện cài lại DATE	58
Hình 4.4: Giao diện cài lại TIME	58
Hình 4.5: Menu chọn báo thức	59
Hình 4.6: Menu chọn báo thức sau khi di chuyển	59
Hình 4.7: Giao diện đặt báo thức của ALARM5	60
Hình 4.8 Giao diện khi kêu báo thức	60
Hình 4.9: Giao diện màn hình chính của hệ thống	61
Hình 4.10: Giao diện menu chính của hệ thống	61
Hình 4.11: Giao diện chỉnh sửa DATE	62
Hình 4.12: Giao diện chỉnh sửa TIME	62
Hình 4.13: Menu chọn báo thức	63
Hình 4.14: Menu chọn báo thức sau khi di chuyển	63
Hình 4.15: Giao diện đặt báo thức của ALARM5	64

Hình 4.16: Giao diện khi kêu báo thức.....	64
Hình 4.17:Màn hình dashboard và màn hình chính của hệ thống .....	65
Hình 4.18: Đặt báo thức trên giao diện dashboard.....	65
Hình 4.19: Giá trị đặt báo thức lưu trên realtime database .....	65
Hình 4.20: ALARM1 trên hệ thống đã cập nhật từ firebase .....	66

## **DANH MỤC BẢNG**

Bảng 2.1: Thông số kĩ thuật IC LM7805CV .....	5
Bảng 2.2 Thông tin chân của IC LM7805CV .....	6
Bảng 2.3: Thông số kĩ thuật của LCD 20x04 kèm I2C Driver .....	7
Bảng 2.4: Thông tin chân của LCD 20x04 kèm I2C Driver .....	7
Bảng 2.5: Thông số kĩ thuật của Module RTC DS1307 .....	8
Bảng 2.6 Thông tin chân của Module RTC DS1307 .....	8
Bảng 2.7: Thông số kĩ thuật của Module DHT22 .....	9
Bảng 2.8: Thông số chân Module DHT22 .....	9
Bảng 2.9: Thông số kĩ thuật của Buzzer.....	10
Bảng 2.10: Thông tin chân của Buzzer .....	10
Bảng 2.11: Thông số kĩ thuật của Module nút nhấn .....	11
Bảng 2.12: Thông tin chân của Module nút nhấn .....	11
Bảng 2.13: Thông số kĩ thuật của NodeMCU ESP8266 .....	12
Bảng 2.14 Thông tin chân của NodeMCU ESP8266 .....	13
Bảng 3.1: Điện áp và dòng điện tiêu thụ của linh kiện .....	29

## **DANH MỤC TỪ VIẾT TẮT**

Từ viết tắt	Từ tiếng anh	Ý nghĩa
IoTs	Internet of Things	Internet vạn vật
RTC	Real Time-Clock	Đồng hồ thời gian thực
UART	Universal Asynchronous Receiver/Transmitter	Giao tiếp nối tiếp không đồng bộ, truyền dữ liệu giữa các thiết bị.
I2C	Inter-Integrated Circuit	Giao tiếp nối tiếp hai dây, kết nối nhiều thiết bị với vi điều khiển.
SDA	Serial Data Line	Dây truyền dữ liệu trong giao tiếp I2C.
SCL	Serial Clock Line	Dây đồng hồ trong giao tiếp I2C định thời cho dữ liệu.
GPIO	General Purpose Input/Output	Chân nhập/xuất tổng quát.
PCB	Printed Circuit Broad	Bảng mạch in
TX	Transmitter	Chân phát
RX	Receiver	Chân nhận

# CHƯƠNG 1 TỔNG QUAN ĐỀ TÀI

## 1.1 GIỚI THIỆU

Trong cuộc sống hiện tại, thời gian là một tài sản vô giá nên con người luôn mong muốn quản lý nó một cách hiệu quả. Do đó, con người đã tạo ra đồ đồng hồ để theo dõi thời gian một cách chính xác. Tuy nhiên, nhu cầu con người ngày càng cao nên đồng hồ để bàn không chỉ đơn thuần dùng để xem giờ mà còn có thể tích hợp nhiều tính năng hữu ích khác. Xuất phát từ lý do đó, nhóm chúng tôi đã thảo luận và đưa ra đề tài “**THIẾT KẾ VÀ THI CÔNG ĐỒNG HỒ ĐỂ BÀN ĐA NĂNG**” để thực hiện trong môn Đồ án 1.

## 1.2 TÍNH CẤP THIẾT

Trong thời buổi hiện tại, người dùng có nhu cầu cao trong việc sử dụng các thiết bị thông minh IoTs trong sinh hoạt và làm việc. Vì thế đồng hồ đa năng tích hợp IoTs là một thiết bị IoTs mang tính cấp thiết cao trong đời sống. Hơn nữa với thiết kế thiết bị nhỏ gọn và dễ sử dụng, điều khiển và quan sát các thông số nhiệt độ, độ ẩm từ xa. Cùng với việc thực hiện đề tài này giúp ứng dụng kiến thức vào thực tiễn, nâng cao kỹ thuật trong lập trình vi điều khiển và tiếp cận gần hơn với các công nghệ IoTs hiện tại. Từ các điều trên đã khiến đề tài “**THIẾT KẾ VÀ THI CÔNG ĐỒNG HỒ ĐỂ BÀN ĐA NĂNG**” là cần thiết để đáp ứng nhu cầu người dùng và nâng cao kỹ năng của sinh viên thực hiện.

## 1.3 MỤC TIÊU NGHIÊN CỨU

Mục tiêu nghiên cứu của đề tài:

- Thiết kế và chế tạo hoàn thiện một đồng hồ để bàn đa năng có khả năng xem thời gian, đặt báo thức, cài lại thời gian, xem được nhiệt độ và độ ẩm môi trường và kết nối được với firebas.
- Cải thiện khả năng lập trình vi điều khiển.
- Biết lập trình thiết kế một dashboard điều khiển các chức năng và hiển thị các dữ liệu.
- Tìm hiểu và ứng dụng các công nghệ phần mềm vào dự án.

- Tìm hiểu và tiếp cận với ứng dụng IoTs vào các thiết bị.

## 1.4 NHIỆM VỤ NGHIÊN CỨU

Đề tài đạt được các mục tiêu nghiên cứu trên đưa ra một số nhiệm vụ sau:

- Phân tích đề tài và lựa chọn vi điều khiển, linh kiện phù hợp
- Tìm hiểu cơ sở lý thuyết các phần cứng đã chọn.
- Thiết kế sơ đồ nguyên lý và vẽ mạch in PCB.
- Lập trình cho vi điều khiển.
- Thiết kế một dashboard(web) cho hệ thống.
- Lắp ráp và chạy thử hệ thiết bị
- Đánh giá hiệu năng và hướng cải thiện thiết bị.

## 1.5 ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

Đối tượng nghiên cứu:

- Về phần cứng: Vi điều khiển NodeMCU ESP8266, module thời gian thực DS1307, module DHT22 (cảm biến nhiệt độ và độ ẩm), LCD 20x4, module I2C driver, module 4 nút nhấn, IC LM7805, Buzzer.
- Về phần mềm: phần mềm Arduino IDE lập trình vi điều khiển, phần mềm Visual Studio Code lập trình thiết kế dashboard(web), phần mềm Altium Designer vẽ sơ đồ nguyên lý và mạch in PCB.

Phạm vi nghiên cứu:

- Đề tài chỉ thiết kế, thi công thành phẩm ở mức độ mô hình.
- Firebase dùng làm trung gian nhận và truyền dữ liệu giữa thiết bị và dashboard (web).
- Tập trung vào các ngôn ngữ lập trình cơ bản (C/C++, HTML, ..).
- Thiết bị điều khiển trực tiếp qua các nút nhấn vật lý để thiết lập thời gian, cài báo thức, chuyển mode.
- Đề tài nghiên cứu dựa trên phạm vi kiến thức lý thuyết được học.

## 1.6 PHƯƠNG PHÁP NGHIÊN CỨU

Phương pháp nghiên cứu:

- Phương pháp thu thập và nghiên cứu tài liệu: để tìm kiếm tài liệu để xây dựng cơ sở lý thuyết cho thiết kế và lập trình hệ thống.
- Phương pháp thiết kế mô hình: Thiết kế sơ đồ nguyên lý và mạch in PCB bố trí vị trí các phần tử trong hệ thống, đồng thời thiết kế giao diện dashboard.
- Phương pháp thực nghiệm: kết nối và lắp ráp các linh kiện vào mô hình, chạy thử mô hình để kiểm thử các chức năng đề ra.
- Phương pháp phân tích – đánh giá: Từ kết quả kiểm thử phân tích tính toán các thông số như năng lượng, công suất,...Từ các kết quả để đánh giá hiệu năng của hệ thống. Sau cùng đưa ra các hướng cải thiện cho hệ thống.

## **1.7 BỘ CỤC ĐỀ TÀI**

Bộ cục đồ án 1 được trình bày 5 chương:

Chương 1: Tổng quan đề tài: Giới thiệu chung, nêu tính cấp thiết cầu đề tài và xác định, đối tượng, phạm vi, phương pháp nghiên cứu.

Chương 2: Cơ sở lý thuyết: Nghiên cứu lý thuyết các phần cứng trong hệ thống, trình bày tổng quan về một số phần cứng của hệ thống, các chuẩn giao tiếp và thư viện lập trình.

Chương 3: Thiết kế và thi công hệ thống: Nêu yêu cầu của đề tài, đưa ra sơ đồ khái của hệ thống, lưu đồ giải thuật. Tính toán xây dựng sơ đồ nguyên lý hệ thống và thi công mạch in và hoàn thành mô hình.

Chương 4: Thực nghiệm và đánh giá: Thực hiện thử nghiệm hệ thống trên mô phỏng và thực tế. Trình bày các kết quả thu được từ quá trình thử nghiệm, đưa ra đánh giá tổng quan.

Chương 5: Kết luận và hướng phát triển: Tổng kết lại các kết quả đạt được và mục tiêu đã hoàn thành. Đề xuất các hướng nghiên cứu và phát triển hệ thống.

## CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

### 2.1 TỔNG QUAN VỀ CÁC PHẦN CỨNG TRONG HỆ THỐNG

#### 2.1.1 Nguồn Pin 9V

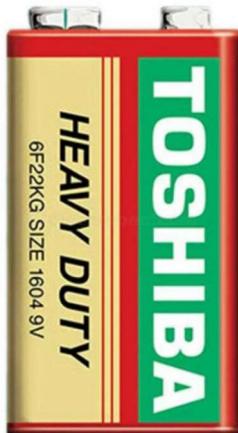
Loại pin: Pin 9V thường là pin kẽm-carbon hoặc pin alkaline (thường thấy là pin hình chữ nhật với 2 đầu tiếp xúc). Một số loại khác có thể là pin NiMH hoặc pin lithium 9V.

Điện áp danh định: 9V (trong điều kiện mới, điện áp thực tế có thể giảm khi pin cạn).

Dung lượng: Thường dao động từ 200mAh đến 600mAh tùy loại (alkaline thường cao hơn kẽm-carbon).

Ứng dụng: Thường dùng trong các thiết bị nhỏ như điều khiển từ xa, máy đo, hoặc các dự án DIY (như Arduino, ESP8266 với bộ chuyển đổi điện áp).

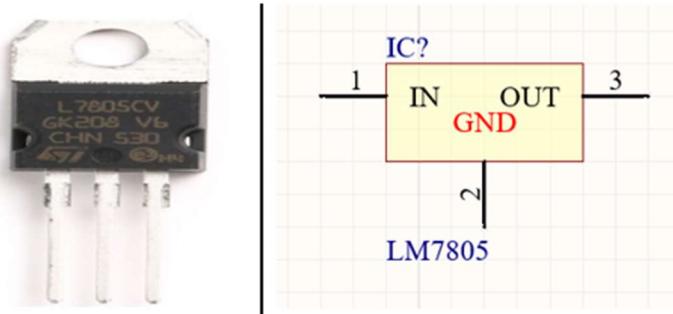
Nguồn cấp cho vi điều khiển: ESP8266 hoạt động tốt với điện áp 3.3V, trong khi pin 9V quá cao. Do đó, nhóm chúng tôi đã sử dụng IC LM7805 (đưa 9V xuống 5V).



Hình 2.1: Nguồn pin 9V

#### 2.1.2 LM7805 IC Ổn Áp 5V/1.5A

IC LM7805CV là một bộ điều chỉnh điện áp tuyến tính (linear voltage regulator) thuộc dòng L78 của STMicroelectronics, được thiết kế để cung cấp điện áp ổn định 5V với dòng tối đa 1.5A. [5]



Hình 2.2: LM7805CV IC Ôn Áp 5V/1.5A

Bảng 2.1: Thông số kỹ thuật IC LM7805CV

Thông số	Giá trị/Mô tả
Điện áp đầu ra (Output Voltage)	$5V \pm 4\%$ (4.8V đến 5.2V tùy điều kiện)
Dòng điện tối đa (Max Output Current)	1.5A (cần tản nhiệt nếu vượt 500mA)
Điện áp đầu vào (Input Voltage Range)	7V đến 25V (khuyến nghị 7.5V-20V)
Độ rơi áp (Dropout Voltage)	Khoảng 2V (tối thiểu Input phải > 7V)
Dòng tiêu thụ không tải (Quiescent Current)	5-8mA
Hiệu suất điều chỉnh dòng (Line Regulation)	$\leq 0.04\% / V$
Hiệu suất điều chỉnh tải (Load Regulation)	$\leq 0.1\%$
Bảo vệ nhiệt (Thermal Shutdown)	Ngắt ở $150^{\circ}C$
Bảo vệ ngắn mạch (Short Circuit Protection)	Có (tự động giới hạn dòng)

Nhiệt độ hoạt động (Operating Temperature)	0°C đến 125°C
Gói chân (Package)	TO-220 (3 chân)

Bảng 2.2 Thông tin chân của IC LM7805CV

Chân số	Tên Chân	Chức năng	Mô tả
1	Input	Điện áp đầu vào	Kết nối với nguồn (7V-25V)
2	Ground	Điểm nối đất (GND)	Kết nối với GND của mạch
3	Output	Điện áp đầu ra 5V	Cung cấp 5V ổn định

### 2.1.3 LCD 20x04 kèm I2C Driver

LCD 20x04 là một màn hình tinh thể lỏng (Liquid Crystal Display) có khả năng hiển thị 20 ký tự trên mỗi dòng và gồm 4 dòng, tổng cộng hiển thị được 80 ký tự. I2C Driver là một mô-đun giao tiếp (thường sử dụng chip như PCF8574) giúp kết nối LCD với vi điều khiển thông qua giao thức I2C, giảm số lượng chân kết nối từ 16 xuống còn 4 (SDA, SCL, VCC, GND), tiết kiệm chân GPIO và đơn giản hóa việc lập trình.

Ứng dụng của LCD 20x04 kèm I2C Driver:

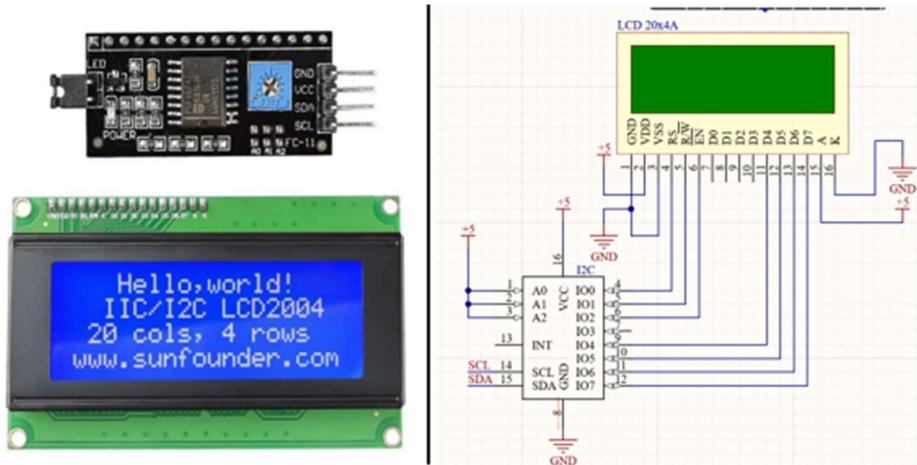
**Hiển thị thông tin trong các dự án điện tử:** Dùng để hiển thị dữ liệu như nhiệt độ, độ ẩm, thời gian trong các hệ thống nhúng (ví dụ: Arduino, Raspberry Pi).

**Giao diện người dùng:** Cung cấp giao diện trực quan cho người dùng trong các thiết bị như máy đo, đồng hồ, hoặc hệ thống điều khiển tự động.

**Hệ thống giám sát:** Hiển thị trạng thái hoạt động của máy móc, cảm biến trong công nghiệp hoặc nhà thông minh.

**Dự án học tập và DIY:** Phổ biến trong các dự án học tập về vi điều khiển để thực hành lập trình và giao tiếp I2C.

**Hiển thị thông báo hoặc menu:** Dùng trong các thiết bị cần hiển thị menu điều khiển hoặc thông báo ngắn gọn, như máy in 3D hoặc thiết bị y tế nhỏ.



Hình 2.3: LCD 20x04 kèm I2C Driver

Bảng 2.3: Thông số kỹ thuật của LCD 20x04 kèm I2C Driver [4]

STT	Kí hiệu	Thông số	Min (nhỏ nhất)	Nom (Chuẩn)	Max (lớn nhất)	Đơn vị
1	VCC	Nguồn cung cấp	4.5	5.0	5.5	V
2	F	Tần số hoạt động	50	100	1000	Hz
3	I	Dòng điện tiêu thụ	1	2	5	mA

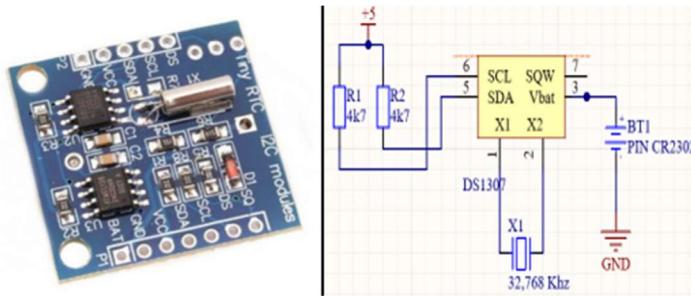
Bảng 2.4: Thông tin chân của LCD 20x04 kèm I2C Driver

Chân	Tên	Mô tả
0	SDA	Dây dữ liệu I2C
1	SCL	Dây đồng hồ I2C
2	VCC	Đầu áp đầu vào 5V
3	GND	Đất

## 2.1.4 Module RTC DS1307

Module RTC DS1307 (Real-Time Clock) là một mô-đun đồng hồ thời gian thực sử dụng chip DS1307, cho phép theo dõi thời gian (giờ, phút, giây, ngày, tháng, năm) một cách chính xác. Module này hoạt động dựa trên thạch anh 32,768 kHz để

tạo xung nhịp thời gian ổn định, đảm bảo độ chính xác cao. DS1307 giao tiếp với vi điều khiển thông qua giao thức I2C, thường đi kèm pin backup (như CR2032) để duy trì thời gian ngay cả khi mất nguồn chính. Ngoài ra, DS1307 có bộ nhớ RAM 56 byte để lưu trữ dữ liệu. [3]



Hình 2.4: Module RTC DS1307

Bảng 2.5: Thông số kỹ thuật của Module RTC DS1307

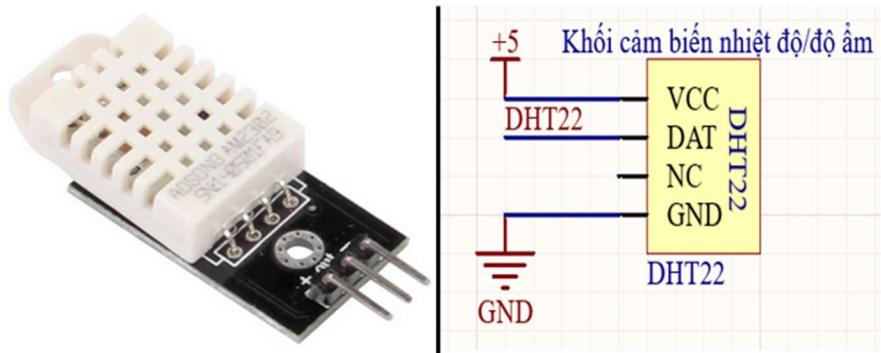
STT	Kí hiệu	Thông số	Min (nhỏ nhất)	Nom (Chuẩn)	Max (lớn nhất)	Đơn vị
1	VCC	Nguồn cung cấp	4.5	5.0	5.5	V
2	F	Tần số hoạt động	_	100	100	Hz
3	I	Dòng điện tiêu thụ	0.5	1.5	2	µA

Bảng 2.6 Thông tin chân của Module RTC DS1307

Chân	Tên	Mô tả
0	SDA	Dây dữ liệu I2C
1	SCL	Dây đồng hồ I2C
2	VCC	Đầu áp đầu vào 5V
3	GND	Đất

## 2.1.5 Module DHT22

Module DHT22 là một cảm biến nhiệt độ và độ ẩm tích hợp, sử dụng chip DHT22 để đo lường nhiệt độ và độ ẩm trong không khí. Cảm biến này hoạt động dựa trên nguyên lý tụ âm và nhiệt điện trở, cung cấp dữ liệu chính xác trong phạm vi giới hạn. DHT22 giao tiếp với vi điều khiển qua giao thức truyền dữ liệu số đơn tuyền (single-wire), không yêu cầu thư viện phức tạp và thường được sử dụng trong các dự án điện tử nhỏ như hệ thống giám sát môi trường. [2]



Hình 2.5: Module DHT22

Bảng 2.7: Thông số kỹ thuật của Module DHT22

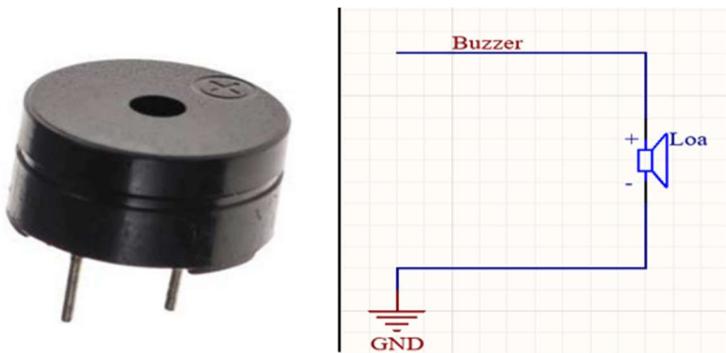
STT	Kí hiệu	Thông số	Min (nhỏ nhất)	Nom (Chuẩn)	Max (lớn nhất)	Đơn vị
1	VCC	Nguồn cung cấp	3.3	5.0	5.5	V
2	F	Tần số lấy mẫu	–	1	1	Hz
3	I	Dòng điện tiêu thụ	0.5	1.0	2.5	mA

Bảng 2.8: Thông số chân Module DHT22

Chân	Tên	Mô tả
0	VCC	Đầu áp đầu vào 5V
1	DATA	Dây dữ liệu
2	GND	Đất

## 2.1.6 Buzzer

Buzzer có 2 đầu âm dương (thường là loại piezo hoặc điện từ không tích hợp mạch điều khiển) là một linh kiện điện tử cơ bản, hoạt động bằng cách biến đổi tín hiệu điện thành âm thanh. Loại này không phải module, không có mạch dao động tích hợp (tương tự buzzer passive), nên cần tín hiệu PWM từ vi điều khiển để tạo tần số âm thanh mong muốn. Hai chân âm dương (dương (+) và âm (-)) được kết nối trực tiếp với nguồn điện hoặc vi điều khiển để phát âm thanh, thường được sử dụng trong các ứng dụng báo động hoặc tín hiệu đơn giản.



Hình 2.6: Buzzer

Bảng 2.9: Thông số kỹ thuật của Buzzer

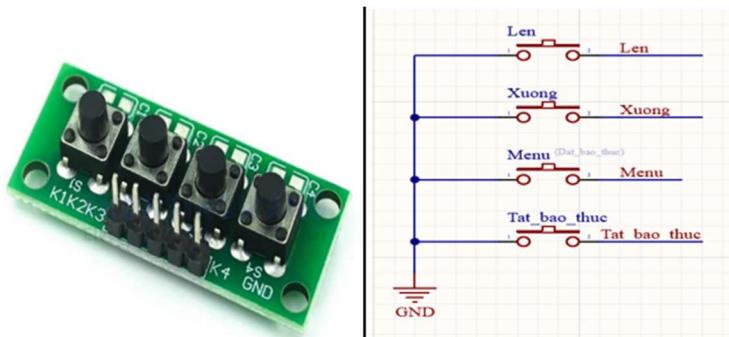
STT	Kí hiệu	Thông số	Min (nhỏ nhất)	Nom (Chuẩn)	Max (lớn nhất)	Đơn vị
1	VCC	Nguồn cung cấp	3.0	5.0	12.0	V
2	F	Tần số hoạt động	1.0	2.0	3.0	kHz
3	I	Dòng điện tiêu thụ	5	20	30	mA

Bảng 2.10: Thông tin chân của Buzzer

Chân	Tên	Mô tả
0	+	Chân dương (đầu vào tín hiệu)
1	-	Chân âm (đất)

## 2.1.7 Module nút nhấn

Module nút nhấn 4 cái là một mô-đun điện tử tích hợp bốn công tắc cơ học (thường là loại tactile switch) trên một bảng mạch in, được thiết kế để kết nối với vi điều khiển như Arduino, ESP32, ESP8266. Mỗi nút nhấn đóng vai trò như một công tắc đơn, cho phép người dùng gửi tín hiệu số (HIGH hoặc LOW) khi được nhấn. Module này thường có các chân kết nối rõ ràng (VCC, GND, và các chân tín hiệu K1-K4), hỗ trợ giao tiếp trực tiếp với vi điều khiển để điều khiển hoặc nhập liệu trong các dự án điện tử.



Hình 2.7: Module nút nhấn

Bảng 2.11: Thông số kỹ thuật của Module nút nhấn

STT	Kí hiệu	Thông số	Min (nhỏ nhất)	Nom (Chuẩn)	Max (lớn nhất)	Đơn vị
1	VCC	Nguồn cung cấp	3.3	5.0	5.5	V
2	F	Tần số nhấn	—	—	100	Lần/giây
3	I	Dòng điện tiêu thụ	0.01	0.02	0.05	mA

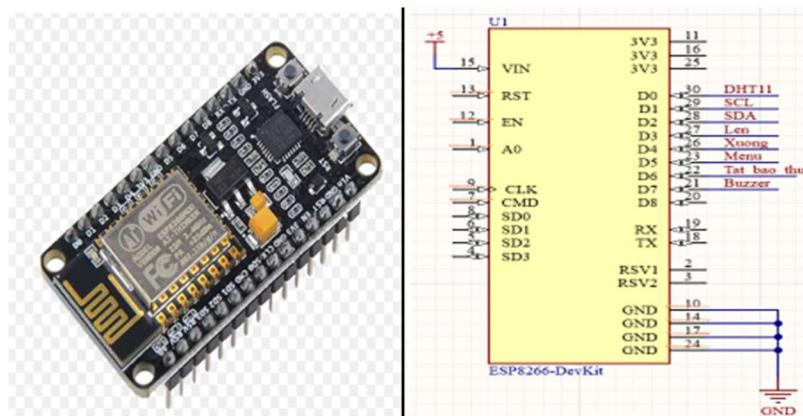
Bảng 2.12: Thông tin chân của Module nút nhấn

Chân	Tên	Mô tả
0	GND	Đất
1	K1	Chân tín hiệu nút 1
2	K2	Chân tín hiệu nút 2

3	K3	Chân tín hiệu nút 3
4	K4	Chân tín hiệu nút 4

### 2.1.8 NodeMCU ESP8266

Module ESP8266 loại có driver CP210x là một mô-đun Wi-Fi dựa trên chip ESP8266, tích hợp khả năng kết nối không dây và xử lý dữ liệu, thường được sử dụng trong các dự án IoTs. Loại này sử dụng chip CP210x (thường là CP2102 hoặc CP2104 từ Silicon Labs) làm cầu nối USB-to-UART, cho phép giao tiếp trực tiếp với máy tính qua cổng USB mà không cần bộ chuyển đổi riêng. Module này thường xuất hiện trên các board phát triển như NodeMCU hoặc bản sao (clone), hỗ trợ lập trình bằng Arduino IDE sau khi cài đặt driver CP210x phù hợp với hệ điều hành. [1]



Hình 2.8: NodeMCU ESP8266

Bảng 2.13: Thông số kỹ thuật của NodeMCU ESP8266

STT	Kí hiệu	Thông số	Min (nhỏ nhất)	Nom (Chuẩn)	Max (lớn nhất)	Đơn vị
1	VCC	Nguồn cung cấp	3.3	3.3	3.6	V
2	F	Tần số WiFi	2.4	2.412	2.484	GHz
3	I	Dòng điện tiêu thụ	20	70	170	mA

Bảng 2.14 Thông tin chân của NodeMCU ESP8266

Chân (Tên trên board)	Tên trên board ESP8266	Mô tả
VIN	VIN	Nguồn đầu vào 5V (qua USB hoặc ngoài)
GND	GND	Đất
3V3	3V3	Nguồn đầu ra 3.3V
GND	GND	Đất
D0	GPIO16	GPIO, không hỗ trợ PWM
D1	GPIO5	GPIO, thường dùng cho I2C (SCL)
D2	GPIO4	GPIO, thường dùng cho I2C (SDA)
D3	GPIO0	GPIO, kéo xuống để flash
D4	GPIO2	GPIO, tích hợp LED trên board
D5	GPIO14	GPIO, thường dùng cho SPI (SCLK)
D6	GPIO12	GPIO, thường dùng cho SPI (MISO)
D7	GPIO13	GPIO, thường dùng cho SPI (MOSI)
D8	GPIO15	GPIO, thường dùng cho SPI (CS)
RX	GPIO3	Chân nhận dữ liệu UART
TX	GPIO1	Chân truyền dữ liệu UART
GND	GND	Đất
3V3	3V3	Nguồn đầu ra 3.3V
EN	EN	Chân kích hoạt (Enable), kéo lên để hoạt động
RST	RST	Chân Reset, kéo xuống để reset

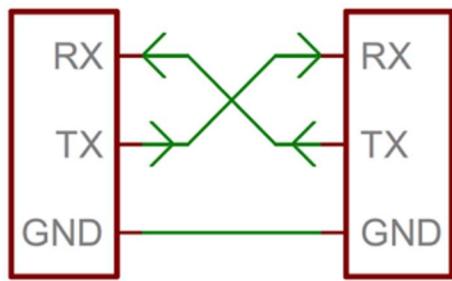
GND	GND	Đất
A0	ADC0	Chân ADC (0-1V)
S3	GPIO10	GPIO, dùng trong SPI Flash
S2	GPIO9	GPIO, dùng trong SPI Flash
SCK	GPIO6	GPIO, dùng trong SPI Flash
SDO	GPIO7	GPIO, dùng trong SPI Flash
SDI	GPIO8	GPIO, dùng trong SPI Flash
SDC	GPIO11	GPIO, dùng trong SPI Flash
RSV	—	Dự trũ (không sử dụng)
RSV	—	Dự trũ (không sử dụng)
3V3	3V3	Nguồn đầu ra 3.3V

## 2.2 GIAO THÚC KẾT NỐI

### 2.2.1 UART

UART viết tắt của từ Universal Asynchronous Receiver–Transmitter là một giao thức truyền thông phần cứng sử dụng giao tiếp bất đồng bộ. Truyền dữ liệu bất đồng bộ có nghĩa là không có xung đồng hồ để đồng bộ hóa các bit đầu ra từ thiết bị truyền đến thiết bị nhận. Để truyền được dữ liệu hệ thống phát và thu phải có mạch tạo dao động xung đồng hồ, hai hệ thống sẽ có 2 mạch tạo dao động độc lập nhưng phải cùng tần số hay cùng tốc độ.

Cách kết nối chân giữa thiết bị phát và thu ở hình 2.9.

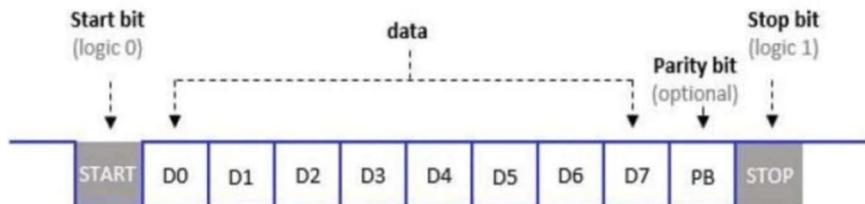


Hình 2.9: Truyền dữ liệu UART

Hai dây đường dây sử dụng trong chuẩn UART:

- TX: Chân phát dữ liệu đi
- RX: Chân thu dữ liệu về

Trong UART dữ liệu được truyền song công, khung dữ liệu chuẩn giao tiếp UART ở hình 2.10.



Hình 2.10: Khung dữ liệu UART

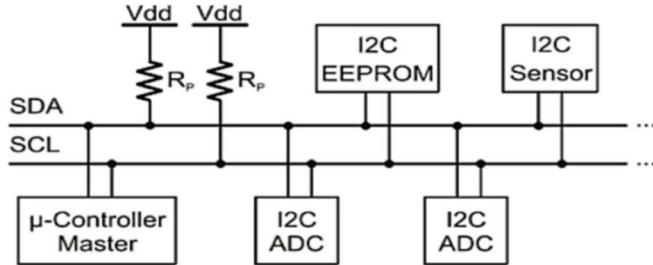
Một khung dữ liệu gồm:

- Start bit: Có 1 bit, là bit bắt đầu truyền dữ liệu, giá trị là 0 (mức thấp).
- Data: Gồm 8 bit dữ liệu cần gửi.
- Parity bit: Có 1 bit, là bit kiểm tra chẵn, lẻ.
- Stop bit: 1 hoặc 2 bit, là bit kết thúc khung dữ liệu, giá trị là 1 (mức cao).

### 2.2.2 I2C

I2C viết tắt của từ Inter-Integrated Circuit là chuẩn truyền thông do hãng điện tử Philips Semiconductor sáng lập, cho phép một thiết bị chủ giao tiếp với nhiều thiết bị tớ với nhau.

Mạch vật lý I2C là mạch cực thu hở, do đó để I2C hoạt động cần mắc hai điện trở kéo lên như hình 2.11, thông thường các giá trị điện trở là  $4k7\Omega$  hoặc  $1k2\Omega$  tùy thuộc vào tốc độ truyền và khoảng cách.



Hình 2.11 Giao tiếp I2C giữa các thiết bị

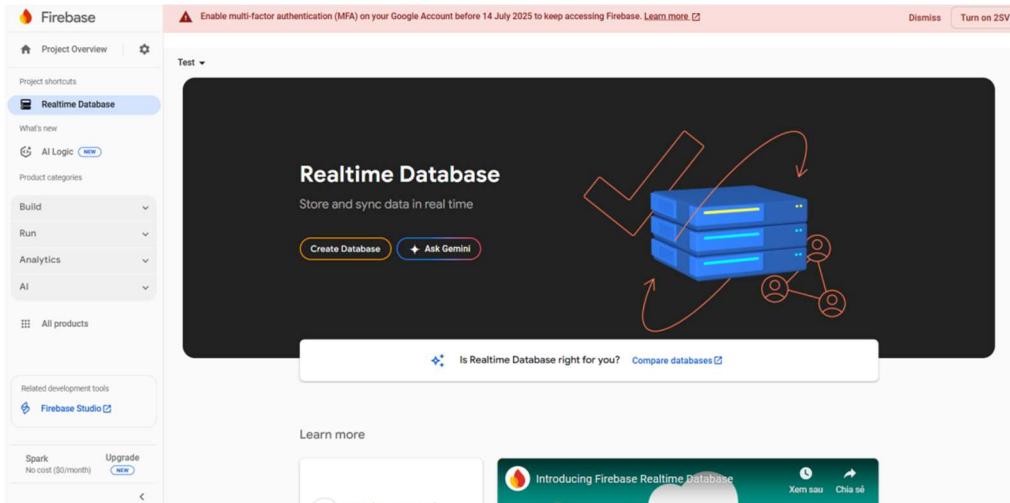
Giao thức I2C dữ liệu truyền song công, hai đường dây tín hiệu là SDA (Serial Data) và SCL (Serial Clock):

- SDA: Truyền tải dữ liệu.
- SCL: Truyền xung đồng hồ để dịch chuyển dữ liệu.

## 2.3 TỔNG QUAN VỀ FIREBASE

Firebase là cơ sở dữ liệu để lưu trữ và đồng bộ dữ liệu hoạt động trên nền tảng đám mây được phát triển bởi Google. Giúp cho người sử dụng dễ dàng phát triển các ứng dụng thông qua đơn giản hóa các thao tác với cơ sở dữ liệu khác nhau.

Firebase Realtime Database là một tính năng của Firebase, với cơ sở dữ liệu thời gian thực, dữ liệu nhận được dưới dạng JSON, dữ liệu luôn được cập nhật nhanh chóng và đồng bộ thời gian thực. Trường hợp mất kết nối Internet dữ liệu sẽ được lưu lại ở cục bộ vì vậy khi có sự thay đổi dữ liệu để tự động cập nhật lên Server của Firebase.



Hình 2.12: Realtime Database của Firebase

## 2.4 TỔNG QUAN VỀ WEB

Website là nơi tập hợp các Web page, cung cấp rất nhiều những nội dung như văn bản, hình ảnh, video clip, cập nhật thông tin mới,... được xác định bằng tên miền cụ thể. Website muốn vận hành hoạt động phải có tên miền và được lưu trữ trên máy chủ Server giúp lưu trữ các dữ liệu của Website.



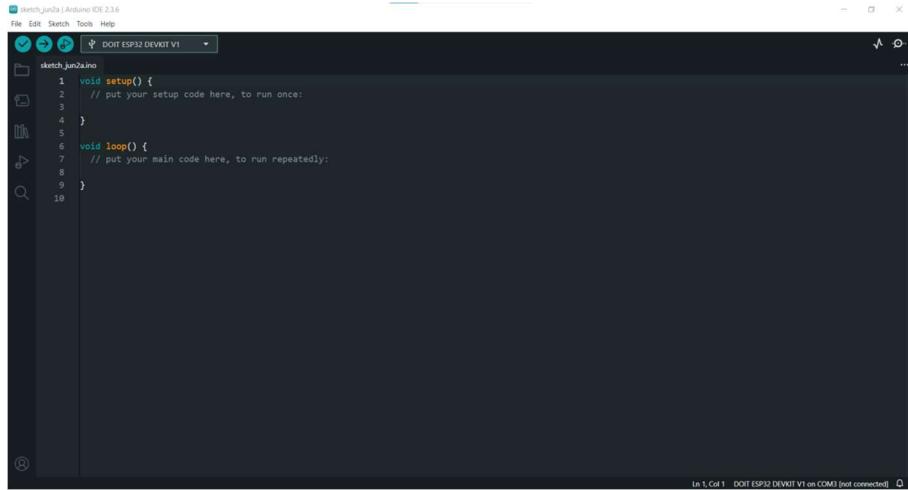
Hình 2.13: Hình ảnh về Web

## 2.5 TỔNG QUAN VỀ CÔNG CỤ PHẦN MỀM

### 2.5.1 Arduino IDE

Arduino IDE là ứng dụng cross-platform (đa nền tảng) được viết bằng Java và IDE, phần mềm được thiết kế để dành cho người mới tập làm quen với lĩnh vực phát

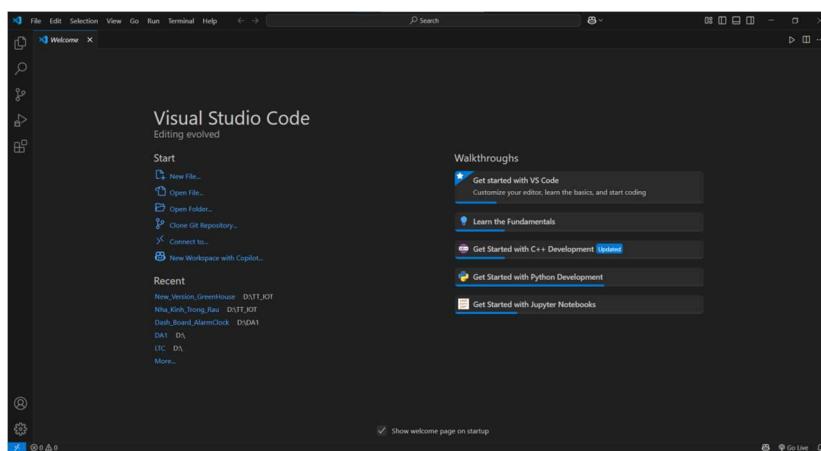
triển phần mềm. Arduino IDE thích hợp với nhiều hệ điều hành: Windows, Linux, macOS,... thông qua ngôn ngữ C, C++. Phần mềm có nhóm phiên bản là 2.1.x, 1.8.x, phiên bản 2.1.x có giao diện đẹp hơn, có trình biên soạn chú thích và biên dịch chương trình nhanh chóng. Arduino IDE là phần mềm rất thông dụng hỗ trợ cho học tập và công việc, cùng với mã nguồn mở dễ dàng lập trình các thiết bị điện tử mà không cần bỏ thời gian dài để viết các thư viện.



Hình 2.14 Giao diện phần mềm Arduino IDE

### 2.5.2 Visual Studio Code

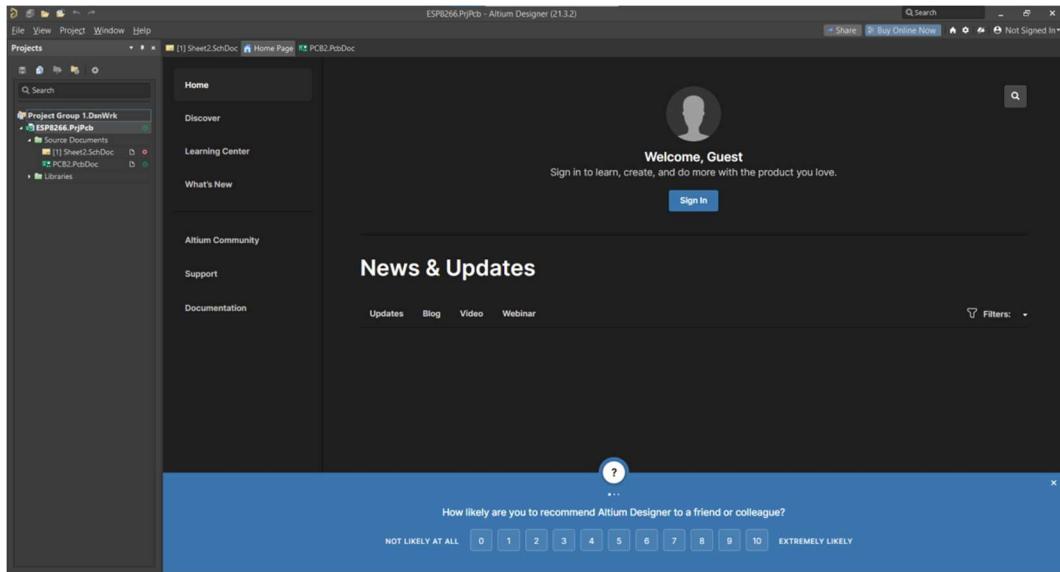
Visual Studio Code là một trình soạn thảo mã nguồn được phát triển bởi Microsoft dành cho Windows, Linux và macOS. Nó hỗ trợ debug, đi kèm với Git, có chức năng nổi bật cú pháp, tự hoàn thành mã thông minh, snippets và cài tiến mã nguồn. Các ngôn ngữ lập trình như: JavaScript, C, CSS, C#, HTML, Python,...



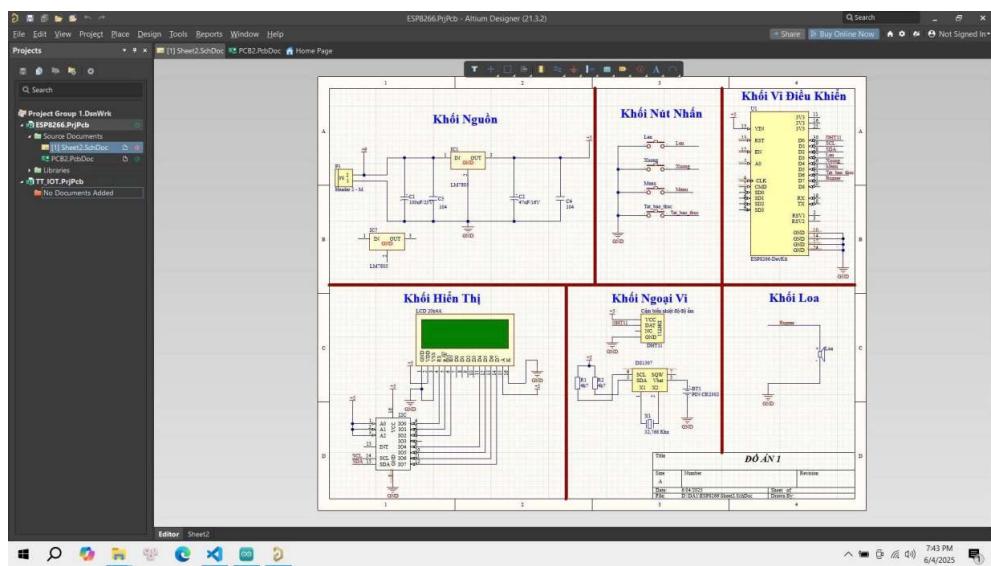
Hình 2.15: Giao diện Visual Studio Code

### 2.5.3 Altium Designer

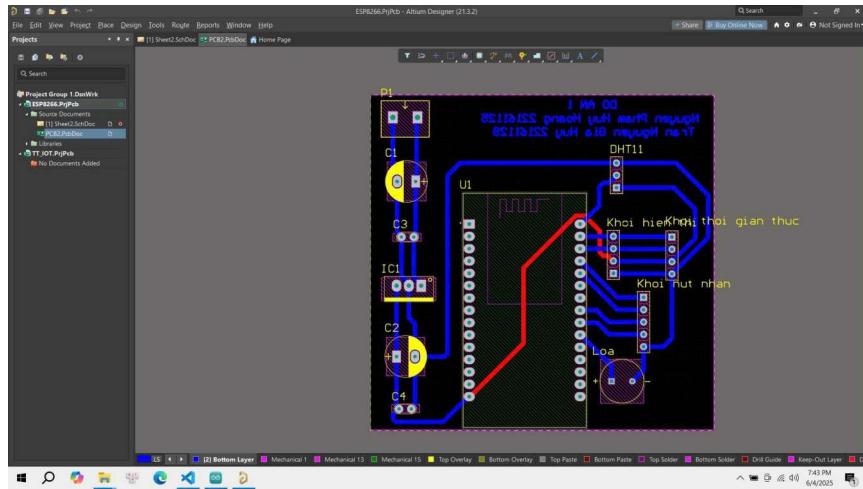
Altium Designer là gói phần mềm tự động hóa thiết kế PCB và điện tử dành cho bản mạch in. Phần mềm được phát triển bởi công ty Altium Limited của Úc. Altium Designer chuyên thiết kế để vẽ sơ đồ nguyên lý và PCB có bộ công cụ rất thông minh và nhiều tính năng đáp ứng nhiều yêu cầu thị trường hiện nay.



Hình 2.16: Giao diện ban đầu phần mềm Altium Designer



Hình 2.17: Giao diện thiết kế schematic của Altium Designer



Hình 2.18: Giao diện vẽ PCB của Altium Designer

## 2.6 CÁC THƯ VIỆN LẬP TRÌNH

### 2.6.1 Thư viện ESP8266WiFi.h

Cú pháp khai báo: #include <ESP8266WiFi.h>

Chức năng: Thư viện này hỗ trợ quản lý kết nối Wifi trên ESP8266

Các hàm sử dụng:

- WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD): Khởi tạo kết nối WiFi với SSID và mật khẩu được định nghĩa.
- WiFi.status(): Trả về trạng thái kết nối WiFi (ví dụ: WL\_CONNECTED).
- WiFi.localIP(): Trả về địa chỉ IP cục bộ của thiết bị sau khi kết nối WiFi thành công.
- WiFi.reconnectWiFi(true): Kích hoạt chế độ tự động kết.

### 2.6.2 Thư viện FirebaseESP8266.h

Cú pháp khai báo: #include<FirebaseESP8266.h>

Chức năng: Thư viện cung cấp các chức năng để giao tiếp với FirebaseDatabase RealtimeDatabase trên ESP8266.

Các hàm sử dụng:

- Firebase.begin(&config, &auth): Khởi tạo kết nối với Firebase bằng cấu hình và thông tin xác thực.

- `Firebase.reconnectWiFi(true)`: Kích hoạt tự động kết nối lại WiFi nếu mất kết nối.
- `Firebase.getJSON(fbdo, path)`: Lấy dữ liệu JSON từ Firebase tại đường dẫn được chỉ định.
- `Firebase.setJSON(fbdo, path, sensorData)`: Gửi dữ liệu JSON lên Firebase tại đường dẫn được chỉ định.
- `Firebase.ready()`: Kiểm tra xem kết nối Firebase đã sẵn sàng hay chưa.

### 2.6.3 Thư viện Wire.h

Cú pháp khai báo: `#include<Wire.h>`

Chức năng: Thư viện hỗ trợ giao tiếp I2C, cần thiết cho các thiết bị như LCD hoặc RTC.

Không có sử dụng các hàm được định nghĩa sẵn trong thư viện Wire.h, nhưng cần thư viện này để hỗ trợ giao tiếp I2C cho hai thư viện khác là LiquidCrystal\_PCF8574 và RTCLib.

### 2.6.4 Thư viện LiquidCrystal\_PCF8574.h

Cú pháp khai báo: `#include<LiquidCrystal_PCF8574.h>`

Chức năng: Thư viện điều khiển màn hình LCD thông qua module PCF 8574 (giao tiếp I2C).

Các hàm sử dụng:

- `lcd.begin(20, 4)`: Khởi tạo màn hình LCD với kích thước 20x4.
- `lcd.setBacklight(255)`: Đặt độ sáng đèn nền của LCD.
- `lcd.clear()`: Xóa nội dung trên màn hình LCD.
- `lcd.setCursor(col, row)`: Đặt vị trí con trỏ trên LCD để hiển thị văn bản.
- `lcd.print(data)`: Hiển thị dữ liệu (chuỗi, số, v.v.) lên LCD.
- `lcd.cursor()`: Hiển thị con trỏ nháy nháy trên LCD.
- `lcd.noCursor()`: Ẩn con trỏ nháy nháy trên LCD.

### 2.6.5 Thư viện RTCLib.h

Cú pháp khai báo: `#include<RTCLib.h>`

Chức năng: Thư viện làm việc với module RTC như DS1307 để quản lí thời gian.

Các hàm sử dụng:

- rtc.begin(): Khởi tạo module RTC.
- rtc.isrunning(): Kiểm tra xem RTC có đang hoạt động hay không.
- rtc.adjust(DateTime(year, month, day, hour, minute, second)): Cài đặt thời gian cho RTC.
- rtc.now(): Lấy thời gian hiện tại từ RTC, trả về đối tượng DateTime.
- Các phương thức của đối tượng DateTime:
  - day(): Lấy ngày.
  - t.month(): Lấy tháng.
  - t.year(): Lấy năm.
  - t.hour(): Lấy giờ.
  - t.minute(): Lấy phút.
  - t.second(): Lấy giây.
  - t.unixtime(): Lấy thời gian dưới dạng Unix timestamp.

### 2.6.6 Thư viện DHTesp.h

Cú pháp khai báo: #include<DHTesp.h>

Chức năng: Thư viện đọc dữ liệu từ cảm biến nhiệt độ và cảm biến DHT (như DHT11, DHT22,...)

Các hàm sử dụng:

- dht.setup(DHTPIN, DHTesp::DHT22): Khởi tạo cảm biến DHT22 trên chân được chỉ định.
- dht.getTemperature(): Lấy giá trị nhiệt độ từ cảm biến.
- dht.getHumidity(): Lấy giá trị độ ẩm từ cảm biến.
- dht.getStatus(): Kiểm tra trạng thái của cảm biến (0 nếu thành công, khác 0 nếu lỗi).
- dht.getStatusString(): Trả về chuỗi mô tả lỗi của cảm biến.

## **CHƯƠNG 3: THIẾT KẾ VÀ THI CÔNG HỆ THỐNG**

### **3.1 YÊU CẦU VÀ SƠ ĐỒ KHỐI HỆ THỐNG**

#### **3.1.1 Yêu cầu của hệ thống**

Đồng hồ để bàn hiển thị thời gian thực:

- Hệ thống cần hiển thị thời gian thực lên màn hình LED đồng hồ để người dùng có thể xem thời gian.

Đặt được thời gian báo thức:

- Hệ thống cần đặt thời gian trực tiếp trên thiết bị và đặt đã được trên dashboard (web).

Màn hình xem được nhiệt độ và độ ẩm phòng.

- Hệ thống đọc được giá trị nhiệt độ và độ ẩm từ cảm biến DHT22, hiển thị trên LCD và gửi dữ liệu lên firebase và hiện thị lên dashboard (web).

Sử dụng pin để hoạt động:

- Hệ thống sử dụng nguồn pin để hoạt động.

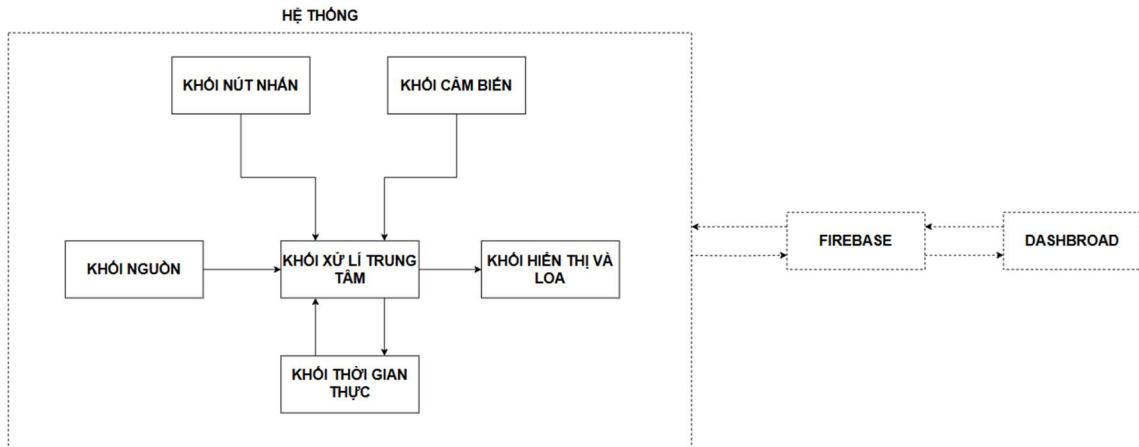
Điều khiển được qua dashboard:

- Hệ thống phải kết nối được với firebase để gửi dữ liệu lên dashboard (web) và nhận dữ liệu từ dashboard (web).

Sản phẩm nhỏ gọn và dễ sử dụng:

- Kích thước sản phẩm nhỏ gọn và hệ thống phải dễ sử dụng và dễ điều khiển.

#### **3.1.2 Sơ đồ khối**



Hình 3.1: Sơ đồ khái niệm hệ thống

### 3.1.3 Chức năng và hoạt động từng khối

#### *Khối nguồn*

Gồm nguồn pin 9V và mạch ổn áp 5V IC LM7805.

Chức năng: Cung cấp điện áp ổn định 5V cho khối xử lý trung tâm và các khối khác hoạt động.

Hoạt động: Nguồn pin 9V vào mạch ổn áp LM7805, nguồn 9V sẽ được giảm áp xuống 5V và áp từ ngõ ra mạch ổn áp LM7805 sẽ cung cấp cho khối xử lý trung tâm và từ khối xử lý trung tâm sẽ cung cấp cho các khối ngoại vi khác.

#### *Khối xử lý trung tâm*

Sử dụng vi điều khiển NodeMCU ESP8266 làm khối xử lý trung tâm.

Chức năng: là trung tâm điều khiển của hệ thống, nhận tín hiệu từ khối cảm biến, khối thời gian thực, khối nút nhấn, gửi dữ liệu đến khối hiển thị, điều khiển loa buzzer, kết nối wifi để gửi và đọc dữ liệu từ firebase realtime database.

Hoạt động: nhận và xử lý tín hiệu từ khối cảm biến và khối thời gian thực, nhận tín hiệu từ khối nút nhấn để chỉnh thời gian và đặt báo thức, điều khiển buzzer kêu khi đến báo thức, gửi dữ liệu lên LCD để hiển thị thông tin, kết nối wifi liên kết firebase realtime database, gửi và đọc dữ liệu từ firebase realtime database.

#### *Khối nút nhấn*

Sử dụng module 4 nút nhấn.

Chức năng: chỉnh các đơn vị thời gian, tắt loa báo thức, chuyển giữa các đơn vị, chọn chế độ đặt báo thức hoặc cài lại thời gian.

Hoạt động: Nhấn nút ‘len’ (D4) tăng đơn vị thời gian và dịch con trỏ từ phải sang trái, nhấn nút ‘xuong’ (D3) giảm đơn vị thời gian và dịch con trỏ từ trái sang phải, nhấn nút ‘tat\_baothuc’ (D6) để tắt báo thức, nhấn nút ‘menu’ (D5) để chuyển đổi giữa các giao diện và nút xác nhận như xác nhận lưu giá trị thời gian hay xác nhận quay lại giao diện trước đó.

#### *Khối cảm biến*

Sử dụng module DHT22.

Chức năng: đo nhiệt độ và độ ẩm từ môi trường ngoài và truyền dữ liệu về khói xử lí trung tâm.

Hoạt động: Đo nhiệt độ và độ ẩm từ môi trường ngoài rồi gửi tín hiệu digital (gồm các bit 0,1) tương ứng cho nhiệt độ và độ ẩm qua khói xử lí trung tâm.

#### *Khối thời gian thực*

Sử dụng module RTC DS1307

Chức năng: cung cấp thời gian thực cho hệ thống.

Hoạt động: gửi các giá trị thời gian đến khói xử lí trung tâm qua I2C và nhận các giá trị được điều chỉnh bằng nút nhấn từ khói xử lí trung tâm để cài lại thời gian.

#### *Khối hiển thị và loa*

Gồm LCD 20x04 và buzzer

Chức năng: Hiển thị các thông tin như thời gian thực, độ ẩm, nhiệt độ,...và báo tiếng píp khi đến thời gian đã cài báo thức.

Hoạt động: nhận dữ liệu từ khói xử lí trung tâm thông qua I2C và nhận tín hiệu từ khói xử lí trung tâm để điều khiển buzzer khi đến thời gian báo thức.

#### *Khối firebase*

Chức năng: trung gian giữa hệ thống và dashboard, lưu các thông tin được gửi lên firebase trong realtime database.

Hoạt động: nhận và lưu dữ liệu từ thiết bị và dashboard, gửi các thông tin thời gian thực, độ ẩm và nhiệt độ lên dashboard và các giá trị thời gian và giá trị biến cho phép báo thức được điều khiển từ dashboard được gửi đến thiết bị.

#### *Khối dashboard*

Chức năng: Theo dõi thông số thời gian thực, độ ẩm và nhiệt độ của hệ thống và điều khiển hệ thống từ xa.

Hoạt động: đọc các thông số thời gian thực, độ ẩm và nhiệt độ từ realtime database và hiển thị lên dashboard, gửi các giá trị điều khiển đến realtime database.

## 3.2 NGUYÊN LÝ HOẠT ĐỘNG CỦA HỆ THỐNG

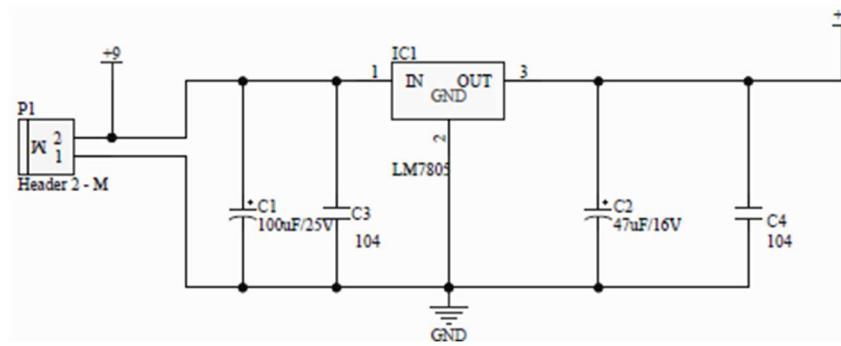
Nguồn pin 9V cấp nguồn vào mạch ổn áp LMM7805 để hạ áp xuống 5V. Áp 5V sẽ được cấp cho ESP8266 và khi có nguồn khởi động ESP8266 sẽ truy vấn vào bộ nhớ flash để lấy code được nạp cuối cùng để chạy. Khi đó ESP8266 sẽ bắt đầu kết nối wifi được định nghĩa trong code. Tiếp theo thu thập dữ liệu từ các khố ngoại vi và đem các dữ liệu đó gửi lên LCD20x4 thông qua I2C, đồng thời cũng gửi các dữ liệu đó lên realtime database để cập nhật các dữ liệu mới lên dashboard. Sau đó ESP8266 sẽ đọc các giá trị báo thức và biến cho phép báo thức được thay đổi trên dashboard ở realtime database và sau đó sẽ đặt báo thức theo các giá trị thời gian trên. Khi có nút ‘menu’ (D5) nhấn, sẽ chuyển màn hình LCD20x4 hiển thị giao diện chọn mode: một là cài lại ngày tháng năm, hai là cài lại giờ phút giây và ba là đặt báo thức. Sau khi điều chỉnh các giá trị bằng nút nhấn, ESP8266 sẽ đọc các giá trị đó nếu mode cài lại thời gian thì các giá trị đó làm thời gian mới, còn mode đặt báo thức thì các giá trị đó sẽ được đặt làm thời gian báo thức. Khi đến thời gian báo thức, ESP8266 sẽ điều khiển buzzer để buzzer kêu và để tắt bằng cách nhấn nút ‘tat\_baothuc’ (D6).

## 3.3 THIẾT KẾ HỆ THỐNG

### 3.3.1 Khối nguồn

Yêu cầu nguồn hoạt động ổn định của ESP8266 là 5V. Nhưng trên thị trường, không có nguồn pin 5V nên chúng tôi chọn nguồn pin 9V và qua mạch ổn áp LM7805.

Mạch ổn áp sử dụng IC ổn áp LM7805 để ổn áp điện áp 5V để cấp nguồn cho ESP8266 hoạt động như sau:



Hình 3.2: Sơ đồ nguyên lý mạch ổn áp dùng LM7805

Điện áp đầu vào từ nguồn pin 9V cấp vào mạch ổn áp thông qua Tblock, ở ngõ ra của mạch ổn áp sẽ có điện áp là 5V để cung cấp vào chân Vin của ESP8266.

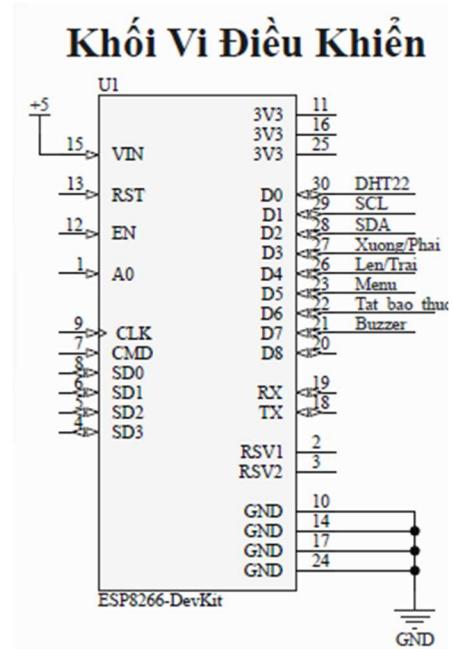
Tụ C1 và C3 để lọc điện áp đầu vào cấp vào chân Vin LM7805, tụ C1 có tác dụng cung cấp điện áp tạm thời cho chân Vin khi nguồn đột ngột bị sụt áp, tụ C3 là tụ gốm nên trở kháng lớn, C2 có tác dụng ngăn nguồn đầu vào tăng áp đột ngột làm dạng sóng điện áp đầu vào có hình răng cưa.

Tụ C2 và C4 để lọc điện áp cấp cho ngõ ra lấy từ chân Vo của LM7805, Tụ C2 có tác dụng cung cấp điện áp tạm thời cho ngõ ra trách tình huống áp tại ngõ ra bị sụt áp, tụ C4 trở kháng lớn và có tác dụng lọc nhiễu điện áp ngõ ra.

### 3.3.2 Khối xử lí trung tâm

Khối xử lí trung tâm trong hệ thống an ninh sẽ sử dụng ESP 8266 để điều khiển, đọc và xử lí các tín hiệu từ khói cảm biến, khói thời gian thực, khói nút nhấn, kết nối wifi và nhận và gửi dữ liệu của realtime database.

Sơ đồ nguyên lý kết nối các chân với khối ngoại vi của ESP8266, như sau:



Hình 3.3 Sơ đồ nối chân ESP8266

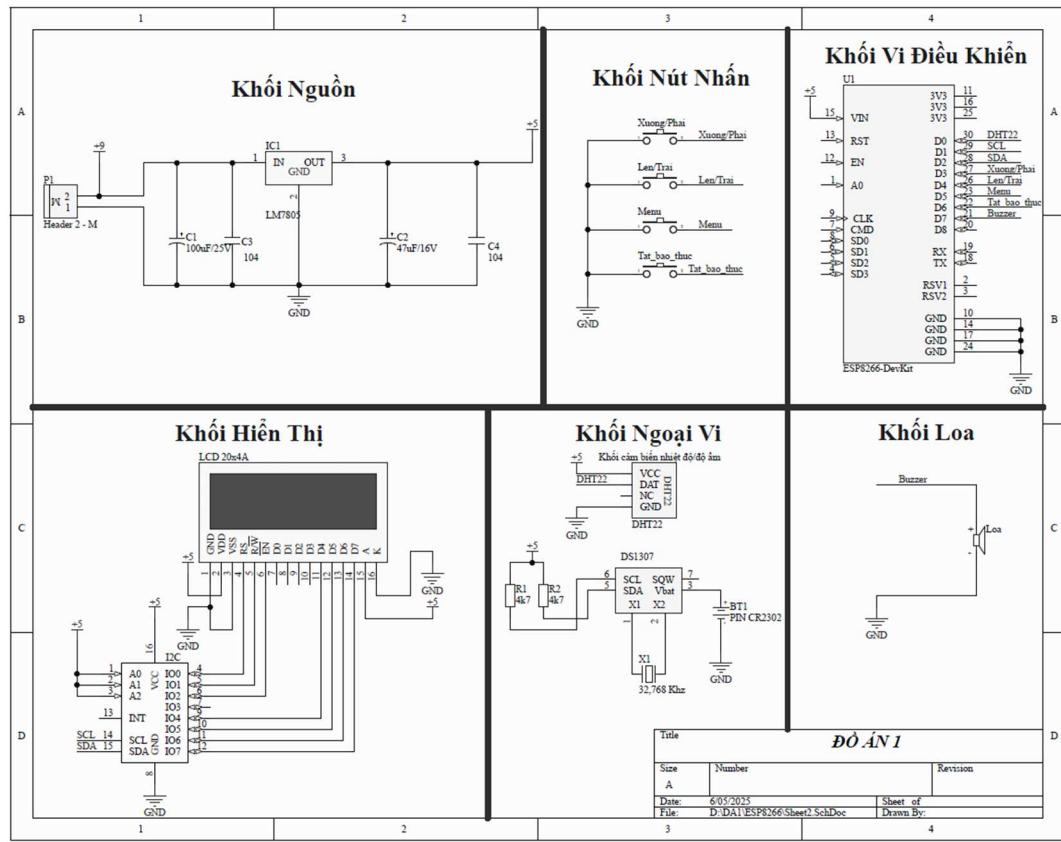
Chân Vin được nối ngõ ra mạch ổn áp LM7805 để

Chân D0 được nối với chân DAT của module DHT22 để nhận tín hiệu từ module DHT22.

Các chân D1, D2 là hai chân giao tiếp I2C được nối với hai chân SCL và SDA của module RTC DS1307 để nhận tín hiệu giá trị thời gian thực, gửi tín hiệu giá trị để cài lại thời gian và LCD20x4 kèm I2C driver để gửi tín hiệu để hiển thị lên màn hình LCD20x4.

Các chân D3, D4, D5, D6 được nối lần lượt với các chân ‘xuong’, ‘len’, ‘menu’, ‘tat\_baothuc’ của module nút nhấn để nhận tín hiệu điều khiển từ nút nhấn.

### 3.4 SƠ ĐỒ NGUYÊN LÝ HỆ THỐNG



Hình 3.4 Sơ đồ nguyên lý hệ thống

### 3.5 TÍNH TOÁN THÔNG SỐ ĐẶC TẢ HỆ THỐNG

Kích thước mô hình: 15x15x15 (cm).

Công suất:

Bảng 3.1: Điện áp và dòng điện tiêu thụ của linh kiện

Linh kiện	Điện áp (V)	Dòng điện tiêu thụ (mA)
ESP32	5V	130
DHT22	5V	2
DS1307	5V	1.94
LCD 20x4	5V	47.7

Tổng dòng điện tiêu thụ của hệ thống:

$$I_{\text{total}} = 130 + 2 + 1.94 + 47.7 = 181.64 \text{ (mA)}$$

Công suất tiêu thụ toàn hệ thống:

$$P = 5.5 \times 181.64 = 0.908 (W)$$

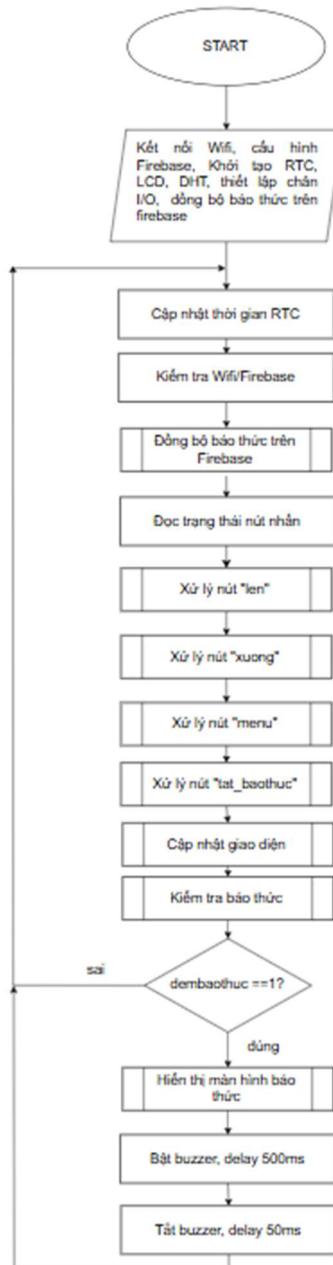
Thời gian hoạt động của hệ thống với dung lượng pin Toshiba 9V 1604S có dung lượng pin lớn nhất khoảng 500 mAh, ta có:

$$\text{Thời gian sử dụng} = \frac{\text{Dung lượng pin (mA)}}{I_{\text{total}}} = \frac{500}{181.64} = 2.75 \text{ giờ}$$

## 3.6 LUU ĐO GIẢI THUẬT

### 3.6.1 Chương trình chính

Lưu đồ giải thuật:



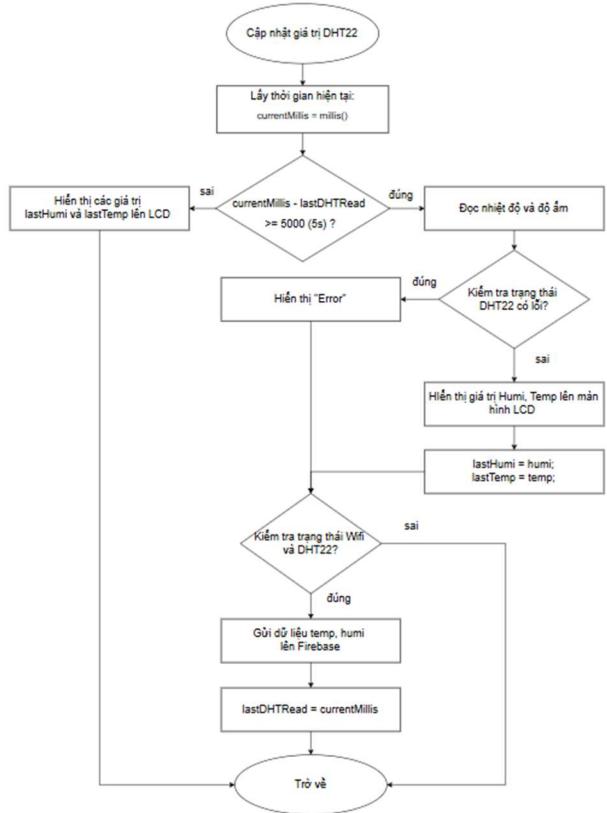
### Hình 3.5: Lưu đồ giải thuật của chương trình chính

Giải thích lưu đồ:

Khi hệ thống được cấp nguồn thì hệ thống kết nối wifi, cấu hình firebase, khởi tạo RTC, LCD, DHT, thiết lập các chân I/O và lấy dữ liệu báo thức trên firebase rồi đồng bộ. Tiếp theo đến hệ thống sẽ cập nhật thời gian RTC (1s cập nhật/1 lần), rồi tiếp theo kiểm tra trạng thái kết nối Wifi và firebase, sau đó đồng bộ các giá trị báo thức trên firebase. Tiếp theo đọc giá trị nút nhấn rồi kiểm tra nút nào nhấn để thực hiện chức năng nút nhấn đó. Tiếp theo kiểm tra và xử lý màn hình LCD sẽ hiện thị giao diện nào. Cuối cùng kiểm tra báo thức nếu đến thời gian báo thức thì dembaothuc = 1 còn không thì dembaothuc = 0 và kiểm tra dembaothuc nếu bằng 0 thì lặp lại vòng lặp còn bằng 1 thì hiển thị màn hình báo thức và bật buzzer trong 0.5s, tắt buzzer trong 0,05s sau đó lặp lại vòng lặp.

#### 3.6.2 Chương trình con doc\_capnhatDHT22()

Lưu đồ giải thuật:



Hình 3.6: Lưu đồ giải thuật của chương trình đọc và cập nhật DHT22

Giải thích lưu đồ:

Sau khi hàm được gọi thì lấy giá trị giây hiện tại cho biến currentMillis. Sau đó kiểm tra khoảng thời gian từ lần cuối cập nhật giá trị đến thời gian hiện tại có lớn hơn hoặc bằng 5s không. Nếu sai thì không đọc giá trị nhiệt độ, độ ẩm mới và hiển thị các giá trị nhiệt độ, độ ẩm của lần cập nhật trước lên màn hình và trở về chương trình đang chạy. Nếu đúng thì đọc giá trị nhiệt độ, độ ẩm mới sau đó kiểm tra trạng thái của cảm biến DHT22 có lỗi, trực trặc gì không. Nếu không có (sai) thì hiển thị giá trị mới đọc lên màn hình LCD và lưu hai giá trị đọc ở trên vào hai biến lastTemp và lastHumi và xuống bước tiếp theo. Còn nếu có lỗi (đúng) thì hiển thị “Error” lên màn hình LCD. Tiếp theo là kiểm tra trạng thái kết nối Wifi và trạng thái hoạt động của DHT22 cả hai hoạt động tốt không. Nếu tốt (đúng) thì cập nhật giá trị nhiệt độ và độ ẩm lên firebase sau đó đặt giá trị thời gian trong biến currentMillis cho biến lastDHTRead làm thời gian lần đọc cuối cùng. Còn nếu không thì trở về chương trình đang chạy.

### **3.6.3 Chương trình con xuli\_nutlen()**

Ý nghĩa giá trị của biến demtong:

0: là màn hình chính hiển thị thời gian, nhiệt độ và độ ẩm.

1: là giao diện menu chỉnh để chọn mục.

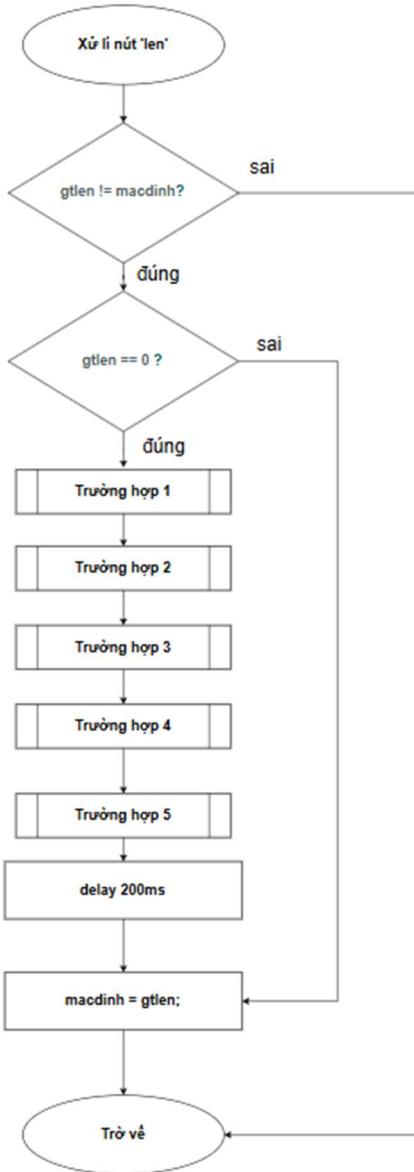
2: là mở giao diện của mục đã chọn.

3: là giao diện chỉnh sửa của mục.

4: là lưu giá trị có thể chỉnh sửa trên giao diện của mục.

5: là quay về giao diện chọn báo thức

*Lưu ý chính:*

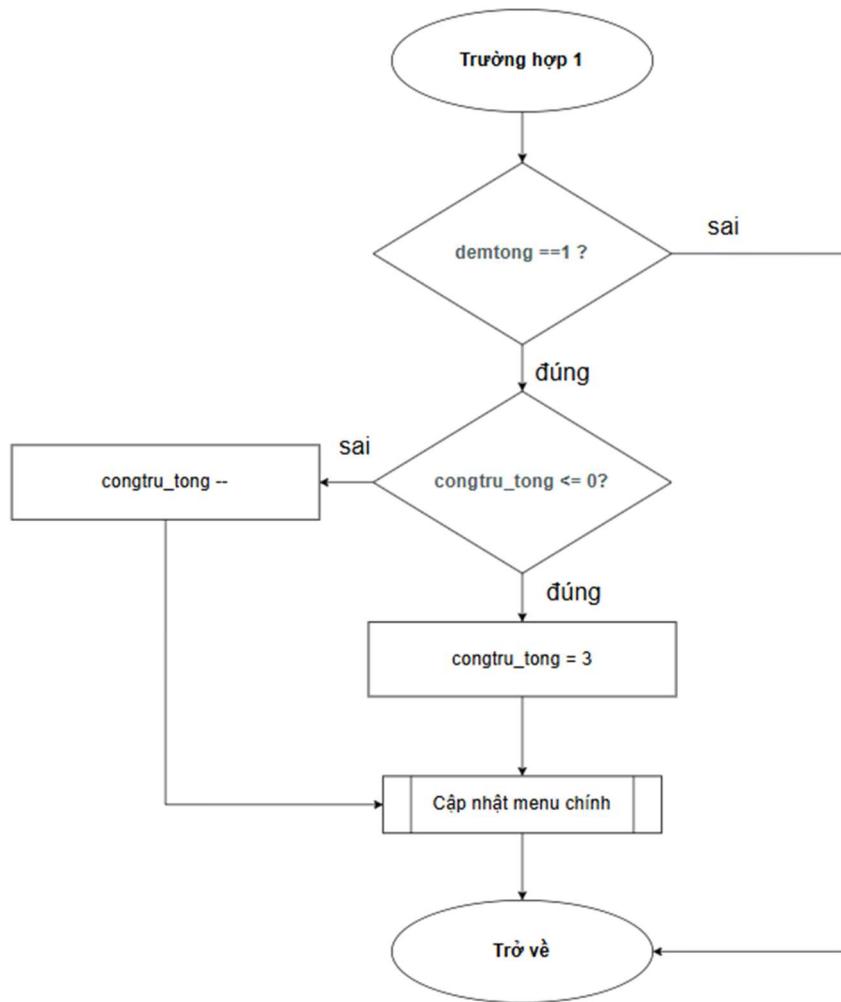


Hình 3.7: Lưu đồ chính của chương trình xuli\_nutlen()

Giải thích lưu đồ:

Khi chương trình xuli\_nutlen(), đầu tiên kiểm tra trạng thái biến gtlen có thay đổi so với trạng thái trước được lưu trong biến macdinh nếu không thì trở về chương trình còn nếu có thì kiểm tra gtlen == 0 (tương đương với nút len được nhấn). Nếu bằng 1 (không có nhấn) thì trở về chương trình chính, còn nếu bằng 0 (có nhấn) thì kiểm tra các trường hợp xử lý nút nhấn để thực hiện chức năng của nút len. Sau khi kiểm tra các trường hợp thì delay 200ms và đặt biến macdinh = gtlen để so sánh cho lần gọi chương trình kế tiếp và cuối cùng trở về chương trình chính.

*Lưu đồ của trường hợp 1:*



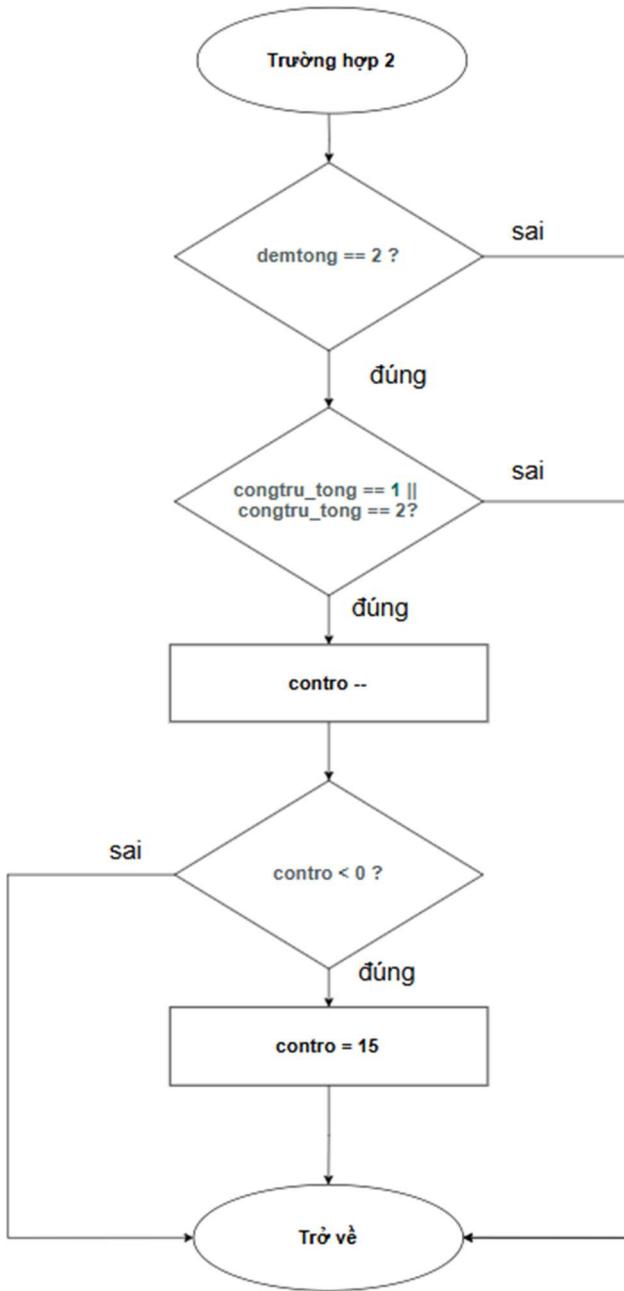
Hình 3.8: Lưu đồ của trường hợp 1 xuli\_nutlen()

Giải thích lưu đồ:

Đầu tiên kiểm tra biến `demptong` bằng 1 hay không, nếu không bằng 1 (sai) thì trở về chương trình `xuli_nutlen()`. Nếu bằng 1 (đúng) thì kiểm tra biến `congtru_tong`  $\leq 0$  hay không, nếu sai thì thực hiện biến `congtru_tong` trừ đi 1 và sau đó thì gọi chương trình con `menu_tong()` để cập nhật giao diện menu chính. Còn nếu đúng thì đặt biến `congtru_tong = 3` sau đó cũng gọi chương trình con `menu_tổng` để cập nhật giao diện menu chính. Cuối cùng là trở về chương trình `xuli_nutlen()`.

Chức năng của trường hợp 1 là điều khiển con trỏ ở giao diện menu chính theo chiều từ “ALARM” đến “TIME” đến “DATE” đến “BACK” rồi quay về “ALARM”.

*Lưu đồ của trường hợp 2:*



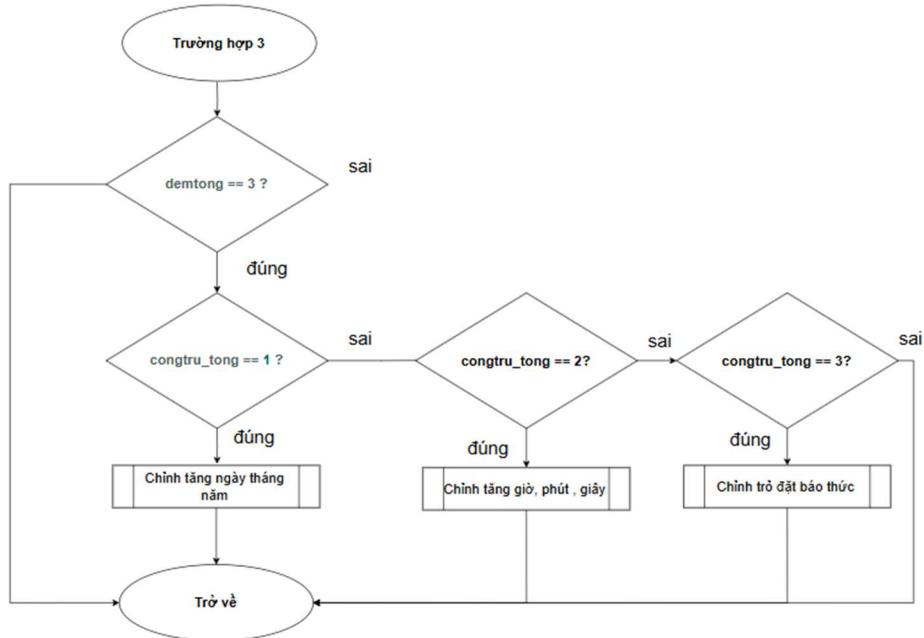
Hình 3.9: Lưu đồ của trường hợp 2 `xuli_nutlen()`

Giải thích lưu đồ:

Đầu tiên kiểm tra biến `demtong` bằng 2 là hay không, nếu không bằng 2 (sai) thì trở về chương trình `xuli_nutlen()`. Nếu bằng 2 (đúng) thì tiếp theo kiểm tra điều kiện là `congtru_tong = 1` hoặc `= 2`. Nếu không bằng hai giá trị 1 hoặc 2 (sai) thì trở về chương trình `xuli_nutlen()`. Còn nếu bằng một trong hai (đúng) thì lấy `contro` trừ đi 1, sau đó kiểm tra `contro < 0` hay không, nếu sai thì trở về chương trình `xuli_nutlen()`, còn nếu đúng thì đặt `contro = 15` và trở về chương trình `xuli_nutlen()`.

Chức năng của trường hợp 2: là di chuyển con trỏ trong các giao diện điều chỉnh DATE, TIME hay giao diện điều chỉnh đặt báo thức, chiều dịch con trỏ theo chiều phải sang trái.

### Lưu đồ trường hợp 3



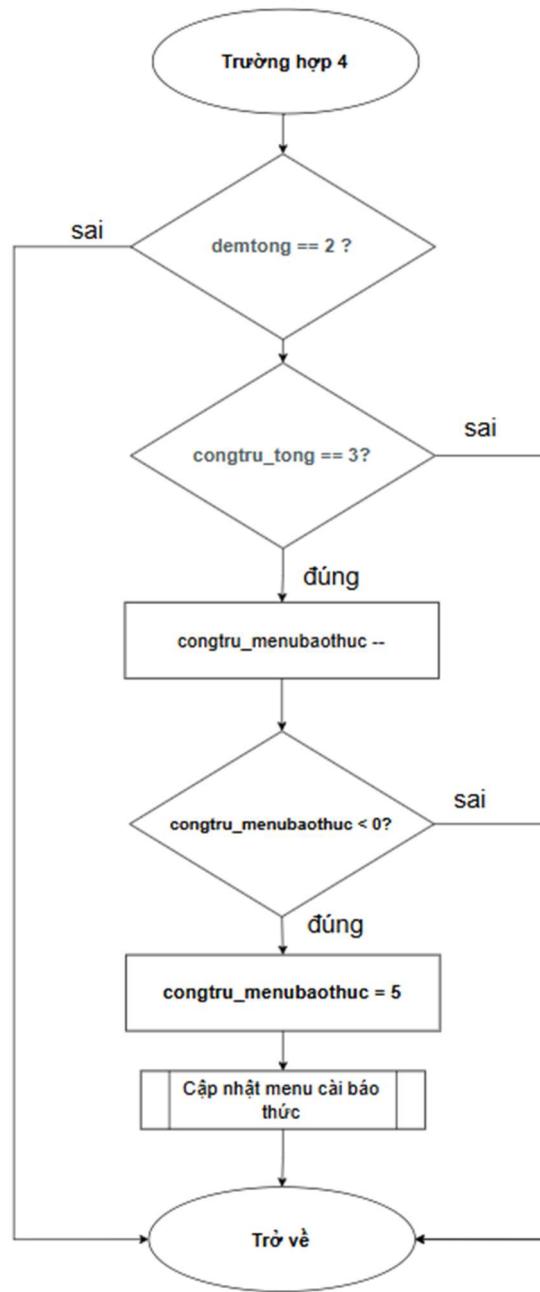
Hình 3.10: Lưu đồ trường hợp 3 của `xuli_nutlen()`

Giải thích lưu đồ:

Đầu tiên kiểm tra biến `demtong` bằng 3 là hay không, nếu không bằng 3 (`sai`) thì trả về chương trình `xuli_nutlen()`. Nếu bằng 3 (`đúng`) thì kiểm tra biến `congtru_tong`. Nếu `congtru_tong = 1` thì so sánh vị trí con trỏ với vị trí ô giá trị ngày tháng năm trong giao diện của mục DATE rồi tăng 1 đơn vị giá trị mà con trỏ đang ở đúng vị trí ô của giá trị đó. Nếu `congtru_tong = 2` thì cũng so sánh vị trí con trỏ với vị trí ô giá trị giờ phút giây trong giao diện của mục TIME rồi tăng 1 đơn vị giá trị mà con trỏ đang ở đúng vị trí của giá trị đó. Nếu `congtru_tong = 3` thì điều chỉnh trỏ đến vị trí giá trị ngày, tháng, giờ, phút ở trong giao diện điều chỉnh báo thức của mục báo thức, nếu nằm trong ba trường hợp của biến `contru_tong` trên thì cuối cùng là trả về chương trình `xuli_nutlen()`. Còn không nằm trong 3 trường hợp liệt kê ở trên thì trả về chương trình `xuli_nutlen()`.

Chức năng của trường hợp 3: chỉnh tăng các giá trị ngày, tháng, năm nếu congtru\_tong =1 (tức là giao diện điều chỉnh của mục DATE), chỉnh tăng các giá trị giờ, phút, giây nếu congtru\_tong =2 (tức là giao diện điều chỉnh của mục TIME), chỉnh con trỏ đến vị trí ô giá trị ngày, tháng, giờ , phút trong giao diện chỉnh sửa của mục ALARMx (x từ 1 đến 5 tùy vào giá trị biến congtru\_menubaothuc) khi congtru\_tong =3.

*Lưu đồ trường hợp 4:*



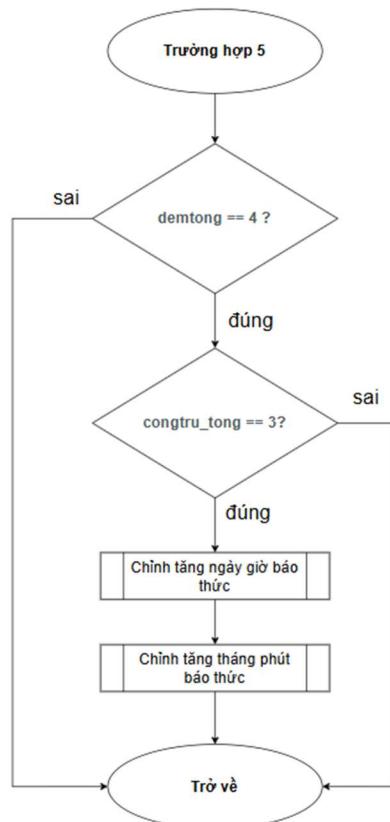
Hình 3.11: Lưu đồ của trường hợp 4 `xuli_nutlen()`

Giải thích lưu đồ:

Đầu tiên kiểm tra biến demtong bằng 2 là hay không, nếu không bằng 2 (sai) thì trở về chương trình xuli\_nutlen(). Nếu bằng 2 (đúng) thì tiếp theo kiểm tra điều kiện là congtru\_tong = 3. Nếu không bằng 3 (sai) thì trở về chương trình xuli\_nutlen(). Còn nếu bằng 3 (đúng) thì lấy congtru\_menubaothuc trừ đi 1, sau đó kiểm tra congtru\_menubaothuc < 0 hay không, nếu sai thì trở về chương trình xuli\_nutlen(), còn nếu đúng thì đặt c congtru\_menubaothuc = 5 rồi gọi chương trình menu\_baothuc() để cập nhật giao diện menu báo thức và trở về chương trình xuli\_nutlen().

Chức năng của trường hợp 4: điều khiển con trỏ trong giao diện menu báo thức để chọn mục theo chiều từ “ALARM5” đến “ALARM4” đến “ALARM3” đến “ALARM2” đến “ALARM1” đến “BACK” rồi quay về “ALARM5”.

Lưu đồ trường hợp 5:



Hình 3.12: Lưu đồ của trường hợp 5 xuli\_nutlen()

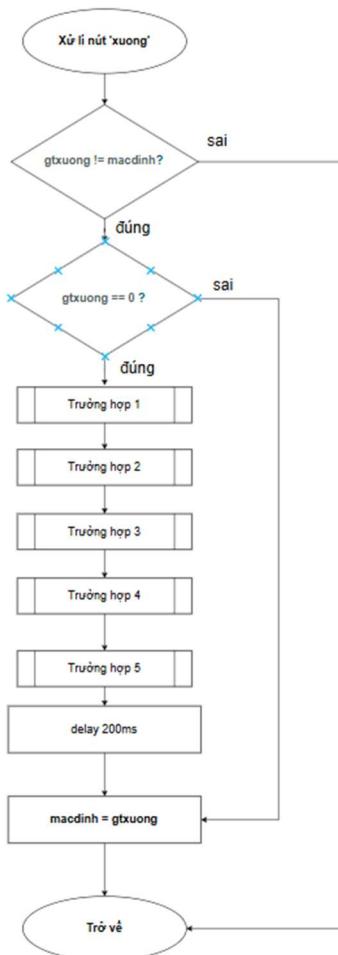
Giải thích lưu đồ:

Đầu tiên kiểm tra điều kiện demtong = 4 hay không, nếu sai thì trở về chương trình xuli\_nutlen(), còn nếu đúng thì kiểm tra điều kiện congtru\_tong = 3 hay không. Nếu sai thì trở về chương trình xuli\_nutlen(). Còn điều kiện đúng thì kiểm tra contro\_bt nếu contro\_bt ở vị trí 6, 7 thì tăng 1 đơn vị cho giá trị ngày nếu ở hàng 0 và cho giá trị giờ nếu ở hàng 2. Còn không thì kiểm tra vị trí contro\_bt có ở vị trí 9, 10 không, có thì tăng 1 đơn vị cho giá trị tháng nếu ở hàng 0 và cho giá trị phút nếu ở hàng 1.

Chức năng của trường hợp 5: là điều chỉnh tăng giá trị ngày, tháng, giờ, phút cho các giao diện đặt báo thức.

### 3.6.4 Chương trình con xuli\_nutxuong()

Lưu đồ chính:

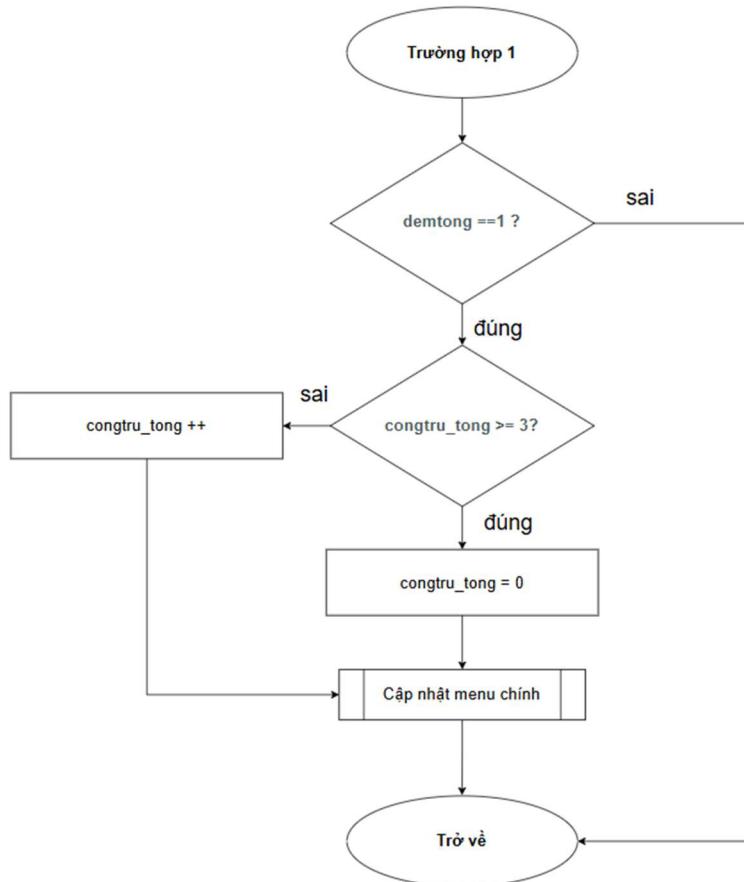


Hình 3.13: Lưu đồ chính của chương trình xuli\_nutxuong()

Giải thích lưu đồ:

Khi chương trình xuli\_nutxuong(), đầu tiên kiểm tra trạng thái biến gtxuong có thay đổi so với trạng thái trước được lưu trong biến macdinh nếu không thì trở về chương trình còn nếu có thì kiểm tra gtxuong == 0 (tương đương với nút len được nhấn). Nếu bằng 1 (không có nhấn) thì trở về chương trình chính, còn nếu bằng 0 (có nhấn) thì kiểm tra các trường hợp xử lí nút nhấn để thực hiện chức năng của nút xuong. Sau khi kiểm tra các trường hợp thì delay 200ms và đặt biến macdinh = gtxuong để so sánh cho lần gọi chương trình kế tiếp và cuối cùng trở về chương trình chính.

*Lưu đồ của trường hợp 1:*



Hình 3.14: Lưu đồ của trường hợp 1 xuli\_nutxuong()

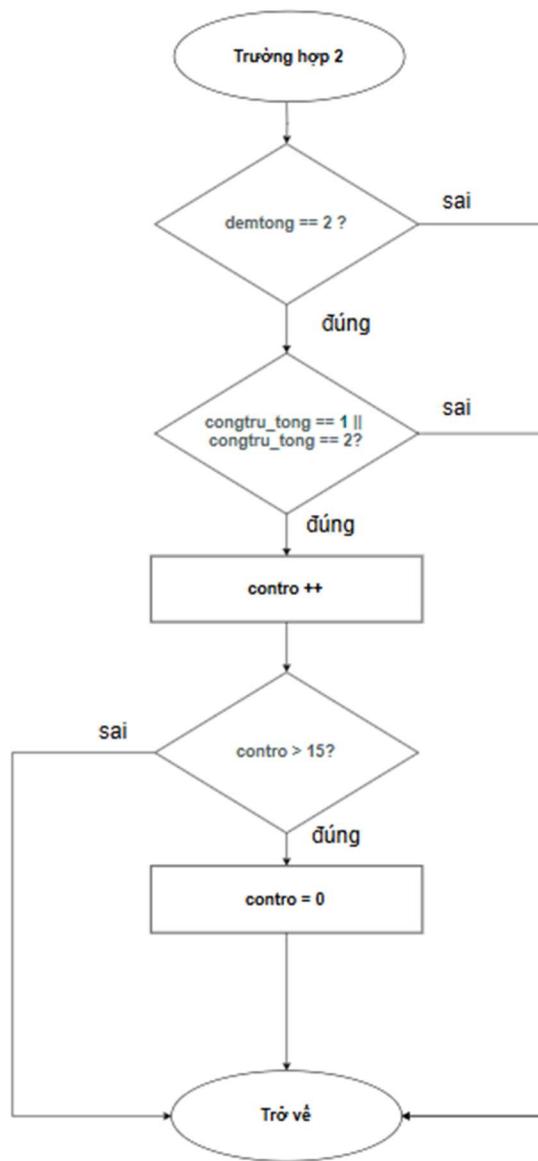
Giải thích lưu đồ:

Đầu tiên kiểm tra biến demtong bằng 1 hay không, nếu không bằng 1 (sai) thì trở về chương trình xuli\_nutxuong(). Nếu bằng 1 (đúng) thì kiểm tra biến

`congtru_tong`  $\geq 3$  hay không, nếu sai thì thực hiện biến `congtru_tong` cộng thêm 1 và sau đó thì gọi chương trình con `menu_tong()` để cập nhật giao diện menu chính. Còn nếu đúng thì đặt biến `congtru_tong` = 0 sau đó cũng gọi chương trình con `menu_tong()` để cập nhật giao diện menu chính. Cuối cùng là trở về chương trình `xuli_nutxuong()`.

Chức năng của trường hợp 1 là điều khiển con trỏ ở giao diện menu chính theo chiều từ “BACK” đến “DATE” đến “TIME” đến “ALARM” rồi quay về “BACK”.

*Lưu đồ của trường hợp 2:*



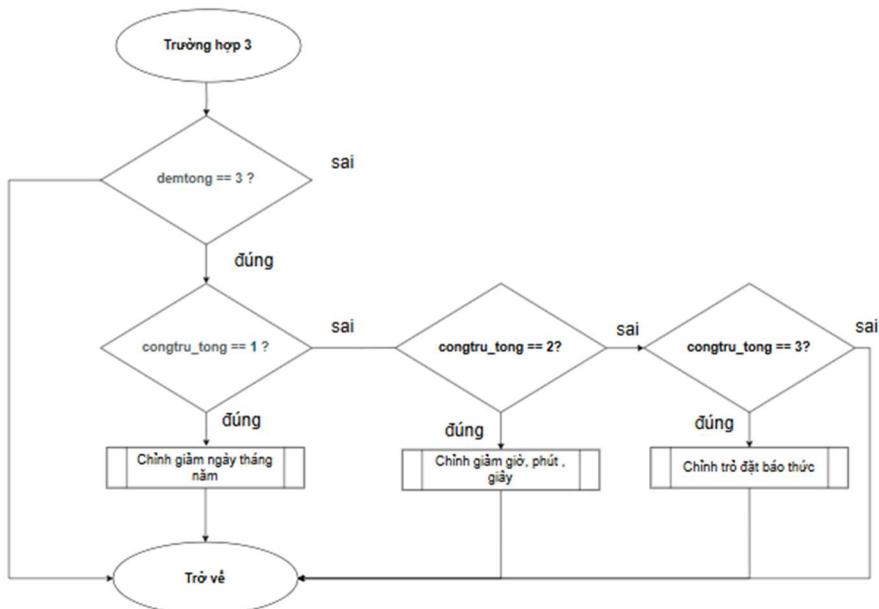
Hình 3.15: Lưu đồ của trường hợp 2 `xuli_nutxuong()`

Giải thích lưu đồ:

Đầu tiên kiểm tra biến demtong bằng 2 là hay không, nếu không bằng 2 (sai) thì trở về chương trình xuli\_nutxuong(). Nếu bằng 2 (đúng) thì tiếp theo kiểm tra điều kiện là congtru\_tong = 1 hoặc = 2. Nếu không bằng hai giá trị 1 hoặc 2 (sai) thì trở về chương trình xuli\_nutxuong(). Còn nếu bằng một trong hai (đúng) thì lấy contro cộng thêm 1, sau đó kiểm tra contro > 15 hay không, nếu sai thì trở về chương trình xuli\_nutxuong(), còn nếu đúng thì đặt contro = 0 và trở về chương trình xuli\_nutxuong().

Chức năng của trường hợp 2: là di chuyển con trỏ trong các giao diện điều chỉnh DATE, TIME hay giao diện điều chỉnh đặt báo thức, chiều dịch con trỏ theo chiều trái sang phải.

### Lưu đồ trường hợp 3



Hình 3.16: Lưu đồ trường hợp 3 của xuli\_nutxuong()

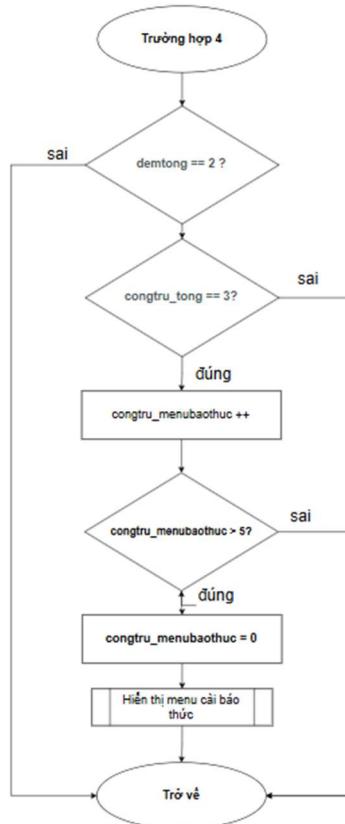
Giải thích lưu đồ:

Đầu tiên kiểm tra biến demtong bằng 3 là hay không, nếu không bằng 3 (sai) thì trở về chương trình xuli\_nutxuong(). Nếu bằng 3 (đúng) thì kiểm tra biến congtru\_tong. Nếu congtru\_tong = 1 thì so sánh vị trí con trỏ với vị trí ô giá trị ngày tháng năm trong giao diện của mục DATE rồi tăng 1 đơn vị giá trị mà con trỏ đang ở đúng vị trí ô của giá trị đó. Nếu congtru\_tong = 2 thì cũng so sánh vị trí con trỏ với

vị trí ô giá trị giờ phút giây trong giao diện của mục TIME rồi tăng 1 đơn vị giá trị mà con trỏ đang ở đúng vị trí của giá trị đó. Nếu `congtru_tong` = 3 thì điều chỉnh trỏ đến vị trí giá trị ngày, tháng, giờ, phút ở trong giao diện điều chỉnh báo thức của mục báo thức, nếu nằm trong ba trường hợp của biến `contru_tong` trên thì cuối cùng là trở về chương trình `xuli_nutxuong()`. Còn không nằm trong 3 trường hợp liệt kê ở trên thì trở về chương trình `xuli_nutxuong()`.

Chức năng của trường hợp 3: chỉnh tăng các giá trị ngày, tháng, năm nếu `congtru_tong` = 1 (tức là giao diện điều chỉnh của mục DATE), chỉnh tăng các giá trị giờ, phút, giây nếu `congtru_tong` = 2 (tức là giao diện điều chỉnh của mục TIME), chỉnh con trỏ đến vị trí ô giá trị ngày, tháng, giờ, phút trong giao diện chỉnh sửa của mục ALARMx (x từ 1 đến 5 tùy vào giá trị biến `congtru_menubaothuc`) khi `congtru_tong` = 3.

*Lưu đồ trường hợp 4:*



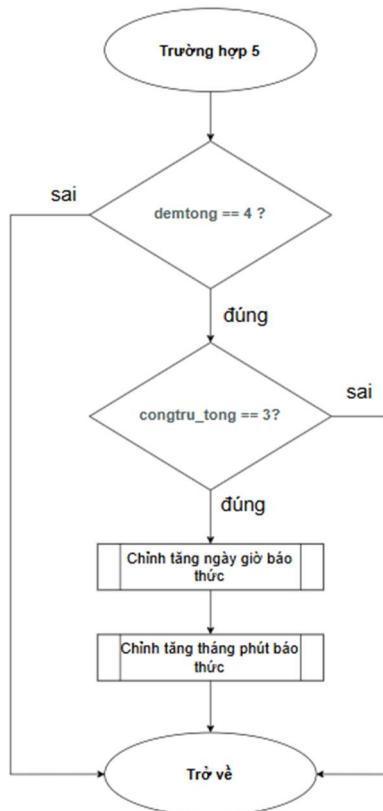
Hình 3.17: Lưu đồ của trường hợp 4 `xuli_nutxuong()`

Giải thích lưu đồ:

Đầu tiên kiểm tra biến demtong bằng 2 là hay không, nếu không bằng 2 (sai) thì trở về chương trình xuli\_nutxuong(). Nếu bằng 2 (đúng) thì tiếp theo kiểm tra điều kiện là congtru\_tong = 3. Nếu không bằng 3 (sai) thì trở về chương trình xuli\_nutxuong(). Còn nếu bằng 3 (đúng) thì lấy congtru\_menubaothuc cộng thêm 1, sau đó kiểm tra congtru\_menubaothuc > 5 hay không, nếu sai thì trở về chương trình xuli\_nutxuong(), còn nếu đúng thì đặt c congtru\_menubaothuc = 0 rồi gọi chương trình menu\_baothuc() để cập nhật giao diện menu báo thức và trở về chương trình xuli\_nutxuong().

Chức năng của trường hợp 4: điều khiển con trỏ trong giao diện menu báo thức để chọn mục theo chiều từ “BACK” đến “ALARM1” đến “ALARM2” đến “ALARM3” đến “ALARM4” đến “ALARM5” rồi quay về “BACK”.

Lưu đồ trường hợp 5:



Hình 3.19: Lưu đồ của trường hợp 5 xuli\_nutxuong()

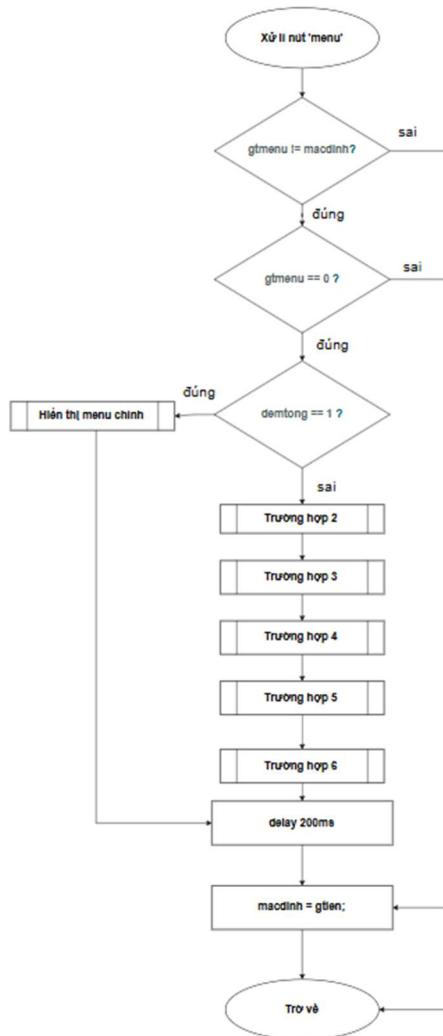
Giải thích lưu đồ:

Đầu tiên kiểm tra điều kiện demtong = 4 hay không, nếu sai thì trở về chương trình xuli\_nutxuong(), còn nếu đúng thì kiểm tra điều kiện congtru\_tong =3 hay không. Nếu sai thì trở về chương trình xuli\_nutxuong(). Còn điều kiện đúng thì kiểm tra contro\_bt nếu contro\_bt ở vị trí 6, 7 thì giảm đi 1 đơn vị cho giá trị ngày nếu ở hàng 0 và cho giá trị giờ nếu ở hàng 2. Còn không thì kiểm tra vị trí contro\_bt có ở vị trí 9, 10 không, có thì giảm đi 1 đơn vị cho giá trị tháng nếu ở hàng 0 và cho giá trị phút nếu ở hàng 1.

Chức năng của trường hợp 5: là điều chỉnh giảm giá trị ngày, tháng, giờ, phút cho các giao diện đặt báo thức.

### 3.6.5 Chương trình con xuli\_nutmenu()

*Lưu đồ chính*

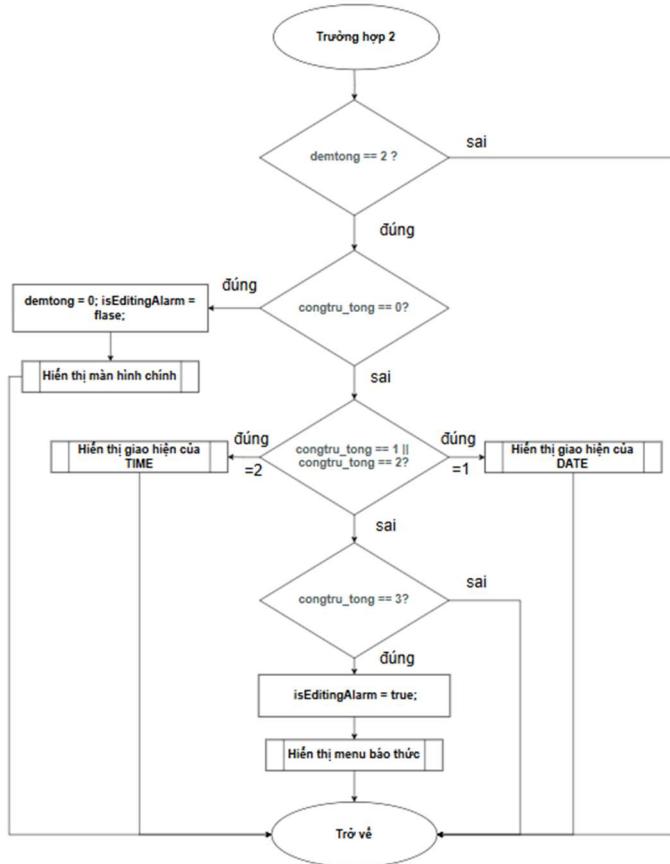


Hình 3.20: Lưu đồ chính của chương trình xuli\_nutmenu()

Giải thích lưu đồ:

Khi chương trình xuli\_nutmenu(), đầu tiên kiểm tra trạng thái biến gtmenu có thay đổi so với trạng thái trước được lưu trong biến macdinh nếu không thì trở về chương trình chính còn nếu có thì kiểm tra gtmenu == 0 (tương đương với nút len được nhấn). Nếu bằng 1 (không có nhấn) thì trở về chương trình chính, còn nếu bằng 0 (có nhấn) thì kiểm tra các trường hợp xử lý nút nhấn để thực hiện chức năng của nút menu. Trường hợp 1 thì kiểm tra demtong = 1 hay không, nếu sai thì kiểm tra tiếp các trường hợp còn lại, nếu đúng thì gọi chương trình menu\_tong() để hiển thị giao diện menu chính rồi delay 200ms. Sau khi kiểm tra các trường hợp thì delay 200ms và đặt biến macdinh = gtmenu để so cho lần gọi chương trình kế tiếp và cuối cùng trở về chương trình chính.

### *Lưu đồ của trường hợp 2*



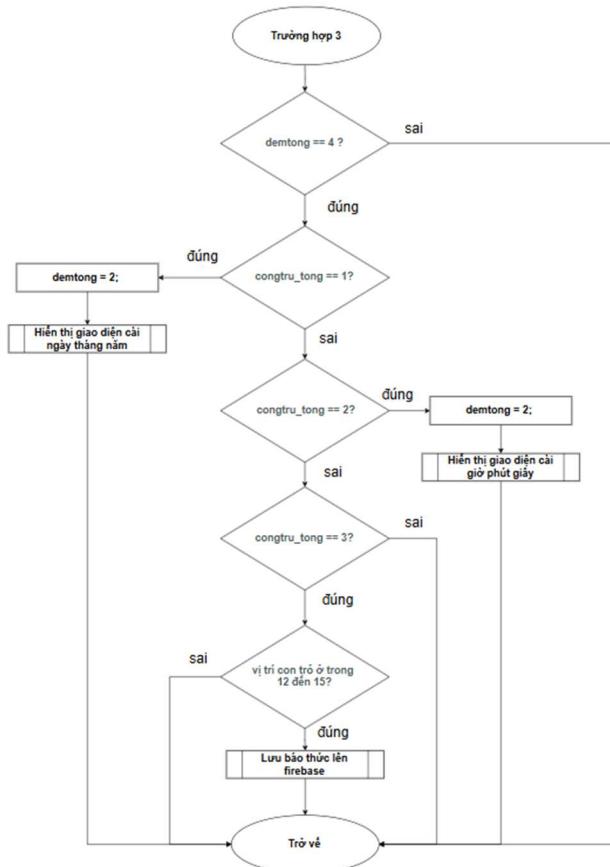
Hình 3.21: Lưu đồ của trường hợp 2 xuli\_nutmenu()

Giải thích lưu đồ:

Đầu tiên kiểm tra biến demtong bằng 2 hay không, nếu không bằng 2 (sai) thì trở về chương trình xuli\_nutmenu(). Nếu bằng 0 (đúng) thì kiểm tra biến congtru\_tong. Nếu congtru\_tong = 0 thì đặt demtong=0, isEditingAlarm = flase và sau đó gọi chương trình con manhinh() để hiển thị giao diện màn hình chính và cuối cùng là trở về chương trình xuli\_nutmenu() còn nếu không bằng 0 thì kiểm tra tiếp. Nếu congtru\_tong bằng 1 hoặc 2 thì gọi chương trình con chonmenu\_tong() bằng 1 thì hiển thị giao diện điều chỉnh DATE, bằng 2 thì hiển thị giao diện điều chỉnh TIME sau đó trở về chương trình xuli\_nutmenu(), nếu còn không trong hai giá trị trên thì kiểm tra tiếp. Nếu congtru\_tong bằng 3 đặt isEditingAlarm = true, sau đó gọi chương trình con menu\_baothuc() để hiển thị giao diện chọn mục ALARM rồi sau đó trở về chương trình xuli\_nutmenu. Nếu không nằm trong 3 trường hợp trên thì trở về chương trình xuli\_nutmenu.

Chức năng của trường hợp 2 là hiển thị giao diện màn hình hiện tại.

### Lưu đồ của trường hợp 3



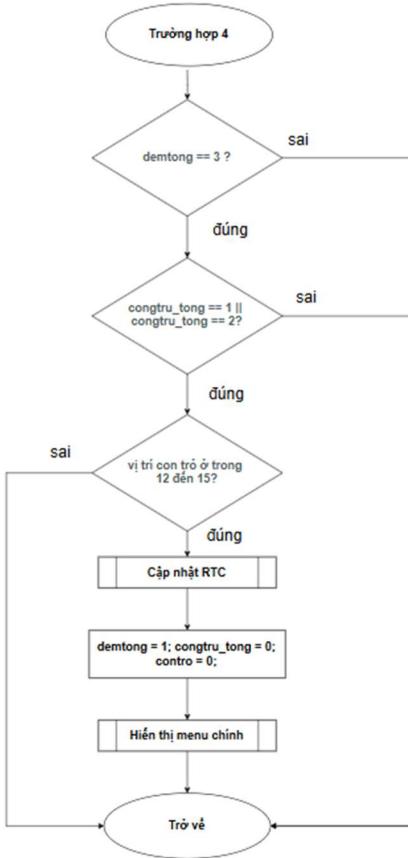
Hình 3.22: Lưu đồ của trường hợp 3 xuli\_nutmenu()

Giải thích lưu đồ:

Đầu tiên kiểm tra biến demtong bằng 4 là hay không, nếu không bằng 4 (sai) thì trở về chương trình xuli\_nutmenu(). Nếu bằng 4 (đúng) thì tiếp theo kiểm tra điều kiện là congtru\_tong = 1. Nếu không bằng 1 (sai) thì kiểm tra điều kiện tiếp theo. Còn nếu bằng một (đúng) thì đặt demtong = 2, gọi chương trình gọi chonmenu\_tong() để quay lại giao diện chỉnh sửa DATE. Nếu congtru\_tong = 2 thì đặt demtong = 2, gọi chương trình gọi chonmenu\_tong() để quay lại giao diện chỉnh sửa TIME rồi trở về chương trình xuli\_nutmenu(), nếu không bằng 2 thì kiểm tra điều kiện tiếp theo. Nếu congtru\_tong = 3 thì kiểm tra vị trí contro\_bt có nằm trong khoảng 12 đến 15 (ở vị trí chữ “SAVE”, nếu đúng cập nhật các giá trị báo thức lên firebase rồi đặt isEditingAlarm = false rồi gọi hàm menu\_baothuc để quay lại menu báo thức rồi đặt các giá trị demtong=2, congtru\_tong = 3, congtr\_bt =6, hang = 0 và tắt con trỏ của lcd rồi đặt biến lastFirebaseSync =0 để kích hoạt đồng bộ dữ liệu báo thức lên Firebase.

Chức năng của trường hợp 3: là để quay lại giao diện cài DATE hoặc TIME sau khi lưu giá trị đã cài lại, còn để cập nhật lên alarm lên Firebase và quay lại menu báo thức sau khi lưu alarm đã cài.

*Lưu đồ trường hợp 4*



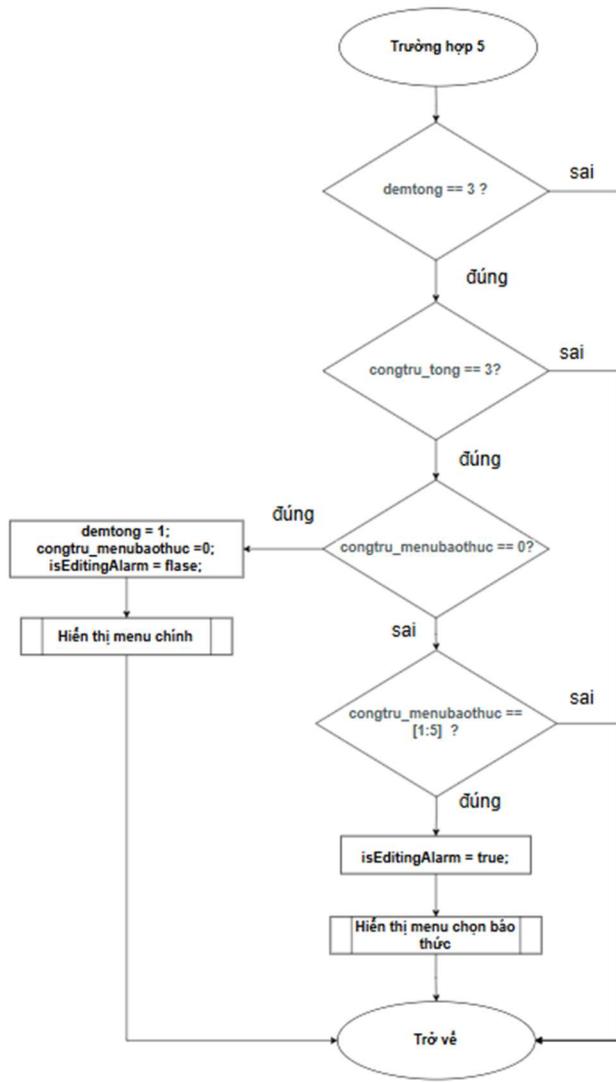
Hình 3.23: Lưu đồ trường hợp 4 của xuli\_nutmenu()

Giải thích lưu đồ:

Đầu tiên kiểm tra biến `demptong` bằng 3 là hay không, nếu không bằng 3 (sai) thì trở về chương trình `xuli_nutmenu()`. Nếu bằng 3 (đúng) thì kiểm tra biến `congtru_tong`. Nếu `congtru_tong` = 1 hoặc = 2 thì kiểm tra vị trí `contro` có nằm trong khoảng 12 đến 15 (ở vị trí chữ “SAVE”), nếu sai thì trở về chương trình `xuli_nutmenu()`, nếu đúng thì gọi hàm `updateRTC()` để lưu giá trị thời gian mới vào RTC rồi đặt `demptong=1`, `congtru_tong =0`, `contro =0`, rồi sau đó gọi chương trình `menu_tong()` để quay lại menu chính rồi tắt con trỏ của LCD và cuối cùng trở về chương trình `xuli_nutmenu()`. Nếu `congtru_tong` không =1 hoặc =2 thì trở về chương trình `xuli_nutmenu()`.

Chức năng của trường hợp 4: : là để cập nhật RTC với giá trị mới cài ở giao diện DATE và TIME rồi quay lại menu chính.

*Lưu đồ trường hợp 5*



Hình 3.24: Lưu đồ của trường hợp 5 xuli\_nutmenu()

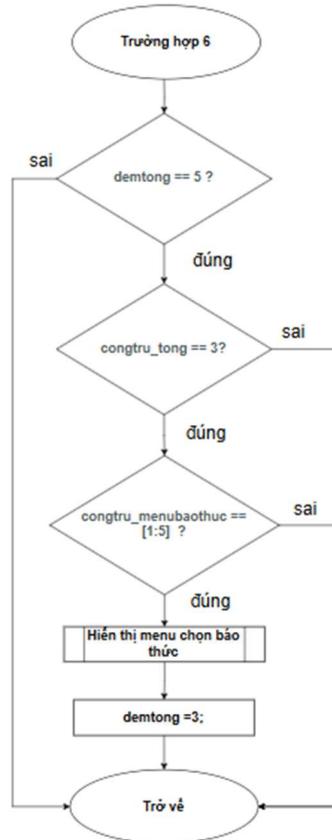
Giải thích lưu đồ:

Đầu tiên kiểm tra biến `demtong` bằng 3 là hay không, nếu không bằng 3 (sai) thì trở về chương trình `xuli_nutmenu()`. Nếu bằng 3 (đúng) thì tiếp theo kiểm tra điều kiện là `congtru_tong = 3`. Nếu không bằng 3 (sai) thì trở về chương trình `xuli_nutmenu()`. Còn nếu bằng 3 (đúng) thì kiểm tra `congtru_menubaothuc = 0` hay không, nếu đúng thì đặt lại `demtong = 1`, `congtru_menubaothuc = 0`, `isEditingAlarm = false` sau đó gọi chương trình con `menu_tong()` để quay lại menu tổng rồi trở về chương trình `xuli_nutmenu()`. Còn sai thì kiểm tra `congtru_menubaothuc` có giá trị trong khoảng 1 đến 5, sai thì trở về chương trình `xuli_nutmenu()`. Còn nếu đúng thì

đặt biến isEditingAlarm = true (cho phép chỉnh sửa báo thức) rồi gọi chương trình con chonmenu\_baothuc() để cập nhật giao diện chỉnh sửa báo thức.

Chức năng của trường hợp 4: để quay trở lại menu chính nếu chọn mục “BACK” và để hiển thị giao diện chỉnh sửa báo thức nếu chọn mục “ALARM(1-5)”.

### Lưu đồ trường hợp 6



Hình 3.25: Lưu đồ của trường hợp 6 xuli\_nutmenu()

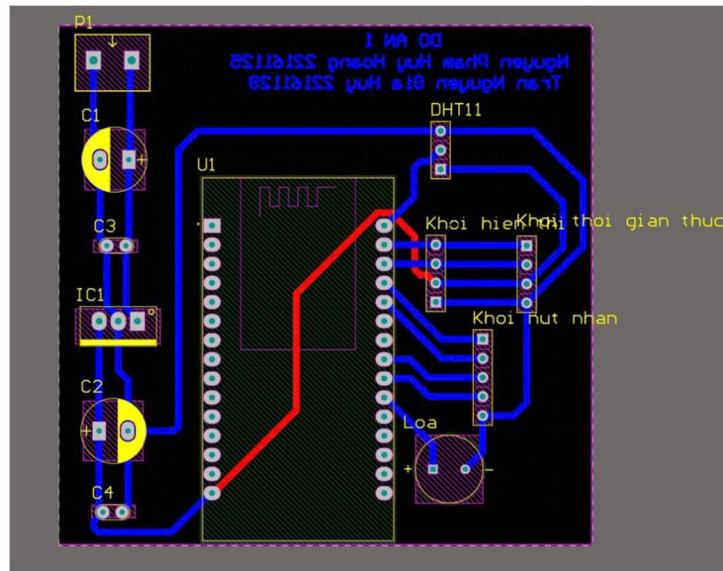
Giải thích lưu đồ:

Đầu tiên kiểm tra điều kiện `demtong = 5` hay không, nếu sai thì trở về chương trình `xuli_nutmenu()`, còn nếu đúng thì kiểm tra điều kiện `congtru_tong = 3` hay không. Nếu sai thì trở về chương trình `xuli_nutmenu()`. Còn nếu đúng thì kiểm tra `congtru_menubaothuc` có giá trị trong khoảng 1 đến 5, sai thì trở về chương trình `xuli_nutmenu()`. Còn nếu đúng thì gọi chương trình con `chonmenu_baothuc()` để hiển thị giao diện chỉnh sửa báo thức và đặt `demtong = 3` và cuối cùng trở về chương trình `xuli_nutmenu()`.

Chức năng của trường hợp 5: để hiển thị lại giao diện chỉnh sửa tại mục hiện tại sau khi lưu giá trị báo thức tại mục đó.

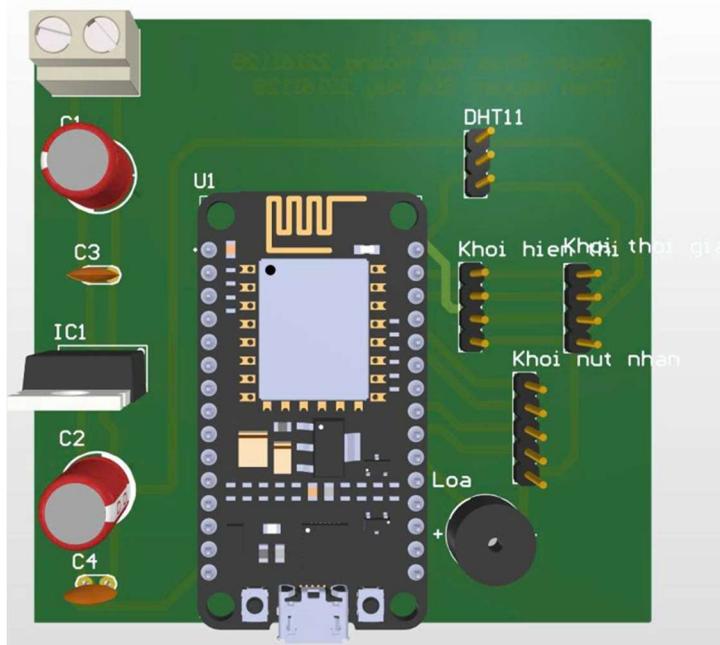
### 3.7 VẼ MẠCH PCB

Thi công mạch PCB trên phần mềm Altium Designer gồm các công việc bố trí và đi dây với các linh kiện với nhau. Chúng tôi đã vẽ được mạch PCB như sau:



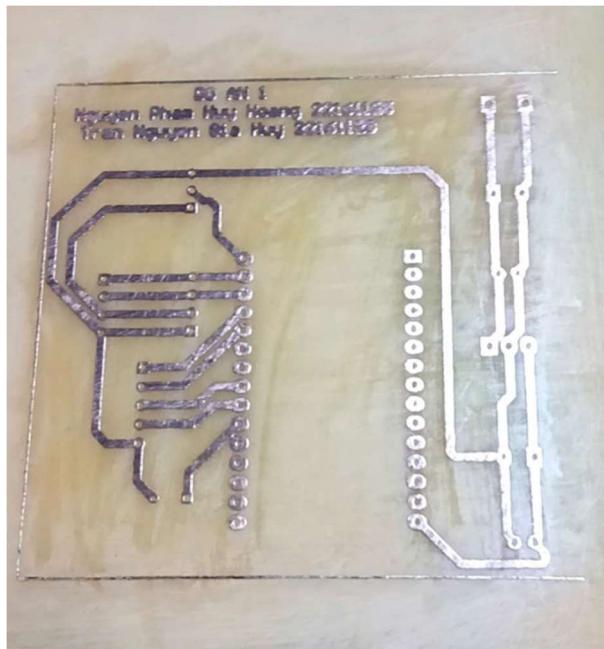
Hình 3.26: Mạch PCB của hệ thống trên Altium Designer

Chuyển qua chế độ 3D, ta được như hình sau:



Hình 3.27: Mạch PCD của hệ thống ở chế độ 3D

Sau khi thiết kế, vẽ xong mạch PCB thì đem đi in ra mạch đồng. Kết quả của công việc in mạch đồng như hình 3.28:

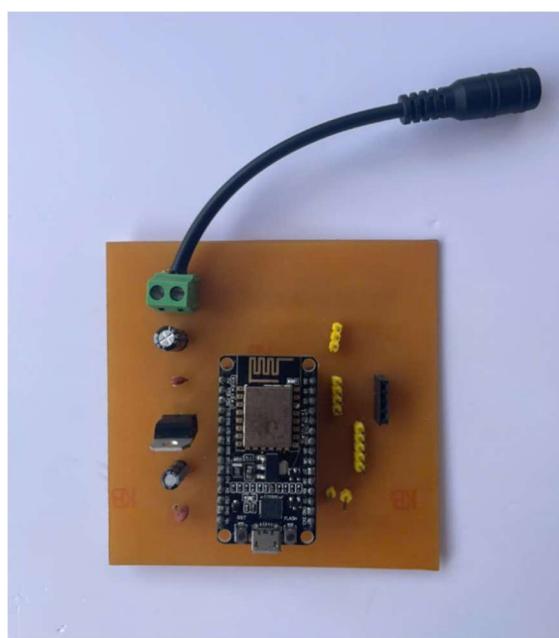


Hình 3.28: Bảng mạch in

### 3.8 THI CÔNG HỆ THỐNG

Công việc đầu tiên là lắp các linh kiện và hàn các linh kiện cố định lên mạch in.

Kết quả của công việc trên, như hình 3.29:



Hình 3.29: Mạch in hoàn thiện của hệ thống

Tiếp theo, thi công một mô hình để kết nối phần cứng trung tâm (mạch in) với các khối ngoại vi, như sau:



Hình 3.30: Góc nhìn từ trên xuống mô hình



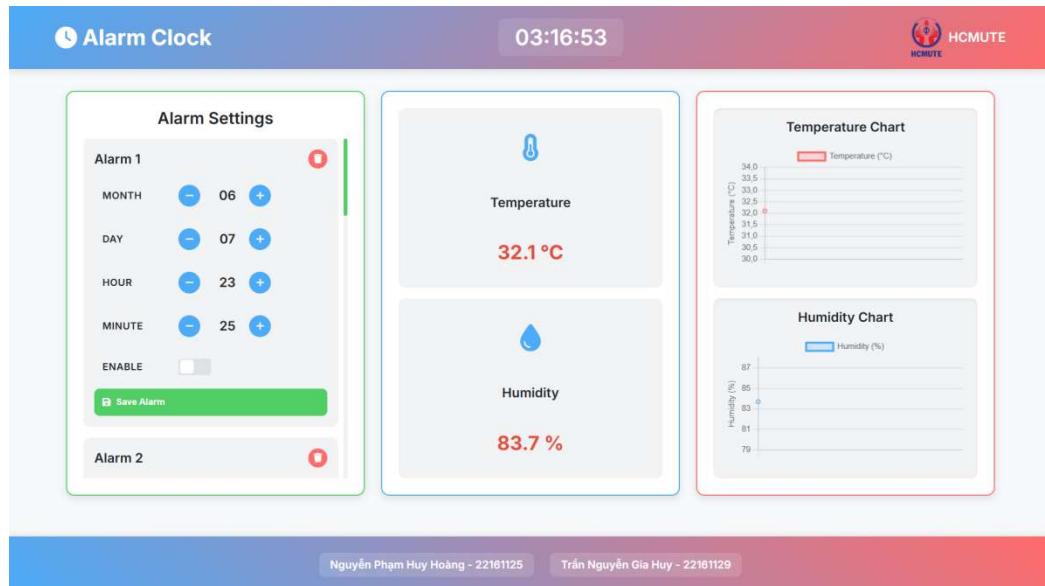
Hình 3.31: Góc nhìn trực diện mô hình

### 3.9 THIẾT KẾ DASHBOARD

Công việc: thiết kế một giao diện web để điều khiển hệ thống từ xa.

Chức năng của dashboard: đọc dữ liệu nhiệt độ và độ ẩm từ realtime database để theo dõi nhiệt độ, độ ẩm của không gian (nơi đặt đồng hồ), điều khiển đặt báo thức từ xa cho hệ thống tối đa đặt được 5 báo thức, tích hợp biểu đồ để hiển thị sự thay đổi nhiệt độ, độ ẩm môi trường.

Dưới đây là giao diện dashboard đã thiết kế:



Hình 3.32: Giao diện Dashboard

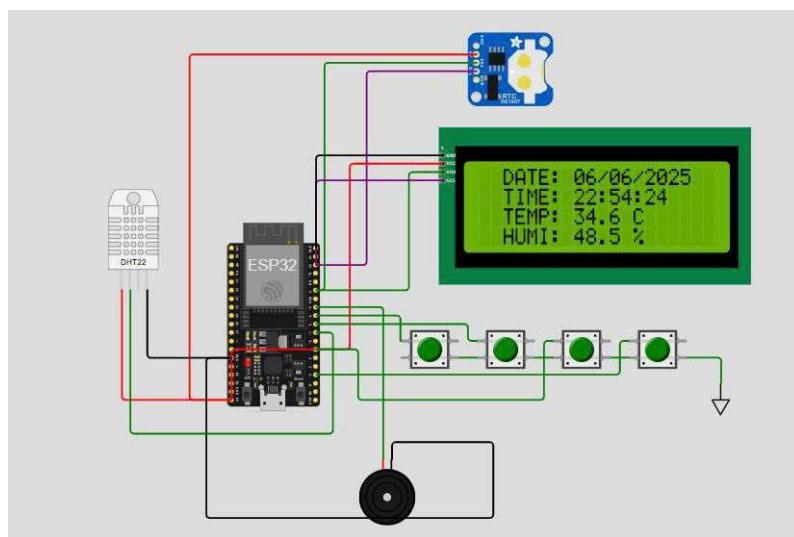
# CHƯƠNG 4 THỰC NGHIỆM VÀ ĐÁNH GIÁ

## 4.1 THỰC NGHIỆM

### 4.1.1 Kết quả mô phỏng

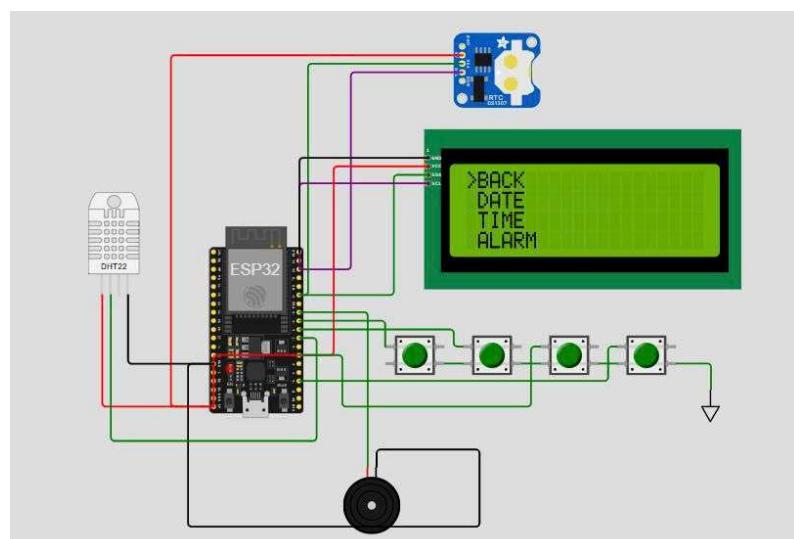
Các kết quả mô phỏng được thực hiện trên web Wokwi.com, nhưng web chỉ hỗ trợ esp32 nên thay đổi về sơ đồ mạch và kết nối chân I/O so với hệ thống thực tế và do mô phỏng nên sẽ không có kết quả điều khiển trên dashboard.

Giao diện màn hình chính hiển thị DATE, TIME, TEMP, HUMI.



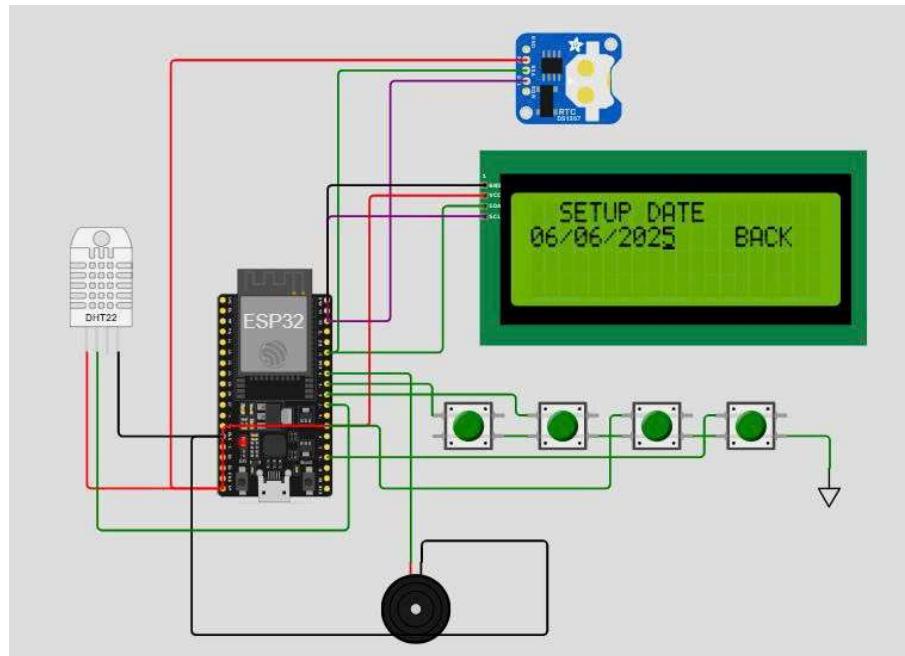
Hình 4.1 Giao diện màn hình chính của hệ thống

Giao diện menu chính để chọn mục của hệ thống.



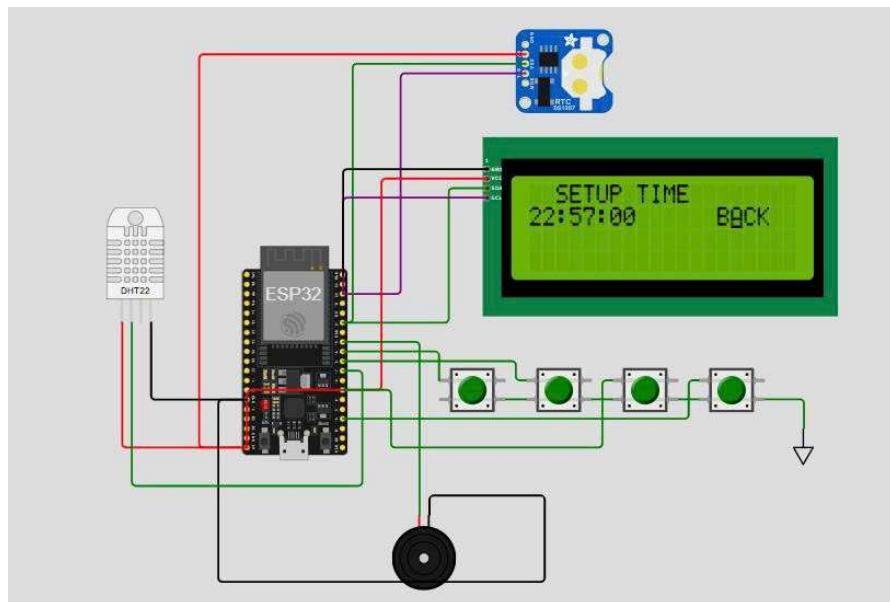
Hình 4.2: Giao diện menu chính của hệ thống

Giao diện cài lại ngày tháng năm sau khi chọn mục DATE.



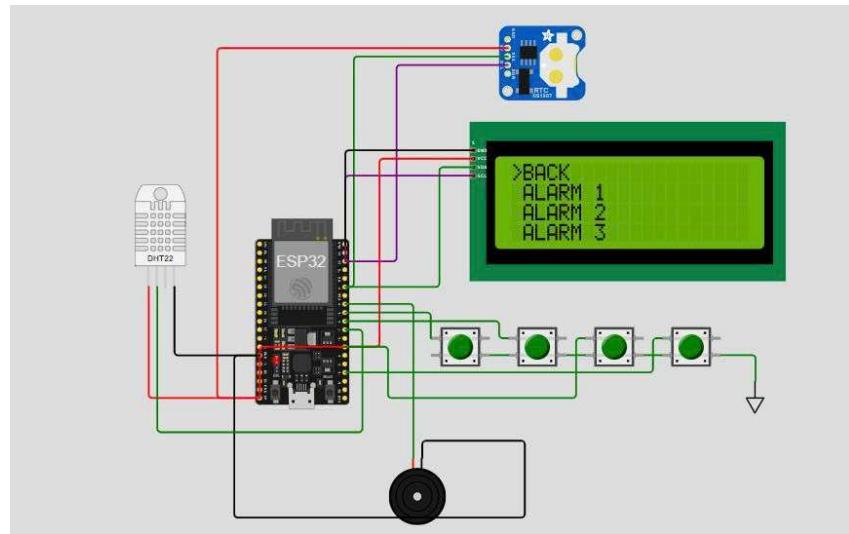
Hình 4.3: Giao diện cài lại DATE

Giao diện cài lại giờ, phút, giây sau khi chọn mục TIME.

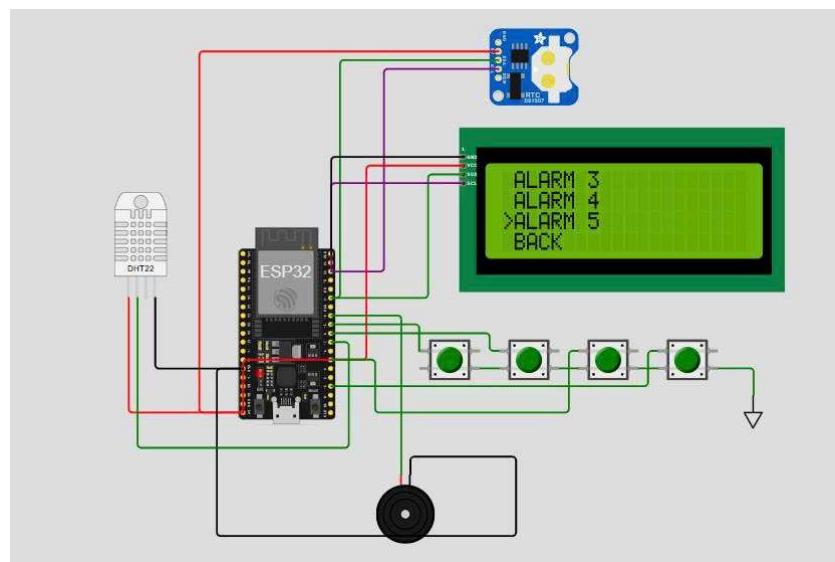


Hình 4.4: Giao diện cài lại TIME

Giao diện menu chọn báo thức sau khi chọn mục ALARM.

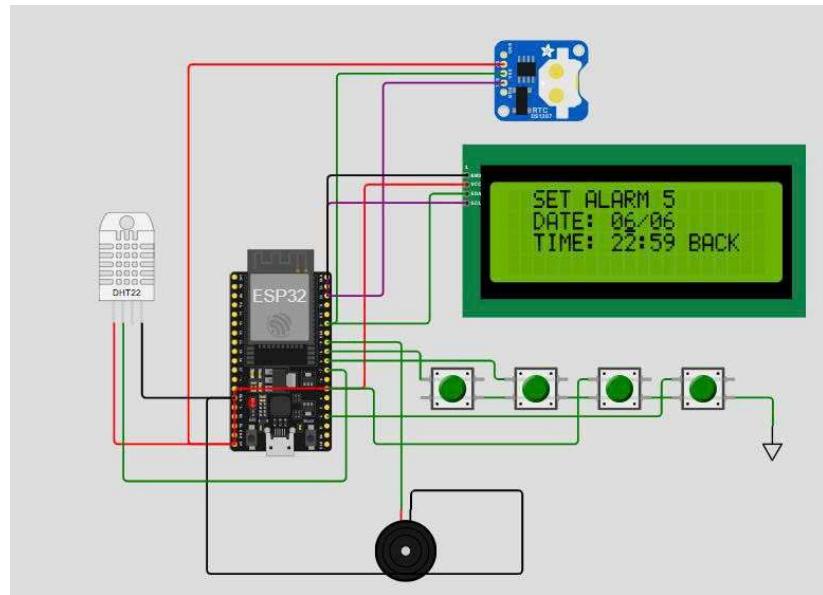


Hình 4.5: Menu chọn báo thức



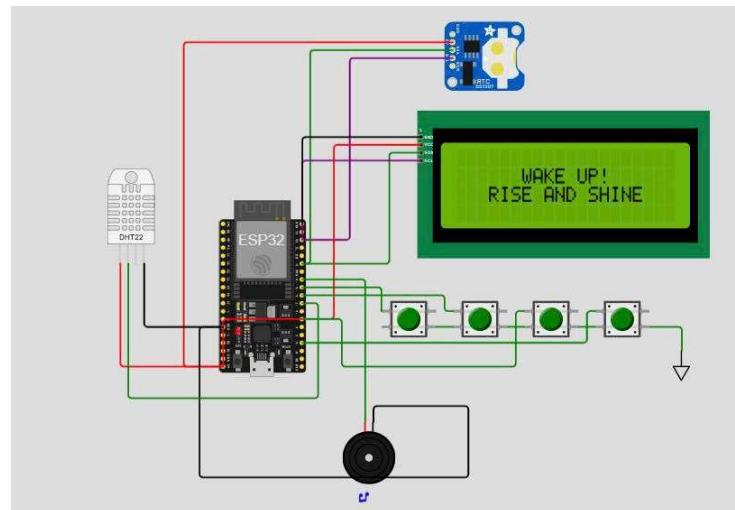
Hình 4.6: Menu chọn báo thức sau khi di chuyển

Giao diện đặt báo thức sau khi chọn mục ALARM5.



Hình 4.7: Giao diện đặt báo thức của ALARM5

Giao diện khi kêu báo thức:



Hình 4.8 Giao diện khi kêu báo thức

#### 4.1.2 Kết quả thực tế

Giao diện màn hình chính hiển thị DATE, TIME, TEMP, HUMI.



Hình 4.9: Giao diện màn hình chính của hệ thống

Giao diện menu chính để chọn mục của hệ thống.



Hình 4.10: Giao diện menu chính của hệ thống

Giao diện cài lại ngày tháng năm sau khi chọn mục DATE.



Hình 4.11: Giao diện chỉnh sửa DATE

Giao diện cài lại giờ, phút, giây sau khi chọn mục TIME.



Hình 4.12: Giao diện chỉnh sửa TIME

Giao diện menu chọn báo thức sau khi chọn mục ALARM.



Hình 4.13: Menu chọn báo thức



Hình 4.14: Menu chọn báo thức sau khi di chuyển

Giao diện đặt báo thức sau khi chọn mục ALARM5.



Hình 4.15: Giao diện đặt báo thức của ALARM5

Giao diện khi kêu báo thức:



Hình 4.16: Giao diện khi kêu báo thức

Điều khiển trên dashboard:

Hiển thị thông tin nhiệt độ, độ ẩm từ Firebase lên dashboard.



Hình 4.17: Màn hình dashboard và màn hình chính của hệ thống

Đặt báo thức trên dashboard:



Hình 4.18: Đặt báo thức trên giao diện dashboard



Hình 4.19: Giá trị đặt báo thức lưu trên realtime database

Các đặt báo thức trên dashboard sẽ được lưu trên realtime database của firebase sẽ hệ thống sẽ đồng bộ báo thức trên firebase:



Hình 4.20: ALARM1 trên hệ thống đã cập nhật từ firebase

## 4.2 ĐÁNH GIÁ

### 4.2.1 Ưu điểm

Hệ thống đáp ứng được các yêu cầu thiết kế đã đề ra.

Hệ thống đáp ứng tốt đối với các tín hiệu từ cảm biến DHT22, module RTC DS1307 và tín hiệu điều khiển từ nút nhấn.

Thao tác điều khiển đơn giản và dễ sử dụng.

### 4.2.2 Nhược điểm

Chí phí phần cứng khá cao.

Tiêu thụ năng lượng lớn.

Tương tác giữa hệ thống và firebase có sự delay cao.

Thuận toán chương trình dài dòng chưa tối ưu.

Cảm biến DHT22 đọc nhiệt độ, độ ẩm có sai số lớn so với nhiệt độ, độ ẩm môi trường ngoài.

Chưa có tái sử dụng nguồn năng lượng.

Mô hình hệ thống thiếu tính thẩm mỹ.

## **CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

### **5.1 KẾT LUẬN**

Sau thời gian nghiên cứu thiết kế, hoàn thành và chạy thử hệ thống để tài “**Đồng hồ đa năng**” người thực hiện hoàn thành mô hình hệ thống đáp ứng được mục tiêu đặt ra như sau:

- Có giao diện cho người dùng trên phần cứng là LCD để hiển thị thời gian và xem nhiệt độ độ ẩm và các nút nhấn để người dùng cài đặt ngày, giờ và đặt báo thức.
- Có giao diện Website để người dùng xem nhiệt độ độ ẩm online và cài báo thức qua Website.
- Có biểu đồ quan sát nhiệt độ độ ẩm
- Có máy chủ lưu trữ dữ liệu cho hệ thống mà không sợ bị mất đi.
- Có giao diện màn hình LCD hiển thị sắc nét, đầy đủ thông tin.

### **5.2 HƯỚNG PHÁT TRIỂN**

Tích hợp thêm app mobile để tiện dụng hơn trong việc cài giờ hoặc đặt báo thức.

Mặc dù có thể đặt báo thức qua Website tuy nhiên cần mở rộng thêm chức năng như chỉnh giờ chính ngày qua Website.

Tùy chỉnh lại giao diện Website thiết kế lại phần chỉnh báo thức để người dùng sử dụng tiện lợi hơn.

Nhận biết có người sử dụng, nếu không có người tự động tắt máy sau một thời gian.

Hệ thống cần phát triển nhỏ gọn hơn.

Thiết kế ít tiêu tốn năng lượng, tối ưu lại khả năng dùng pin, để sử dụng pin một cách lâu dài hơn.

## **TÀI LIỆU THAM THẢO**

- [1] ESPRESSIF SYSTEMS (SHANGHAI) CO.,LTD, ESP8266EX Datasheet, 2023.
- [2] Aosong Electronics Co.,LTD, DHT22 Datasheet.
- [3] Maxim Intergrated Product, DS1307 64 x 8, Serial, I2C Real-Time Clock.
- [4] Hitachi Semiconductor, HD44780U (Dot Matrix Liquid Crystal Display Controller/Driver).
- [5] Texas Instruments, LM7805 Datasheet.