

# LVTN\_GIAMSATDIEUKHIENXEH OIDIENBANGCAN\_KHK (2).pdf

*by* Trương Quang Bảo Khanh

---

**Submission date:** 21-Feb-2025 12:06AM (UTC+0700)

**Submission ID:** 2593887935

**File name:**

46814\_Trương\_Quang\_Bảo\_Khanh\_LVTN\_GIAMSATDIEUKHIENXEHOIDIENBANGCAN\_KHK\_\_2\_\_25435\_967605745.pdf  
(5.79M)

**Word count:** 24302

**Character count:** 106199

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI TP. HỒ CHÍ MINH  
VIỆN CƠ KHÍ



LUẬN VĂN TỐT NGHIỆP

ỨNG DỤNG GIAO THÚC CAN  
TRONG ĐIỀU KHIỂN VÀ GIÁM SÁT XE HƠI ĐIỆN

Ngành: Kỹ thuật điều khiển & Tự động hóa

Chuyên ngành: Tự động hóa Công nghiệp

*Giảng viên hướng dẫn : TS. Khổng Hoài Hưng*

*Sinh viên thực hiện :*

Họ và tên	MSSV	Lớp
-----------	------	-----

- |                           |            |       |
|---------------------------|------------|-------|
| 1. Trương Quang Bảo Khanh | 2051050130 | TD20B |
| 2. Nguyễn Thái Hòa        | 2051050116 | TD20B |
| 3. Hoàng Minh Khôi        | 2051050134 | TD20B |

*TP. Hồ Chí Minh, ngày 22 tháng 1 năm 2025*

Viện: Cơ khí

Ngành: Kỹ thuật điều khiển và tự động hóa

### PHIẾU GIAO ĐỀ TÀI LUẬN VĂN TỐT NGHIỆP

(Phiếu này được dán ở trang đầu tiên của quyển báo cáo LVTN)

#### 1. Họ và tên sinh viên/ nhóm sinh viên được giao đề tài (số lượng: 3):

- (1) Trương Quang Bảo Khanh      MSSV: 2051050130      Lớp: TD20B  
(2) Nguyễn Thái Hòa      MSSV: 2051050116      Lớp: TD20B  
(3) Hoàng Minh Khôi      MSSV: 2051050134      Lớp: TD20B

Ngành : Kỹ thuật điều khiển và tự động hóa .....

Chuyên ngành : Tự động hóa công nghiệp .....

#### 2. Tên đề tài : Ứng dụng giao thức CAN trong điều khiển và giám sát xe hơi điện.

.....  
.....

#### 3. Các dữ liệu ban đầu :

- a) Thiết bị: 3 MCU STM32F1, ST-Link v2 debugger, ILI9341 TFT 2.8inch, Brushless DC with Hall sensor, DC motor 24V, BLDC Driver, air pressure sensor, temperature sensor, CAN Transceiver – CAN Controller.  
b) Phần mềm: Cube IDE, Cube MX, Easy EDA, Picture to Hex converter(web), figma, Logic Analyzer, STM32 ST-Link Unity, Arduino IDE, Visual Code.

#### 4. Các yêu cầu chủ yếu :

- Hiển thị, cài đặt và điều khiển toàn bộ hệ thống thông qua màn hình cảm ứng TFT ILI9341.  
• Hiển thị và điều khiển tốc độ động cơ BLDC.  
• Hiển thị - cài đặt - cảnh báo tình trạng sức căng của lốp xe.

- Hiển thị - cài đặt - cảnh báo nhiệt độ động cơ xe hơi.

#### 5. Kết quả tối thiểu phải có:

- 1) Giao thức CAN được sử dụng để truyền dữ liệu giữa các MCU(STM32F1).
- 2) Động cơ BLDC có thể tăng/giảm tốc, quay thuận/nghịch và bấm sát theo tốc độ cài đặt.
- 3) Các thông số phản hồi như tốc độ động cơ, nhiệt độ ngoài trời, áp suất lốp xe phải được kiểm tra tính chính xác.
- 4) Đọc hiểu toàn bộ source code library cũng như driver code cho các cảm biến và module ngoại vi sử dụng trong đồ án.
- 5) Hoàn thành tối thiểu ¾ các yêu cầu ở mục “4. Các yêu cầu chủ yếu”.

Ngày giao đề tài: 27/11/2024      Ngày nộp báo cáo: 17/02/2025

TP. HCM, ngày 27 tháng 11 năm 2024

**QUẢN LÝ CTĐT**

(Ký và ghi rõ họ tên)

**Giảng viên hướng dẫn**

(Ký và ghi rõ họ tên)



Lê Anh Uyên Vũ

Khổng Hoài Hưng

Viện: Cơ khí

Ngành: Kỹ thuật điều khiển và tự động hóa

## BẢNG NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

### KHÓA LUẬN TỐT NGHIỆP

#### 1. Họ và tên sinh viên/ nhóm sinh viên được giao đề tài (số lượng: 3):

- (1) Trương Quang Bảo Khanh      MSSV: 2051050130      Lớp: TD20B  
(2) Nguyễn Thái Hòa      MSSV: 2051050116      Lớp: TD20B  
(3) Hoàng Minh Khôi      MSSV: 2051050134      Lớp: TD20B

Ngành : Kỹ thuật điều khiển và tự động hóa .....

Chuyên ngành : Tự động hóa công nghiệp .....

#### 2. Tên đề tài : Ứng dụng giao thức CAN trong điều khiển và giám sát xe hơi điện.

#### 3. Tổng quát về KLTN:

Số trang: 104 ..... Số chương: 4 .....

Số bảng số liệu: 0 ..... Số hình vẽ: 4 .....

Số tài liệu tham khảo: 4 ..... Phần mềm tính toán: 0 .....

Số bản vẽ kèm theo: 0 ..... Hình thức bản vẽ: 0 .....

Hiện vật (sản phẩm) kèm theo: 1 .....

#### 4. Nhận xét:

##### a. Về tinh thần, thái độ làm việc của sinh viên:

Tinh thần làm việc tốt, thái độ báo cáo nghiêm túc .....

##### b. Những kết quả đạt được của KLTN:

- Về báo cáo: Nhóm trình bày chi tiết và sâu từng phần, thể hiện tốt độ am hiểu về đồ án

- Về mô hình: Chưa có tính thẩm mỹ cao, cần chỉnh chu về phần cơ khí đẹp hơn, tuy nhiên đảm bảo những cơ cấu và thành phần cần thể hiện trong đồ án.

c. **Những hạn chế của KLTN:**

- Thông số giám sát áp suất lốp xe chưa thể gửi lên màn hình TFT.
- Mô hình chưa được làm mạch in PCB.

**5. Đè nghị:**

Được bảo vệ (hoặc nộp KLTN để chấm)  Không được bảo vệ

**Điểm thi (nếu có): 9.5** .....

*TP. HCM, ngày 18 tháng 02 năm 2025*

**Giảng viên hướng dẫn**

(Ký và ghi rõ họ tên)



**Không Hoài Hưng**

Viện: Cơ khí

Ngành: Kỹ thuật điều khiển và tự động hóa

## BẢNG NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

### KHÓA LUẬN TỐT NGHIỆP

#### 1. Họ và tên sinh viên/ nhóm sinh viên được giao đề tài (số lượng: 3):

- (1) Trương Quang Bảo Khanh      MSSV: 2051050130      Lớp: TD20B  
(2) Nguyễn Thái Hòa      MSSV: 2051050116      Lớp: TD20B  
(3) Hoàng Minh Khôi      MSSV: 2051050134      Lớp: TD20B

Ngành : Kỹ thuật điều khiển và tự động hóa .....

Chuyên ngành : Tự động hóa công nghiệp.....

#### 2. Tên đề tài :

Ứng dụng giao thức CAN trong điều khiển và giám sát xe hơi điện.

#### 3. Nhận xét:

##### a. Những kết quả đạt được của KLTN:

.....  
.....

##### b. Những hạn chế của KLTN:

.....  
.....

#### 4. Đề nghị:

Được bảo vệ       Bổ sung thêm để bảo vệ       Không được bảo vệ

#### 5. Các câu hỏi sinh viên cần trả lời trước Hội đồng:

(1) Làm thế nào để tăng độ chính xác của mô hình? .....

(2) Hiệu chỉnh độ rõ của mô hình bằng cách nào? .....

(3) Ứng dụng của mô hình trong thực tế? .....

.....  
**6. Điểm thi:** .....

TP. HCM, ngày tháng 02 năm 2025

**Giảng viên phản biện**

(Ký và ghi rõ họ tên)

## DANH SÁCH CÁC THÀNH VIÊN THAM GIA

1. Giảng viên hướng dẫn: TS. Không Hoài Hưng, Trường Đại học Giao Thông Vận Tải TP. HCM.
2. Thành viên: Trương Quang Bảo Khanh, viện Công nghệ thông tin và Điện, điện tử, Trường Đại học Giao Thông Vận Tải TP. HCM.
3. Thành viên: Nguyễn Thái Hòa, viện Công nghệ thông tin và Điện, điện tử, Trường Đại học Giao Thông Vận Tải TP. HCM.
4. Thành viên: Hoàng Minh Khôi, viện Công nghệ thông tin và Điện, điện tử, Trường Đại học Giao Thông Vận Tải TP. HCM.

### LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn và tri ân đến thầy Không Hoài Hưng, giảng viên đã tận tâm trong việc hỗ trợ **chúng em hoàn thành khóa luận tốt nghiệp**, từ những giai đoạn chọn đề tài cho đến việc cố vấn cũng như giúp đỡ chúng em giải đáp những thắc mắc trong quá trình hoàn thiện Luận văn tốt nghiệp.

Bên cạnh đó, nhóm chúng em vẫn không thể quên được sự giúp đỡ tận tình từ cô Lê Anh Uyên Vũ, cô là người đã hỗ trợ chúng em trong các thủ tục hoàn thành khóa luận từ giấy tờ cho đến những thông báo chi tiết giúp chúng em thuận lợi trong việc chuẩn bị các khoảng thời gian để hoàn thành khóa luận tốt nghiệp.

Cuối cùng, lời cảm ơn và tri ân đến các thành viên trong nhóm đã không ngừng nỗ lực để đạt được mục tiêu cuối cùng trên giảng đường đại học. Nhờ có sự đoàn kết và quan tâm lẫn nhau, nhóm mới có được sự hoàn thiện và thành quả như bây giờ.

Điều cuối cùng, nhóm chúng em xin gửi lời cảm ơn đến toàn bộ thầy, cô và các bạn sinh viên lớp TD20B đã giúp đỡ và tạo điều kiện để chúng em được học tập và phát triển trong suốt những năm đại học tại Trường Đại học Giao Thông Vận Tải Thành Phố Hồ Chí Minh.

Chúng em xin chân thành cảm ơn!

## TÓM LUỢT KẾT QUẢ NGHIÊN CỨU

### 1. Thông tin chung:

- Tên đề tài: Ứng dụng giao thức CAN trong điều khiển và giám sát xe hơi điện.
- Giảng viên hướng dẫn: TS. Khổng Hoài Hưng.
- Thành viên:
  - + Trương Quang Bảo Khanh – 2051050130.
  - + Nguyễn Thái Hòa – 2051050116.
  - + Hoàng Minh Khôi – 2051050134.
- Tổ chức chủ trì: Trường đại học Giao Thông Vận Tải TP.HCM.
- Thời gian thực hiện: Từ 27/11/2024 đến 17/02/2025.

### 2. Mục tiêu:

- Cải thiện kỹ năng đọc Datasheet/Hardware User Manual, lập trình ngôn ngữ C/C++, cấu trúc dữ liệu và giải thuật để đọc hiểu toàn bộ driver code của giao thức SPI trên vi điều khiển STM32.
- Hiểu về mạng CAN, thực hiện giao tiếp CAN Bus trên nhiều loại vi điều khiển.
- Hiểu về nguyên lý hoạt động của động cơ không chổi than BLDC.
- Sử dụng ít nhất 2 cảm biến công nghiệp cho đề tài.
- Thành thạo một công cụ thiết kế giao diện UX/UI
- Thu thập phản ứng và lựa chọn các cơ cấu chấp hành phù hợp cho đề tài.

### 3. Tính mới và sáng tạo:

#### Truyền dữ liệu giữa các thiết bị qua giao thức CAN

- Sử dụng CAN bus giúp truyền thông nhanh chóng, ổn định giữa các node, giảm bớt hệ thống dây điện và tăng khả năng bảo trì hệ thống.
- Dữ liệu từ động cơ BLDC, màn hình TFT và các cảm biến được trao đổi theo thời gian thực, giảm thiểu khi sử dụng giao thức CAN từ đó tối ưu hóa hiệu suất của xe.

#### Điều khiển động cơ BLDC

- BLDC là loại động cơ được sử dụng ở hầu hết các mô hình xe điện do tính bền bỉ cơ học và hiệu suất cao, chính vì vậy am hiểu về loại động cơ này là một lợi thế lớn trong ngành công nghiệp xe điện.

#### Giám sát áp suất lốp và nhiệt độ động cơ

- Cảm biến áp suất không khí và nhiệt độ giúp theo dõi tình trạng lốp và nhiệt độ động cơ theo thời gian thực, nâng cao độ an toàn.
- Hệ thống có thể cảnh báo sớm các nguy cơ, giúp bảo trì dự đoán trước khi xảy ra hỏng hóc.

#### Màn hình TFT hiển thị thông tin

- Cung cấp giao diện trực quan để giám sát các thông số quan trọng của xe.
- Có thể tích hợp cảm ứng để điều khiển một số chức năng trực tiếp từ màn hình.

#### 4. Kết quả nghiên cứu:

- Giao thức CAN **hoạt động** ổn định giữa các MCU khác nhau, dữ liệu không bị nhiễu khi đặt các CAN Bus sát với động cơ.
- Điều khiển tăng/giảm tốc và quay thuận nghịch động cơ BLDC đồng thời phản hồi tốc độ thực tế liên tục thông qua màn hình cảm ứng TFT ILI9341 2.8 inch.
- Giám sát/điều khiển/cảnh báo nhiệt độ động cơ thông qua màn hình TFT, cơ cấu quạt tản nhiệt hoạt động ổn định.
- Sử dụng được toàn bộ chức năng của màn hình TFT bao gồm xuất dữ liệu hình ảnh thông qua SD Card, xuất dữ liệu hình ảnh được lưu qua mã hex trong vùng RAM của ESP32, cảm ứng chạm phản hồi vị trí chính xác được chạm.
- Hiểu sâu về vi điều khiển STM32, cách lập trình thanh ghi cho chip từ việc đọc Hardware User Manual, cách debug cho STM32 bằng ST-Link(Open OCD).
- Sử dụng thành thạo phần mềm thiết kế UX/UI Figma.

**MỤC LỤC**

<b>LỜI MỞ ĐẦU .....</b>	<b>19</b>
<b>CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN .....</b>	<b>21</b>
1.1.Lý do chọn đề tài .....	21
1.2.Mục tiêu đồ án .....	22
1.3.Phương pháp nghiên cứu.....	22
<b>CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.....</b>	<b>24</b>
2.1.Ứng dụng mạng CAN trong xe hơi điện .....	24
2.1.1. Xe hơi điện là gì? .....	24
2.1.2. Các loại xe hơi điện .....	25
2.1.3. Cấu tạo xe hơi điện .....	26
2.1.4. Giao thức CAN là gì? .....	27
2.1.5. Ưu điểm giao thức CAN .....	28
2.1.6. Ứng dụng giao thức CAN trong xe hơi điện.....	28
2.1.7. Ý tưởng áp dụng vào đồ án.....	29
2.2.Động cơ BLDC và BLDC driver .....	29
2.2.1. Động cơ BLDC .....	29
2.2.2. Mạch lái động cơ BLDC .....	32
2.3.Vi điều khiển STM 32 .....	35
2.3.1. Giới thiệu STM32 .....	35
2.3.2. Các bước hoạt động của STM32 .....	37
2.3.3. Tính năng của vi điều khiển STM32 .....	38
2.3.4. Giới thiệu dòng STM32F103C8T6.....	40
2.4.Vi điều khiển ESP 32 .....	44
2.4.1. Giới thiệu ESP 32 .....	44
2.4.2. Cấu hình ESP 32 .....	45
2.4.3. Sơ đồ chân của DOIT ESP 32 DEVKIT VQ BOARD .....	47
2.4.4. Ưu điểm của ESP 32 so với các thiết bị tương tự .....	47
2.5.Cảm biến nhiệt độ PT100 - MAX31865 .....	48

2.5.1. PT100 - MAX31865 .....	48
2.5.2. Quá trình kết nối PT100 với MAX31865 .....	51
2.5.3. Công thức chuyển đổi công thức Callendar – Van Dusen thành ngôn ngữ lập trình .....	52
2.6. Màn hình cảm ứng TFT ILI 9341 .....	53
2.6.1. Các tính năng chính và thông số kỹ thuật: .....	54
2.6.2. Cấu hình chân cảm (Ví dụ): .....	55
2.7. CAN Controller MCP2515 và Transceiver 2551 .....	56
2.7.1. CAN Controller MCP2515 .....	56
2.7.2. Transceiver 2551 .....	58
2.8. Các linh kiện cần thiết cho cơ cấu chấp hành .....	59
2.8.1. Quạt thông gió .....	59
2.8.2. Module Relay 5V .....	60
2.8.3. Mạch ổn áp 5V và 3.3V .....	61
2.8.4. Nguồn ток 220V AC – 24V DC .....	63
2.8.5. Động cơ máy bơm hơi 12V .....	65
2.8.6. Dây Jumper .....	67
2.8.7. Level Shifter .....	67
2.9. Nền tảng thiết kế giao diện Figma .....	68
2.10. Arduino IDE .....	69
2.10.1.Cài đặt bổ sung ESP 32 board cho Arduino IDE .....	71
2.11. Cube IDE .....	74
2.11.1.Tổng quan về STM32CubeIDE .....	74
2.11.2.Cách tạo Project trên STM32CubeIDE .....	76
2.12. Easy EDA .....	78
2.12.1.Tổng quan về phần mềm Easy EDA .....	78
2.12.2.Cách tạo một project mới trên phần mềm Easy EDA .....	79
2.13. Visual Code .....	80
<b>CHƯƠNG 3. LẬP TRÌNH – THIẾT KẾ VÀ KẾT QUẢ ĐẠT ĐƯỢC.....</b>	<b>82</b>
3.1. Sơ đồ khối – chức năng các khối .....	82

3.2. Lưu đồ thuật toán .....	83
3.3. Cấu hình và lập trình 3 STM32 trên Cube IDE .....	84
3.3.1. Cấu hình và lập trình STM32 điều khiển động cơ BLDC .....	84
3.3.2. Cấu hình và lập trình STM32 Temperature .....	89
3.4. Cấu hình và lập trình ESP32 .....	91
3.4.1. Một số thư viện cần thiết .....	91
3.4.2. Các hàm quan trọng được viết bởi người dùng .....	91
3.5. Khung truyền data CAN giữa các NODE .....	94
3.5.1. Khung truyền CAN giữa ESP32 và STM32 BLDC .....	94
3.5.2. Khung truyền CAN giữa ESP32 và STM32 Temperature .....	95
3.6. Sơ đồ nguyên lý toàn bộ hệ thống .....	97
3.7. Mô hình hoàn thiện sau khi kết nối .....	98
3.7.1. Khối nguồn cấp .....	98
3.7.2. Khối màn hình cảm ứng giám sát và điều khiển .....	99
3.7.3. Can bus và các khối thu thập điều khiển (STM32) .....	100
3.7.4. Tổng quan mô hình .....	103
3.8. Giám sát – Cài đặt – Điều khiển hệ thống xe hơi điện .....	103
3.8.1. Màn hình chính .....	103
3.8.2. Màn hình điều khiển/giám sát động cơ BLDC .....	104
3.8.3. Màn hình điều khiển/giám sát nhiệt độ động cơ .....	106
3.8.4. Kiểm tra tính chính xác của các thông số .....	108
<b>CHƯƠNG 4. KẾT LUẬN.....</b>	<b>111</b>
4.1. Kết quả đề tài .....	111
4.2. Kết luận và hướng phát triển .....	111
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>113</b>

**DANH MỤC ẢNH**

Hình 1.1 Xu thế ngày càng hiện đại hóa .....	21
Hình 1.2 Giao diện điều khiển xe hơi điện .....	22
Hình 2.1 Xe hơi điện.....	24
Hình 2.2 Các loại năng lượng ứng với mỗi loại xe điện.....	26
Hình 2.3 Cấu tạo xe hơi điện .....	26
Hình 2.4 Giao thức CAN kết nối các bộ điều khiển .....	28
<b>Hình 2.5</b> Cấu tạo của động cơ BLDC .....	30
<b>Hình 2.6</b> Hình ưu điểm của động cơ BLDC .....	30
Hình 2.7 Hình nhược điểm của động cơ BLDC .....	30
Hình 2.8 Hình hover board BLDC 24V .....	31
Hình 2.9 Hình cấu tạo hoverboard.....	32
Hình 2.10 Hình mặt trước mạch lái động cơ BLDC 12-36VDC.....	33
Hình 2.11 Hình mặt sau 6 mosfet của BLDC driver .....	33
Hình 2.12 Hình đấu nối BLDC driver trên mô hình.....	35
Hình 2.13 Vi điều khiển STM32 .....	35
Hình 2.14 Mô phỏng vi điều khiển STM32F103C8T6 .....	40
Hình 2.15 Sơ đồ chân của STM32F103C8T6 .....	42
Hình 2.16 Bo mạch phát triển dựa trên vi điều khiển ESP32.....	44
Hình 2.17 Sơ đồ chân của ESP 32 .....	47
Hình 2.18 Tương quan giữa nhiệt độ và điện trở .....	50
Hình 2.19 MAX31865 Adafruit.....	51
Hình 2.20 PT100 và MAX31865.....	52
Hình 2.21 Minh họa công thức Callenda – Van Dusen .....	53
Hình 2.22 Màn hình TFT ILI 9341 .....	54
Hình 2.23 Các chân đấu nối của TFT ILI 9341 .....	54
Hình 2.24 Các chân đấu nối của TFT ILI 9341 .....	55
Hình 2.25 CAN Controller MCP2515 .....	56
Hình 2.26 Cấu hình chân CAN Controller MCP2515 .....	57

Hình 2.27 CAN Transceiver 2551 .....	58
Hình 2.28 Quạt thông gió .....	60
Hình 2.29 Module relay 5V - 2 kênh .....	61
Hình 2.30 Mạch giảm áp XL4015 (5A).....	62
Hình 2.31 Mạch nguồn giảm áp DC mini 3A.....	63
Hình 2.32 Nguồn tổ ong 220VAC-24VDC .....	64
Hình 2.33 Động cơ 775 12VDC 15000rpm.....	65
Hình 2.34 Dây Jumper .....	67
Hình 2.35 Level Shifter .....	68
Hình 2.36 Nền tảng thiết kế giao diện người dùng Figma .....	68
Hình 2.37 Phần mềm Arduino IDE .....	69
<b>Hình 2.38 Cài đặt bổ sung ESP32 board cho arduino IDE</b> .....	<b>71</b>
<b>Hình 2.39 Cài đặt bổ sung ESP32 board cho arduino IDE</b> .....	<b>71</b>
<b>Hình 2.40</b> .....	<b>72</b>
<b>Hình 2.41 Cài đặt bổ sung ESP32 board cho arduino IDE</b> .....	<b>72</b>
<b>Hình 2.42 Cài đặt bổ sung ESP32 board cho arduino IDE</b> .....	<b>73</b>
<b>Hình 2.43 Cài đặt bổ sung ESP32 board cho arduino IDE</b> .....	<b>73</b>
<b>Hình 2.44 Cài đặt bổ sung ESP32 board cho arduino IDE</b> .....	<b>73</b>
<b>Hình 2.45 Cài đặt bổ sung ESP32 board cho arduino IDE</b> .....	<b>74</b>
<b>Hình 2.46 Giao diện làm việc trên STM32CubeIDE</b> .....	<b>76</b>
<b>Hình 2.47 Tạo project trên CubeIDE</b> .....	<b>77</b>
<b>Hình 2.48 Tạo project trên CubeIDE</b> .....	<b>78</b>
<b>Hình 2.49 Giao diện làm việc trên phần mềm Easy EDA</b> .....	<b>79</b>
<b>Hình 2.50 Hướng dẫn tạo project mới trên Easy EDA</b> .....	<b>79</b>
<b>Hình 2.51 Hướng dẫn tạo project mới trên Easy EDA</b> .....	<b>80</b>
<b>Hình 2.52 Nền tảng lập trình Visual Studio Code</b> .....	<b>81</b>
<b>Hình 3.1 Sơ đồ khối</b> .....	<b>82</b>
<b>Hình 3.2 Lưu đồ thuật toán cho khối kiểm tra nhiệt độ</b> .....	<b>83</b>
<b>Hình 3.3 Lưu đồ giải thuật cho khối điều khiển động cơ</b> .....	<b>84</b>

Hình 3.4 Cấu hình timer1 .....	85
Hình 3.5 Mô tả hàm sử dụng timer1 băm xung PWM. ....	86
Hình 3.6 Cấu hình timer2 .....	86
Hình 3.7 Thuật toán tính số vòng quay trên phút. ....	87
Hình 3.8 Cấu hình CAN node .....	88
Hình 3.9 Kích hoạt interrupt cho CAN receive buffer .....	88
Hình 3.10 Ngắt FIFO1 của module CAN .....	89
Hình 3.11 Cấu hình CAN filter cho BLDC node. ....	89
Hình 3.12 Tx frame của BLDC can node.....	89
Hình 3.13 Cấu hình SPI master .....	90
Hình 3.14 Add thư viện LCD vào Project .....	90
Hình 3.15 Cấu hình Tx frame cho Temperature node. ....	91
Hình 3.16 Prototype một số hàm sử dụng trong runtime của ESP32 .....	92
Hình 3.17 Code main .....	93
Hình 3.18 ESP32 setup Tx frame cho BLDC node .....	94
Hình 3.19 Thuật toán xử lý dữ liệu điều khiển BLDC .....	95
Hình 3.20 STM32 BLDC setup data trước khi truyền cho ESP32.....	95
Hình 3.21 ESP32 setup Tx frame cho Temp CAN node .....	96
Hình 3.22 STM32 temperature xử lý dữ liệu nhận được từ ESP32 .....	96
Hình 3.23 STM32 temperature tách dữ liệu nhiệt độ trước khi truyền.....	97
Hình 3.24 Sơ đồ nguyên lý .....	97
Hình 3.25 Nguồn tổ ong cho đề tài .....	98
Hình 3.26 Module hạ áp đầu ra 5V cấp cho ESP32 và STM32 .....	98
Hình 3.27 Nguồn 3.7V cấp cho màn hình TFT ILI3941 .....	99
Hình 3.28 ESP32 – TFT và SD Card – MCP2551 .....	99
Hình 3.29 CAN bus trên hệ thống .....	100
Hình 3.30 Khối thu thập nhiệt độ và điều khiển quạt .....	101
Hình 3.31 Khối điều khiển động cơ BLDC .....	101
Hình 3.32 Khối thu thập áp suất lốp xe STM32 PSI. ....	102

Hình 3.33	Tổng quan mô hình đề tài .....	103
Hình 3.34	Màn hình giao diện chính .....	104
Hình 3.35	Màn hình điều khiển động cơ BLDC-gear 2 - đi lùi.....	105
Hình 3.36	Màn hình điều khiển động cơ BLDC-gear 10 - đi lùi.....	106
Hình 3.37	Màn hình điều khiển động cơ BLDC – gear 8 - đi tiến .....	106
Hình 3.38	Màn hình giám sát nhiệt độ - auto mode - động cơ chưa nóng .....	107
Hình 3.39	Màn hình giám sát nhiệt độ - Auto mode - động cơ nóng.....	107
Hình 3.40	Màn hình giám sát nhiệt độ - manual mode – fan off.....	108
Hình 3.41	Màn hình giám sát nhiệt độ - manual mode – fan on .....	108
Hình 3.42	Tốc độ động cơ đo được từ cảm biến Hall .....	109
Hình 3.43	Tốc độ động cơ đo được từ thiết bị công nghiệp .....	109
Hình 3.44	Kiểm chứng nhiệt độ giữa cảm biến và thiết bị công nghiệp .....	110

## LỜI MỞ ĐẦU

Trong bối cảnh xe điện nhanh chóng nổi lên như một trụ cột của giao thông bền vững, ngành công nghiệp ô tô đang trải qua một bước chuyển mình lớn về thiết kế và công nghệ. Khi các hệ thống của xe điện ngày càng trở nên tinh vi, nhu cầu về một hệ thống truyền thông giữa các bộ phận chính xác, đáng tin cậy và hiệu quả trở nên cần thiết. Tại trung tâm của hệ thống này chính là giao thức Controller Area Network (CAN) – một tiêu chuẩn truyền thông mạnh mẽ, tốc độ cao và có khả năng chống lỗi, cho phép trao đổi dữ liệu theo thời gian thực giữa các đơn vị điều khiển điện tử trên toàn bộ xe.

Trong đề tài nghiên cứu của nhóm chúng em, mục tiêu chính là ứng dụng giao thức CAN trong ô tô điện để tạo điều kiện cho sự kết nối giữa các thành phần quan trọng như hệ thống điều khiển động cơ BLDC, quản lý pin, giám sát cảm biến (bao gồm cảm biến nhiệt độ và áp suất không khí) cũng như giao diện người dùng qua màn hình TFT. Qua việc nghiên cứu sâu về các chi tiết kỹ thuật của mạng CAN, nhóm chúng em hướng đến việc đánh giá hiệu năng của hệ thống trong môi trường thực tế, xác định các thách thức tiềm ẩn và đề xuất các chiến lược tối ưu dữ liệu cũng như khả năng phản hồi của hệ thống.

Mục tiêu của nhóm chúng em không chỉ là nắm vững kiến thức về truyền thông dựa trên CAN trong các hệ thống xe điện, mà còn giúp chúng em bắt kịp xu thế phát triển hiện đại trong ngành ô tô điện. Thông qua đề tài này, chúng em mong muốn đóng góp những hiểu biết có giá trị, từ đó hỗ trợ việc thiết kế ra các xe điện ngày càng đáng tin cậy, hiệu quả và an toàn hơn trong bối cảnh công nghệ không ngừng tiến hóa.

### Những đóng góp chính của luận văn:

- Đề xuất một giải pháp hiệu quả để điều khiển và giám sát xe điện.
- Nâng cao trải nghiệm người dùng thông qua giao diện người dùng trực quan và thân thiện.
- Đóng góp vào việc phát triển công nghệ xe điện tại Việt Nam.

### Luận văn này được cấu trúc thành các chương như sau:

**Chương 1:** Giới thiệu tổng quan;

**Chương 2:** Cơ sở lý thuyết

**Chương 3:** Lập trình - thiết kế và kết quả đạt được

1  
**Chương 4: Kết luận**

## CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN

### 1.1. Lý do chọn đề tài

Ngành công nghiệp ô tô điện ngày càng phát triển và đòi hỏi các giải pháp điều khiển thông minh, thực tế cho thấy các hệ thống truyền thông và giám sát trên xe đang trở thành yếu tố then chốt nhằm đảm bảo hiệu suất hoạt động và an toàn cho người sử dụng. Các sự cố về nhiệt độ động cơ quá cao hay áp suất lốp không đạt chuẩn không chỉ ảnh hưởng nghiêm trọng đến hiệu suất của xe mà còn tiềm ẩn nguy cơ hỏng hóc và tai nạn. Đồng thời, nhu cầu tối giản hệ thống dây dẫn và tăng cường khả năng giao tiếp giữa các bộ phận điều khiển cũng được đặt lên hàng đầu trong quá trình phát triển công nghệ xe điện.

Trên thực tế, nhiều nhà sản xuất xe điện đã và đang đầu tư mạnh mẽ vào các giải pháp tích hợp, trong đó việc áp dụng giao thức CAN bus vào hệ thống điều khiển giúp trao đổi dữ liệu nhanh chóng, ổn định, đồng thời giảm thiểu lỗi truyền thông do nhiễu điện từ. Bên cạnh đó, công nghệ điều khiển động cơ không chổi than (BLDC) với hiệu suất cao và tuổi thọ bền bỉ đã trở thành lựa chọn ưu việt cho các dòng xe điện hiện nay.



Hình 1.1 Xu thế ngày càng hiện đại hóa

Với bối cảnh trên, với mục tiêu nắm bắt xu thế, ứng dụng những kiến thức đã được học cộng hưởng với sự cho phép của giảng viên hướng dẫn thầy Khổng Hoài Hưng, nhóm chúng em quyết định lựa chọn đề tài “Ứng dụng giao thức CAN trong điều khiển xe điện” nhằm giải quyết các vấn đề thực tiễn như giám sát nhiệt độ động cơ, theo dõi áp suất lốp và điều khiển động cơ BLDC thông qua một hệ thống truyền thông tích hợp, đa năng. Hệ thống của nhóm không chỉ sử dụng giao diện người dùng trực quan với màn hình TFT mà còn kết

hợp các cảm biến công nghiệp hiện đại để thu thập và phân tích dữ liệu theo thời gian thực, góp phần tối ưu hóa hiệu suất vận hành và nâng cao độ an toàn cho xe.



Hình 1.2 Giao diện điều khiển xe hơi điện

Thông qua đề tài này, nhóm chúng em mong muốn không chỉ ứng dụng thành thạo các kiến thức về lập trình C/C++ trên vi điều khiển STM32, hiểu sâu hơn về cơ chế hoạt động của mạng CAN và động cơ BLDC, mà còn giúp chúng em nắm vững và bắt kịp xu thế phát triển của ngành công nghiệp xe điện hiện đại.

### 1.2. Mục tiêu đồ án

- Thiết kế và triển khai một giao diện người dùng đồ họa trực quan trên màn hình cảm ứng TFT ILI9341.
- Xây dựng hệ thống truyền thông dựa trên giao thức CAN để kết nối các thành phần điện tử trên xe.
- Tích hợp các chức năng giám sát và cảnh báo các thông số quan trọng của xe, như áp suất lốp và nhiệt độ.
- Đánh giá hiệu quả của hệ thống thông qua các thí nghiệm thực tế.

### 1.3. Phương pháp nghiên cứu

- Đọc tài liệu Datasheet để viết thư viện cho các cảm biến.
- Đọc tài liệu Hardware User Manual để hiểu cách lập trình thanh ghi cho vi điều khiển STM32.
- Sử dụng kiến thức về ngôn ngữ lập trình C/C++ để lập trình cho toàn bộ hệ thống.
- Tham khảo các nguồn tài liệu sách, báo, video nghiên cứu đã được công bố.

- Sử dụng một số thư viện có sẵn dùng cho các thành phần phức tạp như màn hình cảm ứng TFT, bộ thư viện của ESP32 và thư viện HAL của STM32.

## CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

### 2.1. Ứng dụng mạng CAN trong xe hơi điện

#### 2.1.1. Xe hơi điện là gì?

Xe hơi điện là phương tiện giao thông sử dụng động cơ điện để chuyển đổi năng lượng điện thành năng lượng cơ học, tạo ra lực kéo, thay thế hoàn toàn hoặc một phần động cơ đốt trong, động cơ xe được cấp năng lượng từ bộ pin được lắp đặt trên xe và nó sẽ được sạc ở những trạm sạc công cộng hoặc mạng điện tại nhà. Xe điện xuất hiện từ cuối thế kỷ 19 nhưng nhanh chóng bị xe chạy bằng xăng dầu soán ngôi do hạn chế về pin, hạ tầng và chi phí. Nhưng cho đến hiện tại những đột phá trong công nghệ pin lithium-ion, cùng với sự gia tăng quan ngại về biến đổi khí hậu và ô nhiễm môi trường, xe điện đã trở lại và ngày càng phổ biến.



Hình 2.1 Xe hơi điện

**Nguyên lý hoạt động:** Về nguyên lý xe ô tô điện, xe sẽ nhận năng lượng là dòng điện một chiều (DC) từ pin được lắp đặt trên xe. Dòng điện một chiều được chuyển đổi thành dòng điện xoay chiều nhờ bộ biến tần. Khi đó dòng điện xoay chiều sẽ làm cho động cơ và bánh xe hoạt động. Cụ thể đối với một chiếc ô tô điện, khi được vận hành sẽ có nguyên lý như sau:

- Bộ điều khiển lấy và điều chỉnh năng lượng điện từ pin và biến tần.
- Với bộ điều khiển được thiết lập, biến tần sau đó sẽ gửi một lượng năng lượng điện nhất định đến động cơ (theo độ sâu của áp lực lên bàn đạp).
- Động cơ điện chuyển đổi năng lượng điện thành năng lượng cơ học (quay). Vòng quay của roto động cơ làm quay bộ truyền động để các bánh xe quay và sau đó ô tô chuyển động.

- 3  
– Vòng quay của roto động cơ làm quay bộ truyền động để các bánh xe quay và sau đó ô tô chuyên động.

**Ưu điểm:**

- 1  
– **Thân thiện với môi trường:** Không **thải** khí thải **trực tiếp**, giảm hiệu ứng nhà kính.  
– Hiệu suất cao, tiết kiệm năng lượng: Động cơ điện có hiệu suất chuyển đổi năng lượng cao hơn động cơ đốt trong.
- Vận hành êm ái: Không gây tiếng ồn và rung lắc.
  - Vận hành và bảo trì đơn giản đi kèm với chi phí thấp.

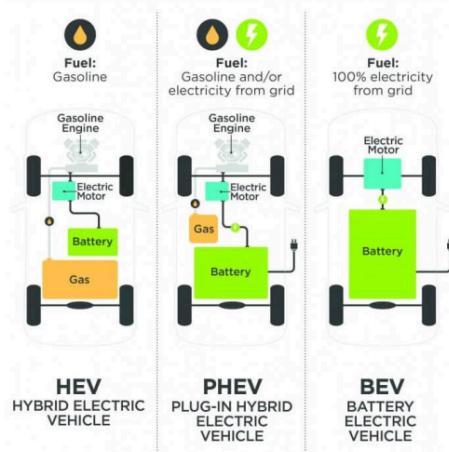
**Hạn chế:**

- Quãng đường di chuyển còn hạn chế.
- Thời gian sạc pin lâu.
- Chi phí ban đầu cao.

**2.1.2. Các loại xe hơi điện**

### Types of Electric Vehicles

If you're looking to purchase an electric vehicle, use this cheat sheet to help determine the various options. Drivers can choose between three types of electric vehicles (EVs). EVs are classed by the amount of electricity that is used as their energy source.

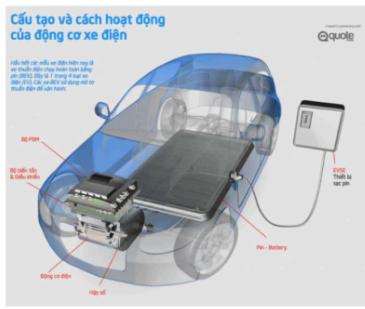


Hình 2.2 Các loại năng lượng ứng với mỗi loại xe điện

- Xe điện thuần túy (BEV): Hoàn toàn sử dụng năng lượng điện.
- Xe hybrid (HEV): Kết hợp động cơ điện và động cơ đốt trong.
- Xe plug-in hybrid (PHEV): Có thể sạc điện nhưng vẫn có động cơ xăng.

### 2.1.3. Cấu tạo xe hơi điện

Cấu tạo xe oto điện bao gồm những thành phần đơn giản hơn so với các loại xe vận hành bằng xăng. Gồm các thành phần chính như sau:



Hình 2.3 Cấu tạo xe hơi điện

– **Ác quy (Pin)**: Là trái tim của xe điện, lưu trữ năng lượng điện dưới dạng hóa học. Khi cần sử dụng, năng lượng hóa học này được chuyển đổi thành dòng điện một chiều (DC) để cung cấp cho các hệ thống khác trong xe.

– **Bộ chuyển đổi DC/DC**: Đây là một thiết bị điện tử đóng vai trò như một "bộ điều áp điện áp". Nó nhận điện áp cao từ pin và chuyển đổi thành điện áp thấp (12V) để cung cấp cho các thiết bị điện tử khác trên xe như đèn, radio, hệ thống điều hòa.

– **Động cơ điện**: Là "trái tim" thứ hai của xe điện, trực tiếp chuyển đổi năng lượng điện thành năng lượng cơ học để làm quay bánh xe. Động cơ điện hoạt động dựa trên tương tác giữa từ trường và dòng điện. Khi dòng điện chạy qua các cuộn dây trong động cơ, nó tạo ra từ trường tương tác với từ trường của nam châm vĩnh cửu, sinh ra lực quay. Đặc biệt, động cơ điện còn có khả năng hoạt động như một máy phát điện khi phanh, chuyển đổi năng lượng động thành điện năng để sạc lại pin.

– **Bộ biến tần:** Bộ phận này đóng vai trò như một "cầu nối" giữa pin và động cơ. Nó biến đổi dòng điện một chiều từ pin thành dòng điện xoay chiều ba pha để cung cấp cho động cơ. Bằng cách điều chỉnh tần số và biên độ của dòng điện xoay chiều, bộ biến tần có thể điều khiển tốc độ và mô-men xoắn của động cơ.

– **Cổng sạc và bộ sạc trên bo mạch:** Cổng sạc là nơi kết nối xe điện với nguồn điện bên ngoài. Bộ sạc trên bo mạch sẽ chuyển đổi điện áp xoay chiều từ nguồn điện bên ngoài thành điện áp một chiều để sạc cho pin.

– **Bộ điều khiển:** Là "bộ não" của xe điện, chịu trách nhiệm điều khiển và quản lý toàn bộ hệ thống. Bộ điều khiển nhận tín hiệu từ các cảm biến và các thiết bị đầu vào khác, sau đó đưa ra các lệnh điều khiển cho các thành phần khác để đảm bảo xe hoạt động ổn định nhất.

– **Ác quy phụ:** Dùng để cung cấp năng lượng cho các thiết bị điện tử trên xe khi xe tắt máy hoặc khi pin chính bị xả quá mức.

– **Hệ thống làm mát:** Giúp duy trì nhiệt độ hoạt động ổn định cho các thành phần quan trọng như pin, động cơ và các thiết bị điện tử, đảm bảo tuổi thọ và hiệu suất của chúng.

– **Truyền động:** Hệ thống truyền động sẽ truyền mô-men xoắn từ động cơ đến bánh xe, giúp xe chuyển động.

#### 2.1.4. Giao thức CAN là gì?

**CAN (Controller Area Network):** Là một giao thức truyền thông công nghiệp, được thiết kế để kết nối các bộ điều khiển điện tử (ECU) trong một hệ thống.

##### Nguyên lý hoạt động:

- Các node (ECU) kết nối với nhau thông qua hai dây bus CANH và CANL.
- Dữ liệu được truyền dưới dạng các khung tin (frame) theo một định dạng chuẩn.
- Cơ chế arbitration đảm bảo rằng chỉ có một node được truyền dữ liệu tại một thời điểm.



Hình 2.4 Giao thức CAN kết nối các bộ điều khiển

### 2.1.5. Ưu điểm giao thức CAN

- Tốc độ truyền dữ liệu cao: Phù hợp với các ứng dụng thời gian thực.
- Khả năng chịu nhiễu tốt: Đảm bảo sự ổn định của hệ thống.
- Dễ dàng mở rộng: Thêm hoặc xóa các node mà không ảnh hưởng đến hệ thống.
- Chi phí thấp: Sử dụng ít dây dẫn hơn so với các giao thức khác.

### 2.1.6. Ứng dụng giao thức CAN trong xe hơi điện

- **Kết nối các ECU:** Các ECU khác nhau trong xe điện (ECU động cơ, ECU pin, ECU ABS,...) liên lạc với nhau qua mạng CAN để trao đổi dữ liệu và điều khiển các chức năng.
  - **Truyền dữ liệu cảm biến:** Các cảm biến đo tốc độ, nhiệt độ, áp suất,... truyền dữ liệu về ECU trung tâm qua mạng CAN.
  - **Điều khiển các bộ thực thi:** ECU trung tâm gửi tín hiệu điều khiển đến các bộ thực thi (động cơ, phanh,...) qua mạng CAN.
  - **Chẩn đoán lỗi:** Khi xảy ra lỗi, các ECU sẽ gửi thông báo lỗi qua mạng CAN để kỹ thuật viên có thể xác định và khắc phục.
  - **Cập nhật phần mềm:** Phần mềm của các ECU có thể được cập nhật qua mạng CAN.
- Tiện lợi và ứng dụng:**
- **Tăng cường độ tin cậy:** Các hệ thống trên xe điện hoạt động đồng bộ và ổn định hơn.
  - **Giảm số lượng dây dẫn:** Giảm trọng lượng và chi phí sản xuất.
  - **Dễ dàng bảo trì và nâng cấp:** Có thể thêm hoặc thay thế các ECU mà không cần thay đổi toàn bộ hệ thống.

– **Cung cấp nền tảng cho các tính năng nâng cao:** Như hỗ trợ lái xe, tự động hóa, kết nối với điện thoại thông minh.

#### Ví dụ cụ thể:

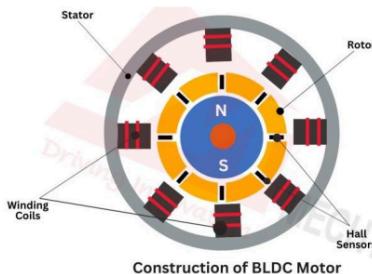
– **Hệ thống phanh ABS:** Các cảm biến đo tốc độ bánh xe truyền dữ liệu về ECU ABS qua mạng CAN. ECU ABS tính toán và điều chỉnh lực phanh trên từng bánh xe để ngăn ngừa bánh xe bị bó cứng.

– **Hệ thống điều khiển hành trình:** ECU điều khiển hành trình nhận dữ liệu tốc độ từ cảm biến và tín hiệu điều khiển từ người lái, sau đó gửi tín hiệu điều khiển đến ECU động cơ để giữ tốc độ ổn định.

Giao thức CAN đóng vai trò quan trọng trong việc kết nối và điều khiển các hệ thống trong xe điện. Nó mang lại nhiều lợi ích như tăng cường độ tin cậy, giảm chi phí và mở ra nhiều khả năng ứng dụng mới cho xe điện.

#### 2.1.7. Ý tưởng áp dụng vào đồ án

Đối với đồ án cho luận văn tốt nghiệp, nhóm chúng em đã đưa ra những áp dụng giao



thức CAN vào xe điện với 3 nhiệm vụ chính, đó là điều khiển và giám sát các chức năng sau:  
Áp suất khí nén lốp xe, tốc độ của bánh xe và nhiệt độ bên ngoài xe hơi điện.

#### 2.2. Động cơ BLDC và BLDC driver

##### 2.2.1. Động cơ BLDC

Động cơ BLDC (Brushless DC Motor) là loại động cơ điện một chiều không sử dụng chổi than và cỗ góp như động cơ DC truyền thống. Thay vào đó, chúng sử dụng nam châm

vĩnh cửu trên rotor và hệ thống điều khiển điện tử để chuyển đổi dòng điện, giúp tăng hiệu suất và độ tin cậy.

**Hình 2.5** Cấu tạo của động cơ BLDC

### Ưu điểm của động cơ BLDC



**Hình 2.6** Hình ưu điểm của động cơ BLDC

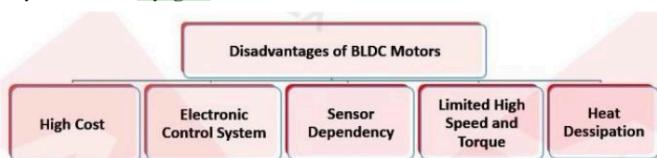
– **Hiệu suất cao:** Động cơ BLDC có hiệu suất vượt trội, tiêu thụ ít năng lượng hơn cho cùng một công suất cơ học, **giúp tiết kiệm năng lượng và giảm chi phí vận hành.**

– **Tuổi thọ dài:** Việc loại bỏ chổi than và cỗ góp giảm thiểu hao mòn cơ học, kéo dài tuổi thọ và giảm nhu cầu bảo trì.

– **Điều khiển chính xác:** Sử dụng hệ thống điều khiển điện tử cho phép kiểm soát tốc độ và mô-men xoắn một cách chính xác, phù hợp với các ứng dụng yêu cầu **độ chính xác cao.**

– **Hoạt động êm ái:** Thiết kế không chổi than giúp giảm tiếng ồn và rung động, cải thiện trải nghiệm người dùng trong các thiết bị gia dụng và công nghiệp.

### Nhược điểm của động cơ BLDC



**Hình 2.7** Hình nhược điểm của động cơ BLDC

– **Chi phí cao:** Do cấu trúc phức tạp và yêu cầu **hệ thống điều khiển điện tử**, động cơ BLDC thường có chi phí sản xuất và bảo trì cao hơn so với động cơ DC truyền thống.

– **Yêu cầu hệ thống điều khiển phức tạp:** Hoạt động của động cơ BLDC phụ thuộc vào bộ điều khiển điện tử để quản lý dòng điện và vị trí rotor, đòi hỏi thiết kế và lập trình phức tạp.

– **Khó sửa chữa:** Khi xảy ra sự cố, việc chẩn đoán và sửa chữa động cơ BLDC có thể phức tạp hơn do cấu trúc và hệ thống điều khiển tinh vi.

#### Ứng dụng của động cơ BLDC

Động cơ BLDC được ứng dụng rộng rãi trong nhiều lĩnh vực nhờ những ưu điểm nổi bật:

– **Phương tiện điện:** Sử dụng trong ô tô điện, xe máy điện và xe đạp điện, cung cấp hiệu suất cao và tuổi thọ dài, góp phần tăng phạm vi hoạt động và giảm chi phí bảo trì.

– **Thiết bị gia dụng:** Áp dụng trong quạt điện, máy điều hòa không khí và máy giặt, giúp giảm tiếng ồn và tăng hiệu quả năng lượng.

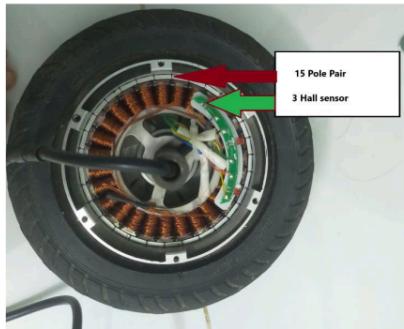
– **Tự động hóa công nghiệp:** Được sử dụng trong robot, băng chuyền và các hệ thống điều khiển chuyển động yêu cầu độ chính xác và độ tin cậy cao.

– **Thiết bị y tế:** Sử dụng trong máy bơm insulin, thiết bị hỗ trợ hô hấp và các dụng cụ phẫu thuật, nơi yêu cầu hoạt động êm ái và chính xác.

Ở đồ án này, nhóm chúng em sử dụng động cơ Hoverboard 24VDC, đây là động cơ tháo từ xe điện cân bằng 2 bánh với khả năng chịu tải cao.



Hình 2.8 Hình hover board BLDC 24V



Hình 2.9 Hình cấu tạo hoverboard

Vì đây là động cơ tháo máy cho nên nhóm chúng em không có tài liệu mô tả kỹ thuật để có thể tra cứu, chính vì vậy nhóm chúng em đưa ra giải pháp là tháo rời động cơ như trên để có thể tự thực hiện **tính toán các thông số cần thiết**.

BLDC hoverboard nhóm chúng em sử dụng gồm có 3 cảm biến Hall và 15 cặp cực Stator. Dựa vào thông số này, khi BLDC quay hết 1 vòng thì mỗi cảm biến Hall sẽ xuất ra 15 tín hiệu mức cao và 15 tín hiệu mức thấp.

=> Như vậy, ta có được kết quả là sẽ có 45 tín hiệu mức cao cho 1 vòng quay, đây là dữ liệu quan trọng để nhóm chúng em thực hiện lập trình điều khiển và đo đặc tốc độ quay của động cơ.

### 2.2.2. Mạch lái động cơ BLDC

Bộ điều khiển này hoạt động trong dải điện áp **DC 12V đến 36V**, đảm bảo cung cấp nguồn ổn định cho các hệ thống động cơ không chổi than. Với **công suất điều khiển tối đa 500W** và **dòng điện hoạt động ≤ 15A**, nó mang lại hiệu suất cao và duy trì hoạt động ổn định cho động cơ.

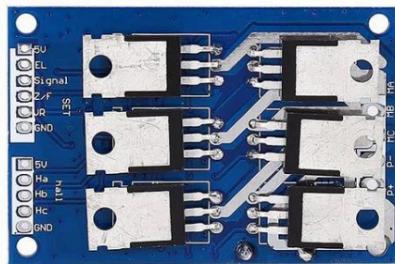
Tích hợp **cảm biến Hall**, bộ điều khiển cho phép quản lý **chính xác tốc độ** và **hướng quay** của động cơ, giúp vận hành êm ái và mượt mà. Ngoài ra, thiết bị còn có **bảo vệ kẹt rotor (stall protection)**, **bảo vệ quá dòng (over-current protection)** và **điều khiển hướng quay (steering control)**, giúp nâng cao độ an toàn và độ tin cậy khi vận hành.

Thiết kế **PCB nhỏ gọn** với kích thước **63x42x17mm**, dễ dàng tích hợp vào nhiều ứng dụng khác nhau, đảm bảo hiệu suất bền bỉ ngay cả trong điều kiện khắc nghiệt.

Với những tính năng vượt trội và độ tin cậy cao, **Bộ điều khiển động cơ không chổi than DC 12V-36V 500W** là một lựa chọn lý tưởng để quản lý và điều khiển động cơ không chổi than trong nhiều ứng dụng khác nhau.



Hình 2.10 Hình mặt trước mạch lái động cơ BLDC 12-36VDC



Hình 2.11 Hình mặt sau 6 mosfet của BLDC driver

#### Kết nối chân (Pin Connection):

- **5V** – Điện áp đầu ra nội bộ của bảng điều khiển
- **Signal** – Cổng xuất tín hiệu xung tốc độ động cơ (tín hiệu xung 5V)

#### Cổng điều khiển (Control port):

- **Z/F** – Cổng điều khiển hướng quay.
  - o Kết nối mức cao “5V” hoặc không kết nối: quay thuận

- Kết nối mức thấp “0V” hoặc nối với GND: quay ngược
- **VR** – Công điều khiển tốc độ.
  - Điều chỉnh tốc độ tuyến tính bằng điện áp analog 0.1V - 5V
  - Điện trở đầu vào: 20KΩ
  - Khi điều chỉnh tốc độ bằng PWM, nối với GND
  - Tần số PWM: 1-20kHz
  - Chu kỳ nhiệm vụ (duty cycle): 0-100%
- **GND** – Chân nối đất cho hệ thống điều khiển

**Cổng nguồn (Power port):**

- **MA** – Pha A của động cơ
- **MB** – Pha B của động cơ
- **MC** – Pha C của động cơ
- **GND** – Cực âm nguồn DC
- **VCC** – Cực dương nguồn DC

**Chân kết nối cảm biến Hall (HALL Terminals):**

- **5V** – Cấp nguồn dương (+) cho cảm biến Hall
- **Ha** – Tín hiệu Hall pha A
- **Hb** – Tín hiệu Hall pha B
- **Hc** – Tín hiệu Hall pha C
- **GND** – Cực âm (-) của nguồn cảm biến Hall

**Lưu ý:**

- Nếu khoảng cách giữa bảng điều khiển và động cơ lớn hơn 50 cm, nên sử dụng dây cáp có lớp chống nhiễu để tránh ảnh hưởng đến hiệu suất điều khiển.
  - Khoảng cách chân cổng điều khiển: 2.54mm
  - Khoảng cách chân cổng nguồn: 3.96mm
  - Cần đảm bảo cách điện giữa MOSFET điều khiển và tản nhiệt hoặc tấm lắp đặt.



Hình 2.12 Hình ảnh mạch BLDC driver trên mô hình

=> Vì động cơ BLDC ăn dòng lớn, nhóm chúng em đã bổ sung thêm một bộ tản nhiệt bằng nhôm cho 6 mosfet được lắp ở mặt sau của BLDC driver

### 2.3. Vi điều khiển STM 32

#### 2.3.1. Giới thiệu STM32

##### STM32 – Vi điều khiển đa năng cho mọi ứng dụng

STM32 là một họ vi điều khiển 32-bit do STMicroelectronics sản xuất, dựa trên kiến trúc ARM Cortex-M. Điểm mạnh của STM32 là sự đa dạng về **chủng loại**, hiệu năng cao, tích hợp nhiều ngoại vi và **khả năng tiết kiệm năng lượng**. Nhờ đó, STM32 được sử dụng rộng rãi trong nhiều lĩnh vực khác nhau.



Hình 2.13 Vi điều khiển STM32

##### Đặc điểm nổi bật

- **Hiệu năng cao:** Dòng vi xử lý ARM Cortex-M mang lại hiệu năng xử lý mạnh mẽ, đáp ứng tốt các yêu cầu của ứng dụng.

- **Tiết kiệm năng lượng:** STM32 được thiết kế để tiêu thụ ít năng lượng, phù hợp cho các ứng dụng chạy bằng pin hoặc yêu cầu thời gian hoạt động dài.
- **Tích hợp nhiều ngoại vi:** STM32 tích hợp nhiều ngoại vi như USB, UART, SPI, I2C, ADC, DAC, Timer, giúp đơn giản hóa thiết kế và giảm chi phí.
- **Đa dạng về chủng loại:** STM32 có <sup>1</sup> nhiều dòng sản phẩm khác nhau, từ dòng F0 (giá rẻ) đến dòng H7 (hiệu năng cao), đáp ứng nhu cầu đa dạng của người dùng.
- **Hỗ trợ phần mềm và công cụ phát triển:** STMicroelectronics cung cấp nhiều công cụ phát triển phần mềm và phần cứng, giúp người dùng dễ dàng làm việc với STM32.

#### Cách thức giao tiếp

STM32 hỗ trợ <sup>4</sup> nhiều giao thức giao tiếp khác nhau, bao gồm:

- **USB:** Giao tiếp với máy tính và các thiết bị USB.
- **UART:** Giao tiếp nối tiếp không đồng bộ.
- **SPI:** Giao tiếp nối tiếp đồng bộ.
- **I2C:** Giao tiếp nối tiếp hai dây.
- **CAN:** Giao tiếp trong mạng lưới.
- **Ethernet:** Giao tiếp mạng.

#### Ứng dụng của STM32

STM32 được sử dụng rộng rãi trong nhiều lĩnh vực khác nhau, bao gồm:

- **IoT (Internet of Things):** Kết nối các thiết bị với internet và điều khiển chúng từ xa.
- **Điện tử tiêu dùng:** Các thiết bị gia dụng, thiết bị đeo thông minh.
- **Công nghiệp:** Hệ thống điều khiển, tự động hóa.
- **Ô tô:** Hệ thống điều khiển động cơ, hệ thống an toàn.
- **Y tế:** Thiết bị y tế, máy móc chẩn đoán.

#### Lợi ích của STM 32

- **Giảm chi phí:** STM32 có nhiều tính năng, giúp giảm chi phí linh kiện và thiết kế.
- **Tăng tốc độ phát triển:** Các công cụ phát triển phần mềm và phần cứng giúp người dùng nhanh chóng triển khai ứng dụng.

– **Nâng cao hiệu năng:** Vi xử lý ARM Cortex-M mang lại hiệu năng cao, đáp ứng tốt các yêu cầu của ứng dụng.

– **Tiết kiệm năng lượng:** STM32 tiêu thụ ít năng lượng, phù hợp cho các ứng dụng chạy bằng pin.

#### Kết luận

STM32 là một dòng vi điều khiển mạnh mẽ, đa năng và được sử dụng rộng rãi trong nhiều lĩnh vực khác nhau. Với hiệu năng cao, tích hợp nhiều ngoại vi, khả năng tiết kiệm năng lượng và sự hỗ trợ tốt từ nhà sản xuất, STM32 là một lựa chọn lý tưởng cho các nhà thiết kế điện tử.

#### 2.3.2. Các bước hoạt động của STM32

##### Khởi động (Booting)

– **Cấp nguồn:** Vi điều khiển STM32 bắt đầu hoạt động khi được cấp nguồn.

– **Nạp chương trình:** Sau khi cấp nguồn, vi điều khiển sẽ thực hiện quá trình nạp chương trình (firmware) từ bộ nhớ Flash. Quá trình này thường được thực hiện thông qua các giao thức như JTAG, SWD hoặc thông qua bootloader.

– **Vector table:** Sau khi nạp chương trình, vi điều khiển sẽ nhảy đến địa chỉ đầu tiên trong vector table để bắt đầu thực thi chương trình. Vector table chứa địa chỉ của các trình xử lý ngắn và các hàm khởi tạo.

##### Thiết lập chế độ (Initialization)

– **Cấu hình thanh ghi:** Trong giai đoạn này, vi điều khiển sẽ thiết lập các thanh ghi (registers) để cấu hình các ngoại vi (peripherals), bộ nhớ và các thành phần khác. Ví dụ, người dùng có thể cấu hình chân GPIO, bộ ADC, bộ UART, bộ Timer, v.v.

– **Khởi tạo biến:** Các biến toàn cục (global variables) và biến tĩnh (static variables) sẽ được khởi tạo.

– **Gọi hàm khởi tạo:** Các hàm khởi tạo (initialization functions) sẽ được gọi để thiết lập các thành phần của hệ thống.

##### Xử lý (Processing)

– **Thực thi chương trình:** Vi điều khiển sẽ thực thi các lệnh trong chương trình đã được nạp vào bộ nhớ Flash. Chương trình này có thể bao gồm các hàm, các lệnh điều khiển, các phép tính toán và các thao tác xử lý dữ liệu.

– **Xử lý ngắt:** Vi điều khiển có thể nhận và xử lý các ngắt (interrupts) từ các nguồn khác nhau, chẳng hạn như ngắt từ bộ hẹn giờ, ngắt từ các thiết bị ngoại vi, ngắt từ người dùng (ví dụ: nhấn nút), v.v. Khi có ngắt, vi điều khiển sẽ tạm dừng thực thi chương trình chính để chạy trình xử lý ngắt (interrupt handler) tương ứng.

– **Gọi hàm:** Chương trình có thể gọi các hàm (functions) để thực hiện các tác vụ cụ thể. Các hàm này có thể được định nghĩa trong chương trình hoặc được cung cấp bởi các thư viện.

#### Đáp ứng yêu cầu (Responding to requests)

– **Giao tiếp với thiết bị khác:** Vi điều khiển có thể giao tiếp với các thiết bị khác thông qua các giao thức như UART, SPI, I2C, USB, CAN, v.v.

– **Điều khiển và giám sát thiết bị ngoại vi:** Vi điều khiển có thể điều khiển và giám sát các thiết bị ngoại vi như cảm biến, động cơ, đèn LED, màn hình LCD, v.v.

– **Đọc và ghi dữ liệu vào bộ nhớ:** Vi điều khiển có thể đọc và ghi dữ liệu vào bộ nhớ RAM hoặc bộ nhớ Flash.

– **Thực hiện các chức năng khác:** Vi điều khiển có thể thực hiện nhiều chức năng khác như đo lường (ví dụ: đo điện áp, dòng điện, nhiệt độ), điều khiển (ví dụ: điều khiển động cơ, điều khiển đèn), xử lý tín hiệu (ví dụ: lọc tín hiệu, khuếch đại tín hiệu), v.v.

#### Vòng lặp chính (Main loop)

Sau khi hoàn thành quá trình khởi tạo, vi điều khiển thường bước vào một vòng lặp chính (main loop) để liên tục thực hiện các tác vụ và đáp ứng các yêu cầu của hệ thống. Vòng lặp chính này có thể được thiết kế theo nhiều cách khác nhau, tùy thuộc vào ứng dụng cụ thể.

#### 2.3.3. Tính năng của vi điều khiển STM32

##### Các tính năng cơ bản của STM32

##### Kiến trúc ARM Cortex-M:

– STM32 sử dụng kiến trúc ARM Cortex-M, một kiến trúc phổ biến trong các ứng dụng nhúng, nổi tiếng với hiệu năng cao và khả năng tiết kiệm năng lượng.

– Có nhiều dòng Cortex-M khác nhau (M0, M3, M4, M7), mỗi dòng có đặc điểm và hiệu năng riêng, cho phép người dùng lựa chọn vi điều khiển phù hợp với yêu cầu của ứng dụng.

#### Số chân GPIO đa dạng:

– STM32 cung cấp nhiều tùy chọn về số lượng chân GPIO (General Purpose Input/Output), cho phép kết nối và điều khiển nhiều thiết bị ngoại vi khác nhau.

– Các chân GPIO có thể được cấu hình để thực hiện nhiều chức năng khác nhau, chẳng hạn như đầu vào, đầu ra, ngắt, v.v.

**Kích thước nhỏ gọn:** STM32 có kích thước nhỏ gọn, giúp tiết kiệm không gian trên mạch in và dễ dàng tích hợp vào các ứng dụng có không gian hạn chế.

#### Dung lượng bộ nhớ lớn:

– STM32 tích hợp bộ nhớ Flash và SRAM với dung lượng khác nhau, đáp ứng nhu cầu lưu trữ chương trình và dữ liệu của nhiều ứng dụng.

– Bộ nhớ Flash được sử dụng để lưu trữ chương trình, trong khi bộ nhớ SRAM được sử dụng để lưu trữ dữ liệu tạm thời trong quá trình hoạt động.

#### Tính năng bảo mật:

– STM32 tích hợp nhiều tính năng bảo mật, giúp bảo vệ dữ liệu và chương trình khỏi các truy cập trái phép.

– Các tính năng bảo mật bao gồm: bảo vệ bộ nhớ, mã hóa dữ liệu, secure boot, v.v.

#### Tính năng giao tiếp đa dạng:

– STM32 hỗ trợ nhiều chuẩn giao tiếp khác nhau, cho phép kết nối với nhiều loại thiết bị ngoại vi.

– Các chuẩn giao tiếp phổ biến bao gồm: USB, I2C, SPI, UART, CAN, Ethernet, v.v.

**Tính năng điều khiển PWM:** STM32 tích hợp các bộ điều khiển PWM (Pulse Width Modulation), cho phép điều khiển tốc độ động cơ, độ sáng đèn LED và nhiều ứng dụng khác một cách dễ dàng và hiệu quả.

**Các tính năng khác:** Ngoài các tính năng trên, STM32 còn tích hợp nhiều tính năng khác như ADC (Analog-to-Digital Converter), DAC (Digital-to-Analog Converter), Timer, RTC (Real-Time Clock), DMA (Direct Memory Access), v.v.

#### 2.3.4. Giới thiệu dòng STM32F103C8T6

##### Tổng quan về Vi điều khiển STM32F103C8T6

STM32F103C8T6 là một vi điều khiển 32-bit mạnh mẽ, thuộc dòng F1 của họ STM32 do STMicroelectronics sản xuất. Nó được xây dựng trên kiến trúc ARM Cortex-M3, mang lại hiệu năng cao và khả năng tiết kiệm năng lượng.



Hình 2.14 Mô phỏng vi điều khiển STM32F103C8T6

##### Đặc điểm nổi bật

###### Lõi ARM Cortex-M3:

– Đây là một trong những kiến trúc phổ biến nhất trong các ứng dụng nhúng, nổi tiếng với sự cân bằng giữa hiệu năng và mức tiêu thụ năng lượng.

– Lõi Cortex-M3 cung cấp các tính năng mạnh mẽ như xử lý ngắt hiệu quả, quản lý năng lượng linh hoạt và các tập lệnh tối ưu cho ứng dụng nhúng.

**Tốc độ xử lý:** STM32F103C8T6 có tốc độ xử lý tối đa 72MHz, cho phép thực hiện các tác vụ phức tạp một cách nhanh chóng.

**Bộ nhớ:** Vi điều khiển này đi kèm với 64KB bộ nhớ Flash để lưu trữ chương trình và 20KB SRAM để lưu trữ dữ liệu trong quá trình chạy.

**Clock, reset và quản lý nguồn:**

– Điện áp hoạt động: STM32F103C8T6 có thể hoạt động ở điện áp từ 2.0V đến 3.6V, phù hợp với nhiều ứng dụng khác nhau.

– Thạch anh ngoài: Hỗ trợ thạch anh ngoài từ 4MHz đến 20MHz để tạo xung clock chính xác cho vi điều khiển.

– Thạch anh nội: Tích hợp bộ dao động RC bên trong, có thể hoạt động ở chế độ 8MHz hoặc 40kHz.

– Chế độ điện áp thấp: Vi điều khiển có các chế độ tiết kiệm năng lượng như Sleep, Stop và Standby.

– Pin VBAT: Chân VBAT cho phép kết nối pin ngoài để duy trì hoạt động của RTC (Real-Time Clock) và lưu trữ dữ liệu khi nguồn chính bị mất.

**ADC (Analog-to-Digital Converter):**

– STM32F103C8T6 có 2 bộ ADC 12-bit, mỗi bộ có 9 kênh.

– Dải giá trị chuyển đổi từ 0V đến 3.6V.

– Hỗ trợ chế độ lấy mẫu một kênh hoặc nhiều kênh.

**DMA (Direct Memory Access):**

– Vi điều khiển có 7 kênh DMA, cho phép truyền dữ liệu giữa bộ nhớ và các ngoại vi mà không cần sự can thiệp của CPU, giúp tăng hiệu suất hệ thống.

– DMA được hỗ trợ cho ADC, UART, I2C và SPI.

**Timer**

– STM32F103C8T6 có 7 bộ timer:

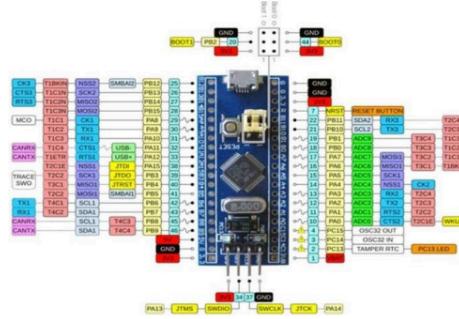
- 3 timer 16-bit hỗ trợ các chế độ Input Capture, Output Compare và PWM.
- 1 timer 16-bit chuyên dụng cho điều khiển động cơ với các chế độ bảo vệ ngắt Input và dead-time.
- 2 watchdog timer nhằm kiểm tra lỗi và bảo toàn.
- 1 SysTick timer 24-bit đếm xuống, thường được sử dụng cho hàm delay.

### Giao tiếp

- Vi điều khiển hỗ trợ nhiều giao thức giao tiếp khác nhau:
  - o 2 bộ I2C
  - o 3 bộ USART (Universal Synchronous/Asynchronous Receiver/Transmitter)
  - o 2 SPI (Serial Peripheral Interface)
  - o 1 CAN (Controller Area Network)
  - o USB 2.0 full-speed interface

### Tính năng khác

- Kiểm tra lỗi CRC (Cyclic Redundancy Check)
- 96-bit ID độc đáo



Hình 2.15 Sơ đồ chân của STM32F103C8T6

### Thông số kỹ thuật

- Nguồn điện: Điện áp cấp 5VDC được chuyển đổi thành 3.3VDC qua IC nguồn và cấp cho vi điều khiển chính. Điều này giúp đơn giản hóa việc cấp nguồn cho vi điều khiển, bạn có thể sử dụng nguồn 5V từ máy tính hoặc bộ sạc USB.

- Tần số:
  - o Tích hợp sẵn thạch anh 8Mhz. Thạch anh này được sử dụng làm nguồn xung clock chính cho vi điều khiển.

- o Tích hợp sẵn thạch anh 32Khz cho các ứng dụng RTC (Real-Time Clock).

Thạch anh này cung cấp xung clock cho RTC, giúp duy trì thời gian thực ngay cả khi vi điều khiển không hoạt động.

– Giao tiếp:

Ra chân đầy đủ tất cả các GPIO (General Purpose Input/Output) và giao tiếp: CAN, I2C, SPI, UART, USB,...

- o + GPIO: Các chân GPIO có thể được cấu hình để làm đầu vào hoặc đầu ra, cho phép kết nối và điều khiển nhiều loại thiết bị ngoại vi khác nhau.

o CAN: Giao thức CAN (Controller Area Network) thường được sử dụng trong các ứng dụng ô tô và công nghiệp để giao tiếp giữa các thiết bị.

- o I2C: Giao thức I2C (Inter-Integrated Circuit) là một giao thức nối tiếp hai dây, thường được sử dụng để kết nối các cảm biến và thiết bị ngoại vi.

o SPI: Giao thức SPI (Serial Peripheral Interface) là một giao thức nối tiếp tốc độ cao, thường được sử dụng để kết nối các thiết bị như bộ nhớ Flash, màn hình LCD, v.v.

o UART: Giao thức UART (Universal Asynchronous Receiver/Transmitter) là một giao thức nối tiếp không đồng bộ, thường được sử dụng để giao tiếp với máy tính và các thiết bị khác.

o USB: Giao thức USB (Universal Serial Bus) cho phép kết nối vi điều khiển với máy tính và các thiết bị USB khác.

– Đèn Led

Tích hợp Led trạng thái nguồn, Led PC13, Nút Reset.

- o Led trạng thái nguồn: Cho biết trạng thái **hoạt động** của vi điều khiển.

o Led PC13: **Led được kết nối với chân** PC13, có thể được sử dụng cho các mục đích khác nhau.

- o Nút Reset: Nút nhấn để khởi động lại vi điều khiển.

– Kích thước: 53.34 x 15.24mm. Kích thước nhỏ gọn giúp dễ dàng tích hợp vi điều khiển vào các ứng dụng có không gian hạn chế.

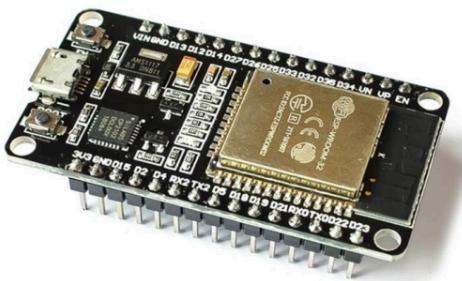
**Thông số kỹ thuật**

Ngoài các thông số trên, STM32F103C8T6 còn có nhiều thông số kỹ thuật quan trọng khác, bao gồm:

- Lõi: ARM Cortex-M3
- Tốc độ: 72MHz
- Bộ nhớ Flash: 64KB
- Bộ nhớ SRAM: 20KB
- ADC: 2 bộ ADC 12-bit
- DMA: 7 kênh DMA
- Timer: 7 bộ timer
- Điện áp hoạt động: 2.0V - 3.6V

#### 2.4. Vi điều khiển ESP 32

##### 2.4.1. Giới thiệu ESP 32



Hình 2.16 Bo mạch phát triển dựa trên vi điều khiển ESP32

ESP32 là một loại chip điều khiển nhỏ gọn, tiết kiệm năng lượng và có giá thành phải chăng. Nó được trang bị cả WiFi và Bluetooth (chế độ kép), cho phép kết nối không dây đa dạng.

Chip <sup>6</sup>ESP32 sử dụng bộ xử lý Tensilica Xtensa LX6, có <sup>6</sup>cả phiên bản lõi đơn và lõi kép. Điểm đặc biệt của nó là tích hợp nhiều thành phần quan trọng như công tắc antenna, RF balun, bộ khuếch đại công suất, bộ khuếch đại thu nhiễu thấp, bộ lọc và module quản lý năng lượng, giúp tối ưu hóa hiệu suất và tiết kiệm không gian.

ESP32 được phát triển bởi công ty Espressif Systems, có trụ sở tại Thượng Hải, Trung Quốc. Quá trình sản xuất chip được thực hiện bởi TSMC với công nghệ 40nm. ESP32 được xem là phiên bản nâng cấp của dòng **vị điều khiển ESP8266** trước đó.

#### 2.4.2. Cấu hình ESP 32

##### Vị xử lý

6

- Loại CPU: Xtensa Dual-Core LX6
- Kiến trúc: 32-bit
- Tốc độ hoạt động: 160MHz (có thể tăng lên đến 240MHz)
- Tốc độ xung đọc từ bộ nhớ Flash: 40MHz (có thể điều chỉnh lên 80MHz trong quá trình lập trình)
  - Bộ nhớ đệm RAM: 520KB SRAM, trong đó:
    - o 8KB RAM RTC tốc độ cao (Real-Time Clock)
    - o 8KB RAM RTC tốc độ thấp (dùng trong chế độ DeepSleep)

##### Kết nối không dây

- Wi-Fi: 802.11 b/g/n/e/i
- Bluetooth: v4.2 BR/EDR và BLE (Bluetooth Low Energy)

##### Giao tiếp ngoại vi

ESP32 hỗ trợ rất nhiều giao thức giao tiếp, bao gồm:

- DAC (Digital to Analog Converter): 2 cổng 8-bit
- ADC (Analog to Digital Converter): 16 cổng 12-bit
- I<sup>2</sup>C: 2 cổng
- UART: 3 cổng
- SPI: 3 cổng (1 cổng dành riêng cho chip Flash)
- I<sup>2</sup>S: 2 cổng
- SD card / SDIO / MMC host
- Slave (SDIO/SPI)
- Ethernet MAC interface with DMA và hỗ trợ IEEE 1588
- CAN bus 2.0

- IR (TX/RX)
- PWM (Pulse Width Modulation): Tất cả các chân GPIO
- Bộ tiền khuếch đại analog công suất cực thấp

**Cảm biến tích hợp**

ESP32 được trang bị sẵn các cảm biến sau:

- Cảm biến Hall (từ trường): 1 cảm biến
- Cảm biến nhiệt độ: 1 cảm biến
- Cảm biến chạm (diện dung): 10 đầu vào

**Tính năng bảo mật**

ESP32 tích hợp nhiều tính năng bảo mật quan trọng:

- Các tính năng bảo mật theo chuẩn IEEE 802.11 (WFA, WPA/WPA2, WAPI)
- Secure boot
- Mã hóa Flash
- Bộ nhớ OTP 1024-bit (One-Time Programmable), tối đa 768-bit cho người dùng
- Tăng tốc phần cứng cho các thuật toán mã hóa (AES, SHA-2, RSA, ECC, RNG)

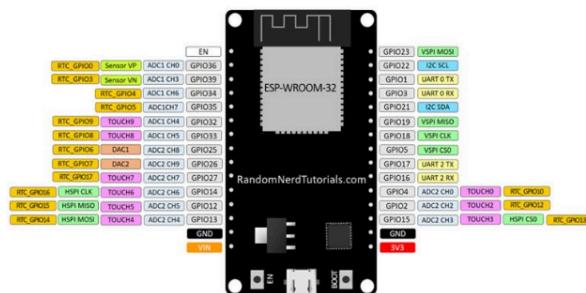
**Nguồn điện và môi trường hoạt động**

- Nhiệt độ hoạt động: -40°C đến 85°C
- Điện áp hoạt động: 2.2V - 3.6V
- Số chân GPIO (General Purpose Input/Output): 34

### 2.4.3. Sơ đồ chân của DOIT ESP 32 DEVKIT VQ BOARD

Phiên bản có 30 GPIO

**ESP32 DEVKIT V1 – DOIT**  
version with 30 GPIOs



Hình 2.17 Sơ đồ chân của ESP 32

### 2.4.4. Ưu điểm của ESP 32 so với các thiết bị tương tự

#### ESP32 - Bước tiến vượt trội so với ESP8266

ESP32 thực sự là một bước nâng cấp đáng kể so với ESP8266, không chỉ về hiệu năng mà còn về tính năng và khả năng kết nối. Trong khi ESP8266 phù hợp với các dự án IoT đơn giản, tiết kiệm chi phí, thì ESP32 lại là lựa chọn lý tưởng cho các ứng dụng phức tạp, đòi hỏi khả năng xử lý mạnh mẽ và tích hợp nhiều ngoại vi.

#### So sánh chi tiết

- Số chân GPIO: ESP32 vượt trội với 30/36 chân GPIO so với 17 chân của ESP8266, mang lại khả năng kết nối linh hoạt hơn với nhiều thiết bị ngoại vi.
- ADC: ESP32 sở hữu 18 kênh ADC 12-bit, cho độ chính xác cao hơn so với 8 kênh ADC 10-bit của ESP8266.
- PWM: ESP32 có 16 kênh PWM mềm, nhiều hơn gấp đôi so với 8 kênh của ESP8266, giúp điều khiển các thiết bị như động cơ, đèn LED một cách linh hoạt hơn.
- Cảm biến tích hợp: ESP32 tích hợp sẵn cảm biến chạm, cảm biến Hall, Ethernet MAC Interface và cảm biến nhiệt độ, giúp đơn giản hóa thiết kế và giảm chi phí cho dự án.

– Bộ nhớ: ESP32 có bộ nhớ lớn hơn với 4MB External Flash và 520KB SRAM, trong đó có 8KB RAM RTC tốc độ cao và 8KB RAM RTC tốc độ thấp, đáp ứng tốt hơn cho các ứng dụng phức tạp.

– Kết nối không dây: ESP32 hỗ trợ cả Bluetooth 4.2 và BLE, mở rộng khả năng kết nối với các thiết bị Bluetooth như bàn phím, chuột, điện thoại, ngay cả khi không có Wi-Fi.

### Ưu điểm nổi bật của ESP32

– Hiệu năng mạnh mẽ: CPU lõi kép Xtensa LX6 cho tốc độ xử lý vượt trội, đáp ứng tốt các yêu cầu tính toán phức tạp.

– Tính năng đa dạng: Tích hợp nhiều ngoại vi và cảm biến, giúp đơn giản hóa thiết kế và mở rộng khả năng ứng dụng.

– Khả năng kết nối linh hoạt: Hỗ trợ Wi-Fi và Bluetooth, cho phép kết nối với nhiều loại thiết bị và mạng.

– Tiết kiệm năng lượng: Chế độ Ultra Low Power giúp giảm mức tiêu thụ điện năng, đặc biệt quan trọng đối với các ứng dụng chạy bằng pin.

### Kết luận

ESP32 là một vi điều khiển mạnh mẽ, đa năng và là sự lựa chọn hoàn hảo cho các dự án IoT phức tạp, đòi hỏi hiệu năng cao, **tính năng đa dạng** và **khả năng kết nối linh hoạt**. Mặc dù có giá thành cao hơn ESP8266, nhưng những ưu điểm vượt trội của ESP32 mang lại giá trị lớn hơn cho người dùng.

## 2.5. Cảm biến nhiệt độ PT100 - MAX31865

### 2.5.1. PT100 - MAX31865

#### MAX31865 - Bộ chuyển đổi RTD sang kỹ thuật số

MAX31865 là bộ chuyển đổi tín hiệu từ điện trở sang dạng kỹ thuật số, được thiết kế đặc biệt để làm việc với các cảm biến nhiệt độ RTD (Resistance Temperature Detector) Platin. Nó giúp đơn giản hóa việc đo và xử lý dữ liệu từ các cảm biến RTD, đặc biệt là PT100.

#### Đặc điểm nổi bật của MAX31865

– Dễ sử dụng: MAX31865 được thiết kế để dễ dàng kết nối và sử dụng với các vi điều khiển.

– Tối ưu hóa cho RTD: Bộ chuyển đổi này được tinh chỉnh để hoạt động tốt nhất với các cảm biến RTD Platin, đảm bảo độ chính xác và ổn định của phép đo.

– Chuyển đổi tín hiệu số: MAX31865 chuyển đổi tín hiệu điện trở từ RTD sang dạng số, giúp dễ dàng xử lý và hiển thị dữ liệu nhiệt độ.

**Cảm biến nhiệt độ PT100:** PT100 là một loại cảm biến nhiệt độ RTD phẳng biến, sử dụng vật liệu Platin để đo nhiệt độ dựa trên nguyên lý thay đổi điện trở.

#### Ý nghĩa của PT100

- PT: Viết tắt của Platin, vật liệu chế tạo cảm biến.
- 100: Giá trị điện trở của cảm biến là 100 ohm ở 0°C.

**Nguyên lý hoạt động:** Khi nhiệt độ thay đổi, điện trở của PT100 cũng thay đổi theo một quy luật nhất định. Sự thay đổi này được MAX31865 chuyển đổi thành tín hiệu số, cho phép xác định nhiệt độ một cách chính xác.

#### Ứng dụng của PT100

PT100 được sử dụng rộng rãi trong các ứng dụng đo nhiệt độ chính xác, bao gồm:

- Hệ thống kiểm soát nhiệt độ công nghiệp
- Phòng thí nghiệm
- Thiết bị y tế
- Các ứng dụng khoa học và nghiên cứu

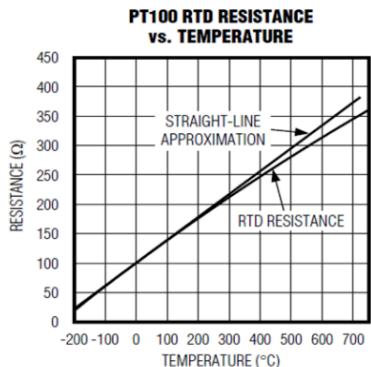
#### Tiêu chuẩn và hệ số alpha

Có hai tiêu chuẩn phổ biến cho PT100, tương ứng với hai giá trị alpha khác nhau:

- IEC 751 (PT100): alpha = 0.00385
- SAMA: alpha = 0.00392

#### Kết hợp MAX31865 và PT100

Sự kết hợp giữa MAX31865 và PT100 tạo thành một hệ thống đo nhiệt độ chính xác và dễ sử dụng. MAX31865 giúp chuyển đổi tín hiệu từ PT100 sang dạng số, cho phép vi điều khiển dễ dàng đọc và xử lý dữ liệu nhiệt độ.



The resistance vs. temperature curve is reasonably linear, but has some curvature, as described by the Callendar-Van Dusen equation:

$$R(T) = R_0(1 + aT + bT^2 + c(T - 100)T^3)$$

where:

$T$  = temperature ( $^{\circ}\text{C}$ )

$R(T)$  = resistance at  $T$

$R_0$  = resistance at  $T = 0^{\circ}\text{C}$

IEC 751 specifies  $a = 0.00385055$  and the following Callendar-Van Dusen coefficient values:

$$a = 3.90830 \times 10^{-3}$$

$$b = -5.77500 \times 10^{-7}$$

$$c = -4.18301 \times 10^{-12} \text{ for } -200^{\circ}\text{C} \leq T \leq 0^{\circ}\text{C}, 0 \text{ for } 0^{\circ}\text{C} \leq T \leq +850^{\circ}\text{C}$$

Hình 2.18 Tương quan giữa nhiệt độ và điện trở

### Đường cong điện trở - nhiệt độ của PT100

Từ biểu đồ trên, đường cong cho thấy mối quan hệ giữa điện trở và nhiệt độ của cảm biến PT100 không hoàn toàn tuyến tính mà có một độ cong nhất định. Điều này được mô tả bằng phương trình Callendar-Van Dusen, cho phép tính toán nhiệt độ dựa trên giá trị điện trở đo được một cách chính xác.

Phương trình Callendar – Van Dusen có dạng sau:

$$R_t = R_0 [1 + At + Bt^2 + C(t-100)t^3]$$

Trong đó:

- $R_t$ : Điện trở của PT100 ở nhiệt độ  $t$
- $R_0$ : Điện trở của PT100 ở  $0^{\circ}\text{C}$  (thường là 100 ohm)
- $t$ : Nhiệt độ ( $^{\circ}\text{C}$ )
- $A, B, C$ : Các hằng số hiệu chuẩn

### Ý nghĩa của phương trình

Phương trình Callendar – Van Dusen cho phép tính toán nhiệt độ một cách chính xác dựa trên giá trị điện trở đo được, bù trừ cho độ cong của đường đặc tuyến.

### Kết nối cảm biến PT100 loại 3 dây với MAX31865

Việc sử dụng kết nối 3 dây giúp giảm thiểu ảnh hưởng của điện trở dây dẫn đến kết quả đo. MAX31865 có khả năng bù trừ điện trở dây dẫn, đảm bảo độ chính xác của phép đo.

**Ưu điểm của kết nối 3 dây**

- Giảm thiểu sai số do điện trở dây dẫn
- Đảm bảo độ chính xác của phép đo

**Dài đo của cảm biến PT100:** Dài đo trung bình của cảm biến PT100 thường dao động từ -40°C đến 200°C. Tuy nhiên, một số loại PT100 có thể đo được nhiệt độ lên đến 400°C.

**Lưu ý khi lựa chọn dài đo**

- Cần lựa chọn cảm biến PT100 có dài đo phù hợp với ứng dụng cụ thể.
- Nhiệt độ đo tối đa không nên vượt quá giới hạn cho phép của cảm biến.

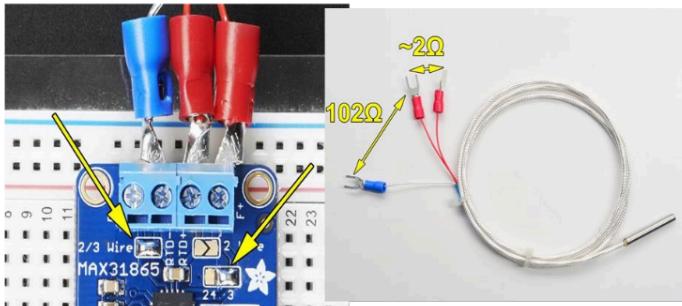
**Kết luận**

Cảm biến PT100 kết hợp với MAX31865 là một giải pháp đo nhiệt độ chính xác và hiệu quả. Việc hiểu rõ về đường cong điện trở - nhiệt độ, cách kết nối và dài đo của PT100 là rất quan trọng để đảm bảo kết quả đo chính xác và tin cậy.

**2.5.2. Quá trình kết nối PT100 với MAX31865**

Hình 2.19 MAX31865 Adafruit

Chân	Tên gọi	Chức năng
Vin	Chân cấp nguồn	Cấp nguồn cho MAX31865 (3V)
3V3	Chân đầu ra 3.3V	Cấp nguồn cho thiết bị khác (tối đa 100mA)
GND	Chân mass	Mass chung
SCK	SPI Clock	Xung nhịp SPI (INPUT)
SDO	Serial Data Out	Dữ liệu từ MAX31865 đến STM32
SDI	Serial Data In	Dữ liệu từ STM32 đến MAX31865
CS	Chip Select	Chọn chip (mức LOW để bắt đầu giao tiếp SPI)



Hình 2.20 PT100 và MAX31865

Sử dụng VOM để xác định các chân cần kết nối. Khi sử dụng loại cảm biến 3 dây, ta cần hàn kết nối tại các điểm mũi tên màu vàng – 2/3 Wire và “4 3”. Và không được quên phải loại bỏ kết nối giữa “2 4”.

Sau khi làm đủ các bước trên, ta đã sẵn sàng để kết nối SPI giữa MAX31865 và STM32 để thực hiện lập trình.

### 2.5.3. Công thức chuyển đổi công thức Callendar – Van Dusen thành ngôn ngữ lập trình

Để có thể biến đổi dữ liệu nhiệt độ nhận từ cảm biến PT100 và MAX31065, ta cần chuyển đổi công thức Callenda – Van Dusen thành ngôn ngữ lập trình để vi điều khiển hiểu được.

Sau khi chuyển đổi ra có giá trị Analog 15 bit được lưu vào biến “fullreg”

$$\text{Công thức (1)} : R_{RTD} = \frac{(ADC\ Code \times R_{Ref})}{2^{15}}$$

$$\Rightarrow RTD = ADC\ Code$$

$$RTD^* = R_{Ref}$$

$$RTD /= 32768(2^{15}) \rightarrow | \text{Ta được: } RTD = R(T)$$

The resistance vs. temperature curve is reasonably linear, but has some curvature, as described by the Callendar-Van Dusen equation:

$$R(T) = R_0(1 + aT + bT^2 + c(T - 100)T^3)$$

where:

$T$  = temperature ( $^{\circ}\text{C}$ )

$R(T)$  = resistance at  $T$

$R_0$  = resistance at  $T = 0^{\circ}\text{C}$

IEC 751 specifies  $\alpha = 0.00385055$  and the following Callendar-Van Dusen coefficient values:

$$a = 3.90830 \times 10^{-3}$$

$$b = -5.77500 \times 10^{-7}$$

$$c = -4.18301 \times 10^{-12} \text{ for } -200^{\circ}\text{C} \leq T \leq 0^{\circ}\text{C}, 0 \text{ for } 0^{\circ}\text{C} \leq T \leq +850^{\circ}\text{C}$$

Hình 2.21 Minh họa công thức Callenda – Van Dusen

Ta cần biến đổi giá trị điện trở  $R(T)$  sang nhiệt độ  $T(^{\circ}\text{C})$

Ta có:

$$R(T) = R_0(1 + aT + bT^2)$$

Đặt biến  $r = R(T); RTDa = a; RTDb = b.$

$$\Leftrightarrow r = 100(1 + RTDaT + RTDbT^2)$$

$$\Leftrightarrow \frac{r}{100} - 1 = RTDaT + RTDbT^2 \text{ (Phương trình bậc 2)}$$

$$\Leftrightarrow T = \frac{-RTDa + \sqrt{RTDa^2 - 4RTDb \times (1 - \frac{r}{100})}}{2RTDb}$$

$$\Leftrightarrow Z_1 = -RTDa$$

$$\Leftrightarrow Z_2 = RTDa^2 - 4RTDb$$

$$\Leftrightarrow Z_3 = \frac{4RTDb}{100}$$

$$\Leftrightarrow Z_4 = 2RTDb$$

$$\Rightarrow T = \frac{Z_1 + \sqrt{Z_2 + Z_3 \cdot r}}{Z_4}$$

Đây sẽ là tiền đề để nhóm lập trình cho cảm biến nhiệt độ PT100 và MAX31065.

## 2.6. Màn hình cảm ứng TFT ILI 9341

ILI9341 là trình điều khiển hệ thống trên chip (SOC) đơn chip cho màn hình tinh thể lỏng (LCD) a-TFT (transistor màn mỏng vô định hình) có độ phân giải 240RGBx320. Đây là

Truong Quang Bảo Khanh – Nguyễn Thái Hòa – Hoàng Minh Khôi

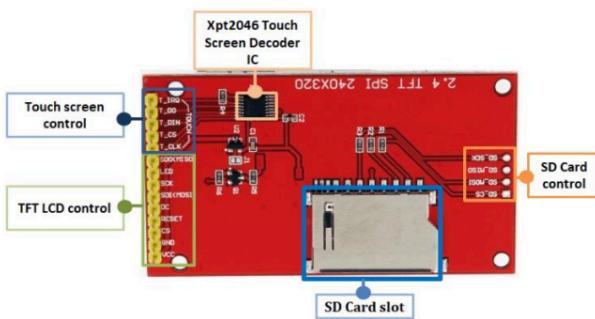
Trang 53

module hiển thị đa năng được sử dụng rộng rãi trong các dự án điện tử, hệ thống nhúng và giao diện màn hình cảm ứng, cho phép hiển thị đồ họa và văn bản rõ nét.



Hình 2.22 Màn hình TFT ILI 9341

#### 2.6.1. Các tính năng chính và thông số kỹ thuật:



Hình 2.23 Các chân đầu nối của TFT ILI 9341

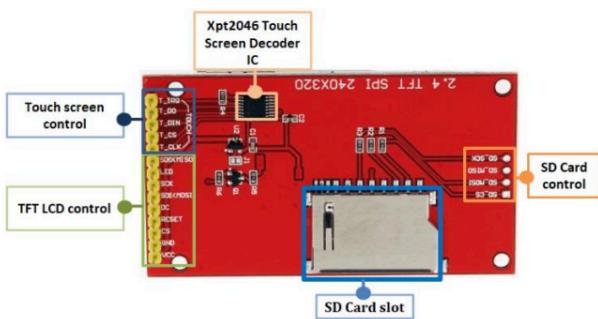
- Độ phân giải: 240xRGB(H) x 320(V)
- Màu hiển thị: Hỗ trợ 262K ở chế độ đầy đủ màu sắc và 8 màu ở chế độ màu giảm.

Một số mô-đun hỗ trợ 65K màu.

- Giao diện: Hỗ trợ giao diện MCU bus dữ liệu song song 8/9/16/18-bit, giao diện RGB bus dữ liệu 6/16/18-bit và giao diện ngoại vi nối tiếp 3/4 dòng (SPI).
- IC trình điều khiển: ILI9341
- Điện áp hoạt động: Điện áp giao diện I/O 1,65V ~ 3,3V. Một số màn hình có thể hoạt động ở mức 3,3V hoặc với đầu vào nguồn 3,3V~5V. Nếu sử dụng MCU 5V, có thể cần bộ dịch mức logic.

- Chức năng trên chip: Bao gồm bộ tạo và điều chỉnh VCOM, bộ tạo thời gian, bộ dao động, bộ chuyển đổi DC/DC và đảo ngược dòng/khung.
- Khu vực hoạt động: Kích thước khu vực hoạt động phẳng biến là 43,2 x 57,6 mm hoặc 36,72 (Rộng) x 48,96 (Cao) mm.
- Kích thước mô-đun: Màn hình có nhiều kích thước mô-đun khác nhau, chẳng hạn như 43 x 72,26 x 3,85 mm hoặc 50,0x86,0 (mm).
- Đèn nền: Thường sử dụng đèn nền LED trắng.

#### 2.6.2. Cấu hình chân cảm (Ví dụ):



Hình 2.24 Các chân đấu nối của TFT ILI 9341

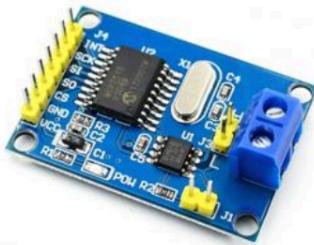
Chân cảm có thể thay đổi đôi chút tùy thuộc vào mô-đun cụ thể, nhưng các chân cảm phẳng biến bao gồm:

- GND: Đất
- VCC: Nguồn điện (thường là 2,5V đến 3,3V hoặc 3,3V-5V)
- CS: Chọn chip cho SPI
- RESET: Chân đặt lại (hoạt động ở mức thấp)
- D/C: Chân điều khiển dữ liệu/lệnh
- MOSI/SDI: Đầu ra chính, đầu vào phụ cho SPI
- SCK: Đồng hồ nối tiếp cho SPI
- LED: Điều khiển đèn nền (cực dương)
- MISO/SDO: Đầu vào chính, đầu ra phụ cho SPI (tùy chọn)

## 2.7. CAN Controller MCP2515 và Transceiver 2551

### 2.7.1. CAN Controller MCP2515

MCP2515 là bộ điều khiển Mạng khu vực điều khiển (CAN) độc lập từ Microchip Technology triển khai thông số kỹ thuật CAN phiên bản 2.0B. Nó được thiết kế để giao tiếp với bus CAN và có khả năng truyền và nhận dữ liệu chuẩn và mở rộng cũng như khung dữ liệu từ xa. MCP2515 giao tiếp với bộ vi điều khiển (MCU) thông qua Giao diện ngoại vi nối tiếp (SPI) tiêu chuẩn công nghiệp.



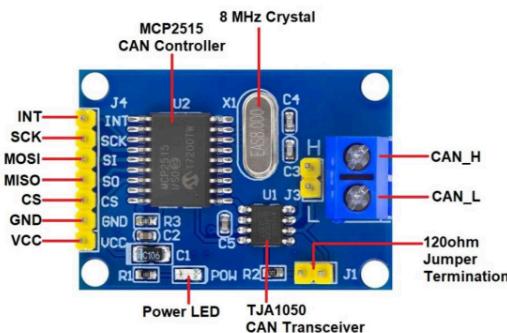
Hình 2.25 CAN Controller MCP2515

Các tính năng chính của MCP2515:

- Triển khai CAN V2.0B ở tốc độ 1 Mb/giây.
- Dữ liệu chuẩn và mở rộng cũng như khung dữ liệu từ xa.
- Hai bộ đệm nhận với bộ lưu trữ tin nhắn được ưu tiên.
- Sáu bộ lọc 29 bit và hai mặt nạ 29 bit.
- Lọc byte dữ liệu trên hai byte dữ liệu đầu tiên, áp dụng cho các khung dữ liệu chuẩn.
- Ba bộ đệm truyền với các tính năng ưu tiên và hủy bỏ.
- Giao diện SPI tốc độ cao (10 MHz). SPI chế độ 0,0 và 1,1.
- Chế độ One-Shot đảm bảo việc truyền tin nhắn chỉ được thực hiện một lần.
- Chân Clock Out có bộ chia tần số thể lập trình, có thể được sử dụng làm nguồn xung nhịp cho các thiết bị khác.
  - Chân đầu ra ngắn có thể lựa chọn.
  - Công nghệ CMOS công suất thấp. Hoạt động từ 2,7V-5,5V.

– MCP2515 có năm chế độ hoạt động: Chế độ cấu hình, Chế độ bình thường, Chế độ ngủ, Chế độ chi nghe và Chế độ lặp lại. Nó bao gồm mặt nạ chấp nhận và bộ lọc để lọc các tin nhắn không mong muốn, giảm chi phí của MCU máy chủ. Mô-đun CAN xử lý tất cả các chức năng để truyền và nhận tin nhắn trên bus CAN. Tin nhắn được truyền bằng cách tái bộ đệm tin nhắn và thanh ghi điều khiển thích hợp, với quá trình truyền được bắt đầu thông qua giao diện SPI hoặc các chân cho phép truyền. Trạng thái và lỗi có thể được kiểm tra bằng cách đọc các thanh ghi thích hợp.

### Cấu hình chân



Hình 2.26 Cấu hình chân CAN Controller MCP2515

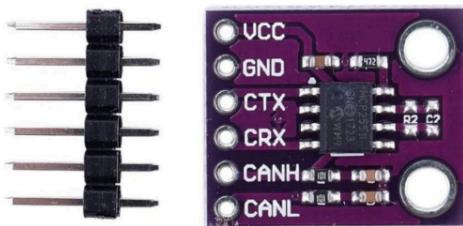
Các chân của MCP2515 gồm:

- TXCAN: Truyền chân đầu ra đến bus CAN.
- RXCAN: Nhận chân đầu vào từ bus CAN.
- CLKOUT: Chân ra xung nhịp với Prescaler có thể lập trình.
- TXORTS: Bộ đệm truyền TXB0 Request-to-Send; 100 kΩ kéo lên bên trong VDD.
- X1RTS: Bộ đệm truyền TXB1 Request-to-Send; 100 kΩ kéo lên bên trong VDD.
- TX2RTS: Bộ đệm truyền TXB2 Request-to-Send; 100 kΩ kéo lên bên trong VDD.
- OSC1: Đầu vào dao động.
- OSC2: Đầu ra dao động.
- Vss: Tham chiếu mặt đất cho chân logic và I/O.

- RX1BF: Chân ngắt RXB1 của bộ đệm nhận hoặc đầu ra kỹ thuật số mục đích chung.
- RX0BF: Chân ngắt RXB0 của bộ đệm nhận hoặc đầu ra kỹ thuật số mục đích chung.
- INT: Chân ra ngắt.
- SCK: Chân đầu vào xung nhịp cho giao diện SPI.
- SI: Chân đầu vào dữ liệu cho giao diện SPI.
- SO: Chân dữ liệu đầu ra cho giao diện SPI.
- CS: Chân đầu vào chọn chip cho giao diện SPI.
- RESET: Thiết bị hoạt động ở mức thấp Thiết lập lại đầu vào.
- VDD: Nguồn cung cấp dương cho chân logic và /0.
- EP: Tấm tản nhiệt lợp ra, kết nối với Vss.
- NC: Không có kết nối bên trong.

### 2.7.2. Transceiver 2551

MCP2551 là bộ thu phát CAN tốc độ cao đóng vai trò là giao diện giữa bộ điều khiển giao thức CAN và bus vật lý. Nó cung cấp khả năng truyền và nhận khác biệt cho bộ điều khiển giao thức CAN. Nó hoàn toàn tương thích với tiêu chuẩn ISO-11898, bao gồm các yêu cầu 24V và hoạt động ở tốc độ lên đến 1 Mb/giây. MCP2551 phù hợp với cả hệ thống 12V và 24V. Nó có trong gói PDIP 8 chân.



Hình 2.27 CAN Transceiver 2551

#### Các tính năng chính của MCP2551

- Hỗ trợ tốc độ lên đến 1 Mb/giây.
- Triển khai các yêu cầu về lớp vật lý theo tiêu chuẩn ISO-11898.
- Phù hợp với hệ thống 12V và 24V.

- Độ dốc được điều khiển bên ngoài để giảm phát xạ RFI.
- Phát hiện lỗi tiếp đất (Chiếm ưu thế vĩnh viễn) trên đầu vào TXD.
- Đặt lại khi bật nguồn và bảo vệ điện áp giật.
- Một nút không có nguồn hoặc sự kiện giảm sẽ không làm nhiễu bus CAN.
- Hoạt động chờ dòng điện thấp.
- Bảo vệ chống hư hỏng do điều kiện ngắn mạch (điện áp pin dương hoặc âm).
- Bảo vệ chống lại các xung điện áp cao.
- Bảo vệ tự động tắt nhiệt.
- Có thể kết nối tối đa 112 nút.
- Khả năng chống nhiễu cao do triển khai bus vi sai.
- Đầu ra CAN của MCP2551 có thể điều khiển tải tối thiểu  $45\Omega$ , cho phép kết nối tối đa 112 nút. Nó có dải điện áp hoạt động từ 4,5V đến 5,5V và dải nhiệt độ môi trường hoạt động từ -40°C đến +125°C.

## 2.8. Các linh kiện cần thiết cho cơ cấu chấp hành

### 2.8.1. Quạt thông gió

Quạt tản nhiệt trong máy hàn đóng vai trò thiết yếu trong việc duy trì giới hạn nhiệt độ hoạt động cho các linh kiện công suất, giúp hạn chế hiện tượng quá nhiệt và gia tăng hiệu suất tổng thể. Với thiết kế khung kim loại chắc chắn và cánh quạt chịu nhiệt cao, quạt thông gió 220V kích thước  $120 \times 120$  mm thường được chọn để lắp đặt trong những bộ nguồn hay tủ điều khiển có yêu cầu khắt khe về độ bền. Bên cạnh yêu tố dòng điện ổn định cùng lưu

lượng gió lớn, độ ồn thấp cũng là điểm mấu chốt để đảm bảo môi trường làm việc êm ái và bền vững cho các thiết bị điện–cơ khí.



Quạt máy hàn - quạt thông gió 220v kích thước 12x12cm

Hình 2.28 Quạt thông gió

#### Thông Số kỹ thuật:

- Điện Áp 220v
- Kích Thước 12x12cmx3.8cm
- Dùng để làm mát máy móc thiết bị
- Thông gió phòng ăn phòng ngủ...
- Quạt Tân Nhiệt ORIX cao cấp
- Các thông số của quạt ORIX 220VAC
- Điện áp: 200V - 240V
- Dòng điện: 0.14A
- Kích thước: 120x120x38mm
- Tốc độ: 2800-3250rpm
- Cánh quạt làm bằng nhựa Plastic cao cấp
- Khung quạt làm bằng kim loại: chắc chắn

#### 2.8.2. Module Relay 5V

Module relay 5V là một trong những linh kiện điện tử quan trọng thường được sử dụng trong các hệ thống điều khiển, tự động hóa. Với khả năng đóng–ngắt các tải công suất cao hoặc cách ly điện áp, module relay 5V giúp đảm bảo an toàn và tăng tính linh hoạt cho các ứng dụng công nghiệp lẫn dân dụng. Nhờ tương thích với dòng điện áp 5V, thiết bị này dễ

dùng tích hợp với mạch vi điều khiển (Arduino, PIC, AVR, Raspberry Pi...), cho phép điều khiển các thiết bị như đèn, motor hay máy bơm bằng tín hiệu số một cách đơn giản.



Hình 2.29 Module relay 5V - 2 kênh

Module được kết nối với các board điều khiển bằng 4 chân header như sau:

- VCC cung cấp nguồn cho các opto.
- GND kết nối với GND của board điều khiển.
- IN1 và IN2 dùng để điều khiển relay 1 và relay 2, tích cực mức thấp

Ngoài ra còn một 3 chân header được dùng để cấp nguồn cho relay, header này sẽ có một jumper dùng để kết nối chân VCC với chân RY\_VCC mục đích dùng chung nguồn VCC (5V) từ header 4 chân cho relay, thông thường jumper được nối lại với nhau. Nếu như muốn cách ly tín hiệu điều khiển với nguồn cấp cho relay thì có thể bỏ jumper này ra và cấp nguồn riêng 5V cho chân RY\_VCC.

### 2.8.3. Mạch ổn áp 5V và 3.3V

#### 2.8.3.1. Mạch ổn áp 5V

Mạch giảm áp XL4015 (5A) là mô-đun chuyển đổi điện áp một chiều dạng “Buck” (hạ áp), cho phép giảm điện áp đầu vào xuống mức thấp hơn theo nhu cầu sử dụng. Với khả năng cung cấp dòng tải lên đến 5 A, mạch XL4015 thích hợp cho nhiều ứng dụng như cấp nguồn cho các thiết bị nhúng, vi điều khiển, động cơ DC hoặc đèn LED công suất nhỏ. So với các mạch tuyến tính, mô-đun XL4015 có hiệu suất cao hơn, ít tỏa nhiệt hơn và dễ dàng chỉnh điện áp ra nhờ biến trở tích hợp.



Hình 2.30 Mạch giảm áp XL4015 (5A)

#### Thông số kỹ thuật

- IC chính: XL4015.
- Tích hợp led báo điện áp đầu ra và nhôm tản nhiệt cho IC chính.
- Điện áp đầu vào: 8~36VDC
- Điện áp đầu ra: 1.25 ~ 32VDC.
- Dòng đầu ra tối đa: 5A
- Hiệu suất : 96%
- Tần số xung: 180KHz
- Tích hợp Mosfet đóng ngắt tần số cao.
- Maximum Duty Cycle: 100%
- Minimum Drop Out: 0.3VDC
- Nhiệt độ làm việc : -40 ~ 125 độ C
- Kích thước: 54 x 23 x 18mm

#### 2.8.3.2. Mạch ổn áp 3.3V

Mạch giảm áp DC mini 3A là một bộ chuyển đổi DC-DC dạng Buck (hạ áp) với kích thước nhỏ gọn, thuận tiện để tích hợp vào nhiều dự án và thiết bị điện tử. Nó cho phép chuyển điện áp đầu vào ( $V_{in}$  có thể cao hơn) xuống điện áp đầu ra ( $V_{out}$ ) thấp hơn một cách linh hoạt, qua đó cấp nguồn cho vi điều khiển, mô-đun cảm biến, đèn LED hoặc motor DC công suất nhỏ.



Hình 2.31 Mạch nguồn giảm áp DC mini 3A

#### Thông số kỹ thuật

- Điện áp đầu vào: 4.5 ~ 28V
- Điện áp đầu ra: 0.8 ~ 20V (có thể điều chỉnh)
- Dòng ra: Tối đa 3A
- Dòng liên tục 2A
- Tần số chuyển đổi: 1MHz (bình thường), 1.5Mhz (tối đa)
- Độ ngọt đầu ra: < 30mV
- Hiệu suất chuyển đổi: 96% (tối đa)
- Nhiệt độ hoạt động: (-40°C đến +85°C)
- Kích thước: 22 x 17 x 4 mm
- Trọng lượng: 2g

#### 2.8.4. Nguồn tổ ong 220V AC – 24V DC

Nguồn tổ ong 24V là bộ chuyển đổi điện áp từ lưới điện xoay chiều (AC) 220V xuống điện áp một chiều (DC) ở mức 24V định mức. Nhờ sử dụng công nghệ switching (chuyển mạch xung), nó có kích thước nhỏ gọn, hiệu suất cao hơn so với nguồn tuyển tĩnh. Được ứng dụng rộng rãi trong các hệ thống điều khiển công nghiệp, đèn LED, máy in 3D, bơm huỳnh quang, camera giám sát, và nhiều thiết bị điện tử khác.



Hình 2.32 Nguồn tổ ong 220VAC-24VDC

#### Thông số kỹ thuật

- Điện áp đầu vào: 110V - 220VAC (50-60Hz)
- Điện áp đầu ra: 24VDC
- Dòng điện: Có nhiều loại với các mức dòng khác nhau, phổ biến nhất là 1A, 5A, 10A, và 20A.

**Công suất:** Tùy thuộc vào loại, ví dụ:

- Nguồn 24V - 1A: 25W
- Nguồn 24V - 5A: 120W
- Nguồn 24V - 20A: 480W

**Kích thước:** Thay đổi theo công suất, ví dụ:

- Nguồn 5A có kích thước khoảng 199 x 97 x 42 mm
- Nguồn 20A có kích thước khoảng 215 x 113 x 50 mm
- Trọng lượng: Khoảng từ 500g đến hơn 1kg tùy thuộc vào công suất.

#### Cấu tạo

Nguồn tổ ong bao gồm các thành phần chính sau:

- Biến áp xung: Sử dụng lõi ferit, giúp hoạt động hiệu quả ở tần số cao.
- Cầu chì: Bảo vệ mạch khỏi ngắn mạch.
- Đổi chinh lưu: Chuyển đổi AC thành DC.

- Sò công suất: Điều khiển quá trình đóng/cắt điện.
- Tụ lọc: Tích trữ năng lượng và ổn định điện áp đầu ra.
- IC quang và IC TL431: Duy trì điện áp ổn định cho tải tiêu thụ.

#### Nguyên lý hoạt động

Khi nguồn AC được cấp vào, biến áp sơ cấp sẽ được điều khiển đóng/cắt bởi sò công suất, tạo ra từ trường biến thiên. Từ trường này sẽ sinh ra điện áp ở cuộn thứ cấp. Điện áp này sau đó được chỉnh lưu và lọc qua các tụ điện để cung cấp dòng điện DC ổn định cho các thiết bị sử dụng<sup>245</sup>.

#### Ưu điểm

Nguồn tần số có nhiều ưu điểm so với các loại máy biến áp truyền thống:

- Kích thước nhỏ gọn và nhẹ, giúp tiết kiệm không gian.
- Hiệu suất cao, với khả năng hoạt động tốt trong điều kiện khắc nghiệt.
- Giá thành hợp lý, phù hợp với nhu cầu sử dụng đa dạng của người dùng.

#### 2.8.5. Động cơ máy bơm hơi 12V

Động cơ 775 là dòng motor DC dạng vỏ kim loại hình trụ, cấp công suất trung bình, thường dùng trong các ứng dụng cần mô-men xoắn tương đối lớn, tốc độ quay cao. Kích thước tiêu chuẩn: Đường kính thân ~42mm, chiều dài thân ~66–70mm (không tính trực). Trục đầu ra: Đường kính phô biến 5mm, phù hợp lắp pulley, bánh răng, khớp nối. Lỗ thông gió ở mặt sau, giúp tản nhiệt khi hoạt động ở tải cao.



Hình 2.33 Động cơ 775 12VDC 15000rpm

#### Thông số kỹ thuật

- Điện áp định mức: 12VDC
- Tốc độ: Khoảng 15000 vòng/phút
- Công suất: Thường khoảng 150W (có thể dao động tùy theo nhà sản xuất)

- Dòng không tải: Khoảng 1.2A ở 12V
- Dòng điện roto bị khóa: Khoảng 4.7A
- Kích thước: Kích thước cụ thể có thể khác nhau nhưng thường là khoảng 70mm chiều dài và 40mm đường kính.
- Trọng lượng: Khoảng từ 300g đến 500g tùy thuộc vào thiết kế.

### Cấu tạo

Động cơ này thường được trang bị:

- Bạc đạn: Có thể có một hoặc hai bạc đạn, giúp giảm ma sát và tăng tuổi thọ.
- Quạt làm mát: Một số phiên bản có quạt làm mát để duy trì hiệu suất hoạt động.
- Trục dài: Thích hợp cho các ứng dụng cần lắp đặt linh hoạt.

### Nguyên lý hoạt động

Động cơ hoạt động dựa trên nguyên lý điện từ, khi dòng điện chạy qua cuộn dây bên trong, nó tạo ra từ trường làm quay rotor. Tốc độ quay của rotor phụ thuộc vào điện áp cấp vào và tải trọng.

### Ứng dụng

Động cơ 775 được sử dụng rộng rãi trong:

- Chế tạo robot và mô hình học tập.
- Máy khoan mini, máy cắt, và máy bơm nước.
- Các dự án DIY và chế tạo đồ chơi điện tử.

### Ưu điểm

- Hiệu suất cao với tốc độ lớn.
- Kích thước nhỏ gọn, dễ dàng lắp đặt trong nhiều ứng dụng.
- Giá thành hợp lý, phù hợp cho nhiều dự án khác nhau.

### 2.8.6. Dây Jumper



Hình 2.34 Dây Jumper

Dây Jumper là loại dây điện có đầu cắm (connector) nhô, dùng để kết nối các linh kiện, module, vi điều khiển với nhau trên breadboard, hoặc cắm trực tiếp vào các chân header (GPIO) trên bo mạch. Nhờ tính linh hoạt và khả năng tháo lắp nhanh, dây Jumper được ưa chuộng cho các thử nghiệm mạch, nguyên mẫu (prototype), hoặc tìm hiểu vi mạch mà không cần hàn cố định.

#### Cách sử dụng:

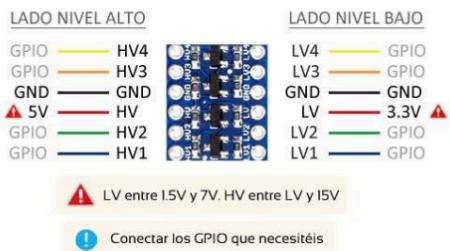
Xác định kiểu chân cần kết nối (đực hay cái), từ đó chọn loại dây Jumper đực-đực, cái-cái, hay đực-cái phù hợp. Ghim chắc chắn dây vào đúng cực (+, -) hoặc đúng chân tín hiệu để tránh chập mạch. Khi chạy thử mạch, nếu không thấy tín hiệu hoạt động, kiểm tra lại các dây Jumper xem có bị lỏng hay cắm sai vị trí trên breadboard.

#### Lưu ý:

Dây Jumper chỉ phù hợp cho tín hiệu điện áp/thấp (ví dụ 3.3V, 5V, 12V). Với điện áp cao hoặc dòng lớn, cần bộ dây dẫn chuyên dụng có tiết diện lớn hơn và đầu cắm chắc chắn hơn. Tránh uốn dây quá mạnh hoặc giật mạnh làm hỏng đầu connector. Bố trí đường dây gọn gàng, tránh cắm chéo, đan nhiều tangle gây khó quan sát, dễ nhầm lẫn.

### 2.8.7. Level Shifter

Mạch level shifter là một thành phần trong điện tử ~~được sử dụng để chuyển đổi~~ mức độ logic (voltage level) giữa hai hệ thống hoặc thiết bị khác nhau. Trong một số trường hợp, các thành phần hoặc hệ thống ~~có thể sử dụng mức độ logic khác nhau~~, và để chúng ~~có thể~~ tương tác với nhau, một mạch level shifter được sử dụng để đảm bảo rằng tín hiệu được hiểu đúng.



Hình 2.35 Level Shifter

Các mức độ logic thường được đo bằng điện áp, và mạch level shifter sẽ thực hiện chuyển đổi giữa các mức độ này.

## 2.9. Nền tảng thiết kế giao diện Figma

Figma là công cụ thiết kế giao diện (UI/UX) dựa trên nền tảng web, cho phép nhiều người cùng cộng tác, chỉnh sửa thiết kế theo thời gian thực. Figma cho phép người dùng có thể sử dụng trực tiếp trên trình duyệt mà không bắt buộc cài đặt phần mềm, đồng thời vẫn có phiên bản ứng dụng desktop cho các hệ điều hành phổ biến.



Hình 2.36 Nền tảng thiết kế giao diện người dùng Figma

### Tính năng nổi bật

- Cộng tác trực tuyến: Giống như Google Docs, Figma cho phép nhiều người dùng cùng làm việc trên một tài liệu trong thời gian thực. Điều này giúp tăng cường khả năng hợp tác giữa các designer và developer.
- Kho tài nguyên phong phú: Figma cung cấp một kho plugins và tài nguyên phong phú, hỗ trợ người dùng trong quá trình thiết kế và phát triển sản phẩm.
- Thiết kế đa nền tảng: Người dùng có thể truy cập Figma từ bất kỳ thiết bị nào có kết nối internet mà không cần cài đặt phần mềm, giúp tiết kiệm thời gian và công sức.

### Ứng dụng của Figma

Figma được sử dụng rộng rãi trong nhiều lĩnh vực khác nhau:

- Thiết kế UI/UX: Phục vụ cho việc thiết kế giao diện cho ứng dụng web và di động.
- Marketing: Tạo các ấn phẩm truyền thông như tờ rơi, áp phích và đồ họa thông tin.
- Chế tạo prototype: Giúp các nhà phát triển kiểm tra ý tưởng và tương tác trước khi đưa vào sản xuất thực tế.

### Ưu điểm

- Giao diện thân thiện: Dễ dàng sử dụng ngay cả với những người mới bắt đầu.
- Tính linh hoạt cao: Hỗ trợ làm việc từ xa và cộng tác hiệu quả.
- Chi phí hợp lý: Có phiên bản miễn phí với nhiều tính năng cơ bản, phù hợp cho cá nhân và nhóm nhỏ.

### 2.10. Arduino IDE



5 Hình 2.37 Phần mềm Arduino IDE

### Arduino IDE - Môi trường lập trình đơn giản và mạnh mẽ

Arduino IDE (Integrated Development Environment) là một phần mềm mã nguồn mở, được sử dụng rộng rãi để viết và biên dịch mã cho các bo mạch Arduino. Điểm mạnh của Arduino IDE là sự đơn giản, dễ sử dụng, phù hợp cho cả người mới bắt đầu và người có kinh nghiệm.

#### Các thành phần chính của Arduino IDE

– Phần cứng: Arduino cung cấp một loạt các bo mạch được thiết kế sẵn với nhiều loại cảm biến và linh kiện khác nhau. Hiện nay, có hơn 300.000 board mạch Arduino có sẵn trên thị trường, đáp ứng nhu cầu đa dạng của người dùng.

– Phần mềm: Arduino IDE là phần mềm giúp người dùng tận dụng tối đa các cảm biến và linh kiện trên bo mạch Arduino. Phần mềm này cung cấp một môi trường lập trình thân thiện, cho phép người dùng viết mã, biên dịch và tải lên bo mạch một cách dễ dàng.

#### **Ưu điểm Arduino IDE**

– Mã nguồn mở: Arduino IDE là phần mềm mã nguồn mở, cho phép người dùng tự do sử dụng, chỉnh sửa và phát triển.

– Dễ sử dụng: Giao diện của Arduino IDE rất trực quan và dễ hiểu, ngay cả người không có kiến thức kỹ thuật sâu rộng cũng có thể bắt đầu lập trình với Arduino.

– Hỗ trợ đa nền tảng: Arduino IDE có thể chạy trên nhiều hệ điều hành khác nhau như Windows, macOS và Linux.

– Cộng đồng lớn mạnh: Arduino có một cộng đồng người dùng rất lớn và nhiệt tình, sẵn sàng chia sẻ kiến thức và giúp đỡ lẫn nhau.

– Tài liệu phong phú: Arduino cung cấp rất nhiều tài liệu hướng dẫn, ví dụ và thư viện, giúp người dùng dễ dàng học tập và làm việc với Arduino.

#### **Cách thức hoạt động của Arduino IDE**

– Viết mã: Người dùng viết mã chương trình bằng ngôn ngữ C/C++ trên Arduino IDE.

– Biên dịch: Arduino IDE biên dịch mã nguồn thành mã máy, có thể chạy trên bo mạch Arduino.

– Tải lên: Người dùng kết nối bo mạch Arduino với máy tính và sử dụng Arduino IDE để tải mã máy lên bo mạch.

– Chạy chương trình: Sau khi mã máy được tải lên, bo mạch Arduino sẽ thực hiện chương trình.

#### **Ứng dụng của Arduino**

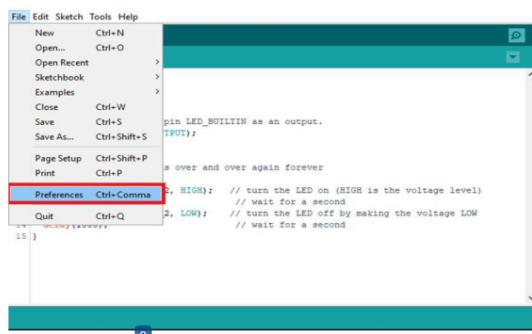
Arduino được sử dụng rộng rãi trong nhiều lĩnh vực khác nhau, bao gồm:

– IoT (Internet of Things): Kết nối các thiết bị với internet và điều khiển chúng từ xa.

- Robot: Xây dựng và điều khiển robot.
- Điện tử DIY: Tạo ra các thiết bị điện tử tự chế.
- Giáo dục: Dạy và học về lập trình và điện tử.

### 2.10.1. Cài đặt bổ sung ESP 32 board cho Arduino IDE

Mở arduino IDE chọn File>Preferences



Hình 2.38 Cài đặt bổ sung ESP32 board cho arduino IDE

Nhập [https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json) vào trường “additional Board Manager URLs” trong hình sau đó nhấn “OK”.



Hình 2.39 Cài đặt bổ sung ESP32 board cho arduino IDE

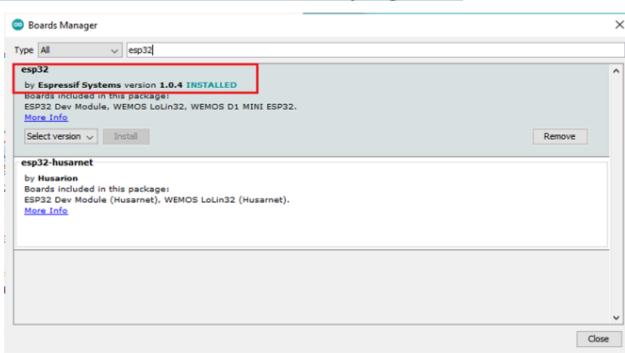
Mở Board Manager > Tools > Board > Boards Manager...



Hình 2.40

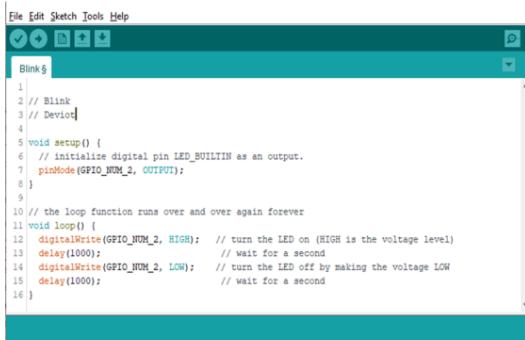
### 8 Cài đặt bổ sung ESP32 board cho arduino IDE

8 Tim ESP32 và nhấn cài đặt cho “ESP32 by Espressif”



Hình 2.41 Cài đặt bổ sung ESP32 board cho arduino IDE

Thử một chương trình với ESP32.Tìm đến example Blink của Arduino



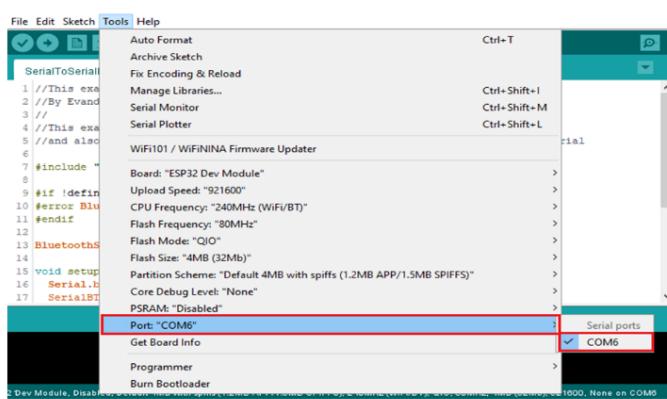
Hình 2.42 Cài đặt bổ sung ESP32 board cho arduino IDE

Chọn board là **ESP32 Dev Module** hoặc chọn **DOIT ESP32 DEVKIT V1**



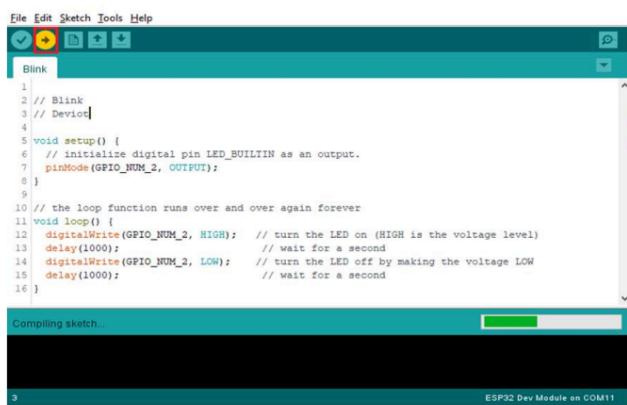
Hình 2.43 Cài đặt bổ sung ESP32 board cho arduino IDE

Chọn Port ,nếu như đã cắm ESP32 vào máy tính mà bạn không thấy cổng COM trong Arduino IDE của mình thì bạn hãy cài đặt **CP210x USB UART Bridge VCP Drivers** .



Hình 2.44 Cài đặt bổ sung ESP32 board cho arduino IDE

Upload chương trình chạy , quan sát LED xanh trên Board.



Hình 2.45 Cài đặt bổ sung ESP32 board cho arduino IDE

## 2.11. Cube IDE

### 2.11.1. Tổng quan về STM32CubeIDE

STM32CubeIDE là một môi trường phát triển tích hợp (IDE) mạnh mẽ và miễn phí do STMicroelectronics cung cấp, được thiết kế đặc biệt để hỗ trợ việc phát triển ứng dụng nhúng trên vi điều khiển STM32.

#### Các tính năng nổi bật trên STM32CubeIDE

- Dựa trên Eclipse: STM32CubeIDE được xây dựng dựa trên nền tảng Eclipse, một IDE mã nguồn mở phổ biến và mạnh mẽ, cung cấp nhiều tính năng và plugin hữu ích cho việc phát triển phần mềm.
- Hỗ trợ đa nền tảng: STM32CubeIDE có thể chạy trên nhiều hệ điều hành khác nhau như Windows, macOS và Linux, giúp người dùng linh hoạt hơn trong việc lựa chọn nền tảng phát triển.
- Tích hợp STM32CubeMX: STM32CubeIDE tích hợp sẵn STM32CubeMX, một công cụ đồ họa mạnh mẽ cho phép cấu hình vi điều khiển STM32 một cách trực quan và dễ dàng. Với STM32CubeMX, bạn có thể lựa chọn vi điều khiển, thiết lập các chân GPIO, cấu hình các ngoại vi, tạo mã khởi tạo và nhiều hơn nữa.

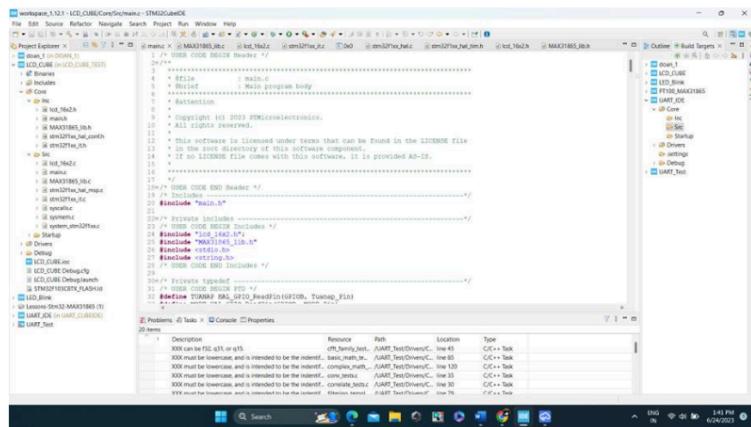
– Trình soạn thảo mã và gỡ lỗi: STM32CubeIDE cung cấp một trình soạn thảo mã mạnh mẽ với nhiều tính năng hỗ trợ như tô màu cú pháp, tự động hoàn thành mã, kiểm tra lỗi cú pháp, giúp bạn viết mã C/C++ một cách nhanh chóng và chính xác. Bên cạnh đó, IDE này cũng tích hợp trình gỡ lỗi mạnh mẽ, cho phép bạn theo dõi và kiểm soát quá trình thực thi chương trình, tìm và sửa lỗi một cách hiệu quả.

– Trình biên dịch và nạp chương trình: STM32CubeIDE tích hợp trình biên dịch GCC, cho phép bạn biên dịch mã nguồn thành mã máy để chạy trên vi điều khiển STM32. IDE này cũng hỗ trợ nạp chương trình vào vi điều khiển thông qua nhiều giao thức khác nhau như ST-LINK, JTAG, SWD.

– Các thư viện và ví dụ: STM32CubeIDE đi kèm với một loạt các thư viện HAL (Hardware Abstraction Layer) và các ví dụ mẫu, giúp bạn dễ dàng trong quá trình phát triển ứng dụng. Các thư viện HAL cung cấp các hàm API đơn giản và dễ sử dụng để tương tác với các ngoại vi của vi điều khiển, trong khi các ví dụ mẫu giúp bạn nhanh chóng làm quen với các tính năng và cách sử dụng của STM32.

– Hỗ trợ RTOS: STM32CubeIDE hỗ trợ phát triển ứng dụng với các hệ điều hành thời gian thực (RTOS) như FreeRTOS, cho phép bạn xây dựng các ứng dụng phức tạp với nhiều tác vụ chạy song song.

– Cập nhật thường xuyên: STMicroelectronics liên tục cập nhật và cải tiến STM32CubeIDE, bổ sung các tính năng mới, cải thiện hiệu suất và sửa lỗi, đảm bảo bạn luôn có một công cụ phát triển mạnh mẽ và hiện đại.



Hình 2.46 Giao diện làm việc trên STM32CubeIDE

## 2.11.2. Cách tạo Project trên STM32CubeIDE

### 1. Mở STM32CubeIDE

Khởi động phần mềm STM32CubeIDE trên máy tính của bạn.

### 2. Tạo Project mới

- Trên thanh menu, chọn **File > New > STM32 Project**.
- Hoặc, bạn có thể nhấp chuột phải vào vùng Project Explorer và chọn **New > STM32 Project**.

### 3. Chọn Target

Trong cửa sổ **Target Selection**, bạn có thể chọn vi điều khiển theo nhiều cách:

- MCU/MPU Selector: Chọn theo dòng vi điều khiển (ví dụ: STM32F1Cx), sau đó tìm kiếm và chọn chip cụ thể (ví dụ: STM32F103C8T6).
- Board Selector: Nếu bạn sử dụng một board mạch phát triển có sẵn (ví dụ: Nucleo, Discovery), bạn có thể chọn board mạch đó để tự động cấu hình các thiết lập phần cứng.
- Part Number Search: Nhập trực tiếp tên chip (ví dụ: STM32F103C8T6) vào ô tìm kiếm.

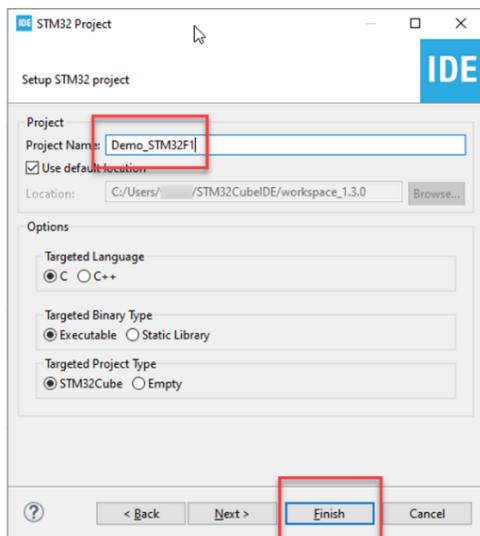
Sau khi chọn được vi điều khiển mong muốn, nhấp vào **Next**.

#### 4. Đặt tên và cấu hình Project

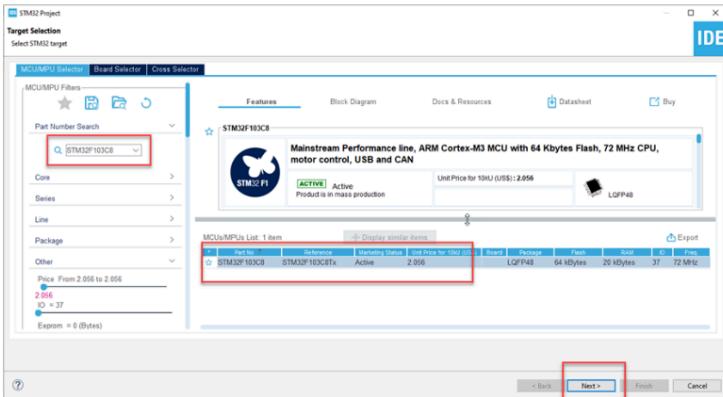
Trong cửa sổ **Project Name**, bạn cần:

- Nhập tên cho project của bạn (ví dụ: MyProject).
- Chọn đường dẫn lưu trữ project.
- Chọn ngôn ngữ lập trình (C hoặc C++).

Nhấp vào **Finish** để hoàn tất quá trình tạo project.



Hình 2.47 Tạo project trên CubeIDE



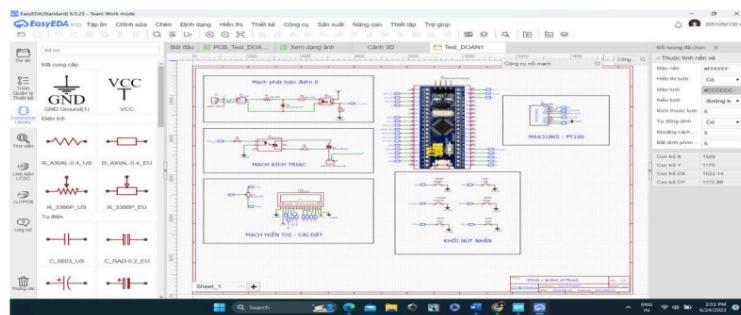
Hình 2.48 Tạo project trên CubeIDE

## 2.12. Easy EDA

### 2.12.1. Tổng quan về phần mềm Easy EDA:

EasyEDA là một phần mềm thiết kế mạch điện trực tuyến (EDA - Electronic Design Automation) mạnh mẽ và dễ sử dụng. Nó cung cấp một môi trường đồ họa để bạn có thể thiết kế mạch điện, mô phỏng, và làm việc với các linh kiện điện tử.

Phần mềm Easy EDA cho phép bạn thiết kế mạch điện một cách trực quan, mô phỏng và mô phỏng ngược các linh kiện trong mạch điện trước khi tiến hành sản xuất, thiết kế bo mạch in PCB một cách chính xác,... Nhìn chung, đây là phần mềm phù hợp và dễ sử dụng đối với những người làm điện tử.

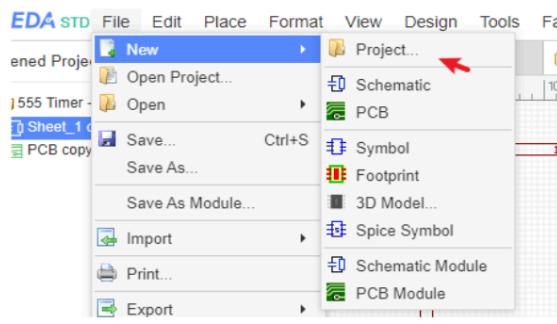


Hình 2.49 Giao diện làm việc trên phần mềm Easy EDA

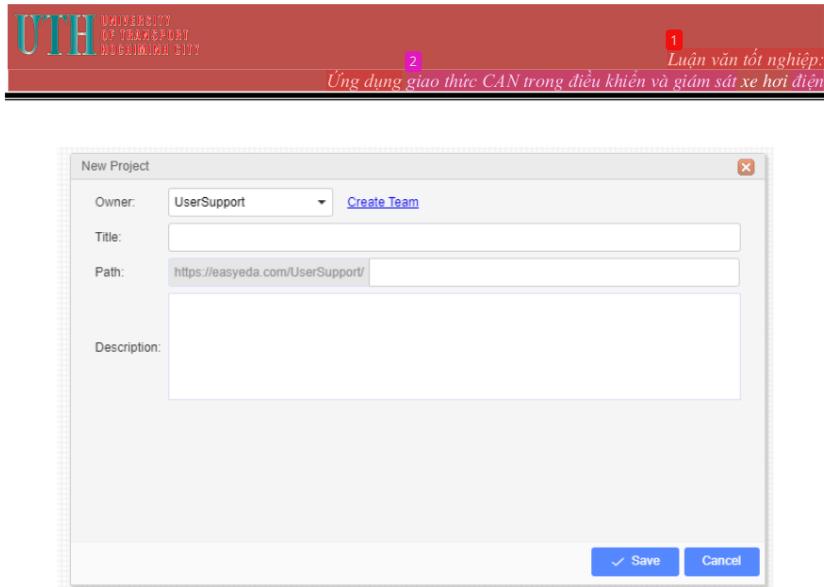
### 2.12.2. Cách tạo một project mới trên phần mềm Easy EDA:

Sau khi mở phần mềm, để tạo một project mới, ta thực hiện các thao tác:

**File > New > Create a new project/Schematic..etc**



Hình 2.50 Hướng dẫn tạo project mới trên Easy EDA



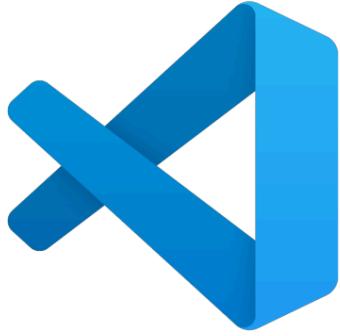
Hình 2.51 Hướng dẫn tạo project mới trên Easy EDA

Trong đó:

- Owner: Đây là mục cho phép thay đổi chủ sở hữu của dự án; ta có thể chuyển chủ sở hữu cho nhóm mà mình đã tham gia.
- Title: Tiêu đề hiển thị của dự án.
- Path: EasyEDA cho phép ta đặt đường dẫn cho dự án, giúp dễ dàng chia sẻ với mọi người.
- Description: Thêm một mô tả ngắn giúp bất kỳ ai bạn chia sẻ dự án này hiểu rõ về nội dung của dự án.

### 2.13. Visual Code

Visual Studio Code là trình soạn thảo mã nguồn (code editor) miễn phí, đa nền tảng, được phát triển bởi Microsoft. Nền tảng này hỗ trợ nhiều ngôn ngữ lập trình khác nhau như JavaScript, TypeScript, Python, C++, Java, C#, Go... đáp ứng các nhu cầu lập trình của người dùng.



Hình 2.52 Nền tảng lập trình Visual Studio Code

4  
**Tính năng nổi bật**

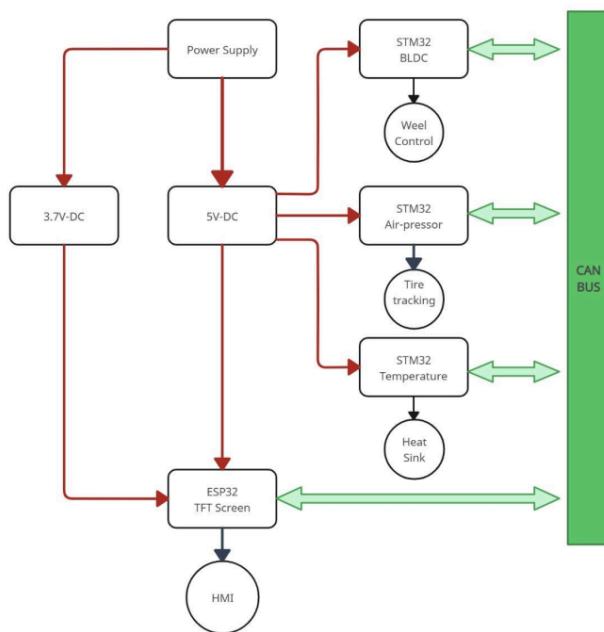
- Giao diện người dùng thân thiện: VS Code có giao diện tối giản nhưng dễ sử dụng, giúp lập trình viên dễ dàng tìm kiếm và quản lý mã nguồn.
- Hỗ trợ mở rộng: Người dùng có thể cài đặt các tiện ích mở rộng từ Marketplace để bổ sung thêm tính năng cho trình soạn thảo.
  - Tích hợp Git: VS Code cho phép người dùng quản lý mã nguồn và thực hiện các thao tác với Git ngay trong ứng dụng.
  - Tính năng gỡ lỗi mạnh mẽ: Hỗ trợ gỡ lỗi cho nhiều ngôn ngữ lập trình với khả năng đặt breakpoint, theo dõi biến và kiểm tra call stack.
  - Tự động hoàn thành mã: Cung cấp khả năng tự động hoàn thành thông minh dựa trên ngữ cảnh của mã đang viết.

**Ưu điểm**

- Nhẹ và nhanh: VS Code có dung lượng nhỏ hơn so với nhiều IDE khác, giúp tiết kiệm tài nguyên hệ thống.
- Cộng đồng lớn: Với sự phát triển mạnh mẽ từ cộng đồng lập trình viên, người dùng có thể dễ dàng tìm thấy tài liệu hướng dẫn và hỗ trợ trực tuyến.
- Miễn phí và mã nguồn mở: Người dùng có thể tải về và sử dụng mà không phải trả phí.

### CHƯƠNG 3. LẬP TRÌNH – THIẾT KẾ VÀ KẾT QUẢ ĐẠT ĐƯỢC.

#### 3.1. Sơ đồ khối – chức năng các khối



Hình 3.1 Sơ đồ khối

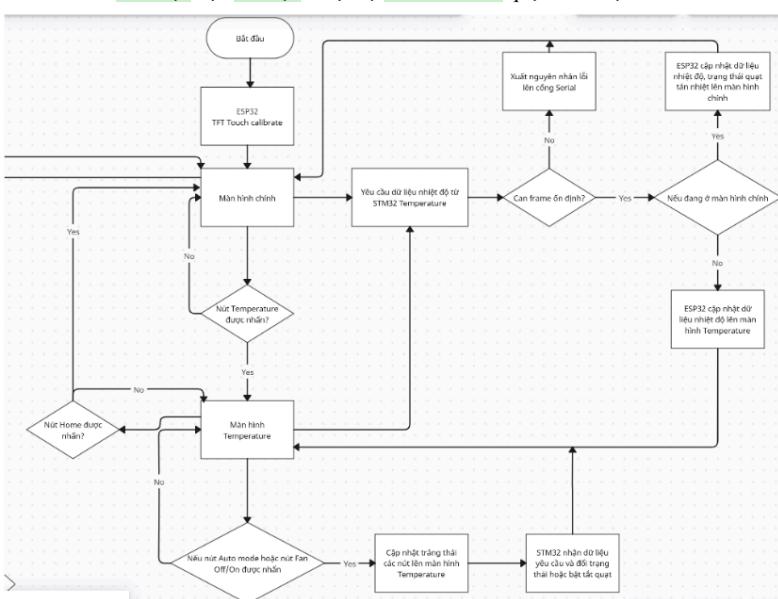
##### Chức năng các khối

- Power supply: Chuyển nguồn 220VAC thành 24VDC cấp cho khối 5V và 3.7V
- 5V-DC: Cấp nguồn 5V cho tất cả MCU, CAN transceiver, CAN controller.
- 3.7VDC: Nguồn riêng cấp cho màn hình TFT ILI9341, giảm nhiễu trong quá trình hoạt động.
- ESP32 TFT Screen: ESP32 thu thập dữ liệu từ các STM32 CAN node đồng thời gửi dữ liệu điều khiển đến từng CAN node thông qua màn hình TFT tương tác với người dùng.

- STM32 Air-pressure: Thu thập thông số áp suất lốp xe bằng cảm biến áp suất không khí và gửi về ESP32 thông qua CAN Bus.
- STM32 BLDC: nhận tín hiệu điều khiển từ ESP32 từ đó điều khiển động cơ thông qua mạch lái động cơ 500w. Khi động cơ hoạt động, liên tục gửi dữ liệu tốc độ động cơ về ESP32 thông qua CAN Bus.
- STM32 Temperature: thu thập dữ liệu nhiệt độ động cơ, thực hiện tản nhiệt qua cát chàp hành quạt tản nhiệt đồng thời gửi dữ liệu nhiệt độ về ESP32 thông qua CAN Bus.
- CAN BUS: là khu vực kết nối CANH và CANL theo chuẩn giao thức CAN với điện trở 120 Ohm ở 2 đầu CAN Bus.

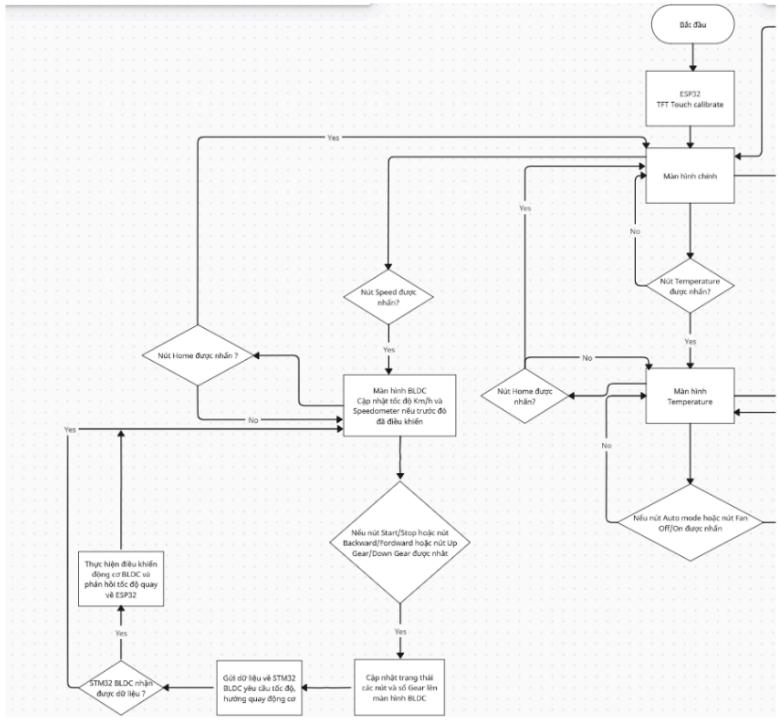
### 3.2. Lưu đồ thuật toán

Lưu đồ cho việc đọc tín hiệu nhiệt độ và điều khiển quạt tản nhiệt.



Hình 3.2 Lưu đồ thuật toán cho khởi kiểm tra nhiệt độ

### Lưu đồ thuật toán khói điều khiển động cơ



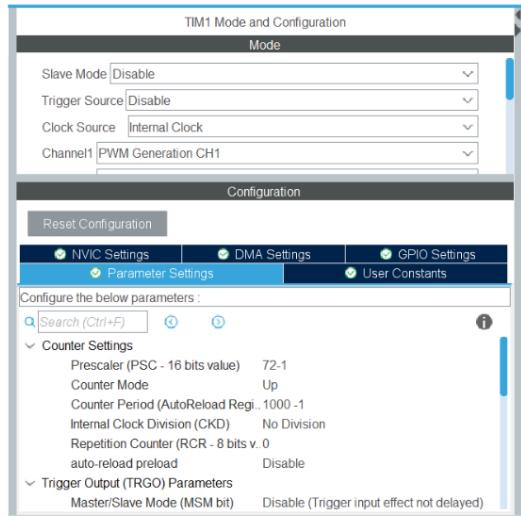
Hình 3.3 Lưu đồ giải thuật cho khói điều khiển động cơ

### 3.3. Cấu hình và lập trình 3 STM32 trên Cube IDE

#### 3.3.1. Cấu hình và lập trình STM32 điều khiển động cơ BLDC

STM32 sử dụng ngoại vi timer1, timer2 và CAN.

Cấu hình Timer1:



Hình 3.4 Cấu hình timer1

Timer 1 được sử dụng để tạo xung PWM với tần số 1000Hz, pinout PA8 được lựa chọn để điều khiển tốc độ quay của động cơ tùy thuộc vào độ rộng ở mức cao và mức thấp với tần số 1 mili giây.

Dưới đây là hàm wheelMoveForward(), hàm này sẽ truyền vào đối số Gear nhận được từ ESP32, từ đó băm xung PWM với khoảng giá trị từ 0 đến 1000.

Ví dụ: Nếu người dùng gửi xuống Gear 8, hàm \_\_HAL\_TIM\_SET\_COMPARE() sẽ truyền vào đối số thứ 3 là  $8 * 100 = 800$ .

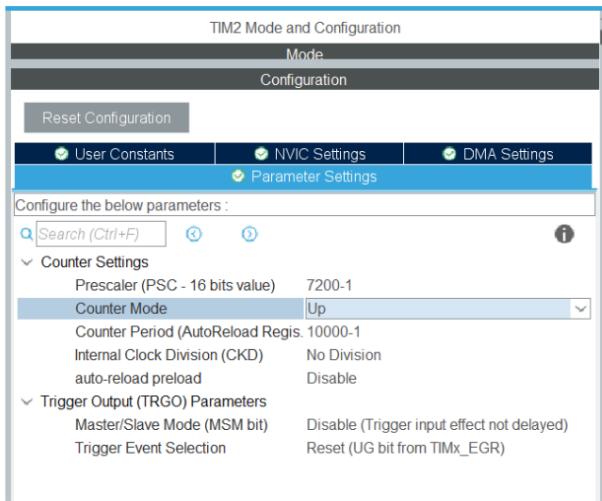
=> Điều này có nghĩa chân PA8 được băm xung tần số 1000HZ với 80% thời gian mức cao và 20% thời gian ở mức thấp.

```
void wheelMoveForward(uint8_t speed)
{
    HAL_TIM_Set_COMPARE(&htim1, TIM_CHANNEL_1, 0);
    HAL_GPIO_WritePin(GPIOA, En_Pin, DISABLE);
    HAL_GPIO_WritePin(GPIOA, Dir_Pin, FORWARD);
    HAL_GPIO_WritePin(GPIOA, En_Pin, ENABLE);

    __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, speed*100);
}
```

Hình 3.5 Mô tả hàm sử dụng timer1 bấm xung PWM.

Cấu hình Timer2:



Hình 3.6 Cấu hình timer2

Timer 2 được sử dụng với mục đích tạo một ngắt timer với chu kỳ 1 giây. Mỗi lần xảy ra interrupt, hệ thống sẽ đếm số lượng ngắt được tạo ra trước đó, từ đó tính toán số vòng quay trên phút (rpm) của động cơ.

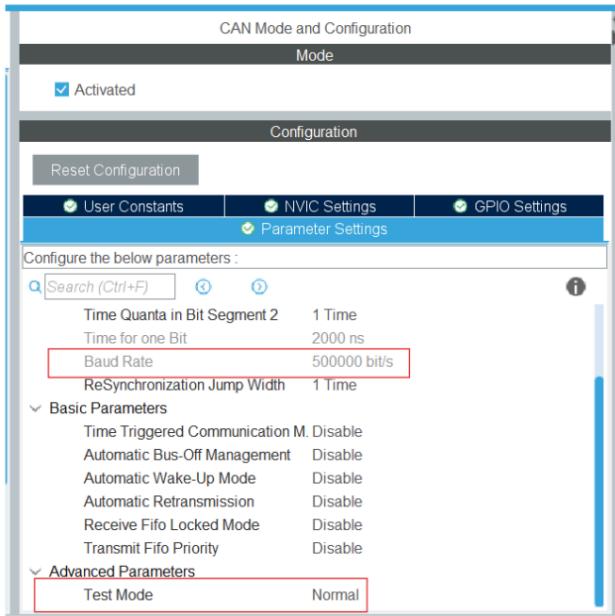
```
1 /*This function calculate the number of pulse per second.*/
2 void TIM2_IRQHandler(void)
3 {
4     /* USER CODE BEGIN TIM2_IRQHandler 0 */
5
6     /* USER CODE END TIM2_IRQHandler 0 */
7     HAL_TIM_IRQHandler(&htim2);
8     /* USER CODE BEGIN TIM2_IRQHandler 1 */
9     rpm = (hall_pulse * 60) / 45;
0
1     // Reset bộ đếm xung cho lần tính tiếp theo
2     hall_pulse = 0;
3
4     //updateLevelSmoothly();
5
6     /* USER CODE END TIM2_IRQHandler 1 */
7 }
```

Hình 3.7 Thuật toán tính số vòng quay trên phút.

Biến hall\_pulse được tăng theo thời gian dựa vào chân Pulse của mạch lái động cơ. Như đã phân tích ở chương 2, vì động cơ có 3 cảm biến hall và có 15 cặp cực, khi động cơ quay hết 1 vòng, ta có được 45 interrupt.

Cấu hình ngoại vi CAN:

Ở module CAN, ta cần cấu hình tốc độ truyền cho CAN là 500000bit/s, tốc độ này phải đồng bộ với tất cả CAN node còn lại trong hệ thống. Mode sử dụng là Normal mode.



Hình 3.8 Cấu hình CAN node

User Constants	NVIC Settings	GPIO Settings
Parameter Settings		
NVIC Interrupt Table	Enabled	Preemption Priority
USB high priority or CAN TX interrupts	<input type="checkbox"/>	0
USB low priority or CAN RX0 interrupts	<input checked="" type="checkbox"/>	0
CAN RX1 interrupt	<input checked="" type="checkbox"/>	0
CAN SCE interrupt	<input type="checkbox"/>	0

Hình 3.9 Kích hoạt interrupt cho CAN receive buffer

Khi nhận được can message từ ESP32, STM32 sẽ xảy ra ngắn, từ đó ta cho phép module CAN nhận thông tin và gửi ngược message (tốc độ quay) về ESP32.

```
96 void HAL_CAN_RxFifo1MsgPendingCallback(CAN_HandleTypeDef *hcan)
97 {
98     error = HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO1, &RxHeader, RxData);
99     if (RxHeader.DLC == 3)
100    {
101        datacheck = 1;
102    }
103 }
```

Hình 3.10 Ngắt FIFO1 của module CAN

Cấu hình CAN filter, bằng việc sử dụng Can filter những mess CAN không liên quan đến từ các node CAN khác sẽ không được STM32 BLDC nhận. Cấu hình bên dưới cho thấy STM32 BLDC node sẽ chỉ nhận những CAN frame có địa chỉ là 0x037.

```
}
```

```
/* USER CODE BEGIN CAN_Init_2 */
CAN_FilterTypeDef canfilterconfig;

canfilterconfig.FilterActivation = CAN_FILTER_ENABLE;
canfilterconfig.FilterBank = 10; // which filter bank to use
canfilterconfig.FilterFIFOAssignment = CAN_FILTER_FIFO1;
canfilterconfig.FilterIdHigh = 0x037<<5;
canfilterconfig.FilterIdLow = 0;
canfilterconfig.FilterMaskIdHigh = 0xFFFF<<5;
canfilterconfig.FilterMaskIdLow = 0x0000;
canfilterconfig.FilterMode = CAN_FILTERMODE_IDMASK;
canfilterconfig.FilterScale = CAN_FILTERSCALE_32BIT;
canfilterconfig.SlaveStartFilterBank = 0; // doesn't matter in

HAL_CAN_ConfigFilter(&hcan, &canfilterconfig);
```

Hình 3.11 Cấu hình CAN filter cho BLDC node.

Cấu hình TX frame cho BLDC node, địa chỉ của BLDC node sẽ là 0x104.

```
TxHeader.DLC = 3; // data length
TxHeader.IDE = CAN_ID_STD;
TxHeader.RTR = CAN_RTR_DATA;

TxHeader.StdId = 0x104; // ID BLDC
```

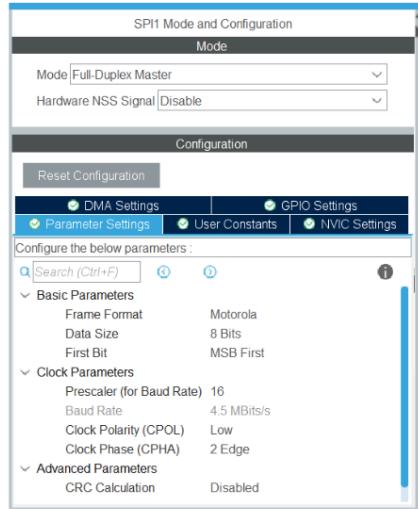
Hình 3.12 Tx frame của BLDC can node.

### 3.3.2. Cấu hình và lập trình STM32 Temperature

STM32 sử dụng ngoại vi giao tiếp SPI và CAN

Cấu hình SPI Master Full-Duplex sử dụng để giao tiếp với MAX-31865 PT100:

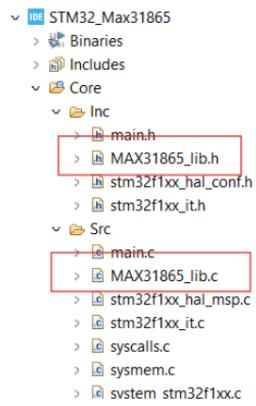
Tốc độ truyền là 4,5Mbits/s, Data size là 1 byte.



Hình 3.13 Cấu hình SPI master

Thư viện MAX31865.

Ta cần add thư viện Max31865 vào cây thư mục của Project.



Hình 3.14 Add thư viện LCD vào Project

Vào Inc để add thư viện “.h” và vào Src để add thư viện “.c”.

Cấu hình ngoại vi CAN, chuẩn bị Tx frame có địa chỉ là 0x446

```
TxHeader.DLC = 2; // data length
TxHeader.IDE = CAN_ID_STD;
TxHeader.RTR = CAN_RTR_DATA;
TxHeader.StdId = 0x446; // ID
```

Hình 3.15 Cấu hình Tx frame cho Temperature node.

Phần cấu hình còn lại tương tự như BLDC Can node.

### 3.4. Cấu hình và lập trình ESP32

#### 3.4.1. Một số thư viện cần thiết

```
#include <SPI.h> => Sử dụng thư viện SPI để giao tiếp với màn hình TFT và MCP2551.
#include <mcp2515.h> => Thư viện sử dụng cho module Can controller MCP2551.
#include <PNGdec.h> => Thư viện dùng để giải mã hex hình PNG hiển thị lên TFT
#include <TFT_eSPI.h> => bộ thư viện API dùng cho màn hình TFT
#include "Free_Fonts.h" => bộ thư viện lưu trữ font chữ dùng cho TFT
#include <JPEGDecoder.h> => Thư viện dùng để đọc định dạng ảnh JPEG trong SD Card.
```

#### 3.4.2. Các hàm quan trọng được viết bởi người dùng.

Hàm trong ngôn ngữ lập trình C là một khối mã lệnh được định nghĩa để thực hiện một nhiệm vụ cụ thể. Việc sử dụng hàm giúp tái sử dụng mã nguồn, tổ chức chương trình rõ ràng hơn và dễ bảo trì.

Tác dụng của hàm trong lập trình:

- Tái sử dụng mã nguồn – Viết một lần, gọi nhiều lần, giúp tránh lặp mã.
- Chia nhỏ chương trình – Giúp chương trình dễ hiểu và dễ quản lý hơn.
- Dễ dàng bảo trì – Khi cần thay đổi thuật toán, chỉ cần sửa trong một hàm thay vì nhiều đoạn mã lặp lại.
- Tăng tính module – Giúp chương trình dễ mở rộng bằng cách thêm các hàm mới.
- Hỗ trợ gỡ lỗi tốt hơn – Tập trung kiểm tra lỗi từng hàm thay vì cả chương trình lớn.

Một số hàm quan trọng:

```
void send_receive_bldc_mess();
void tx_can_frame_temp();
void tx_can_frame_bldc();
void tx_can_frame_psi();
void tft_temp_print();
void fan_rotate();
void updateFanAnimation();
void menu_check_button();
void bldc_check_button();
void bldc_png_default_layout(); // used when
void temp_png_default_layout();
void temp_check_button();
void psi_check_button();
void UpdateSpeedometerEvery50ms();
void UpdateTemperateStatusEvery50ms();
void bldc_png_update_layout_after_touch();
void temp_png_update_layout_after_touch();
//-----For TFT---
```

Hình 3.16 Prototype một số hàm sử dụng trong runtime của ESP32

Code main:

Bằng việc sử dụng các hàm, code main trở nên rất ngắn và dễ dàng bảo trì..

```
void loop() {
    while (menu_scr == 1) {
        menu_check_button();
        if(menu_scr == 0)
        {
            break;
        }
        if(bldc_enable == 1)
        {
            send_receive_bldc_mess();
        }

        send_receive_temp_mess();

        fan_rotate();
        Serial.println("MENU");
    }
    while (bldc_scr == 1)
    {
        bldc_check_button();
        if(menu_scr == 1 && bldc_scr == 0)
        {
            prv_knh = 300;
            bldc_scr_comback = 1;
            break;
        }
        send_receive_temp_mess();
        fan_rotate();
    }
    while(temp_scr == 1)
    {
        temp_check_button();
        if(menu_scr == 1 && temp_scr == 0)
        {
            prv_temperature = 300;
            temp_scr_comback = 1;
            break;
        }
        fan_rotate();
    }
}
```

Hình 3.17 Code main

Giải thích ý nghĩa flow code:

- Biến toàn cục: menu\_scr, bldc\_scr, temp\_scr được sử dụng để chuyển đổi qua lại giữa các thao tác lựa chọn của người dùng trên màn hình cảm ứng TFT.
- Hàm xxx\_check\_button: được sử dụng để liên tục kiểm tra điểm chạm của người dùng trên màn hình TFT, đồng thời gọi đến các hàm như send\_bldc\_mes(), send\_temp\_mess() để truyền-nhận dữ liệu CAN. Cuối cùng gọi đến hàm xxx\_png\_update\_layout\_after\_touch() để cập nhật lại trạng thái các nút nhấn cũng như cập nhật dữ liệu nhiệt độ hoặc tốc độ động cơ.
- Hàm fan\_rotate(): được dùng ở tất cả màn hình, có nhiệm vụ liên tục kiểm tra trạng thái quạt tản nhiệt. Khi quạt ở hệ thống bắt đầu quay, quạt trên màn hình cảm ứng cũng sẽ quay và ngược lại.

### 3.5. Khung truyền data CAN giữa các NODE

#### 3.5.1. Khung truyền CAN giữa ESP32 và STM32 BLDC.

ESP32 cập nhật Tx frame trước khi truyền:

- Can\_id = 0x037, đây là địa chỉ của ESP32 CAN mà STM32 BLDC sẽ nhận
- Can\_dlc = 3, độ rộng dữ liệu là 3 bytes.
- Data[0] = 0x01 yêu cầu khởi động động cơ BLDC, ngược lại thì yêu cầu Tắt.
- Data[1] = 0x01 yêu cầu động cơ chạy tiến về phía trước
- Data[2] = 0xXX cho biết gear mong muốn. Gear từ 0 đến 10

```
void tx_can_frame_bldc() {  
    canMsg.can_id = 0x037; // CAN ID  
    canMsg.can_dlc = 3; // Data length code (number of bytes)  
    if (bldc_enable == 1) {  
        if (bldc_direction == 1) {  
            canMsg.data[0] = 0x01; //start bldc  
            canMsg.data[1] = 0x01; // move forward  
            canMsg.data[2] = bldc_gear1; // gear power speed  
        } else {  
            canMsg.data[0] = 0x01; //start bldc  
            canMsg.data[1] = 0x00; // move backward  
            canMsg.data[2] = bldc_gear1; // gear power speed  
        }  
    } else {  
        canMsg.data[0] = 0x00; // stop bldc  
        canMsg.data[1] = 0x01; // move forward  
        canMsg.data[2] = 0x00; // gear power speed = 0  
    }  
}
```

Hình 3.18 ESP32 setup Tx frame cho BLDC node

STM32 nhận dữ liệu và phản hồi tốc độ động cơ về ESP32.

```
// DẠCH VĂN HỌC VỀ KỸ THUẬT Ô TÔ  
if (RxData[0] != prevRxData[0] || RxData[1] != prevRxData[1] || RxData[2] != prevRxData[2])  
{  
    if (RxData[0] != 0)  
    {  
        if (RxData[1] == 1)  
        {  
            if (RxData[1] != prevRxData[1])  
            {  
                changeLevel((uint8_t)0);  
                wheelMoveForward(0);  
                while (previousValue != 0);  
            }  
            wheelMoveForward(RxData[2]);  
            changeLevel(RxData[2]);  
        }  
        else  
        {  
            if (RxData[1] != prevRxData[1])  
            {  
                wheelMoveBackward(0);  
                changeLevel((uint8_t)0);  
                while (previousValue != 0);  
            }  
            wheelMoveBackward(RxData[2]);  
            changeLevel(RxData[2]);  
        }  
    }  
    else  
    {  
        wheelStop();  
        changeLevel(RxData[2]); // ESP32 should send gear 0 at this time  
    }  
  
    // Cập nhật giá trị cũ  
    prevRxData[0] = RxData[0];  
    prevRxData[1] = RxData[1];  
    prevRxData[2] = RxData[2];  
}
```

Hình 3.19 Thuật toán xử lý dữ liệu điều khiển BLDC

```
datacheck = 0; // Đặt lại cờ?  
  
// Gửi phản hồi qua CAN  
TxData[0] = (previousValue >> 8) & 0xFF; // Byte cao  
TxData[1] = previousValue & 0xFF; // Byte thấp  
  
error = HAL_CAN_AddTxMessage(&hcan, &TxHeader, TxData, &TxMailbox);  
}
```

Hình

Hình 3.20 STM32 BLDC setup data trước khi truyền cho ESP32

### 3.5.2. Khung truyền CAN giữa ESP32 và STM32 Temperature

ESP32 cập nhật Tx frame trước khi truyền

- Can\_id = 0x047, đây là địa chỉ của ESP32 CAN mà STM32 Temperature sẽ nhận
- Can\_dlc = 2, độ rộng dữ liệu là 2 bytes.
- Data[0] = 0x01. Điều khiển quạt tản nhiệt chạy manual mode ngược lại là auto mode

- Data[1] = 0x01 yêu cầu bật quạt.(chỉ có tác dụng ở manual mode).

```
void tx_can_frame_temp() {  
    canMsg.can_id = 0x047; // CAN ID  
    canMsg.can_dlc = 2; // Data length code  
    if (temp_fan_mode == 1) {  
        if (fan_button == 1) {  
            canMsg.data[0] = 0x01; //Manual control  
            canMsg.data[1] = 0x01; //Fan ON  
        } else {  
            canMsg.data[0] = 0x01; //Manual control  
            canMsg.data[1] = 0x00; //Fan OFF  
        }  
    } else {  
        canMsg.data[0] = 0x00; //Auto control ON  
        canMsg.data[1] = 0x00; //Fan OFF  
    }  
}
```

Hình 3.21 ESP32 setup Tx frame cho Temp CAN node

STM32 nhận dữ liệu và phản hồi nhiệt độ của Max31865 về ESP32

```
    readTemperatureNonBlocking();  
    if(RxData[0] == 0)  
    {  
        if(PT100_Temperature > 33)  
        {  
            HAL_GPIO_WritePin(GPIOA, Fan_Relay_Pin, 1);  
        }  
        else  
        {  
            HAL_GPIO_WritePin(GPIOA, Fan_Relay_Pin, 0);  
        }  
    }  
    else  
    {  
        if(RxData[1] == 0)  
        {  
            HAL_GPIO_WritePin(GPIOA, Fan_Relay_Pin, 0);  
        }  
        else  
        {  
            HAL_GPIO_WritePin(GPIOA, Fan_Relay_Pin, 1);  
        }  
    }  
}
```

Hình 3.22 STM32 temperature xử lý dữ liệu nhận được từ ESP32

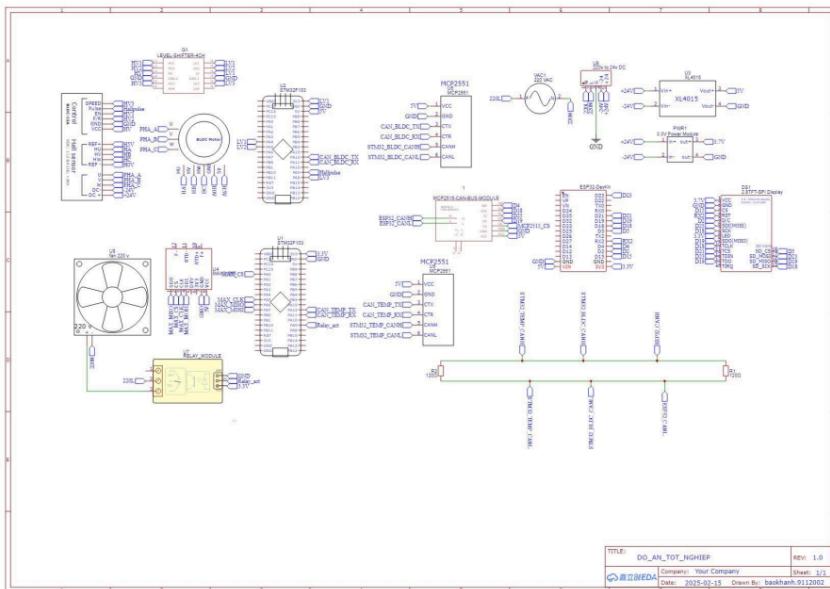
```

if (datacheck)
{
    datacheck = 0;
    temp = (uint32_t)(PT100_Temperature * 100); // Nhận với 100 để giữ 2 chữ số thập phân
    // Tách thành 2 phần 16-bit
    TxData[0] = (uint8_t)(temp >> 8); // Lấy byte cao
    TxData[1] = (uint8_t)(temp & 0xFF); // Lấy byte thấp
    HAL_CAN_AddTxMessage(&hcan, &TxHeader, TxData, 4TxMailbox);
}

```

Hình 3.23 STM32 temperature tách dữ liệu nhiệt độ trước khi truyền.

### 3.6. Sơ đồ nguyên lý toàn bộ hệ thống



Hình 3.24 Sơ đồ nguyên lý

### 3.7. Mô hình hoàn thiện sau khi kết nối

#### 3.7.1. Khởi nguồn cấp

Nguồn tổ ong 220VAC – 24VDC

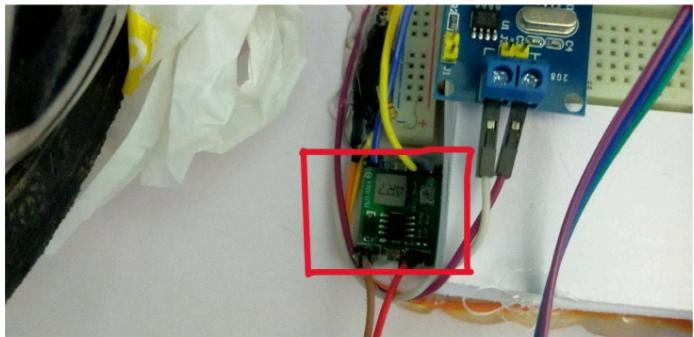


Hình 3.25 Nguồn tổ ong cho đèn tài

Hình nguồn 5V cấp cho tất cả MCU

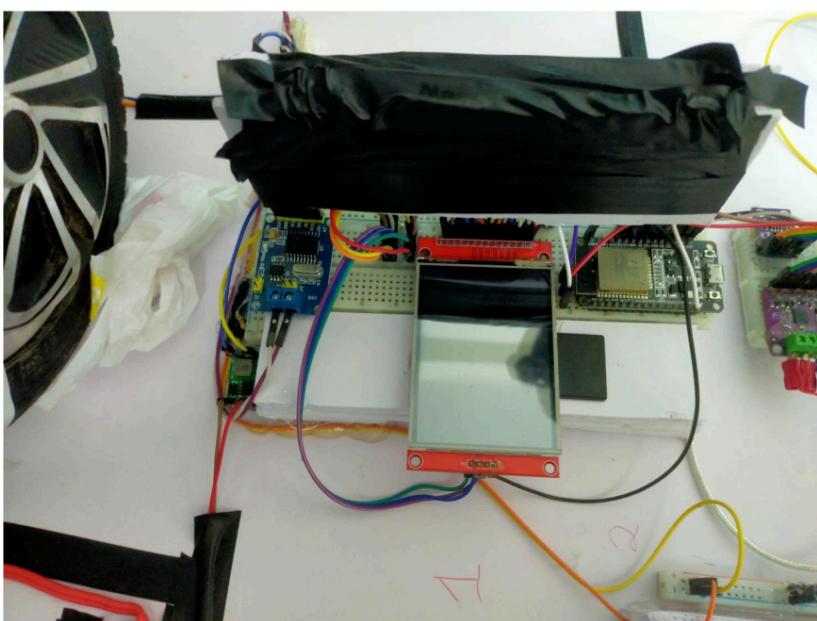


Hình 3.26 Module hạ áp đầu ra 5V cấp cho ESP32 và STM32



Hình 3.27 Nguồn 3.7V cấp cho màn hình TFT ILI3941

### 3.7.2. Khối màn hình cảm ứng giám sát và điều khiển



Hình 3.28 ESP32 – TFT và SD Card – MCP2551

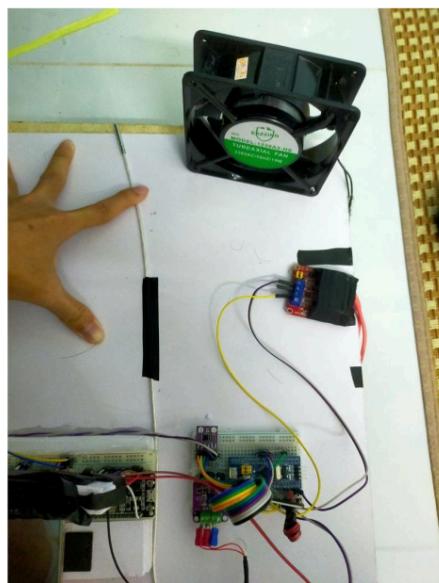
### 3.7.3. Can bus và các khối thu thập điều khiển (STM32)

Can bus:



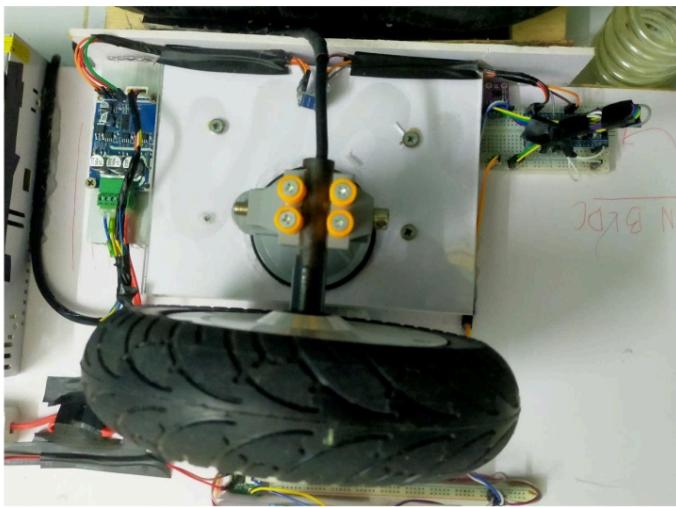
Hình 3.29 CAN bus trên hệ thống

Khối thu thập nhiệt độ và điều khiển quạt STM32 Temperature:



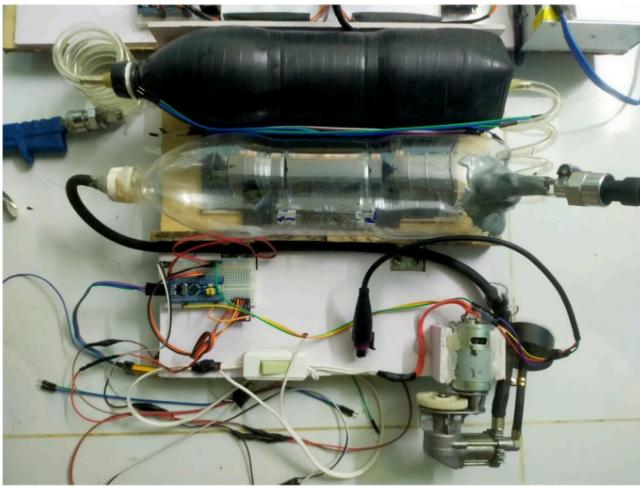
Hình 3.30 Khối thu thập nhiệt độ và điều khiển quạt

Khối điều khiển động cơ STM32 – BLDC – BLDC driver



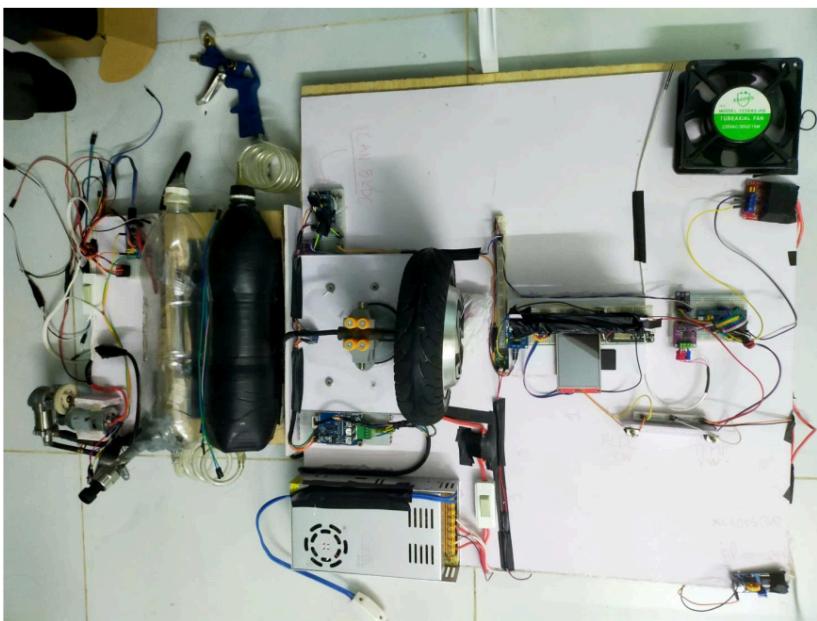
Hình 3.31 Khối điều khiển động cơ BLDC

Hình khối thu thập áp suất lốp xe



Hình 3.32 Khối thu thập áp suất lốp xe STM32 PSI.

### 3.7.4. Tổng quan mô hình



Hình 3.33 Tổng quan mô hình đè tài

## 3.8. Giám sát – Cài đặt – Điều khiển hệ thống xe hơi điện

### 3.8.1. Màn hình chính

Hiển thị:

- Hiển thị nhiệt độ động cơ, trạng thái quạt tản nhiệt
- Hiển thị tốc độ động cơ nếu động cơ đang hoạt động
- Hiển thị thông tin về các thông số khi người dùng nhấn vào nút “i” bên phải trên cùng.

Gồm 4 nút:

- Nút Temperature đưa người dùng giám sát và cài đặt trạng thái quạt tản nhiệt
- Nút Speed đưa người dùng vào chế độ điều khiển động cơ

- Nút PSI đưa người dùng xem trạng thái hiện tại của lốp xe
- Nút "i" ở góc bên phải trên cùng cung cấp thông tin về các chức năng của xe.



Hình 3.34 Màn hình giao diện chính

### 3.8.2. Màn hình điều khiển/giám sát động cơ BLDC

Gồm 5 nút:

- Nút Start/Stop: Khởi động hoặc dừng động cơ
- Nút Backward/Fordward: Đảo chiều động cơ.
- Nút Up gear(U): tăng tốc.
- Nút Down gear(D): giảm tốc
- Nút Home: đưa người dùng về màn hình chính

Hiển thị:

- Hiển thị trạng thái các nút khi người dùng thay đổi.
- Hiển thị tốc độ km/h khi động cơ quay
- Hiển thị bảng đồng hồ tốc độ tương ứng với km/h.

- Hiển thị số gear hiện tại.



Hình 3.35 Màn hình điều khiển động cơ BLDC-gear 2 - di lùi



Hình 3.36 Màn hình điều khiển động cơ BLDC-gear 10 - di lùi



Hình 3.37 Màn hình điều khiển động cơ BLDC – gear 8 - di tiến

### 3.8.3. Màn hình điều khiển/giám sát nhiệt độ động cơ

Gồm 3 nút:

– Nút Auto/Manual: chọn trạng thái điều khiển quạt. Với auto mode nếu nhiệt độ lớn hơn 33 độ C, quạt sẽ tự động quay và ngược lại thì quạt tự động tắt. Với manual mode, nếu người dùng nhấn Fan ON thì quạt quay, Fan OFF thì quạt sẽ tắt.

– Nút Fan On/Off: điều khiển quạt ở trạng thái manual. Ở auto mode, nút này không phản hồi.

– Nút home: đưa người dùng về lại màn hình chính.

Hiển thị:

– Hiển thị trạng thái các nút sau khi người dùng tương tác.

– Hiển thị nhiệt độ động cơ và thanh báo trạng thái



Hình 3.38 Màn hình giám sát nhiệt độ - auto mode - động cơ chưa nóng



Hình 3.39 Màn hình giám sát nhiệt độ - Auto mode - động cơ nóng



Hình 3.40 Màn hình giám sát nhiệt độ - manual mode – fan off



Hình 3.41 Màn hình giám sát nhiệt độ - manual mode – fan on

### 3.8.4. Kiểm tra tính chính xác của các thông số

#### 3.8.4.1. Kiểm tra tốc độ động cơ.

Tốc độ vòng trên phút của động cơ đo được từ cảm biến Hall và thuật toán trên STM32 đo được khi chạy full tốc độ là 350 – 351 vòng trên phút:

Expression	Type	Value
rpm	volatile uint32_t	350
PT100_Temperature		Failed to evaluate expression
psi		Failed to evaluate expression
analog_12bit		Failed to evaluate expression
voltage		Failed to evaluate expression
RxData		Failed to evaluate expression

Hình 3.42 Tốc độ động cơ đo được từ cảm biến Hall

Tốc độ vòng trên phút của động cơ đo được từ máy bắn tốc độ công nghiệp là 360 vòng trên phút:



Hình 3.43 Tốc độ động cơ đo được từ thiết bị công nghiệp

#### 3.8.4.2. Kiểm tra nhiệt độ

Nhiệt độ đo được từ cảm biến trả về 30.1 và từ thiết bị đo nhiệt độ công nghiệp trả về 30.3:



Hình 3.44 Kiểm chứng nhiệt độ giữa cảm biến và thiết bị công nghiệp

## CHƯƠNG 4. KẾT LUẬN

### 4.1. Kết quả đề tài

1  
Kết quả đạt được của đề tài:

- Hiển thị, cài đặt và điều khiển toàn bộ hệ thống thông qua màn hình cảm ứng TFT ILI9341.
- Hiển thị tốc độ động cơ và điều khiển động cơ BLDC.
- Hiển thị - cài đặt - cảnh báo nhiệt độ động cơ.
- Giao thức CAN được sử dụng để giao tiếp giữa các MCU như ESP32 và STM32, đồng thời dữ liệu CAN không bị chia cắt vì đã có CAN filter cho các STM32 node.
- Tốc độ động cơ và nhiệt độ động cơ đã được kiểm tra tính chính xác.

1  
Kết quả chưa đạt được của đề tài:

- Thông số giám sát áp suất lốp xe chưa thể gửi lên màn hình TFT.
  - Mô hình chưa được làm mạch in PCB.
- Một vài sự cố phát hiện của đề tài:
- Khi bật tắt relay 5V liên tục, quạt tản nhiệt sẽ có dòng điện chảy ngược(chế độ máy phát khi tắt quạt) khiến màn hình cảm ứng bị nhiễu tín hiệu.

Giải pháp: khi làm việc với động cơ, cần phải có một số linh kiện bảo vệ MCU từ dòng chảy ngược như diode, relay bảo vệ, v.v...

### 4.2. Kết luận và hướng phát triển

Sau quá trình nghiên cứu và thực nghiệm, nhóm chúng em đã hoàn thành việc thiết kế và triển khai hệ thống điều khiển và giám sát xe điện dựa trên giao thức CAN. Hệ thống đã đáp ứng hầu hết các mục tiêu đề ra, bao gồm hiển thị và điều khiển thông số xe thông qua màn hình cảm ứng TFT ILI9341, điều khiển động cơ BLDC, giám sát và cảnh báo nhiệt độ động cơ. Giao thức CAN đã được áp dụng thành công để kết nối các vi điều khiển ESP32 và STM32, với cơ chế lọc CAN giúp đảm bảo dữ liệu truyền nhận chính xác, không bị chia cắt.

Các kết quả thực nghiệm cho thấy hệ thống có độ ổn định cao, đo lường chính xác tốc độ và nhiệt độ động cơ. Tuy nhiên, đề tài vẫn còn một số hạn chế cần khắc phục, như chưa thể hiển thị thông tin áp suất lốp lên màn hình TFT và chưa triển khai mạch in PCB. Ngoài

ra, trong quá trình thực hiện, nhóm cũng phát hiện hiện tượng nhiễu tín hiệu khi bật/tắt relay 5V liên tục do dòng chảy ngược từ quạt tản nhiệt. Nhóm đã đề xuất giải pháp sử dụng diode và relay bảo vệ để hạn chế vấn đề này, nhằm đảm bảo tính ổn định của hệ thống.

**Hướng phát triển:**

Nhóm chúng em định hướng phát triển và hoàn thiện hệ thống theo các hướng sau:

– **Tích hợp chức năng hiển thị áp suất lốp lên màn hình TFT**, giúp người dùng theo dõi trực quan hơn và nâng cao tính an toàn của xe.

– **Thiết kế và chế tạo mạch in PCB chuyên dụng**, giúp hệ thống gọn gàng hơn, tăng độ ổn định và giảm nhiễu trong quá trình hoạt động.

– **Tối ưu giao diện người dùng**, cải thiện trải nghiệm điều khiển và hiển thị thông tin trực quan hơn.

– **Mở rộng hệ thống giám sát**, bổ sung thêm các cảm biến để theo dõi các thông số khác như điện áp pin, dòng điện tiêu thụ, hoặc các cảnh báo lỗi quan trọng khác.

– **Thử nghiệm hệ thống trên mô hình xe thực tế**, để đánh giá độ tin cậy và khả năng ứng dụng trong thực tiễn.

Nhóm chúng em tin rằng với những cải tiến trên, hệ thống sẽ ngày càng hoàn thiện và có thể áp dụng vào thực tế trong các mô hình xe điện thông minh.

## TÀI LIỆU THAM KHẢO

### Sách tham khảo:

- [1] MAX31865, Maxim Integrated and the Maxim Integrated logo are trademarks of Maxim Integrated Products, Inc.
- [2] RM0008 Reference manual, PM0056, STM32F10xxx Cortex®-M3 programming manual.
- [3] Brushless DC Motor Fundamentals Application Note, July 2011, by Jian Zhao/Yangwei Yu.
- [4] ILI9341, a-Si TFT LCD Single Chip Driver 240RGBx320 Resolution and 262K color, ILITEK.

# LVTN\_GIAMSATDIEUKHIENXEHOIDIENBANGCAN\_KHK (2).pdf

## ORIGINALITY REPORT

**15%**  
SIMILARITY INDEX

**13%**  
INTERNET SOURCES

**7%**  
PUBLICATIONS

**7%**  
STUDENT PAPERS

## PRIMARY SOURCES

- |          |  |           |
|----------|--|-----------|
| <b>1</b> | <a href="http://www.ctu.edu.vn">www.ctu.edu.vn</a><br>Internet Source            | <b>6%</b> |
| <b>2</b> | Submitted to Vietnam Maritime University<br>Student Paper                        | <b>5%</b> |
| <b>3</b> | Submitted to Ho Chi Minh City University of Transport<br>Student Paper           | <b>1%</b> |
| <b>4</b> | <a href="http://kotlinnews.com">kotlinnews.com</a><br>Internet Source            | <b>1%</b> |
| <b>5</b> | <a href="http://www.slideshare.net">www.slideshare.net</a><br>Internet Source    | <b>1%</b> |
| <b>6</b> | <a href="http://text.123docz.net">text.123docz.net</a><br>Internet Source        | <b>1%</b> |
| <b>7</b> | <a href="http://www.vietnic.vn">www.vietnic.vn</a><br>Internet Source            | <b>1%</b> |
| <b>8</b> | Submitted to Ho Chi Minh University of Technology and Education<br>Student Paper | <b>1%</b> |

Exclude quotes      On  
Exclude bibliography      On

Exclude matches      < 1%