**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA ĐIỆN**

Ảnh có chứa văn bản, vòng tròn, biểu tượng, Phông chữ

Mô tả được tạo tự động

**ĐỒ ÁN TỐT NGHIỆP**

**KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HÓA**

**ĐỀ TÀI:**

**NGHIÊN CỨU VÀ ỨNG DỤNG CHUẨN TRUYỀN THÔNG CANOPEN CHO ĐIỀU KHIỂN ĐỘNG CƠ**

Thông tin sinh viên:  **HUỲNH ĐỨC TRUNG**

**NGUYỄN LƯƠNG PHONG**

Lớp: **20TDHCLC2**

Thông tin Cán bộ hướng dẫn :

GVHD: **TS. NGUYỄN KHÁNH QUANG**

CBHD1: **TS. NGUYỄN ĐĂNG KHOA**

***Đà Nẵng – 06/2025***

**TÓM TẮT**

1. **Mục đích thực hiện đề tài**

Hiện nay, trong lĩnh vực điều khiển tự động, yêu cầu về việc trao đổi dữ liệu nhanh chóng, tin cậy giữa các thiết bị ngày càng trở nên cấp thiết. Chuẩn truyền thông CANopen, phát triển từ giao thức CAN, đã và đang được ứng dụng rộng rãi trong các hệ thống công nghiệp, đặc biệt là trong điều khiển động cơ, nhờ khả năng đáp ứng yêu cầu giao tiếp tốc độ cao và tính đồng bộ

Trước thực tế đó, đề tài “Nghiên cứu và ứng dụng chuẩn truyền thông CANopen cho điều khiển động cơ” được thực hiện nhằm tìm hiểu sâu về nguyên lý hoạt động, đặc điểm của CANopen, đồng thời triển khai ứng dụng thực tế vào bài toán điều khiển động cơ, góp phần nâng cao tính linh hoạt và hiệu quả trong các hệ thống điều khiển công nghiệp.

1. **Mục tiêu của đề tài**

* Nghiên cứu lý thuyết giao thức CAN và chuẩn truyền thông CANopen.
* Thiết kế và xây dựng mô hình thực nghiệm điều khiển động cơ sử dụng giao thức CANopen.
* Đánh giá hiệu quả của việc sử dụng CANopen trong việc điều khiển và giám sát hoạt động của động cơ.

1. **Phạm vi và đối tượng nghiên cứu**

* Tìm hiểu và ứng dụng chuẩn truyền thông CANopen cho điều khiển các loại động cơ trong môi trường mô phỏng và thực tế.
* Đối tượng nghiên cứu và các hệ thống nhúng, hệ thống điều khiển các loại động cơ sử dụng giao thức CANopen.

1. **Phương pháp nghiên cứu**

Nghiên cứu tài liệu lý thuyết về CAN, CANopen và các chuẩn kỹ thuật liên quan.

Thiết kế phần cứng và lập trình phần mềm điều khiển động cơ.

Tiến hành mô phỏng và thực nghiệm trên mô hình thực tế.

Đánh giá và phân tích kết quả thực nghiệm và đề xuất hướng cải tiến.

1. **Kết cấu của đồ án tốt nghiệp**

Mở đầu

Chương 1: Cơ sở lý thuyết

Chương 2: Thiết kế hệ thống truyền thông

Chương 3: Thiết kế cấu hình Object Dictionary

|  |  |
| --- | --- |
| ĐẠI HỌC ĐÀ NẴNG  TRƯỜNG ĐẠI HỌC BÁCH KHOA  **KHOA ĐIỆN** | **CỘNG HÒA XÃ HÔI CHỦ NGHĨA VIỆT NAM**  Độc lập - Tự do - Hạnh phúc |

Phụ lục 01

**NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

Họ tên sinh viên: Nguyễn Lương Phong Số thẻ sinh viên: 105200420

Huỳnh Đức Trung 105200433

Lớp: 20TDHCLC2 Khoa: Điện Ngành: Kỹ thuật Điều khiển và Tự động hóa

1. *Tên đề tài đồ án:*

Nghiên cứu và ứng dụng chuẩn truyền thông CANopen cho điều khiển động cơ.

*Đề tài thuộc diện:* ☐*Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*

1. *Các số liệu và dữ liệu ban đầu:*

* Datasheet vi điều khiển STM32F446RE
* Tài liệu về giao thức CAN và chuẩn CANopen
* Tài liệu kỹ thuật động cơ BLDC
* Thư viện CANopenNode và CANopenNodeSTM32

Board mạch Nucleo STM32, driver simpleFOC, CAN transceiver, phần mềm mô phỏng

*Nội dung các phần thuyết minh và tính toán:*

* Tìm hiểu chuẩn giao thức CAN và giao thức CANopen
* Nghiên cứu thư viện CANopenNode và cấu trúc Object Dictionary
* Cấu hình truyền thông CAN trên vi điều khiển STM32
* Thiết kế hệ thống truyền nhận dữ liệu với động cơ BLDC
* Cấu hình TPDO/RPDO truyền nhận tốc độ, dòng, tham chiếu,…
* Thiết kế phần mềm giám sát trên máy tính bằng QTCreator
* Đánh giá và phân tích kết quả và hoàn thiện thuyết minh

1. *Các bản vẽ, đồ thị ( ghi rõ các loại và kích thước bản vẽ ):*

* Sơ đồ khối của hệ thống
* Sơ đồ kết nối phần cứng
* Lưu đồ chương trình điều khiển CANopen
* Giao diện phần mềm giám sát QT
* Biểu đồ kết quả truyền nhận tốc độ/vị trí từ động cơ

|  |  |
| --- | --- |
| 1. *Họ tên người hướng dẫn:* | *Phần/ Nội dung:* |
| TS. Nguyễn Khánh Quang | Giám sát tiến độ thực hiện  Kiểm tra và cho ý kiến dự trên phần công việc đã hoàn thành |
| TS. Nguyễn Đăng Khoa | Hướng dẫn kiến thức thực tế  Cung cấp cách tài liệu tham khảo |

1. *Ngày giao nhiệm vụ đồ án:*
2. *Ngày hoàn thành đồ án:*

|  |  |
| --- | --- |
|  | *Đà Nẵng, ngày tháng năm 2025* |
| **Trưởng Bộ môn Tự động hóa**  TS. Giáp Quang Huy | **Người hướng dẫn** |

|  |  |
| --- | --- |
| ĐẠI HỌC ĐÀ NẴNG  TRƯỜNG ĐẠI HỌC BÁCH KHOA  **KHOA ĐIỆN** | **CỘNG HÒA XÃ HÔI CHỦ NGHĨA VIỆT NAM**  Độc lập - Tự do - Hạnh phúc |

Phụ lục 02

**NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TT | Họ tên sinh viên | Số thẻ SV | Lớp | Ngành |
| 1 | Nguyễn Lương Phong | 105200420 | 20TDHCLC2 | Kỹ thuật Điều khiển và Tự động hóa |
| 2 | Huỳnh Đức Trung | 105200433 | 20TDHCLC2 | Kỹ thuật Điều khiển và Tự động hóa |

1. *Tên đề tài đồ án:*

Nghiên cứu và ứng dụng chuẩn truyền thông CANopen cho điều khiển động cơ

*Đề tài thuộc diện:* ☐*Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*

1. *Các số liệu và dữ liệu ban đầu:*

* Datasheet vi điều khiển STM32F446RE
* Tài liệu về giao thức CAN và chuẩn CANopen
* Tài liệu kỹ thuật động cơ BLDC
* Thư viện CANopenNode và CANopenNodeSTM32

1. *Nội dung các phần thuyết minh và tính toán:*
2. *Phần chung:*

|  |  |  |
| --- | --- | --- |
| TT | Họ tên sinh viên | Nội dung |
| 1 | Nguyễn Lương Phong | Tìm hiểu chuẩn truyền thông CANopen và nguyên lý điều khiển động cơ BLDC bằng thuật toán FOC.  Phân tích hệ thống điều khiển tổng thể của hệ thống.  Đề xuất giải pháp ứng dụng chuẩn CANopen trong điều khiển động cơ BLDC.  Phân chia nhiệm vụ, phối hợp giữa hai thành viên để hoàn thiện hệ thống truyền thông – giám sát – điều khiển theo chuẩn công nghiệp. |
| 2 | Huỳnh Đức Trung |

1. *Phần riêng:*

|  |  |  |
| --- | --- | --- |
| TT | Họ tên sinh viên | Nội dung |
| 1 | Nguyễn Lương Phong | Thiết kế giao diện giám sát điều khiển động cơ bằng phần mềm QTCreator.  Thực hiện lập trình thu – gửi dữ liệu từ giao diện tới STM32 bằng thư viện CAN trên máy tính.  Tối ưu giao diện thân thiện người dùng, dễ vận hành và giám sát. |
| 2 | Huỳnh Đức Trung | Nghiên cứu chuẩn truyền thông CANopen, giao thức CAN và ứng dụng trong điều khiển động cơ.  Cấu hình và sử dụng thư viện CANopenNode cho STM32F446RE.  Tổ chức truyền nhận dữ liệu STM32 và phần mềm giám sát trên máy tính thông qua CAN USB adapter.  Viết chương trình truyền thông trên STM32 sử dụng HAL + CANopenNode để kết nối với giao diện điều khiển. |

1. *Các bản vẽ, đồ thị ( ghi rõ các loại và kích thước bản vẽ ):*
2. *Phần chung:*

| TT | Họ tên sinh viên | Nội dung |
| --- | --- | --- |
|  |  |  |
|  |  |

1. *Phần riêng:*

|  |  |  |
| --- | --- | --- |
| TT | Họ tên sinh viên | Nội dung |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
| 1. *Họ tên người hướng dẫn:* | *Phần/ Nội dung:* |
|  |  |
|  |  |
|  |  |

1. *Ngày giao nhiệm vụ đồ án:*
2. *Ngày hoàn thành đồ án:*

|  |  |
| --- | --- |
|  | *Đà Nẵng, ngày tháng năm 2025* |
| **Trưởng Bộ môn Tự động hóa**  TS. Giáp Quang Huy | **Người hướng dẫn** |

|  |  |
| --- | --- |
| ĐẠI HỌC ĐÀ NẴNG  **TRƯỜNG ĐẠI HỌC BÁCH KHOA**  KHOA ĐIỆN | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  Độc lập - Tự do - Hạnh phúc |

**PHIẾU KIỂM SOÁT TIẾN ĐỘ LÀM ĐỒ ÁN TỐT NGHIỆP**

Phụ lục 03

(Phiếu dành cho người hướng dẫn/sinh viên)

Họ tên sinh viên: Nguyễn Lương Phong Số thẻ SV : 105200420

Huỳnh Đức Trung 105200433

Tên đề tài ĐATN: Nghiên cứu và ứng dụng chuẩn truyền thông CANopen điều khiển động cơ

Họ tên người HD: T.S Nguyễn Khánh Quang Đơn vị: Trường Đại học Bách Khoa Đà Nẵng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tuần** | **Ngày** | **Khối lượng** | | **GVHD**  **ký tên** |
| **đã thực hiện (%)** | **tiếp tục thực hiện (%)** |  |
| 1 | 13/3/2025 | Tìm hiểu đề tài tốt nghiệp, xác định phạm vi nhóm phụ trách (CANopen) | Đọc tài liệu về giao thức truyền thông CAN và CANopen |  |
| 2 | 20/3/2025 | Tìm hiểu chuẩn CAN, nguyên lý hoạt động và mô hình truyền thông | Nghiên cứu cấu trúc OD, PDO, SDO trong CANopen |  |
| 3 |  | Tìm hiểu thư viện CANopenNode và cách tích hợp với STM32 | Cấu hình cơ bản CANopen trên STM32 Nucleo F446RE |  |
| 4 |  | Duyệt lần 1: Đánh giá khối lượng hoàn thành \_\_\_\_\_ % :  Được tiếp tục làm ĐATN Không tiếp tục thực hiện ĐATN | |  |
| 5 |  | Tạo OD mẫu bằng phần mềm CANopenEditor, cấu hình các biến cơ bản | Tạo và test TPDO gửi từ STM32 đến PC (hiển thị dữ liệu mẫu) |  |
| 6 |  | Cấu hình RPDO được nhận giá trị vận tốc từ PC xuống STM32 | Giao tiếp CAN thử nghiệm với nhóm FOC (test nhận dữ liệu) |  |
| 7 |  | Giao tiếp TPDO/RPDO hoạt động ổn định, bắt đầu thiết kế phần mềm quan sát | Nghiên cứu QTCreator và giao tiếp CAN trên máy tính |  |
| 8 |  | Duyệt lần 2: Đánh giá khối lượng hoàn thành \_\_\_\_\_ % :  Được tiếp tục làm ĐATN Không tiếp tục thực hiện ĐATN | |  |
| 9 |  | Thiết kế giao diện QT hiện thị tốc độ và dòng điện đọc từ động cơ BLDC | Viết chương trình QT gửi lệnh điều khiển tốc độ về STM32 |  |
| 10 |  | Hoàn thiện giao tiếp QT – STM32 qua CAN (test gửi/nhận) | Bắt đầu viết chương 1, chương 2 cho báo cáo |  |
| 11 |  | Viết chương 3: phân tích giao thức CANopen và mô hình hệ thống | Viết chương 4: Cấu hình Object Dictionary, PDO mapping |  |
| 12 |  | Duyệt lần 3: Đánh giá khối lượng hoàn thành \_\_\_\_\_ % :  Được tiếp tục làm ĐATN Không tiếp tục thực hiện ĐATN | |  |
| 13 |  | Hoàn thiện chương 5: giao tiếp với hệ FOC, kết quả thử nghiệm thực tế | Viết chương 6: Kết luận và hướng phát triển |  |
| 14 |  | Rà soát toàn bộ báo cáo, kiểm tra lỗi chính tả, định dạng | In ấn báo cáo, chuẩn bị thuyết trình |  |
| 15 |  | Nộp báo cáo, trình bày thuyết trình |  |  |
| 16 |  | Bảo vệ đồ án trước hội đồng |  |  |

# LỜI NÓI ĐẦU

Trong suốt quá trình học tập tại Trường Đại học Bách Khoa – Đại học Đà Nẵng, chúng em đã nhận được sự giảng dạy tận tình, sự hướng dẫn chu đáo cùng sợ hỗ trợ nhiệt tình từ các giảng viên Khoa Điện cũng như toàn thể cán bộ giảng viên của nhà trường. Chúng em xin chân thành cảm ơn tất cả các thầy cô đã truyền đạt cho chúng em những kiến thức chuyên môn cùng những kinh nghiệm thực tiễn quý báu, giúp chúng em tự tin bước vào cuộc sống.

Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc đến Thầy Nguyễn Khánh Quang, Tiến sĩ – Giảng viên Khoa Điện – Trường Đại học Bách khoa – Đại học Đà Nẵng và Thầy Nguyễn Đăng Khoa, Tiến sĩ – Giám đốc Công ty Sybotix Solution, đã tận tình hướng dẫn, giải đáp những thắc mắc, đóng góp ý kiến và tạo mọi điều kiện thuận lợi cho chúng em trong suốt quá trình thực hiện và hoàn thành đồ án tốt nghiệp. Nhờ sự hỗ trợ quý báu của thầy, chúng em đã hoàn thành đồ án đúng tiến đồ đề ra.

Chúng em cũng xin gửi lời cảm ơn chân thành đến gia đình và bạn bè đã luôn động viên, khích lệ và hỗ trợ chúng em cả về vật chất lẫn tinh thần trong suốt quá trình học tập và thực hiện đồ án tốt nghiệp này.

Mặc dù đã cố gắng hoàn thành đồ án với tinh thần trách nhiệm cao nhất, song chắc chắn không tránh khỏi các thiết sót. Tôi rất mong nhận được sự cảm thông từ quý Thầy Cô và huy vọng sẽ nhận được những ý kiến đóng góp quý báu để hoàn thiện hơn nữa đồ án của chúng em.

Xin chân thành cảm ơn!

LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT

Chúng tôi xin cam đoan rằng đồ án tốt nghiệp với đề tài *“Nghiên cứu và ứng dụng chuẩn truyền thông CANopen trong điều khiển động cơ”* là kết quả làm việc trung thực và nghiêm túc của hai thành viên. Toàn bộ nội dung trong đồ án này được xây dựng dựa trên quá trình nghiên cứu, tìm hiểu tài liệu, thiết kế và thực nghiệm thực tế do chính chúng tôi thực hiện dưới sự hướng dẫn của giảng viên hướng dẫn.

Chúng tôi cam kết không sao chép, không sử dụng trái phép nội dung của bất kỳ cá nhân hay tổ chức nào. Nếu phát hiện có sự gian lận hay vi phạm quy định về liêm chính học thuật, chúng tôi xin hoàn toàn chịu trách nhiệm trước nhà trường và pháp luật.

Mọi tài liệu tham khảo đều được trích dẫn rõ ràng, đúng quy định. Chúng tôi hiểu rằng việc tuân thủ các nguyên tắc về liêm chính học thuật là yêu cầu bắt buộc và thể hiện đạo đức nghề nghiệp của người làm khóa học.

Sinh viên thực hiện

{Chữ ký, họ và tên sinh viên}

**MỤC LỤC**

[LỜI NÓI ĐẦU ix](#_Toc198474624)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 1](#_Toc198474625)

[1.1 Tổng quan về mạng CAN 1](#_Toc198474626)

[1.1.1 Giới thiệu về mạng CAN 1](#_Toc198474627)

[1.1.2 Miền ứng dụng của CAN 2](#_Toc198474628)

[1.1.3 Cấu trúc mạng và kỹ thuật truyền dẫn trong mạng CAN 3](#_Toc198474629)

[1.1.4 Cơ chế truyền nhận trên dây 5](#_Toc198474630)

[1.1.5 Cấu trúc bức điện 7](#_Toc198474631)

[1.1.6 Cấu trúc chi tiết gói tin trong giao thức CAN 8](#_Toc198474632)

[1.1.7 Cơ chế tranh quyền truy cập bus (Arbitration) trong giao thức CAN 12](#_Toc198474633)

[1.2 Tổng quan về chuẩn truyền thông CANopen 14](#_Toc198474634)

[1.2.1 Giới thiệu về chuẩn truyền thông CANopen 14](#_Toc198474635)

[1.2.2 Cấu trúc của CANopen 16](#_Toc198474636)

[CHƯƠNG 2: THIẾT KẾ HỆ THỐNG TRUYỀN THÔNG 30](#_Toc198474637)

[2.1 Sơ đồ kết nối phần cứng 30](#_Toc198474638)

[2.1.1 Giới thiệu tổng quan 30](#_Toc198474639)

[2.1.2 Sơ đồ kết nối phần cứng 30](#_Toc198474640)

[2.1.3 Phân tích phần cứng của hệ thống 31](#_Toc198474641)

[2.2 Sơ đồ khối nguyên lí hoạt động 42](#_Toc198474642)

[2.2.1 Giới thiệu tổng quan 42](#_Toc198474643)

[2.2.2 Sơ đồ khối nguyên lí hoạt động 42](#_Toc198474644)

[2.2.3 Mô tả chi tiết nguyên lí hoạt động 43](#_Toc198474645)

[CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN FOC 44](#_Toc198474646)

[3.1 Giới thiệu tổng thể vòng điều khiển 44](#_Toc198474647)

[3.2 Điều khiển dòng điện (Id, Iq) 45](#_Toc198474648)

[3.3 Biến đổi Clark và Park 46](#_Toc198474649)

[3.3.1 Phép biến đổi Clark 46](#_Toc198474650)

[3.3.2 Phép biến đổi Park 47](#_Toc198474651)

[3.4 Điều khiển tốc độ (vòng ngoài) 47](#_Toc198474652)

[3.5 Điều khiển vị trí (vòng ngoài cùng) 48](#_Toc198474653)

[3.6 Điều chế sin PWM 48](#_Toc198474654)

[CHƯƠNG 4: THIẾT KẾ VÀ CẤU HÌNH OBJECT DICTIONARY BẰNG CANopenEditor 50](#_Toc198474655)

[4.1 Giới thiệu CANopenEditor 50](#_Toc198474656)

[4.2 Giao diện và cấu trúc cơ bản 50](#_Toc198474657)

[4.3 Ứng dụng tạo Object Dictionary cho hệ thống điều khiển động cơ BLDC. 52](#_Toc198474658)

[4.3.1 Xác định các tham số cần thiết trong hệ thống điều khiển BLDC 53](#_Toc198474659)

[4.3.2 Tạo mục TPDO trong OD 53](#_Toc198474660)

[4.3.3 Tạo mục RPDO trong OD 54](#_Toc198474661)

[CHƯƠNG 5: TRIỂN KHAI PHẦN MỀM TRÊN STM32 55](#_Toc198474662)

[5.1 Cấu hình TIM1 – Phát xung PWM 3 pha. 55](#_Toc198474663)

[5.1.1 Chức năng 55](#_Toc198474664)

[5.1.2 Cấu hình các kênh của TIM1 56](#_Toc198474665)

[5.2 Cấu hình SPI – Giao tiếp với Encoder AS5147U 57](#_Toc198474666)

[5.2.1 Chức năng 57](#_Toc198474667)

[5.2.2 Cấu hình SPI 57](#_Toc198474668)

[5.2.3 Khung truyền SPI và quá trình đọc góc 58](#_Toc198474669)

[5.3 Cấu hình ADC1 – Đo dòng điện 3 pha 58](#_Toc198474670)

[5.3.1 Bộ khuếch đại INA240 59](#_Toc198474671)

[5.3.2 Nguyên lý đo dòng điện 59](#_Toc198474672)

[5.3.3 Sử dụng ADC injected mode 59](#_Toc198474673)

[5.3.4 Trình tự hoạt động 60](#_Toc198474674)

[5.4 Cấu hình ADC2 – Đo điện áp nguồn. 60](#_Toc198474675)

[5.4.1 Mạch chia phân áp 60](#_Toc198474676)

[5.4.2 Sơ đồ kết nối 61](#_Toc198474677)

[5.4.3 Cấu hình ADC2 62](#_Toc198474678)

[5.5 Giao tiếp CAN 63](#_Toc198474679)

[5.5.1 Cấu hình CAN 63](#_Toc198474680)

[CHƯƠNG 6: XÂY DỰNG PHẦN MỀM ĐIỀU KHIỂN ĐỘNG CƠ BẰNG QT 65](#_Toc198474681)

[6.1 Mục tiêu của phần mềm ứng dụng 65](#_Toc198474682)

[6.2 Công cụ và công nghệ sử dụng 65](#_Toc198474683)

[6.2.1 Giới thiệu về phần mềm QT Creator 65](#_Toc198474684)

[6.2.2 Cấu trúc chương trình giao diện 66](#_Toc198474685)

[6.2.3 Cơ chế truyền nhận dữ liệu 67](#_Toc198474686)

[6.2.4 Các tab đồ thị giám sát 67](#_Toc198474687)

[KẾT LUẬN CHUNG 70](#_Toc198474688)

[TÀI LIỆU THAM KHẢO 71](#_Toc198474689)

DANH SÁCH HÌNH ẢNH

[Hình 1.1. Tính ổn định của CAN 1](#_Toc197959662)

[Hình 1.2. Ứng dụng mạng CAN trong điều khiển xe hơi 2](#_Toc197959663)

[Hình 1.3. Cấu tạo phần cứng của một NodeCAN 3](#_Toc197959664)

[Hình 1.4. Các Node trên hệ thống CAN 5](#_Toc197959665)

[Hình 1.5. Logic 0 và Logic 1 trên CANBus 6](#_Toc197959666)

[Hình 1.6. Khả năng kháng nhiễu của giao tiếp CAN 6](#_Toc197959667)

[Hình 1.7. Cấu trúc gói tin Standard CAN 9](#_Toc197959668)

[Hình 1.8. Cấu trúc gói tin Extended CAN 10](#_Toc197959669)

[Hình 1.9. Hình ảnh mô tả một tình huống tranh quyền truy cập bus giữa ba Node. 13](#_Toc197959670)

[Hình 1.10. Ứng dụng CANopen trong điều khiển thiết bị công nghiệp 15](#_Toc197959671)

[Hình 1.11. Các lớp giao thức truyền thông CANopen 16](#_Toc197959672)

[Hình 1.12. Lớp vật lý 17](#_Toc197959673)

[Hình 1.13. Định dạng khung chuẩn CAN 17](#_Toc197959674)

[Hình 1.14. Sơ đồ phân bổ định danh mặc định 18](#_Toc197959675)

[Hình 1.15. Giao tiếp SDO 21](#_Toc197959676)

[Hình 1.16. Truy cập vào Object Dictionary 22](#_Toc197959677)

[Hình 1.17. Giao tiếp PDO 23](#_Toc197959678)

[Hình 1.18. Write PDO và Read PDO 24](#_Toc197959679)

[Hình 1.19. Cơ chế đồng bộ SYCN 25](#_Toc197959680)

[Hình 1.20. Cơ chế truyền thông báo lỗi khẩn cấp EMCY 26](#_Toc197959681)

[Hình 1.21. Cấu trúc mạng NMT 27](#_Toc197959682)

[Hình 1.22. Trạng thái NMT Slave 28](#_Toc197959683)

[Hình 2.1. Sơ đồ hệ thống điều khiển động cơ BLDC sử dụng giao thức CANopen 30](#_Toc197959684)

[Hình 2.2. STM32F446RE 32](#_Toc197959685)

[Hình 2.3. USB CAN Adapter 33](#_Toc197959686)

[Hình 2.4. SN65HVD230 34](#_Toc197959687)

[Hình 2.5. MKS SimpleFOC Shield V2.0.4 35](#_Toc197959688)

[Hình 2.6. Magnetic Encoder Development Board 37](#_Toc197959689)

[Hình 2.7. Khung truyền nhận 16 bit 38](#_Toc197959690)

[Hình 2.8. Sơ đồ kết nối AS5147U 40](#_Toc197959691)

[Hình 2.9. Sơ đồ khối nguyên lý hoạt động của hệ thống 43](#_Toc197959692)

[Hình 3.1. Phép biến đổi Clarke 46](#_Toc197959693)

[Hình 3.2. Phép biến đổi Park 47](#_Toc197959694)

[Hình 4.1. Phần mềm CANopenEditor 50](#_Toc197959695)

[Hình 4.3. Giao diện phần mềm CANopenEditor 51](#_Toc197959696)

[Hình 4.4. Thanh công cụ chính của CANopenEditor 51](#_Toc197959697)

[Hình 4.5. Danh sách sub-index 52](#_Toc197959698)

[Hình 5.1. Timer 1 Channel 1,2,3 55](#_Toc197959699)

[Hình 5.2. PWM Center Aligned Mode 1 và Trigger cho ADC Injected 56](#_Toc197959700)

[Hình 5.3. Cấu hình PWM Mode 2 57](#_Toc197959701)

[Hình 5.4. Cấu hình SPI1 58](#_Toc197959702)

[Hình 5.5. Sơ đồ kết nối INA240 59](#_Toc197959703)

[Hình 5.6 Cấu trúc cầu phân áp 61](#_Toc197959704)

[Hình 5.7. Cấu hình AD2\_IN1 62](#_Toc197959705)

[Hình 5.8. Cấu hình DMA cho ADC2 63](#_Toc197959706)

[Hình 5.9. Cấu hình CAN trên STM32 63](#_Toc197959707)

[Hình 6.1. Phần mềm QT Creator 65](#_Toc197959708)

[Hình 6.2. Giao diện chính của phần mềm điều khiển trên máy tính 66](#_Toc197959709)

# CƠ SỞ LÝ THUYẾT

## Tổng quan về mạng CAN

### Giới thiệu về mạng CAN

Controller Area Network (CAN) là giao thức giao tiếp nối tiếp hỗ trợ mạnh cho những hệ thống điều khiển thời gian thực phân bố (Distributed Realtime Control System) với độ ổn định, bảo mật và đặc biệt chống nhiễu cực kỳ tốt.

Việc phát triển mạng CAN được bắt đầu từ năm 1983 bởi “Công ty Robert Bosch GmbH”. Sau đó, giao thức được chính thức công bố vào năm 1986 tại đại hội Hiệp hội Kĩ sư ô tô “Society of Automotive Engineers, SAE” ở Detroit, Michigan, Mỹ. Những chip CAN controller đầu tiên được Intel sản xuất vào năm 1987, và sau đó là bởi Philips. Xe Mercedes-Benz W140 là xe đầu tiên được trang bị CAN.

Ngay từ khi mới ra đời, mạng CAN đã được chấp nhận và ứng dụng một cách rộng rãi trong các lĩnh vực công nghiệp, chế tạo ô tô, xe tải. Với thời gian, CAN càng trở nên thông dụng hơn vì tính hiệu quả, ổn định, đơn giản và đặc biệt là chi phí rẻ. Nó được sử dụng với với việc truyền dữ liệu lớn, đáp ứng thời gian thực và trong các môi trường khác nhau. Cuối cùng, truyền tốc độ cao rất ổn đỉnh. Đó là lý do tại sao chúng được sử dụng trong nhiều ngành công nghiệp khác ngoài xe hơi như các máy nông nghiệp, tàu ngầm, các dụng cụ y khoa, máy dệt.

Điểm nổi trội nhất ở chuẩn CAN là tính ổn định và an toàn. Nhờ cơ chế phát hiện và xử lý lỗi cực mạnh, lỗi CAN messages hầu như được phát hiện. Theo thống kê, xác suất để một message của CAN bị lỗi không được phát hiện là:



Hình 1.1. Tính ổn định của CAN

Ví dụ: Cho rằng nếu giả sử cứ 0.7s thì môi trường tác động lên đường truyền CAN làm lỗi 1 bit. Và giả sử tốc độ truyền là 500kbit/s. Hoạt động 8h/ngày và 356 ngày/năm. Thì trong vòng 1000 năm trung bình sẽ có một frame bị lỗi mà không phát hiện.

### Miền ứng dụng của CAN

Mạng truyền thông CAN đã trở thành tiêu chuẩn quan trọng trong các hệ thống nhúng hiện đại, đặc biệt là trong ngành công nghiệp ô tô. Với khả năng truyền thông tin nhanh, tin cậy và giảm thiểu số lượng dây nối, CAN không chỉ được dùng trong xe hơi mà còn được ứng dụng rộng rãi trong tự động hóa công nghiệp, thiết bị y tế, điều khiển robot, và các hệ thống giám sát từ xa.

Trong các dòng xe hiện đại, thường tồn tại hai mạng CAN riêng biệt:

* CAN tốc độ cao (High-speed CAN): hoạt động ở tốc độ lên đến 1Mbps, phụ vục cho các chức năng quan trọng như thời gian thực như: điều khiển động cơ, hộp số, hệ thống phanh ABS,…
* CAN tốc độ thấp (Low-speed CAN): tốc độ thấp hơn (125kbps hoặc thấp hơn), sử dụng cho các hệ thống không yêu cầu thời gian thực như: gương chiếu hậu, cửa kính điện, điều chỉnh ghế, điều hòa không khí,…



Hình 1.2. Ứng dụng mạng CAN trong điều khiển xe hơi

Như hình 1.2, CANbus kết nối các bộ điều khiển điện tử ECU với nhau, hình thành một hệ thống mạng linh hoạt và hiệu quả, nơi các ECU có thể giao tiếp với nhau mà không cần bộ điều khiển trung tâm. Điều này giúp tăng độ tin cậy, giảm chi phí dây dẫn và dễ dàng mở rộng hệ thống.

### Cấu trúc mạng và kỹ thuật truyền dẫn trong mạng CAN

#### Cấu trúc mạng

Mỗi ECU trong mạng CANbus gồm ba thành phần chính: Vi điều khiển (MCU), CAN Controller, và CAN Transceiver.



Hình 1.3. Cấu tạo phần cứng của một NodeCAN

Dưới đây là chi tiết chức năng và cách hoạt động của từng thành phần

##### **Vi điều khiển (MCU)**: Vi điều khiển là “bộ não” của ECU, chịu trách nhiệm xử lý dữ liệu và điều khiển luồng thông tin. Nó nhận và phân tích dữ liệu từ CAN Controller sau khi thông tin được truyền qua BUS và chuẩn bị dữ liệu để gửi đến các ECU khác thông qua CAN Controller.

* **Quy trình hoạt động của MCU:**

**Khi nhận dữ liệu**

Dữ liệu từ bus CAN được CAN Controller giải mã và gửi đến vi điều khiển.

MCU sẽ:

* Giải mã nội dung tin nhắn (message).
* Xử lý và ra quyết định dựa trên thông tin nhận được (ví dụ: điều khiển động cơ, bật đèn...).

**Khi truyền dữ liệu**

* Vi điều khiển tạo ra thông điệp (message) và gửi thông điệp đó đến CAN Controller để truyền đi trên BUS.

##### **CAN Controller:** CAN Controller là giao diện giữa Vi điều khiển và bus CAN. Nó đảm bảo việc truyền nhận dữ liệu tuân thủ theo giao thức truyền thông CAN.

* **Quy trình hoạt động của CAN Controller:**

**Khi truyền dữ liệu**

* Nhận dữ liệu từ vi điều khiển, định dạng lại thành một khung CAN (CAN frame), gồm các trường như ID, DLC, Data Field, CRC, EOF....
* Theo dõi bus CAN (qua cơ chế Arbitration) để đảm bảo bus đang rảnh (không bị chiếm dụng).
* Truyền dữ liệu nối tiếp trên bus thông qua CAN Transceiver.

**Khi nhận dữ liệu**

* Lắng nghe tín hiệu trên bus CAN.
* Lưu trữ các bit nhận được và kiểm tra các trường hợp như lỗi CRC, lỗi khung.
* Khi nhận đủ một khung CAN hoàn chỉnh (được đánh dấu bằng EOF - End of Frame), thông báo cho vi điều khiển rằng có dữ liệu mới cần xử lý.

##### **CAN Transceiver:** CAN Transceiver là bộ chuyển đổi tín hiệu giữa CAN Controller và bus CAN vật lý. Nó đảm bảo tín hiệu truyền trên bus phù hợp với đặc tính của giao thức CAN.

* **Quy trình hoạt động của CAN Transceiver:**

**Khi nhận dữ liệu từ bus**

* CAN bus sử dụng tín hiệu vi sai (differential signals) giữa hai dây CAN\_H và CAN\_L.
* CAN Transceiver chuyển đổi tín hiệu vi sai này thành tín hiệu mức logic (nhị phân) mà CAN Controller có thể hiểu được (bit 0 hoặc 1).
* Chuyển tín hiệu nhị phân đến CAN Controller để xử lý.

**Khi truyền dữ liệu từ bus**

* Nhận dữ liệu nhị phân từ CAN Controller.
* Chuyển đổi dữ liệu nhị phân thành tín hiệu vi sai phù hợp với tiêu chuẩn điện áp trên bus CAN.
* Truyền tín hiệu vi sai này lên bus CAN, đảm bảo tương thích với các ECU khác.

##### **Tổng quan về truyền và nhận dữ liệu trên ECU**

* **Khi truyền dữ liệu**
* Vi điều khiển: Tạo ra thông điệp (data) và gửi đến CAN Controller.
* CAN Controller: Định dạng thông điệp thành khung CAN và gửi dữ liệu nhị phân đến CAN Transceiver.
* CAN Transceiver: Chuyển đổi dữ liệu nhị phân thành tín hiệu vi sai trên bus CAN để truyền đi.
* **Khi nhận dữ liệu**
* CAN Transceiver: Chuyển tín hiệu vi sai trên bus CAN thành dữ liệu nhị phân.
* CAN Controller: Lưu trữ các bit nhận được, kiểm tra lỗi, và thông báo khi nhận đủ một khung hoàn chỉnh.
* Vi điều khiển: Nhận dữ liệu từ CAN Controller, xử lý, và thực hiện các hành động cần thiết.

### Cơ chế truyền nhận trên dây

CANbus sử dụng hai dây để giao tiếp. Các dây được gọi là CAN High và CAN Low. CANbus kết nối với tất cả các thành phần trên mạng thông qua hai dây này. Trong CANbus không hề phân biệt Master và Slave như một số giao thức khác.



Hình 1.4. Các Node trên hệ thống CAN

Các điểm đầu và cuối của Canbus thường được nối bằng bằng điện trở 120 Ohm, điện trở này được gọi là Terminator Resistor.

Không giống như một số giao thức khác trong việc sử dụng mức điện áp trên dây so với GND để quy đổi ra bit logic, CAN sử dụng mức chênh lệch điện áp giữa 2 dây để quy đổi. Những thay đổi về mức điện áp này sau đó được dịch sang mức logic.



Hình 1.5. Logic 0 và Logic 1 trên CANBus

Thông thường, giá trị điện áp cao nhất ở dây CAN High là 3.5V, điện áp thấp nhất là 2.5V.  Ở dây CAN low điện áp thấp nhất là 1.5V và cao nhất là 2.5V:

* **Mức logic 0:** Khi VCANH – VCANL = 2.5V. Hay mức chênh lệch điện áp 2 dây là 2.5V.
* **Mức logic 1:** Khi VCANH – VCANL = 0V. Hay khi cả hai dây đều có điện áp 2.5V.

Đó chính là lí do tại sao CAN lại có khả năng kháng nhiễu rất tốt. Đơn giản, khi có nhiễu tác động lên CANbus điều khiển mức điện áp giữa hai dây cùng thay đổi, cùng tăng hoặc cùng giảm nhưng mức chênh lệch điện áp giữa 2 dây không thây đổi nhiều.



Hình 1.6. Khả năng kháng nhiễu của giao tiếp CAN

### Cấu trúc bức điện

Trong mạng CAN dữ liệu giữa các thiết bị được trao đổi thông quan các gói tin được gọi là bức điện (CAN Frame). Mỗi bức điện bao gồm nhiều trường (Field) cụ thể nhằm đảm bảo việc giao tiếp được ổn định, đúng thứ tự và phát hiện lỗi một cách hiệu quả.

#### Các loại khung truyền

Trong giao thức CAN, dữ liệu được truyền giữa các thiết bị dưới dạng bức điện (Message Frame). Bức điện chứa các trường (field) cụ thể để đảm bảo giao tiếp hiệu quả, đáng tin cậy và đúng thứ tự. Các loại bức điện (CAN Frames) bao gồm:

##### **Data Frame (Khung dữ liệu)**

* **Chức năng:** Gửi dữ liệu thực tế giữa các thiết bị ECU.
* **Đặc điểm:**
* Chứa trường dữ liệu chính: Data Field (0 đến 8 byte).
* Có Identifier (ID) để xác định độ ưu tiên truyền thông.
* **Ý nghĩa:** Đây là Frame phổ biến nhất, dùng để truyền thông tin cảm biến, trạng thái, lệnh điều khiển giữa các thiết bị trong mạng CAN.

##### **Remote Frame (Khung yêu cầu dữ liệu)**

* **Chức năng:** Một node yêu cầu một node khác gửi dữ liệu.
* **Đặc điểm:**
* Không có trường Data Field.
* Bit RTR (Remote Transmission Request) được đặt thành 1.
* **Ứng dụng:** Ví dụ, thiết bị giám sát muốn yêu cầu dữ liệu từ cảm biến thì nó sẽ gửi một remote frame, thiết bị kia sẽ phản hồi bằng data frame.

##### **Error Frame (Khung báo lỗi)**

* **Chức năng:** Báo hiệu có lỗi trên bus CAN.
* **Đặc điểm:**
* Gửi bởi thiết bị phát hiện lỗi (bất kỳ thiết bị nào cũng có thể gửi).
* Chứa các Error Flags để cảnh bảo lỗi.
* **Cơ chế hoạt động:** Mạng CAN có khả năng tự phát hiện và sửa lỗi, nếu một thiết bị nhận thấy lỗi CRC, lỗi định dạng,… nó sẽ phát một Error Frame để toàn bộ mạng được biết.

##### **Overload Frame (Khung quá tải)**

* **Chức năng:** Tạm dừng giao tiếp khi thiết bị cần thêm thời gian xử lý.
* **Đặc điểm:**
* Gửi bởi thiết bị đang bị quá tải xử lý.
* Chứa Overload Flags (6-12 bit).
* **Ứng dụng:** Trong trường hợp thiết bị nhận chưa kịp xử lý dữ liệu, nó sẽ gửi Overload Frame để trì hoãn việc truyền tiếp theo, giúp hệ thống không bị nghẽn.

Bảng 1.1. Các loại Frame trong giao tiếp CAN

|  |  |  |
| --- | --- | --- |
| **Loại bức điện (Frame)** | **Chức năng chính** | **Đặc điểm** |
| **Data Frame** | Truyền dữ liệu thực tế giữa các ECU | - Chứa dữ liệu trong trường Data Field (0-8 byte).  - ID xác định ưu tiên truyền thông. |
| **Remote Frame** | Yêu cầu dữ liệu từ một ECU khác trên bus | - Không có Data Field.  - Bit RTR được đặt thành 1 (Remote). |
| **Error Frame** | Báo hiệu rằng một lỗi xảy ra trên bus CAN | - Được gửi bởi thiết bị phát hiện lỗi.  - Bao gồm trường lỗi đặc biệt (Error Flags) để thông báo. |
| **Overload Frame** | Tạm dừng giao tiếp do thiết bị cần thêm thời gian để xử lý | - Được gửi khi thiết bị nhận cảm thấy quá tải.  - Bao gồm Overload Flags (6-12 bit). |

### Cấu trúc chi tiết gói tin trong giao thức CAN

Gói truyền trên mạng CAN bao gồm nhiều trường khác nhau để đảm bảo truyền thông chính xác, phát hiện lỗi và quản lý quyền ưu tiên. Mỗi trường đóng vai trò riêng biệt:

Bảng 1.2. Cấu trúc và ý nghĩa các trường trong Frame CAN

|  |  |  |
| --- | --- | --- |
| **Thành phần** | **Số bit** | **Mô tả** |
| **SOF (Start of Frame)** | 1 bit | - Bit bắt đầu của khung truyền, luôn có giá trị **0**.  - Thông báo rằng một ECU muốn truyền dữ liệu. |
| **Arbitration Field** | 12 hoặc 30 bit | ID (Identifier): - Định danh của ECU gửi dữ liệu.  - Quyết định mức độ ưu tiên (ID càng thấp, ưu tiên càng cao). |
| RTR (Remote Transmission Request): - 1 bit xác định loại khung.  - 0: Data Frame.  - 1: Remote Frame. |
| **Control Field** | 1 bit | IDE (Identifier Extension): - Kiểm tra định dạng gói tin  0: Standard Frame 1: Extended Frame. |
| **Data Field** | 0-8 byte | Chứa dữ liệu thực tế được truyền (ví dụ: giá trị cảm biến, lệnh điều khiển). |
| **CRC Field** | 15 bit | Mã kiểm tra lỗi để phát hiện dữ liệu hỏng trong quá trình truyền. |
| **ACK Field** | 2 bit | - Bit xác nhận từ ECU nhận.  - ACK Slot: ECU nhận dữ liệu đặt bit này về 0 để báo nhận thành công. |
| **EOF (End of Frame)** | 7 bit | Đánh dấu kết thúc khung truyền, chuẩn bị cho bus quay lại trạng thái nhàn rỗi. |

#### Gói tin Standard CAN (chuẩn 11-bit ID)

Gói tin dạng chuẩn (Standard CAN) là định dạng cơ bản nhất của giao thức CAN, sử dụng mã định danh dài 11 bit. Đây là lựa chọn phổ biến trong các hệ thống có số lượng node vừa phải như mạng truyền thông nội bộ trong ô tô.



Hình 1.7. Cấu trúc gói tin Standard CAN

Bảng: Cấu trúc gói tin của Standard CAN

|  |  |  |
| --- | --- | --- |
| **Thành phần** | **Số bit** | **Chức năng** |
| Start of Frame (SOF) | 1 | Báo hiệu bắt đầu một gói tin |
| Identifier | 11 | Mã định danh thông điệp, đồng thời là căn cứ để xác định mức ưu tiên khi tranh quyền |
| RTR | 1 | Bit yêu cầu dữ liệu:   * 0 cho Data Frame * 1 cho Remote Frame |
| IDE | 1 | Xác định kiểu gói tin: “0” biểu thị khung chuẩn |
| r0 | 1 | Bit dự phòng cho khả năng mở rộng sau này |
| DLC (Data Length Code) | 4 | Chỉ định số byte dữ liệu (0 đến 8). |
| Data Field | 0-64 | Dữ liệu thực tế (tối đa 8 byte) |
| CRC | 15+1 | Kiểm tra lỗi bằng mã CRC, gồm chuỗi CRC và bit phần cách |
| ACK | 2 | Phản hồi từ các node nhận: nếu nhận thành công sẽ ghì đè bit “0” |
| EOF (End of Frame) | 7 | Đánh dấu kết thúc gói tin |
| IFS (Interframe Space) | ≥3 | Khoảng nghỉ trước gói tiếp theo |

Đặc điểm nổi bật của Stardand CAN là có mã định danh 11 bit đáp ứng phần lớn ứng dụng công nghiệp thông thường có thể tạo 2048 ID khác nhau. Có tốc độ xử lí nhanh hơn vì khung dữ liệu ngắn và phù hợp với mạng truyền thông có quy mô nhỏ hoặc trung bình.

#### Gói tin Extended CAN (Mở rộng 29-bit ID)

Khi hệ thống cần phân biệt nhiều loại thông điệp hoặc có số lượng node lớn, gói tin mở rộng sẽ được sử dụng. Định dạng này cho phép sử dụng 29-bit để làm mã định danh, mở rộng không gian địa chỉ hóa lên đến hơn 500 triệu ID.



Hình 1.8. Cấu trúc gói tin Extended CAN

Bảng: Cấu trúc gói tin Extended CAN

|  |  |  |
| --- | --- | --- |
| **Thành phần** | **Số bit** | **Chức năng** |
| SOF (Start of Frame) | 1 | Bắt đầu khung dữ liệu |
| 11-bit Base ID | 11 | Phần đầu của mã định danh |
| SRR (Substitute RTR) | 1 | Bit thay thế cho RTR trng khung mở rộng luôn luôn là “1” |
| IDE (Identifier Extension) | 1 | Đặt là “1” để báo hiệu đây là gói tin mở rộng |
| 18-bit Extended ID | 18 | Phần mở rộng giúp hoàn chỉnh mã định danh 29 bit |
| RTR | 1 | Tương tự khung chuẩn:   * 0 là truyền dữ liệu * 1 là yêu cầu dữ liệu |
| r1, r0 | 2 | Các bit dự phòng, gửi là 0 |
| DLC | 4 | Mã hóa số lượng byte dữ liệu gửi đi |
| Data Field | 0-64 | Tối đa 8 byte dữ liệu. |
| CRC | 15+1 | Kiểm tra lỗi truyền thông |
| ACK | 2 | Phản hồi từ các node nhận dữ liệu thành công |
| EOF | 7 | Kết thúc gói tin |
| IFS | ≥3 | Khoảng nghĩ giữa hai gói |

Đặc điểm nổi bật của Extended CAN là có tổng cộng 29-bit mã định danh nên có thể tạo được hơn 500 triệu mã ID khác nhau. Hữu ích trong các mạng lớn, hệ thống phân lớp thông điệp phức tạp. Vì mã định danh dài hơn nên thời gian truyền cũng lâu hơn so với khung chuẩn.

#### So sánh giữa Standard và Extended CAN.

Trong giao thức CAN, việc lựa chọn sử dụng định dạng khung chuẩn (Standand Frame) hay khung mở rộng (Extendend Frame) ảnh hưởng trực tiếp đến khả năng định danh, tốc độ truyền và hiệu quả hệ thống. Cả hai loại đều tuân thủ quy tắc truyền dữ liệu CAN, nhưng có một số khác biệt quan trọng về cấu trúc và ứng dụng:

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Standard CAN** | **Extended CAN** |
| Độ dài ID | 11 bit | 29 bit |
| Số lượng ID khả dụng | 2.048 | Hơn 500 triệu |
| Trường bổ sung | Không | Có SRR, IDE, Extended ID |
| Thời gian tranh quyền | Ngắn hơn | Dài hơn |
| Tốc độ truyền | Nhanh hơn | Chậm hơn một chút |
| Phù hợp với | Mạng nhỏ hoặc phản ứng nhanh | Hệ thống lớn, cần phần loại ID |
| Khả năng mở rộng hệ thống | Giới hạn | Rất cao |

### Cơ chế tranh quyền truy cập bus (Arbitration) trong giao thức CAN

Trong hệ thống truyền thông CAN, tất cả các node đều có khả năng chủ động quyền truyền dữ liệu bất cứ lúc nào khi bus đang ở trạng thái rỗi. Tuy nghiên, để tránh va chạm khi nhiều node cùng cố gắng truyền đồng thời, CAN áp dụng một cơ chế tranh quyền thông minh gọi là arbitration. Mục tiêu của arbitration là xác định node nào sẽ giành quyền truyền dữ liệu trước, đảm bảo tính toàn vẹn tín hiệu và không làm mất thông điệp nào trong quá trình tranh quyền.

#### Cơ chế hoạt động

Mỗi khung dữ liệu trong CAN đều có một trường Identifier (ID) dài 11bit (chuẩn CAN 2.0A) hoặc 29bit (chuẩn CAN 2.0B mở rộng). Trường này không chỉ định danh thông điệp mà còn đại diện cho mức độ ưu tiên của thông điệp đó. Quy tắc ưu tiên được xác định như sau: ID càng nhỏ, mức độ ưu tiên càng cao.

Quá trình arbitration được thực hiện từng bit một, theo cơ chế so sánh bit mức điện áp:

* Bit logic “0” được gọi là dominant (ưu tiên hơn).
* Bit logic “1” được gọi là recessive (ít ưu tiên hơn).
* Khi hai node cùng gửi, nếu một node gửi “1” mà trên bus xuất hiện “0”, node đó hiểu rằng mình đã thua và ngừng truyền ngay lập tức.

Mỗi node vừa truyền vừa lắng nghe bus (tự kiểm tra phản hồi). Nếu node phát hiện giá trị trên bus khác với bit mình gửi (ví dụ đang gửi “1” nhưng bus là “0”), node đó sẽ rút lui khỏi quá trình truyền để tránh xung đột.



Hình 1.9. Hình ảnh mô tả một tình huống tranh quyền truy cập bus giữa ba Node.

Hình ảnh dưới đây mô tả quá trình tranh quyền truy cập bus (arbitration) giữa ba node trong mạng CAN. Mỗi node đều có một khung dữ liệu frame cần gửi và bắt đầu truyền đồng thời sau khi bus trở về trạng thái lỗi. Cơ chế arbitration trong CAN đảm bảo rằng node có thông điệp quan trọng nhất (có mã định danh – ID – nhỏ nhất) sẽ được quyền truyền trước, còn các node khác sẽ tạm ngưng mà không gây va chạm trên bus.

Ba node có ID là 11bit có ID như sau:

Node 1: 11100110011

Node 2: 11100111111

Node 3: 11100110001

Từ trái sang phải, các bit ID được truyền từ bit quan trọng nhất (bit 1) đến bit ít quan trọng nhất (bit 11). Tại mỗi thời điểm,các node vừa truyền vừa theo dỗi tín hiệu trên bus. Bit “0” (dominant) sẽ ghi đè bit “1” (recessive) nếu có tranh chấp tại cùng thời điểm. Nếu một node gửi bit “1” nhưng phát hiện bus đang ở mức “0”, nó sẽ lập tức dừng truyền – nghĩa là node đó đã thua trong quá trình arbitration.

#### Phân tích từng bước

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bit số** | **Node 1** | **Node 2** | **Node 3** | **Tín hiệu trên bus** | **Kết luận** |
| 1-6 | 111001 | 111001 | 111001 | Theo dõi giống nhau tất cả tiếp tục |  |
| 7 | 1 | 1 | 0 | 0 (dominant) | Node 1 và 2 gửi 1 nhưng bus là 0 nên thua |
| 8-11 | - | - | 0001 | Node 3 tiếp tục | Node 3 thắng và truyền toàn bộ khung |

Tại bit thứ 7, Node 3 gửi bit dominant (0) trong khi Node 1 và Node 2 gửi bit recessive (1). Do tín hiệu thực tế trên bus là 0, cả Node 1 và Node 2 đều phát hiện mâu thuẫn và phải rút khỏi quá trình truyền. Kể từ thời điểm đó, chỉ còn Node 3 tiếp tục truyền phần còn lại của khung dữ liệu.

#### Nhận xét

Qua quá trình arbitration, có thể rút ra các đặc điểm sau:

* Cơ chế tranh quyền diễn ra bit-by-bit, bắt đầu từ bit quan trọng nhất của ID.
* Node có ID nhỏ nhất (nhiều bit 0 hơn ở phía đầu) sẽ luôn có ưu tiên cao hơn.
* Arbitration trong CAN không làm mất dữ liệu, chỉ hoãn truyền đối với các node có ưu tiên thấp hơn.
* Quá trình này diễn ra hoàn toàn trong phần cứng, đảm bảo tốc độ và độ tin cậy cao trong hệ thống thời gian thực.

## Tổng quan về chuẩn truyền thông CANopen

### Giới thiệu về chuẩn truyền thông CANopen

CANopen là một giao thức truyền thông dựa trên mạng CAN (Controller Area Network), được phát triển đặc biệt cho các ứng dụng trong tự động hóa công nghiệp, ô tô, và các hệ thống điều khiển thiết bị. CANopen được thiết kế để sử dụng trong các môi trường yêu cầu tính ổn định cao, khả năng chịu tải lớn và độ tin cậy cao.

Giao thức này được xây dựng để điều khiển các thiết bị trong mạng CAN, cho phép các thiết bị trong hệ thống giao tiếp với nhau một cách hiệu quả và đồng bộ.

Ảnh có chứa văn bản, thiết kế, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình 1.10. Ứng dụng CANopen trong điều khiển thiết bị công nghiệp

Trong lĩnh vực truyền thông công nghiệp, có nhiều giao thức khác nhau được sử dụng để đảm bảo việc trao đổi dữ liệu giữa các thiết bị một các nhanh chóng và tin cậy. Tuy nhiên, mỗi giao thức đều có ưu nhược điểm riêng tùy theo yêu cầu ứng dụng. Để lựa chọn giao thức phù hợp cho hệ thống điều khiển động cơ BLDC, cần tiến hành phân tích và so sánh CANopen với một số giao thức phổ biến khác như Modbus, Profibus và EtherCAT.

Bảng 1.3. Bảng so sánh các giao thức truyền thông công nghiệp

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tiêu chí** | **CANopen** | **Modbus RTU** | **Profibus** | **EtherCAT** |
| Tầng vật lí | Bus CAN (2 dây, chống nhiễu cao) | RS-485 (2 dây, chống nhiều trung bình) | RS-485 cải tiến  (cáp đặc biệt) | Ethernet (cáp đôi xoắn hoặc cáp quang) |
| Mô hình truyền thông | Multi-master, tranh quyền bus | Master-Slave | Master-Slave | Master-Slave tối ưu |
| Tính thời gian thực | Tốt (ms-level) | Trung bình (10-100ms) | Tốt (vài ms) | Rất tốt (us-level) |
| Độ tin cậy | Cao | Trung bình | Cao | Rất cao |
| Khả năng mở rộng mạng | 127 node lí thuyết | 32 thiết bị mỗi đoạn RS-485 | Hàng trăm node | Hàng nghìn node |
| Ứng dụng | Điều khiển động cơ, robot di động, y tế | Đọc cảm biến, scada đơn giản | Dây chuyền sản xuất lớn | Máy CNC, robot, tự động hóa cao cấp |

### Cấu trúc của CANopen

Giao thức CANopen được mô tả trong mô hình OSI, chủ yếu bao gồm lớp ứng dụng và lớp giao tiếp, trong khi các lớp liên kết dữ liệu và lớp vật lý được kế thừa từ giao thức CAN. Điều này giúp CANopen tập trung vào quy trình giao tiếp và ứng dụng, còn các lớp cơ sở đảm nhận việc truyền dẫn và đảm bảo tính ổn định mạng.



Hình 1.11. Các lớp giao thức truyền thông CANopen

#### Physical Layer

Lớp vật lý của CANopen dựa trên mạng CAN (ISO 11898), kết nối các nút qua bus chung sử dụng cáp xoắn đôi vi sai với hai tín hiệu CANH và CANL. Mạng yêu cầu trở kháng 120Ω ở mỗi đầu bus để đảm bảo tín hiệu chính xác và giảm nhiễu, giúp duy trì kết nối ổn định.



Hình 1.12. Lớp vật lý

#### Data Link Layer

Lớp liên kết dữ liệu (Data Link Layer) của CANopen, cũng được tiêu chuẩn hóa trong ISO 11898, quản lý việc truyền tải dữ liệu giữa các thiết bị trên mạng CAN. Lớp này bao gồm hai phần chính:

* LLC (Logical Link Control): Đảm bảo truyền dữ liệu an toàn, kiểm tra và sửa lỗi trong quá trình truyền.
* MAC (Medium Access Control): Quản lý truy cập vào môi trường truyền thông, sử dụng quy tắc CSMA/CD để tránh xung đột khi gửi dữ liệu.

Các cơ chế này giúp đảm bảo tính chính xác và độ tin cậy trong việc truyền tải thông tin qua mạng CANopen.



Hình 1.13. Định dạng khung chuẩn CAN

##### **Sơ đồ phân bổ mã định danh**

Hệ thống phân bổ ID trong CANopen bao gồm hai thành phần chính:

* Function code: xác định loại đối tượng giao tiếp, giúp phân loại các thông điệp dựa trên chức năng cụ thể.
* Module ID: dùng để nhận diện từng thiết bị slave trong mạng, và chi tiết cấu trúc của nó được mô tả trong Bảng 2.1.

Hệ thống này cho phép master kết nối và trao đổi thông tin với tối đa 127 thiết bị slave, đảm bảo khả năng mở rộng và linh hoạt.



Hình 1.14. Sơ đồ phân bổ định danh mặc định

Bảng 1.4. Bảng ánh xạ Function Code và COB-ID trong giao thức CANopen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Đối tượng** | **Function Code**  **(binary)** | **Kết quả COB-ID** | | **Thông số truyền thông tại chỉ mục** |
| NPM | 0000 | 0 |  | - |
| SYNC | 0001 | 128 | (0080h) | 1005h |
| EMERGENCY |  | 129…255 | (0081h-00FFh) | 1014h |
| PDO1(tx) | 0011 | 385…511 | (0181h-01FFh) | 1800h |
| PDO1(rx) | 0100 | 513…639 | (0201h-027Fh) | 1400h |
| PDO2(tx) | 0101 | 641…8767 | (0281h-02FFh) | 1801h |
| PDO2(rx) | 0110 | 769…895 | (0301h-037Fh) | 1401h |
| PDO3(tx) | 0111 | 897…1023 | (0381h-03FFh) | 1802h |
| PDO3(rx) | 1000 | 1025…1151 | (0401h-047Fh) | 1402h |
| PDO4(tx) | 1001 | 1153…1279 | (0481h-04FFh) | 1803h |
| PDO4(rx) | 1010 | 1281…1407 | (0501h-057Fh) | 1403h |
| SDO1(tx) | 1011 | 1409…1535 | (0581h-05FFh) | 1200h |
| SDO1(rx) | 1100 | 1537…1663 | (0601h-067Fh) | 1200h |

#### Quy tắc CSMA/CD

CSMA/CD (Carrier Sense Multiple Access with Collision Detection) là một kỹ thuật được sử dụng trong mạng CAN để kiểm soát xung đột trong quá trình truyền dữ liệu. Mặc dù CSMA/CD chủ yếu liên quan đến CAN bus, nó vẫn đóng vai trò quan trọng trong việc quản lý cách các nút trên mạng CAN và CANopen tương tác với nhau. Dưới đây là một mô tả chi tiết về cách CSMA/CD hoạt động trong mạng CANopen.

##### **Carrier Sense (Cảm nhận tín hiệu)**

Trước khi một nút gửi dữ liệu, nó sẽ kiểm tra xem bus có đang có tín hiệu truyền từ nút khác hay không. Nếu bus đang rảnh rỗi, nút sẽ bắt đầu gửi dữ liệu. Nếu bus đang có tín hiệu, nút sẽ đợi cho đến khi bus trống rồi mới gửi.

##### **Multiple Access (Truy cập nhiều lần)**

Trong mạng CAN, nhiều nút có thể chia sẻ một bus. Khi có nhiều nút muốn gửi dữ liệu cùng lúc, mạng sẽ ưu tiên các nút có ID thấp hơn (ưu tiên cao hơn). Điều này giúp xác định nút nào có quyền gửi trước. Nếu hai nút cùng gửi dữ liệu đồng thời, nút có ID thấp hơn sẽ thắng và gửi dữ liệu trước.

##### **Collision Detection (Phát hiện xung đột)**

Nếu hai nút gửi dữ liệu cùng lúc, một xung đột sẽ xảy ra. Khi đó, các nút sẽ phát hiện sự không khớp giữa dữ liệu đã gửi và tín hiệu trên bus. Khi xung đột được phát hiện, tất cả các nút sẽ dừng lại ngay lập tức để tránh gửi dữ liệu sai.

##### **Backoff (Dừng và thử lại)**

Sau khi xung đột, các nút sẽ không gửi lại ngay mà sẽ chờ một khoảng thời gian ngẫu nhiên. Điều này giúp tránh việc các nút gửi dữ liệu cùng lúc lần nữa, giảm thiểu khả năng xung đột lặp lại.

##### **Điều khiển lỗi**

Khi xung đột xảy ra, hệ thống sẽ gửi một Error Frame để thông báo lỗi cho các nút khác. Các nút sẽ ngừng truyền, xử lý lỗi và tiếp tục hoạt động bình thường sau khi hết lỗi.

#### Application Layer

Lớp Application Layer là lớp giao tiếp cao nhất của giao thức CANopen, quản lý việc trao đổi dữ liệu và các chức năng ứng dụng giữa các thiết bị trên mạng CAN. Lớp này sử dụng các Communication Objects (COBs) và bao gồm các thành phần như Object Dictionary, PDO, SDO, NMT và EMCY. Các thành phần này giúp quản lý dữ liệu, tham số hệ thống, đồng bộ hóa thời gian, thông báo lỗi và quản lý trạng thái thiết bị.

#### Object Dictionary

**Object Dictionary** là kết nối chính của các đối tượng trong một thiết bị. Đây là nơi tất cả các đối tượng được cấu trúc theo dạng bảng rõ ràng. Việc nhóm các đối tượng được xác định bởi các đặc tả của giao thức.

Mỗi đối tượng trong Object Dictionary được định địa chỉ bằng một **index 16-bit**, được biểu thị dưới dạng số thập lục phân gồm bốn chữ số.

Bảng 1.5. Địa chỉ index của một số nhóm đối tượng

|  |  |
| --- | --- |
| **Index (hex)** | **Nhóm đối tượng** |
| 0000h | Dành riêng (không được sử dụng trong CANopen) |
| 0001 h – 009F h | Các kiểu dữ liệu tĩnh và phức tạp |
| 00A0 h – 0FFF h | Dành riêng (không được sử dụng trong CANopen) |
| 1000 h – 1FFF h | Hồ sơ giao tiếp (Ví dụ: DS 301, DS 302) |
| 2000 h – 5FFF h | Hồ sơ thiết bị dành cho nhà sản xuất |
| 6000 h – 9FFF h | Hồ sơ thiết bị được tiêu chuẩn hoá |
| A000 h – FFFF h | Dành riêng (không được sử dụng trong CANopen) |

#### Các đối tượng giao tiếp

Các đối tượng giao tiếp CANopen được định nghĩa thông qua các dịch vụ và giao thức, giúp quản lý và trao đổi dữ liệu giữa các thiết bị trong mạng. Các đối tượng này được phân loại cụ thể như bảng sau:

Bảng 1.6. Các đối tượng giao tiếp trong CANopen

|  |  |
| --- | --- |
| **Các đối tượng giao tiếp** | |
| Process Data Objects (PDO) |  |
| Service Data Objects (SDO) |  |
| Special Function Objects Synchronization Objects (SYNC) | Time Stamp Objects |
| Emergency Objects (EMCY) |
| Network Management Objects | NMT Message |
| Node Guarding Object |

##### **Đối tượng SDO**

Service Data Objects (SDOs) được sử dụng để truyền dữ liệu không thời gian thực, thường là cấu hình và tham số hóa thiết bị. SDO cho phép truy cập vào các mục trong Object Dictionary của thiết bị. Một SDO sử dụng hai khung dữ liệu CAN (CAN Data Frames) với các định danh khác nhau để đảm bảo giao tiếp được xác nhận (confirmed communication).

SDO thiết lập một kênh giao tiếp peer-to-peer giữa hai thiết bị, trong đó thiết bị sở hữu Object Dictionary được truy cập đóng vai trò là server của SDO. Một thiết bị có thể hỗ trợ nhiều SDO, nhưng ít nhất một SDO là bắt buộc và được cấu hình mặc định.



Hình 1.15. Giao tiếp SDO

Đặc điểm dữ liệu của SDO:

* Tốc độ chậm hơn: Do yêu cầu xác nhận giữa thiết bị gửi và nhận (client/server).
* Có khả năng truyền dữ liệu lớn: Dữ liệu được chia thành các gói nhỏ, không giới hạn ở 8 byte như PDO.
* Truy vấn trực tiếp Object Dictionary: SDO cho phép đọc và ghi vào các mục trong Object Dictionary của CANopen.

Truy cập đọc và ghi vào Object Dictionary của CANopen được thực hiện thông qua SDOs. Bộ chỉ định lệnh Client/Server Command Specifier chứa các thông tin sau:

* Tải xuống/tải lên (download/upload),
* Yêu cầu/phản hồi (request/response),
* Truyền phân đoạn/nhanh (segmented/expedited transfer),
* Số byte dữ liệu (number of data bytes),
* Chỉ báo kết thúc (end indicator),
* Bit chuyển đổi xen kẽ cho mỗi phân đoạn tiếp theo (alternating toggle bit).

Trong một mạng CANopen, có thể sử dụng tối đa 256 kênh SDO, mỗi kênh yêu cầu hai định danh CAN.



Hình 1.16. Truy cập vào Object Dictionary

**Vai trò của SDO:**

* Truy cập dữ liệu không thời gian thực:Cho phép đọc/ghi tham số cấu hình hoặc giá trị lớn hơn 8 byte.
* Thiết lập cấu hình thiết bị: Dùng để cài đặt hoặc hiệu chỉnh thiết bị qua Object Dictionary.
* Hỗ trợ client/server: Tương tác trực tiếp giữa hai thiết bị với xác nhận rõ ràng.

Ứng dụng:

* Đọc/ghi tham số cấu hình như ID thiết bị, tốc độ truyền CAN, hoặc giá trị hiệu chỉnh.

##### **Đối tượng PDO**

Process Data Object (PDO) truyền dữ liệu real-time giữa các thiết bị trong mạng CANopen, như trạng thái cảm biến hoặc tín hiệu điều khiển. PDO hoạt động theo mô hình producer/consumer, nơi dữ liệu được truyền từ một thiết bị (producer) đến một hoặc nhiều thiết bị (consumer) qua broadcast. PDO không yêu cầu xác nhận (unacknowledged communication), với TxPDO của producer tương ứng với RxPDO của thiết bị nhận.



Hình 1.17. Giao tiếp PDO

**Có hai loại PDO:**

* Write PDO được ánh xạ vào một CAN Data frame duy nhất.
* Read PDO được ánh xạ vào một CAN Remote Frame, và sẽ nhận phản hồi qua CAN Data Frame tương ứng.

Đặc điểm dữ liệu của PDO:

* + Payload nhỏ: Trường dữ liệu của một PDO có thể chứa tối đa 8 byte dữ liệu quá trình.
  + Tốc độ nhanh: Không cần xác nhận (acknowledge), sử dụng giao tiếp broadcast.
  + Không yêu cầu truy vấn: PDO được cấu hình trước và được truyền khi có sự kiện (Event-Driven) hoặc theo chu kỳ (Cyclic).

Cấu trúc ánh xạ PDO:

* Kiểu dữ liệu và ánh xạ của các đối tượng ứng dụng vào PDO được định nghĩa bởi cấu trúc ánh xạ PDO trong Object Dictionary, được mô tả tại:
* "1600h": Dành cho R\_PDO đầu tiên.
* "1A00h": Dành cho T\_PDO đầu tiên.
* Mạng CANopen có thể hỗ trợ tối đa 512 T\_PDO và 512 R\_PDO.



Hình 1.18. Write PDO và Read PDO

**Hồ sơ giao tiếp CANopen phân biệt ba chế độ kích hoạt thông điệp:**

Kích hoạt theo sự kiện (Event-driven): Việc truyền thông điệp được kích hoạt bởi sự kiện cụ thể của đối tượng được xác định trong hồ sơ thiết bị.

Thăm dò bằng khung yêu cầu từ xa (Polling by remote frames):PDO không đồng bộ được truyền khi nhận yêu cầu từ xa từ thiết bị khác.

Đồng bộ hóa (Synchronized): Các PDO đồng bộ được kích hoạt khi hết khoảng thời gian truyền cụ thể, được đồng bộ hóa bằng việc nhận đối tượng SYNC.

**Vai trò của PDO trong mạng CANopen:**

Truyền dữ liệu real-time: Đảm bảo việc trao đổi dữ liệu nhanh chóng giữa các thiết bị.

Hỗ trợ đồng bộ hóa: Đồng bộ dữ liệu thông qua SYNC Object để đảm bảo tính nhất quán.

Phát dữ liệu: Cho phép một thiết bị (producer) truyền dữ liệu đến nhiều thiết bị khác (consumer).

Liên kết Object Dictionary: Làm giao diện giữa dữ liệu quá trình và các mục trong Object Dictionary.

Hỗ trợ chế độ truyền: Bao gồm Event-driven, Polling, và Synchronized để phù hợp với từng ứng dụng.

**Ứng dụng:**

Điều khiển động cơ, đọc tín hiệu cảm biến, hoặc gửi tín hiệu báo lỗi nhanh chóng.

##### **Đối tượng SYNC**

SYNC Producer gửi tín hiệu đồng bộ cho các SYNC Consumer, giúp các thiết bị bắt đầu thực hiện nhiệm vụ đồng bộ theo đúng thời gian. Điều này đảm bảo tất cả các thiết bị trong mạng CANopen thực hiện công việc đúng thời điểm, đặc biệt với cảm biến và bộ chấp hành.



Hình 1.19. Cơ chế đồng bộ SYCN

Truyền đồng bộ của một PDO có nghĩa là việc truyền dữ liệu được cố định về mặt thời gian liên quan đến việc truyền đối tượng SYNC. PDO đồng bộ được truyền trong một cửa sổ thời gian nhất định ("synchronous window length") liên quan đến việc truyền SYNC và, tối đa là một lần cho mỗi chu kỳ của SYNC. Thời gian giữa các đối tượng SYNC được chỉ định bởi tham số "communication cycle period".

CANopen phân biệt các chế độ truyền sau:

* Truyền đồng bộ: PDO đồng bộ được truyền trong cửa sổ đồng bộ sau tín hiệu SYNC, với ưu tiên cao hơn PDO bất đồng bộ
* Truyền bất đồng bộ: PDO bất đồng bộ và SDO có thể được truyền bất kỳ lúc nào, tùy vào mức độ ưu tiên, và chúng cũng có thể xuất hiện trong cửa sổ đồng bộ.

##### **Đối tượng EMCY**

Các thông điệp khẩn cấp (Emergency messages) được kích hoạt khi có lỗi nghiêm trọng bên trong thiết bị. Chúng được truyền từ thiết bị gặp sự cố đến các thiết bị khác với mức độ ưu tiên cao, vì vậy chúng rất phù hợp cho các cảnh báo lỗi kiểu ngắt.

Một thông điệp khẩn cấp chỉ có thể được gửi một lần cho mỗi "sự kiện lỗi", tức là các thông điệp khẩn cấp không được lặp lại. Miễn là không có lỗi mới xảy ra trên thiết bị ở trạng thái "Enter Pre-Operational", không có thông điệp khẩn cấp nào cần được gửi thêm.

Các mã lỗi trong bảng lỗi cũng như thông tin cụ thể của thiết bị được chỉ định trong các hồ sơ thiết bị thông qua các mã lỗi khẩn cấp được định nghĩa theo CANopen Communication Profile.



Hình 1.20. Cơ chế truyền thông báo lỗi khẩn cấp EMCY

##### **Dịch vụ NMT (Network Management Services)**

Dịch vụ Quản lý Mạng (NMT - Network Management) trong CANopen được sử dụng để quản lý trạng thái của các thiết bị trong mạng. Các dịch vụ NMT cung cấp các cơ chế để kiểm soát và điều phối các thiết bị trong mạng CANopen, đảm bảo rằng hệ thống hoạt động trơn tru và đồng bộ.

CANopen sử dụng mô hình master/slave để quản lý mạng, với một thiết bị thực hiện chức năng NMT Master và các thiết bị còn lại là NMT Slaves.

**Quản lý mạng cung cấp các nhóm chức năng sau:**

* Dịch vụ điều khiển mô-đun (Module Control Services) để khởi tạo các NMT Slaves muốn tham gia vào ứng dụng phân tán.
* Dịch vụ điều khiển lỗi (Error Control Services) để giám sát trạng thái giao tiếp của các nút và mạng.
* Dịch vụ điều khiển cấu hình (Configuration Control Services) cho phép tải lên và tải xuống dữ liệu cấu hình từ và đến mô-đun mạng.

Một NMT Slave đại diện cho phần của một nút chịu trách nhiệm về chức năng NMT của nút đó. Nó được xác định duy nhất bởi mã số mô-đun (module ID).



Hình 1.21. Cấu trúc mạng NMT

Các thiết bị NMT Slave của CANopen triển khai một máy trạng thái tự động đưa mỗi thiết bị vào trạng thái “Pre-Operational” khi được cấp nguồn và khởi tạo.

Trong trạng thái “Pre-Operational”, nút có thể được cấu hình và tham số hóa thông qua SDO (ví dụ: sử dụng công cụ cấu hình), nhưng giao tiếp PDO không được phép. NMT Master có thể chuyển từ trạng thái “Pre-Operational” sang “Operational” và ngược lại.

Trong trạng thái “Operational”, giao tiếp PDO được phép và thiết bị có thể thực hiện các hành vi ứng dụng cụ thể, như được định nghĩa trong hồ sơ thiết bị. Object Dictionary có thể được truy cập qua SDO. Tuy nhiên, một số đối tượng có thể bị tắt hoặc chỉ cho phép đọc khi thiết bị đang hoạt động.

Khi thiết bị chuyển sang trạng thái “Stopped”, giao tiếp PDO và SDO sẽ ngừng hoạt động.



Hình 1.22. Trạng thái NMT Slave

Bảng 1.7. Các lệnh phổ biến trong NMT

|  |  |  |  |
| --- | --- | --- | --- |
| **Lệnh NMT** | **Mã lệnh (Command)** | **Trạng thái sau khi thực thi** | **Mô tả** |
| Start Remote Node | 0x01 | Operational | Chuyển thiết bị sang trạng thái *Operational*, bắt đầu truyền PDO. |
| Stop Remote Node | 0x02 | Stopped | Dừng thiết bị, chỉ nhận lệnh NMT, không truyền PDO hay SDO. |
| Enter  Pre-Operational | 0x80 | Pre-Operational | Đưa thiết bị vào trạng thái *Pre-Operational* để cấu hình hoặc chuẩn bị hoạt động. |
| Reset Communication | 0x82 | Initialization  (Pre-Operational) | Reset giao tiếp của thiết bị, tính toán lại các định danh và gửi thông điệp khởi động. |
| Reset Node | 0x81 | Initialization  (Pre-Operational) | Reset toàn bộ thiết bị, tương đương với khởi động lại. |

**Ứng dụng:**

* Sử dụng trong các hệ thống tự động hóa công nghiệp để đảm bảo tất cả thiết bị được quản lý và hoạt động nhất quán.
* Khả năng reset node từ xa giúp khắc phục sự cố mà không cần can thiệp vật lý vào thiết bị.

# THIẾT KẾ HỆ THỐNG TRUYỀN THÔNG

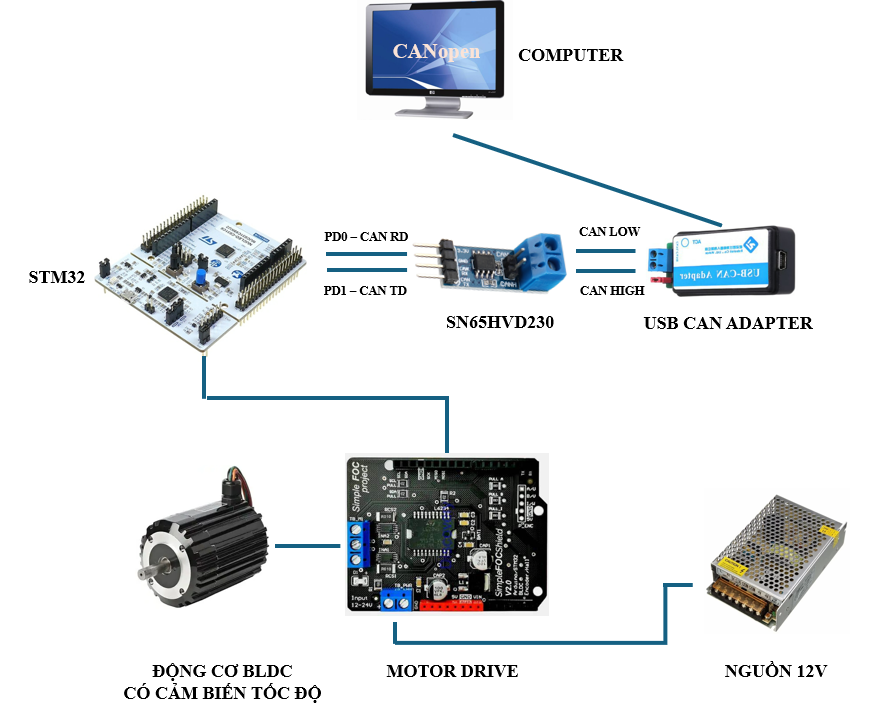
## Sơ đồ kết nối phần cứng

### Giới thiệu tổng quan

Hệ thống điều khiển động cơ BLDC sử dụng chuẩn truyền thông CANopen được thiết kế để đảm bảo khả năng truyền dữ liệu ổn định, thời gian thực, và khả năng giám sát, điều khiển từ xa. Hệ thống bao gồm nhiều thành phần phần cứng và phần mềm được tích hợp và hoạt động đồng bộ nhằm đảm bảo khả năng điều khiển chính xác và phần hồi thời gian thực, với mục tiêu chính là điều khiển tốc độ, dòng điện hoặc vị trí của động cơ một cách chính xác thông qua mạng CAN.

### Sơ đồ kết nối phần cứng

Hệ thống điều khiển động cơ BLDC sử dụng giao thức truyền thông CANopen được mình họa trong Hình 4.1 dưới đây. Các thành phần chính bao gồm:



Hình 2.1. Sơ đồ hệ thống điều khiển động cơ BLDC sử dụng giao thức CANopen

Bảng 2.1. Danh sách các thành phần chính trong hệ thống

|  |  |  |
| --- | --- | --- |
| **STT** | **Thành phần** | **Chức năng/ Vai trò** |
| 1 | Máy tính | Phần mềm tự phát triển bằng QT. Gửi lệnh điều khiển hoặc truy vấn dữ liệu từ xa |
| 2 | USB-CAN Adapter | Chuyển đổi tín hiệu USB từ máy tính sang giao thức CAN vật lý để giao tiếp với vi điều khiển |
| 3 | SN65HVD230 (CAN Transceiver) | Bộ chuyển đổi mức điện áp giữa vi điều khiển và mạng CAN. Đảm bảo truyền thông ổn định trên bus CAN |
| 4 | STM32 Nucleo-F446RE | Đóng vai trò như một Slave Node trong hệ CANopen. Xử lý lệnh điều khiển, phản hồi dữ liệu tốc độ/động cơ, và điều khiển Motor Driver. |
| 5 | Motor Driver (SimpleFOC) | Nhận tín hiệu điều khiển từ STM32 và cấp dòng điện phù hợp cho các cuộn dây của động cơ BLDC để tạo mô-men quay. |
| 6 | Động cơ BLDC có cảm biến | Động cơ không chổi than có tính hợp cảm biến tốc độ (Hall Sensor) giúp phản hồi vị trí và tốc độ về STM32 |
| 7 | Nguồn DC 12V | Cung cấp nguồn cho Motor Driver và động cơ BLDC. |

### Phân tích phần cứng của hệ thống

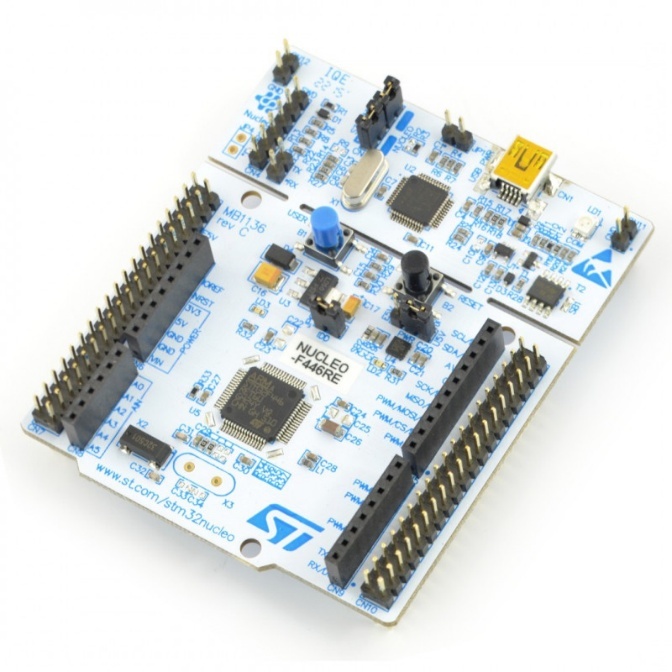
Trong hệ thống truyền thông CANopen phục vụ điều khiển động cơ BLDC, phần cứng đóng vai trò vô cùng quan trọng. Các thành phần phần cứng không chi đảm bảo khả năng kết nối và truyền dữ liệu một cách ổn định mà còn giúp hệ thống hoạt động chính xác, an toàn theo thời gian thực.

Ở mục này, chúng ta sẽ đi sâu vào từng thành phần phần cứng chính trong hệ thống. Việc lựa chọn và cấu hình đúng các phần từ phần cứng là tiền đề quan trọng để đảm bảo quá trình thiết lập và vận hành mạng CANopen diễn ra thuận lợi.

#### Vi điều khiển STM32F446RE

Vi điều khiển STM32F446RE là bộ xử lí trung tâm trong hệ thống. Đây là dòng vi điều khiển hiệu năng cao của STMMicrolectronics, sử dụng nhân ARM Cortex-M4 với bộ xử lý tín hiệu số DSP và đơn vị tính toán dấu chấm động FPU, rất phù hợp với các ứng dụng điều khiển động cơ.

Một trong những lý do lựa chọn STM32F446RE là vì nó tích hợp sẵn bộ điều khiển CAN, cho phép thiết lập giao tiếp truyền thông CAN dễ dàng mà không cần thêm chip điều khiển ngoài.



Hình 2.2. STM32F446RE

Bảng 2.2. Thông số kỹ thuật của STM32F446RE

|  |  |
| --- | --- |
| **Thông số kỹ thuật** | **Giá trị** |
| Loại vi xử lí | ARM Cortex-M4 (tích hợp FPU và DSP) |
| Tốc độ xử lý | Tối đa 180 MHz |
| Bộ nhớ Flash | 512KB |
| Bộ nhớ SRAM | 128KB |
| Số chân GPIO | Khoảng 50 chân có thể lập trình |
| ADC | 3 ADC 12-bit, tối đa 16 kênh |
| Timer | 14 Timer |
| Giao tiếp | 3xSPI, 3xI2C, 4xUART, 2x CAN, USB OTG FS |
| Điện áp hoạt động | 3.3V (cấp qua USB hoặc VIN) |
| Lập trình và debug | Cổng MicroUSB tích hợp ST-link/V2-1 |
| Tương thích | Hỗ trợ STM32CubeMX, Arduino |
| CAN tích hợp | Có sẵn 2 kênh bxCAN (CAN1 và CAN2) |

#### USB CAN Adapter

USB CAN Adapter là một thiết bị ngoại vi được sử dụng để chuyển đổi giao tiếp giữa chuẩn USB (trên máy tính) và chuẩn truyền thông CAN, cho phép máy tính có thể gửi và nhận dữ liệu qua mạng CAN. Thiết bị này đóng vai trò rất quan trọng trong việc giám sát, cấu hình hoặc kiểm thử giao tiếp CAN thông qua các phần mềm hỗ trợ như CANopenTool, PCAN View,…

A close up of a device

Description automatically generated

Hình 2.3. USB CAN Adapter

Bảng 2.3. Thông số kỹ thuật của USB CAN Adapter

|  |  |
| --- | --- |
| **Thông số** | **Giá trị/ Mô tả** |
| Chuẩn giao tiếp | USB 2.0 |
| Giao thức CAN | CAN 2.0A và CAN 2.0B |
| Tốc độ truyền CAN | 5Kbps đến 1Mbps |
| Kết nối CAN | CAN\_H và CAN\_L |
| Điện áp hoạt động | Lấy nguồn trực tiếp cổng USB |
| Ứng dụng | Giám sát mạng CAN, giao tiếp STM32 và cấu hình thiết bị CANopen |

#### SN65HVD230

SN65HVD230 là một CAN transceiver (bộ chuyển đổi tín hiệu CAN) dùng để giao tiếp giữa vi điều khiển như STM32 và busCAN. Thiết bị này chuyển đổi mức tín hiệu logic TTL (thường là 3.3V hoặc 5V) sang tín hiệu vi sai phù hợp với chuẩn CAN, đồng thời cũng thực hiện việc thu nhận tín hiệu CAN và chuyển về mức logic cho vi điều khiển đọc. Transceiver này được sử dụng rộng rãi trong các ứng dụng giao tiếp CAN vì giá thành rẻ, dễ tích hợp và có hiệu suất ổn định.

A close-up of a blue circuit board

Description automatically generated

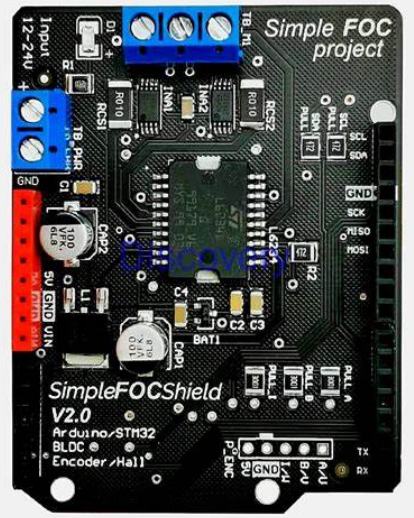
Hình 2.4. SN65HVD230

Bảng 2.4. Thông số kỹ thuật của SN65HVD230

|  |  |
| --- | --- |
| **Thông số** | **Giá trị/ Mô tả** |
| Điện áp hoạt động | 3.3V |
| Giao thức hỗ trợ | CAN 2.0A và CAN 2.0B |
| Tốc độ truyền dữ liệu | Tối đa là 1Mbps |
| Dòng tiêu thụ | <10 mA |
| Ứng dụng | Giao tiếp STM32 với CANbus |

#### Driver MKS SimpleFOC Shield V2.0.4

SimpleFOC Shield là một bo mạch mở rộng được thiết kế điều khiển động cơ BLDC (Brushless DC) hoặc gimbal bằng phương pháp Field-Oriented Control (FOC). Bo mạch này được phát triển với mục tiêu đơn giản hóa việc điều khiển động cơ một cách hiệu quả, linh hoạt và dễ sử dụng.



Hình 2.5. MKS SimpleFOC Shield V2.0.4

Bảng 2.5. Thông số kỹ thuật của Motor Driver FOC

|  |  |
| --- | --- |
| **Thông số** | **Mô tả** |
| Điện áp đầu vào | 12V – 35VDC |
| Dòng điện tối đa | 5A |
| Công suất tối đa | 120W |
| Chip điều khiển động cơ | L6234 |
| Cảm biến dòng | INA240 |
| Tương thích với bo mạch | Arduino UNO, STM32 Nucleo |
| Giao tiếp hỗ trợ | SPI, I2C, Analog |

Kết nối phần cứng:

* **Kết nối phần cứng với STM32**

Shield sử dụng chuẩn chân Arduino Uno R3, có thể cắm trực tiếp lên các bo vi điều khiển STM32 Nucleo-64.

Tương thích với hầu hết các chân giao tiếp như PWM, SPI, UART.

* **Kết nối phần cứng với động cơ**

Shield cung cấp ba cổng kết nối đầu ra (U, V, W) để cấp dòng điện điều khiển cho các pha của động cơ BLDC.

* **Kết nối phần cứng với cảm biến**

Hỗ trợ các cảm biến như encoder quay, cảm biến Hall hoặc cảm biến từ để đo lường vị trí và tốc độ của rotor.

Cổng giao tiếp của cảm biến thường dùng giao thức SPI hoặc I2C.

* **Kết nối phần cứng với nguồn cấp**

Shield yêu cầu một nguồn cấp riêng 12V đến 24V để cung cấp điện áp cho động cơ và driver.

* **Kết nối chân cắm**
* PWM1, PWM2, PWM3: Tín hiệu PWM để điều khiển các pha U, V, W của động cơ.
* SEN1, SEN2, SEN3: Chân cảm biến dòng để đo dòng điện qua các pha.
* SPI/I2C/UART: Giao tiếp với vi điều khiển hoặc cảm biến vị trí.
* GND và VIN: Nguồn cấp cho driver và động cơ.

#### Cảm biến vị trí AS5147U

AS5047U là một cảm biến từ Hall sử dụng công nghệ CMOS. Các cảm biến Hall này chuyển đổi thành phần từ trường vuông góc với bề mặt chip thành tín hiệu điện áp.

Tín hiệu từ các cảm biến Hall được khuếch đại và lọc bởi khối xử lý tín hiệu tương tự phía trước (AFE - Analog Front-End), trước khi được chuyển đổi bởi bộ chuyển đổi tương tự–số (ADC).

Dữ liệu đầu ra của ADC được xử lý bởi khối CORDIC (Coordinate Rotation Digital Computer) phần cứng để tính toán góc và biên độ vector từ. Biên độ từ trường này được sử dụng bởi khối điều khiển tự động hệ số khuếch đại (AGC) để điều chỉnh mức độ khuếch đại, nhằm bù lại các biến đổi do nhiệt độ và biến thiên từ trường.

AS5047U liên tục tạo ra thông tin góc, thông tin này có thể được truy xuất thông qua các giao diện khác nhau của thiết bị. Độ phân giải nội bộ 14-bit có thể đọc được thông qua thanh ghi qua giao tiếp SPI. Ngoài ra, độ phân giải tại đầu ra ABI có thể được lập trình từ 10 đến 14 bit.

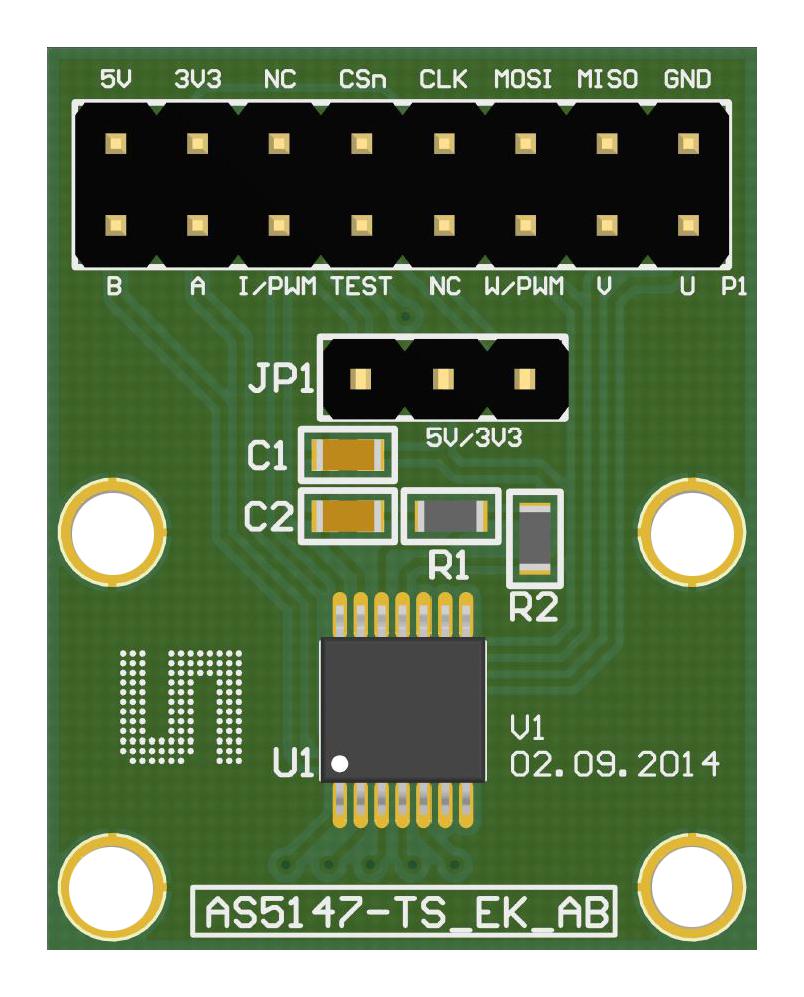
Khối Bù sai số góc động (Dynamic Angle Error Compensation) sẽ hiệu chỉnh góc được tính toán để bù lại độ trễ bằng thuật toán dự đoán tuyến tính. Ở tốc độ quay không đổi, thời gian trễ sẽ được AS5047U bù trừ nội bộ, giúp giảm sai số góc động trên các đầu ra SPI, ABI và UVW.

Một bộ lọc thích ứng (adaptive filter) được tích hợp sau khối bù trễ để giảm nhiễu chuyển tiếp tại tốc độ quay thấp. Thông tin ổn định sau lọc sẽ có sẵn tại các đầu ra SPI, ABI và UVW.

AS5047U cho phép lựa chọn giữa đầu ra UVW hoặc giao diện mã hóa xung PWM tại chân W.

Các thiết lập không thay đổi sau khi cấp nguồn (non-volatile settings) có thể được lập trình qua giao tiếp SPI, không cần bộ lập trình chuyên dụng.

AS5047U hỗ trợ các ứng dụng tốc độ cao lên tới 28.000 vòng/phút (28krpm).



Hình 2.6. Magnetic Encoder Development Board

###### **Khung truyền nhận SPI 16 bit**

Ảnh có chứa văn bản, ảnh chụp màn hình, hàng, số

Nội dung do AI tạo ra có thể không chính xác.

Hình 2.7. Khung truyền nhận 16 bit

**Khung dữ liệu 16-bit (16-bit Command Frame)**

Một khung dữ liệu gồm 16 bit được gửi đến AS5047U qua giao tiếp SPI để thực hiện các thao tác như đọc hoặc ghi thanh ghi. Cấu trúc của khung lệnh này thường bao gồm:

* 1 bit R/W: Xác định thao tác là đọc 1 hay là ghi 0.
* 14 bit địa chỉ thanh ghi: Xác định thanh ghi muốn truy cập.
* 1 bit kiểm tra lỗi (Parity): Dùng để kiểm tra tính toàn vẹn dữ liệu.

Bảng: Bảng mô tả thanh ghi bộ nhớ khả biến

|  |  |  |  |
| --- | --- | --- | --- |
| **Địa chỉ** | **Tên thanh ghi** | **Giá trị mặc định** | **Mô tả** |
| 0x0000 | NOP | 0x0000 | Không thực hiên thao tác nào |
| 0x0001 | ERRFL | 0x0000 | Thanh ghi lỗi |
| 0x0003 | PROG | 0x0000 | Thanh ghi lập trình |
| 0x3FF5 | DIA | 0xX3C2 hoặc 0xXBC2 cho chế độ 3.3V | Chuẩn đoán (DIAGNOSTIC) |
| 0x3FF9 | AGC | 0x0000 | Giá trị AGC (Tự động điều chỉnh độ lợi) |
| 0x3FFA | Sin-data | 0x0000 | Dữ liệu thô kênh sin |
| 0x3FFB | Cos-data | 0x0000 | Dữ liệu thô kênh cosin |
| 0x3FFC | VEL | 0x0000 | Vận tốc |
| 0x3FFD | MAG | 0x0000 | Biên độ CORDIC |
| 0x3FFE | ANGLEUNC | 0x0000 | Góc đo được không bù lỗi góc động |
| 0x3FFF | ANGLECOM | 0x0000 | Góc đo được đã bù lỗi góc động |
| 0x00D1 | ECC\_Checksum | 0x0000 | Checksum ECC được tính theo cấu hình thực tế của thanh ghi |

**Khung dữ liệu 16-bit (16-Bit Data Frame)**

Khung dữ liệu gồm 16 bit được AS5047U gửi trả lại qua giao tiếp SPI sau khi nhận được một khung lệnh hợp lệ. Cấu trúc của khung dữ liệu này thường bao gồm:

* 14 bit dữ liệu: Chứa thông tin cần truyền, ví dụ như giá trị góc.
* 1 bit lỗi (Error flag): Cho biết nếu có lỗi xảy ra trong quá trình truyền hoặc xử lý.
* 1 bit kiểm tra chẵn lẻ (Parity bit): Được sử dụng để phát hiện lỗi trong dữ liệu.

###### **Nguyên lí đo góc**

AS5147U sử dụng **hiệu ứng Hall từ tính theo trục dọc (Vertical Hall Effect)** để đo từ trường phát ra từ một **nam châm vĩnh cửu hình trụ** được gắn thẳng hàng với trục quay. Cảm biến chứa mảng các phần tử Hall đặt theo 2 trục X-Y, cho phép cảm nhận thành phần từ trường dọc phát sinh từ nam châm quay.

Góc quay (θ): Tính bằng công thức

 (4.1)

Trong đó:

* n: Số lượng xung
* PPR (Pulses Per Revolution) là số lượng xung phát ra cho mỗi vòng quay của trục.

###### **Nguyên lí đo tốc độ**

Cảm biến AS5147U **không trực tiếp đo tốc độ góc (angular velocity)** mà chỉ cung cấp giá trị **góc tuyệt đối** tại mỗi thời điểm. Để tính tốc độ quay, hệ thống phải **tính toán vi sai góc giữa hai lần đọc gần nhau** và chia cho khoảng thời gian đo:

Tốc độ góc (ω): Được tính bằng công thức:



hoặc tính theo **radian** nếu cần thiết:



* : hiệu số góc giữa hai lần đo liên tiếp
* : Thời gian đo

Số lượng xung nhận được là tổng số xung trong một khoảng thời gian đo.

Thời gian đo có thể là một giây hoặc một khoảng thời gian ngắn hơn tuỳ vào yêu cầu đo tốc độ.

Tốc độ góc tính được từ encoder sẽ cho phép xác định được tốc độ quay của động cơ hoặc hệ thống cơ khí mà encoder đang giám sát.

###### **Sơ đồ kết nối**

A diagram of a power supply

Description automatically generated

Hình 2.8. Sơ đồ kết nối AS5147U

#### Động cơ gimbal BLDC

Động cơ sử dụng trong đề tài là loại động cơ không chổi than ba pha (BLDC) chuyên dụng cho hệ thống ổn định gimbal, có tên gọi THREE PHASE BRUSHLESS GIMBAL STA BLDC Motor, với các đặc điểm nổi bật phù hợp cho điều khiển chính xác theo phương pháp FOC.

A green motor with orange wire

Description automatically generated

Hình 2.9. Động cơ BLDC

Bảng thông số kĩ thuật của động cơ BLDC

|  |  |
| --- | --- |
| **Thông số** | **Giá trị** |
| Điện áp định mức ​ | 7.4 V |
| Tốc độ không tải ​ | 2500 rpm |
| Dòng không tải | 120 mA |
| Tốc độ định mức ​ | 1934 rpm |
| Mô-men xoắn định mức ​ | 7.47 mN·m |
| Dòng định mức | 0.410 A |
| Công suất định mức ​ | 1.51 W |
| Mô-men xoắn khi đứng yên ​ | 33.0 mN·m |
| Dòng khi đứng yên ​ | 1.4 A |
| Công suất đầu ra tối đa (lý thuyết) ​ | 2.2 W |
| Hiệu suất tối đa | 49.9 % |

Bảng đặc tính cơ điện của động cơ BLDC

|  |  |
| --- | --- |
| **Thông số** | **Giá trị** |
| Điện trở pha-pha | 5.29 Ω |
| Cảm kháng pha-pha | 1.69 mH |
| Hằng số mô-men xoắn ​ | 25.78 mN·m/A |
| Hằng số vận tốc ​ | 370 rpm/V |
| Độ dốc tốc độ/mô-men | 75.9 rpm/mN·m |
| Hằng số thời gian cơ học ​ | 26.24 ms |
| Quán tính rotor | 33.0 g·cm² |

###### **Nguyên lí hoạt động**

Động cơ BLDC gimbal là loại động cơ điện điều khiển theo tín hiệu xoay chiều ba pha, không sử dụng chổi than mà dùng cảm biến từ tính (trong đề tài sử dụng AS5147U) để xác định vị trí rotor. Trong quá trình vận hành:

* Rotor gắn nam châm vĩnh cửu quay quanh trục.
* Stator chứa các cuộn dây ba pha được điều khiển bằng sóng điện áp sin (hoặc trapezoid).
* Việc thay đổi pha dòng điện được điều chỉnh theo vị trí rotor (dò từ encoder) nhằm tạo lực điện từ quay liên tục.

###### **Ưu điểm của động cơ gimbal BLDC**

Độ mượt và chính xác cao: nhờ vào mô-men quay nhỏ, độ rung thấp – phù hợp với các ứng dụng như gimbal camera, tay máy robot, v.v.

Hiệu suất cao: do không có ma sát chổi than.

Thích hợp với điều khiển FOC: khả năng phản ứng nhanh với thuật toán điều khiển véc-tơ.

Kích thước nhỏ gọn, dễ tích hợp trong các hệ thống nhúng và cơ điện tử.

## Sơ đồ khối nguyên lí hoạt động

### Giới thiệu tổng quan

Hệ thống điều khiển động cơ BLDC sử dụng phương pháp điều khiển hướng từ thông FOC kết hợp truyền thông CANopen cho phép điều khiển và giám sát động cơ từ xa một các hiệu quả và chính xác. Hệ thống bao gồm các thành phần chính như vi điều khiển STM32, driver điều khiển động cơ, cảm biến dòng và cảm biến rotor, tất cả được đồng bộ thông qua thuật toán FOC. Ngoài ra, giao tiếp CANopen giúp kết nối hệ thống với máy tính trung tâm để gửi lệnh và nhận dữ liệu phản hồi theo thời gian thực.

### Sơ đồ khối nguyên lí hoạt động

Nguyên lí hoạt động của hệ thống được mình họa trong Hình . dưới đây. Sơ đồ này thể hiện cách các khối chức năng chính phối hợp với nhau trong quá trình điều khiển và giám sát động cơ.



Hình 2.10. Sơ đồ khối nguyên lý hoạt động của hệ thống

### Mô tả chi tiết nguyên lí hoạt động

Hệ thống bao gồm các khối chức năng chính như sau:

* PC GUI (CANopen):Đóng vai trò là trung tâm điều khiển từ xa, gửi lệnh và nhận dữ liệu phản hồi từ hệ thống động cơ qua mạng CANopen.
* USB-CAN Adapter và CAN Transceiver:Kết nối giao tiếp giữa máy tính và vi điều khiển STM32. Bộ chuyển đổi này giúp truyền tải dữ liệu qua lại giữa CAN\_H và CAN\_L và STM32.
* STM32 Nucleo F446RE:Là bộ vi điều khiển trung tâm chịu trách nhiệm nhận lệnh từ CANopen, xử lí thuật toán FOC và điều khiển động cơ BLDC. Ngoài ra, nó cũng xử lí dữ liệu phản hồi từ cảm biến dòng và cảm biến vị trí để hiệu chỉnh điều khiển theo thời gian thực.
* MKS SimpleFOC:Đảm nhận việc tạo tín hiệu điều khiển PWM để cấp cho động cơ BLDC dựa trên các thông số của STM32 và dữ liệu cảm biến.
* BLDC Motor:Động cơ không chổi than 3 pha nhận tín hiệu điều khiển từ MKS SimpleFOC và quay theo lệnh yêu cầu. Động cơ được tích hợp cảm biến vị trí (AS5147U) để đo góc rotor.
* AS5147U (Cảm biến vị trí):Cảm biến này cung cấp thông tin vị trí chính xác của rotor về STM32 qua giao tiếp SPI, giúp thuật toán FOC xác định được góc quay rotor theo thời gian thực.
* INA240 + Shunt Resistor (Cảm biến dòng):Đo dòng điện chạy qua các pha động cơ và gửi dữ liệu về STM32 để phục vụ quá trình điều khiển và bảo vệ hệ thống.
* Nguồn cấp (7.4V-12V):Cung cấp nguồn điện chính cho toàn bộ hệ thống, đảm bảo động cơ và các thành phần hoạt động ổn định.

**Nguồn tín hiệu hoạt động của hệ thống:**

Khi người dùng gửi lệnh điều khiển động cơ từ máy tính, lệnh được truyền qua mạng CANopen tới STM32. Vi điều khiển xử lý lệnh và thực hiện điều khiển thông qua driver MKS Simple FOC để xuất tín hiệu PWM điều khiển động cơ BLDC. Trong quá trình vận hành, dữ liệu về dòng điện (từ INA240) và vị trí rotor (từ AS5147U) liên tục được gửi về STM32. Vi điều khiển sử dụng các dữ liệu này để tính toán và điều chỉnh thuật toán FOC nhằm đảm bảo động cơ vận hành chính xác và hiệu quả nhất. Thông tin trạng thái sẽ được phản hồi lại từ máy tính qua CANopen để người dùng giám sát.

# PHÂN TÍCH THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN FOC

## Giới thiệu tổng thể vòng điều khiển

Hệ thống điều khiển động cơ BLDC trong đề tài sử dụng phương pháp FOC với cấu trúc ba vòng lồng nhau: vị trí, tốc độ và dòng điện. Mục tiêu là điều khiển mượt, chính xác và ổn định mô-men động cơ, phù hợp với các ứng dụng đòi hỏi độ chính xác cao như camera gimbal, robot định vị hoặc thiết bị quang học.

* Vòng vị trí: Đưa rotor đến vị trí đặt trước (θref​), đầu ra là tốc độ tham chiếu (ωref​), dùng PID vị trí.
* Vòng tốc độ:Đảm bảo tốc độ rotor đạt yêu cầu (ωref​), đầu ra là dòng Iq tham chiếu, dùng PID tốc độ.
* Vòng dòng điện: Điều khiển mo-men qua dòng Iq và khử từ dư bằng dòng Iq, đầu ra là điện áp Uq, Ud (PWM), dùng PID dòng điện.

## Điều khiển dòng điện (Id, Iq)

Điều khiển dòng điện là bước nền tảng trong FOC, với mục tiêu điều khiển chính xác mô-men động cơ. Dòng điện stator được tách ra làm hai thành phần:

* + I-d (d-axis): điều khiển từ thông (thường đặt bằng 0 với BLDC không cần kích thích từ trường phụ).
  + Iq (q-axis): điều khiển mô-men.

Quy trình điều khiển dòng gồm các bước:

1. Đo dòng pha:
   * Hai trong ba dòng pha (Ia và Ib) được đo trực tiếp qua điện trở shunt.
   * Pha còn lại được suy ra: Ic = -(Ia+Ib)
2. Biến đổi Clarke: Chuyển dòng từ hệ 3 pha sang hệ tọa độ α-β (tĩnh).



1. Biến đổi Park: Dùng góc rotor θ để chuyển từ hệ α-β sang d-q (gắn theo rotor).



1. Bộ điều khiển PID dòng: Tính sai số giữa dòng đặt và dòng đo. Từ đó tính Ud, Uq làm đầu ra điều khiển.



1. Biến đổi Park ngược + điều chế PWM:

* Chuyển Ud, Uq trở lại Uα, Uβ, rồi Uabc (ba pha).
* Áp dụng kỹ thuật điều chế PWM để tạo xung điều khiển cho ba pha động cơ.

1. Điều chế sin PWM (Sine PWM)

## Biến đổi Clark và Park

### Phép biến đổi Clark

Dòng điện ba pha của động cơ BLDC được chuyển đổi thông qua phép biến đổi Clarke thành hai dòng điện trực giao (𝑖α, 𝑖β). Phép biến đổi Clarke có thể được biểu diễn qua phương trình ().

A diagram of a straight line

Description automatically generated

Hình 3.1. Phép biến đổi Clarke

Phương trình chuyển đổi Clarke:



Trong đó, 𝑖𝛼 và 𝑖𝛽 là các dòng điện trong hệ tọa độ tĩnh trực giao, còn 𝑖𝑎, 𝑖𝑏, 𝑖𝑐 là các dòng điện của hệ ba pha. Các dòng điện vừa được chuyển đổi này được biểu diễn như dòng điện tạo mô-men xoắn và dòng điện sinh từ thông tương ứng. Mặc dù dòng điện pha đã được chuyển đổi thành các thành phần của từ thông và mô-men xoắn, nhưng các thành phần này vẫn có dạng hình sin, khiến việc điều khiển trở nên khó khăn do chúng thay đổi liên tục.

### Phép biến đổi Park

Với phép biến đổi Park, hai dòng điện xoay chiều (𝑖𝛼, 𝑖𝛽) sẽ được chuyển thành dòng điện một chiều (𝑖𝑞, 𝑖𝑑).

Phép biến đổi Park thay đổi hệ quy chiếu tĩnh từ góc nhìn của stato sang hệ quy chiếu quay từ góc nhìn của rotor, thông qua phương trình

A diagram of a straight line

Description automatically generated

Hình 3.2. Phép biến đổi Park

Phương trình chuyển đổi Park:



## Điều khiển tốc độ (vòng ngoài)

Nhận đầu vào là tốc độ đặt ω\_ref.

Tốc độ thực tế ω được tính từ chênh lệch góc qua cảm biến AS5147U:



Sử dụng PID để tính dòng Iq\_ref:



Iq\_ref được đưa vào vòng điều khiển dòng điện.

Lưu ý: Id\_ref vẫn có thể đặt bằng 0 để tối ưu mô-men.

## Điều khiển vị trí (vòng ngoài cùng)

Đầu vào là góc vị trí đặt θ\_ref.

Cảm biến AS5147U cung cấp θ thực tế.

Sai số vị trí:



Sử dụng PID để tính tốc độ tham chiếu:



Tốc độ ω\_ref sau đó đưa vào vòng điều khiển tốc độ.

## Điều chế sin PWM

Điều chế sin PWM là một kỹ thuật điều chế tín hiệu được sử dụng để tạo các tín hiệu điện áp ba pha có dạng sóng hình sin từ tín hiệu điện áp một chiều DC. Kỹ thuật này giúp động cơ hoạt động êm ái, giảm nhiễu điện từ và tổn thất năng lượng, đặc biệt phù hợp với các động cơ BLDC sử dụng trong hệ thống gimbal cần độ chính xác và mượt mà cao.

Trong sin PWM, các tín hiệu PWM điều khiển từng pha được tạo ra bằng cách so sánh sóng hình sin (dạng điện áp tham chiếu theo Id, Iq) với sóng tam giác (sóng mang). Kết quả là độ rộng xung của tín hiệu PWM thay đổi theo biên độ sóng sin, từ đó tạo ra điện áp tương ứng trên từng pha.

Sau khi có điện áp Ud, Uq:

* 1. Park ngược:



* 1. Chuyển về ba pha:



* 1. Chuẩn hóa điện áp pha: Các giá trị điện áp Ua, Ub, Uc được chuẩn hóa về dải 0-1 (hoặc -1 đến 1) tùy theo mức điện áp DC của hệ thống.
  2. Tạo sóng sin PWM: Giá trị điện áp pha sau khi chuẩn hóa được ánh xạ sang độ rộng xung PWM sử dụng chức năng PWM mode 2 của TIM1 ở chế độ Center-Aligned Mode 1.
  3. Tạo xung PWM ba pha: Các xung PWM tương ứng với từng pha (U, V, W) sẽ được xuất ra các chân điều khiển cổng MOSFET của mạch cầu ba pha trong driver (SimpleFOC Shield), từ đó tạo ra dòng điện ba pha tương ứng cho động cơ.

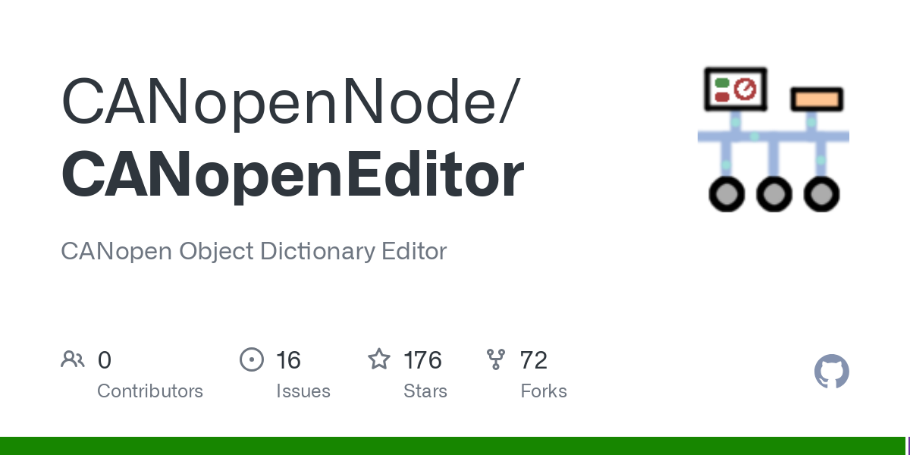
# THIẾT KẾ VÀ CẤU HÌNH OBJECT DICTIONARY BẰNG CANopenEditor

## Giới thiệu CANopenEditor

CANopenEditor là công cụ mã nguồn mở do công đồng phát triển, được dùng để tạo và quản lý Object Dictionary (OD) cho các thiết bị CANopen. OD là một thành phần cốt lõi trong kiến thức CANopen, chứa tất cả các thông tin cấu hình, dữ liệu vận hành, thông số truyền nhận (PDO, SDO) và trạng thái thiết bị.

Phần mềm CANopenEditor cung cấp giao diện trực quan, cho phép người dùng dễ dàng:

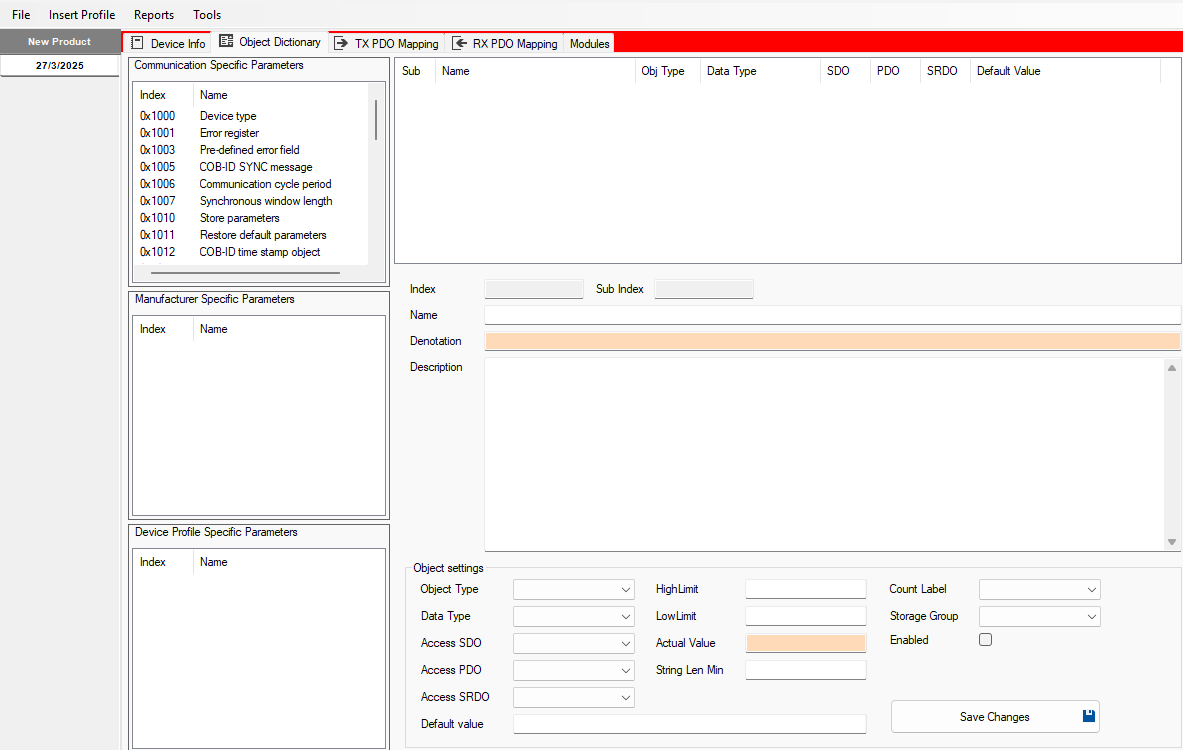
* Tạo mới hoặc chỉnh sửa các mục trong OD(entry)
* Định nghĩa các chỉ số (Index), Sub-Index, kiểu dữ liệu và quyền truy cập
* Gán mối liên hệ với PDO, SDO
* Xuất file dữ liệu dưới dạng .c, .h, .eds phục vụ cho firmware và phần mềm giám sát.



Hình 4.1. Phần mềm CANopenEditor

## Giao diện và cấu trúc cơ bản

Sau khi mở phần mềm CANopenEditor, người dùng sẽ làm việc trên một giao diện đồ hoạ trực quan, giúp dễ dàng và chỉnh sửa Object Dictionary (OD) của thiết bị CANopen.



Hình 4.3. Giao diện phần mềm CANopenEditor

Giao diện được chia thành các khu vực chính sau:

* **Thanh công cụ chính:**

****

Hình 4.4. Thanh công cụ chính của CANopenEditor

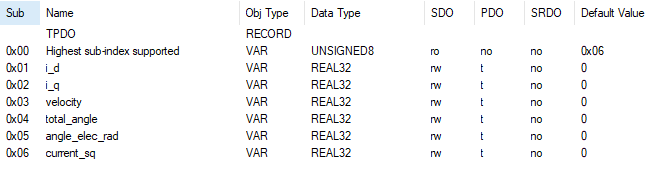
Gồm các tab chính:

* File, Insert Profile, Reports, Tools: chứa các chức năng về tạo, mở, lưu project, xuất file .c/.h .eds
* Tab làm việc
* **Device Info:** Nhập thông tin thiết bị như Vendor ID, Product Code,…
* **Object Dictionary:** Giao diện chính để xây dựng cấu trúc OD.
* **TX PDO Mapping:** Quản lý các object được ánh xạ vào TPDO
* **RX PDO Mapping:** Quản lý object ánh xạ vào RPDO
* **Modules:** Dùng cho các cấu trúc mô-đun
* **Cây Object Dictionary**

Hiển thị tất cả object đã tạo trong OD, phân chia theo từng vùng:

* **Communication Specific Parameters (0x1000 – 0x1FFF):** Các object giao thức cơ bản như Device Type, Error Register, SYNC,…
* **Manufacturer Specific Parameters (0x2000 – 0x5FFF):** Vùng do người dùng tự định nghĩa.
* **Device Profile Specific Parameters (0x6000 – 0x9FFF):** Dùng cho chuẩn hóa từng loại thiết bị
* **Danh sách Sub-Index**

Khi chọn một object kiểu RECORD hay ARRAY, danh sách Sub-Index sẽ hiển thị ở bảng bên phài: Ví dụ như object 0x2200 TPDO trong hình có các sub như:



Hình 4.5. Danh sách sub-index

* **Khu vực chỉnh sửa Object**

Khi chọn một Object, người dùng có thể sửa chi tiết:

* **Index, Sub-Index**
* **Name:** Đặt tên cho object
* **Object Settings:**
* Object Type: Variable, Record, Array
* Data Type: UNSIGNED8, INTEGER16, REAL32, …
* Default Value
* Storage Group: chọn nhóm lưu trữ
* Enabled: kích hoạt object trong xuất file

## Ứng dụng tạo Object Dictionary cho hệ thống điều khiển động cơ BLDC.

Sau khi tìm hiểu lý thuyết và giao diện phần mềm CANopenEditor, bước tiếp theo là tiến hành tạo Object Dictionary (OD) phù hợp với điều khiển và giám sát hệ thông BLDC sử dụng giao thức CANopen. OD là nói lưu trữ toàn bộ thông tin cấu hình, các tham số truyền nhận dữ liệu cũng như quyền truy cập và kiểu dữ liệu được hỗ trợ.

### Xác định các tham số cần thiết trong hệ thống điều khiển BLDC

Dựa trên thiết kế của hệ thống, các tham số chính cần truyền và nhận thông qua CANopen bao gồm:

Bảng 3.1. Các tham số truyền nhận trong hệ thống điều khiển động cơ BLDC

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| velocity | Vận tốc hiện tại của động cơ |
| total\_angel | Góc quay tổng của rotor |
| angle\_elec\_rad | Góc điện (radian) |
| i\_d | dòng điện d trong khung tọa độ dq |
| i\_q | dòng điện q trong khung tọa độ dq |
| velocity\_ref | Vận tốc tham chiếu (điều khiển từ Master) |

Các biến này sẽ được sắp xếp trong OD như sau:

* **0x2200 TPDO:** Gói dữ liệu truyền từ Slave (STM32) truyền đến Master (PC).
* **0x2000 velocity\_ref:** Biến nhận từ Master truyền Slave thông qua RPDO.

### Tạo mục TPDO trong OD

Trong phần mềm CANopenEditor, thực hiện các bước sau:

* Bước 1: Chọn mục Manufacturer Specific Parameters
* Bước 2: Thêm một entry mới:
* Index: 0x2200
* Name: TPDO
* Object Type: RECORD
* Storage Group: PERSIST\_COMM
* Bước 3: Thêm các sub-index cho TPDO:

Bảng 3.2. Bảng Sub-Index của TPDO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sub-index** | **Name** | **Data Type** | **Access** | **Ghi chú** |
| 0x00 | Highest sub-index | UNSIGNED8 | RO | Tổng số sub-index |
| 0x01 | i\_d | REAL32 | RW | dòng điện i\_d |
| 0x02 | i\_q | REAL32 | RW | dòng điện i\_q |
| 0x03 | velocity | REAL32 | RW | vận tốc hiện tại |
| 0x04 | total\_angle | REAL32 | RW | góc quay tổng |
| 0x05 | angle\_elec\_rad | REAL32 | RW | góc điện |
| 0x06 | current\_sq | REAL32 | RW | dòng q |

#### Cấu hình TXPDO Mapping

Trong hệ thống truyền thông CANopen, TXPDO là đối tượng dùng để gửi dữ liệu từ thiết bị Node lên mạng CAN.

Các Object được ánh xạ (Mapping) bao gồm:

Bảng 3.3. Cấu hình ánh xạ TXPDO cho các tham số truyền dữ liệu

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **COB-ID (hex)** | **Index/Sub-index** | **Nội dung dữ liệu truyền** |
| 1 | 180h | 0x2200/01 | Dòng điện d (i\_d) |
|  |  | 0x2200/02 | Dòng điện q (i\_q) |
| 2 | 280h | 0x2200/03 | Vận tốc – velocity |
|  |  | 0x2200/06 | current\_sq |
| 3 | 380h | 0x2200/04 | total\_angle |
|  |  | 0x2200/05 | angle\_elec\_rad |

### Tạo mục RPDO trong OD

Ta cần tạo biến velocity\_ref để điều khiển tốc độ.

* Index: 0x2000
* Name: velocity\_ref
* Object Type: VAR
* Data type: REAL32
* Access SDO: RW
* Access PDO: R (vì Master gửi, Slave nhận)
* Storage Group: PERSIST\_COMM

Đây là biến được truyền thông qua RPDO để điều khiển vận tốc động cơ từ máy tính.

#### Cấu hình RXPDO Mapping

# TRIỂN KHAI PHẦN MỀM TRÊN STM32

Trong dự án điều khiển động cơ BLDC theo phương pháp FOC vi điều khiển STM32 đóng vai trò trung tâm trong việc xử lý và điều khiển hệ thống. Các ngoại vi tích hợp trên STM32 được cấu hình để thực hiện các nhiệm vụ chính như phát xung PWM 3 pha, đo dòng điện pha, giao tiếp với encoder từ tính AS5147U qua giao thức SPI, và quản lý tín hiệu thời gian thực. Ngoài ra, để đảm bảo khả năng giao tiếp điều khiển từ xa trong hệ thống điều khiển phân tán, giao thức truyền thông CANopen cũng được triển khai trên STM32. CANopen giúp hệ thống dễ dàng trao đổi dữ liệu điều khiển và trạng thái vận hành với thiết bị chủ (master), đồng thời hỗ trợ phát hiện và xử lý lỗi một cách nhanh chóng.

Chương này trình bày chi tiết về cách cấu hình các module chính trên STM32, bao gồm:

Cấu hình TIM1 để phát xung PWM 3 pha.

Cấu hình module CAN và giao thức CANopen để đảm bảo truyền thông ổn định và hiệu quả trên busCAN.

Các phần mềm hỗ trợ khác cần thiết cho quá trình điều khiển động cơ theo thời gian thực.

## Cấu hình TIM1 – Phát xung PWM 3 pha.

### Chức năng

TIM1 là bộ định thời tiên tiến (Advanced Timer) được sử dụng để phát xung PWM 3 pha ở chế độ Center-Aligned Mode 1. Các kênh CH1, CH2 và CH3 của TIM1 lần lượt tạo ra xung PWM điều khiển ba pha của động cơ thông qua driver L6234.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.

Hình 5.1. Timer 1 Channel 1,2,3

Center-aligned mode giúp xung PWM **cân đối và đồng đều về hai phía thời gian**, giảm **nhiễu xung dòng** và tối ưu cho việc lấy mẫu dòng đúng tại điểm trung tâm của xung PWM (nơi điện áp tuyến tính nhất, ít gợn).

Ưu điểm của center-aligned mode trong điều khiển BLDC/FOC:

* Tạo ra sóng PWM dạng tam giác cân.
* Điểm kích ADC (trigger) tại đỉnh giữa xung PWM cho phép đo dòng chính xác hơn.
* Giảm EMI (nhiễu điện từ) so với Edge-aligned mode.

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

Hình 5.2. PWM Center Aligned Mode 1 và Trigger cho ADC Injected

### Cấu hình các kênh của TIM1

Chế độ PWM: PWM Mode 2 (OCxM = 0b111)

Trong PWM mode 2, đầu ra ở mức HIGH khi CNT > CCRx, và ở mức LOW khi CNT < CCRx. Phù hợp với driver như L6234 vì cho phép tạo ra xung nghịch với độ rộng điều chế được kiểm soát.

PWM mode 2 – center-aligned mode (symmetric PWM): giúp tín hiệu PWM đối xứng, giảm nhiễu, hỗ trợ chính xác thời điểm đo dòng ở giữa chu kỳ PWM.

Kích hoạt preload cho CCRx: Đảm bảo việc cập nhật giá trị CCRx không gây nhiễu tức thời, chỉ cập nhật tại điểm Update Event.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình 5.3. Cấu hình PWM Mode 2

## Cấu hình SPI – Giao tiếp với Encoder AS5147U

### Chức năng

Đọc giá trị góc rotor từ encoder từ tính AS5147U qua giao tiếp SPI (giao tiếp ở chế độ SPI Slave, CPOL = 0, CPHA = 1 theo datasheet của AS5147U).

### Cấu hình SPI

SPI Mode: Mode 1 (CPOL = 0, CPHA = 1)

Tốc độ baudrate: cấu hình để đảm bảo thời gian đọc vị trí nhanh hơn chu kỳ điều khiển (<10 MHz).

Chiều truyền: Full duplex

Data size: 16-bit (AS5147U trả về dữ liệu 14-bit góc, 2-bit CRC) và chân giao tiếp: SPIx\_MOSI, MISO, SCK, và chân CS điều khiển riêng bằng GPIO.

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

Hình 5.4. Cấu hình SPI1

### Khung truyền SPI và quá trình đọc góc

Mỗi lần đọc vị trí từ AS5147U được thực hiện bằng cách gửi một khung lệnh 16-bit đến encoder. Trong đó, 6 bit đầu là địa chỉ thanh ghi, 1 bit là R/W (1 để đọc), 1 bit parity. Sau khi gửi lệnh, dữ liệu góc sẽ được trả về trong lần truyền tiếp theo.

Quy trình đọc góc thực tế:

1. Kéo chân CS (chip select) xuống mức thấp.
2. Gửi lệnh đọc 16-bit đến địa chỉ 0x3FFF để yêu cầu đọc dữ liệu góc.
3. Kéo chân CS lên lại mức cao, kết thúc truyền.
4. Sau một chu kỳ delay ngắn (~350 ns), thực hiện một lần truyền SPI 16-bit thứ hai, để nhận lại dữ liệu góc.
5. Dữ liệu trả về được lọc parity, kiểm tra lỗi, và trích xuất 14-bit dữ liệu góc.

## Cấu hình ADC1 – Đo dòng điện 3 pha

Để thực hiện điều khiển dòng điện trong phương pháp Field-Oriented Control (FOC), cần đo chính xác dòng điện chảy qua các pha của động cơ BLDC. Trong dự án này, dòng điện được đo gián tiếp thông qua điện trở shunt được khuếch đại bởi **IC INA240**, sau đó đưa vào **ADC của STM32** để số hóa và xử lý.

### Bộ khuếch đại INA240

INA240-Q1 là một bộ khuếch đại đo dòng điện đầu ra điện áp, đạt chuẩn ô tô (automotive-qualified), với khả năng khử nhiễu PWM được cải thiện, cho phép đo sụt áp trên điện trở shunt trong một dải điện áp chế độ chung rất rộng từ –4 V đến 80 V, độc lập với điện áp nguồn cấp.

Thiết bị hoạt động với nguồn đơn từ 2.7 V đến 5.5 V, tiêu thụ dòng điện tối đa chỉ 2.4 mA. INA240-Q1 có 4 tùy chọn hệ số khuếch đại cố định: 20 V/V, 50 V/V, 100 V/V và 200 V/V.

A diagram of a supply chain

Description automatically generated

Hình 5.5. Sơ đồ kết nối INA240

### Nguyên lý đo dòng điện

Mỗi pha động cơ BLDC được nối qua một điện trở shunt có giá trị nhỏ (ví dụ: 10mΩ).

Điện áp rơi trên điện trở shunt phản ánh dòng điện tức thời của pha.

Do điện áp rơi rất nhỏ, tín hiệu này được khuếch đại tuyến tính bởi INA240 với hệ số khuếch đại (gain) khoảng 50.

Tín hiệu khuếch đại được đưa vào chân ADC analog của STM32.

### Sử dụng ADC injected mode

Trong dự án này, việc đo dòng điện được thực hiện tại điểm giữa của xung PWM (center of center-aligned PWM cycle) – thời điểm dòng điện ít bị nhiễu nhất. Để làm được điều này, ADC được cấu hình ở chế độ Injected Mode, và được kích hoạt bằng trigger từ TIM1.

Cấu hình cụ thể:

* ADC mode: Injected conversion
* Trigger source: TIM1\_TRGO (Trigger Output)
* Trigger event: Update hoặc Capture Compare (tùy theo thiết kế)
* Sampling time: Càng ngắn càng tốt nhưng vẫn đủ độ chính xác (ví dụ: 15 cycles)
* Số kênh: 2 hoặc 3 kênh tùy số lượng pha được đo (2 kênh là đủ với giả định dòng 3 pha tổng bằng 0)

### Trình tự hoạt động

1. TIM1 hoạt động ở chế độ center-aligned, tạo ra xung PWM cho 3 pha điều khiển driver L6234.
2. Ở giữa chu kỳ PWM, TIM1 sẽ tạo ra tín hiệu trigger cho ADC injected.
3. ADC1 injected sẽ tự động bắt đầu chuyển đổi trên 2 hoặc 3 kênh đã cấu hình.
4. Khi ADC hoàn tất chuyển đổi, có thể sử dụng ngắt hoặc đọc trực tiếp kết quả từ thanh ghi JDRx.
5. Sau đó, dòng điện dạng số sẽ được xử lý (lọc, chuyển đổi sang Id/Iq…).

## Cấu hình ADC2 – Đo điện áp nguồn.

### Mạch chia phân áp

Trong hệ thống điều khiển động cơ BLDC, việc giám sát điện áp nguồn là một yêu cầu quan trọng nhằm đảm bảo an toàn vận hành và điều chỉnh phù hợp thuật toán điều khiển trong các điều kiện điện áp thay đổi. Do điện áp nguồn trong hệ thống có thể dao động từ 5V đến 20V, vượt quá giới hạn đầu vào của bộ chuyển đổi tương tự – số (ADC) trên vi điều khiển Nucleo F446RE (tối đa 3.3V), cần phải sử dụng mạch chia điện áp để đưa điện áp về mức thích hợp cho ADC.

Nguyên lý hoạt động:

Mạch chia điện áp sử dụng hai điện trở mắc nối tiếp để giảm điện áp đầu vào theo công thức:



Trong đó:

* : điện áp nguồn cần đo (từ 5V đến 20V)
* :điện áp đưa vào chân ADC (tối đa 3.3V)
* :hai điện trở phân áp

Lựa chọn điện trở:

Để đo chính xác điện áp tối đa 20V, và đảm bảo VADC≤3.3V, chọn:

* R1=10kΩ
* R2=2kΩ

Khi đó:

Ảnh có chứa văn bản, Phông chữ, số, đồng hồ

Nội dung do AI tạo ra có thể không chính xác.



### Sơ đồ kết nối

Đầu vào Vin​: nối vào nguồn 5V–20V

Điểm giữa R1 và R2: nối vào chân ADC của vi điều khiển

Đầu còn lại của R2: nối với GND



Hình 5.6 Cấu trúc cầu phân áp

### Cấu hình ADC2

Trong hệ thống, chân PA0 (ADC2\_IN1) được sử dụng để đo điện áp sau khi qua mạch chia áp. Để tối ưu hiệu suất, DMA (Direct Memory Access) được sử dụng để tự động lưu trữ giá trị ADC vào bộ nhớ mà không chiếm tài nguyên CPU.

Kênh ADC: IN1 (PA0)

Độ phân giải: 12-bit (0–4095)

Chế độ chuyển đổi: Continuous Conversion Mode

Data Alignment: Right

Ảnh có chứa văn bản, ảnh chụp màn hình, thực đơn, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Hình 5.7. Cấu hình AD2\_IN1

#### Cấu hình DMA cho ADC2

DMA Request: ADC2

Stream: chọn theo mặc định CubeMX gán cho ADC2 (tùy vào MCU cụ thể)

Direction: Peripheral to Memory

Priority: Medium

Mode: Normal (có thể chọn Circular nếu đo liên tục)

Peripheral Increment: Disable

Memory Increment: Enable

Data Width: Half Word cho cả Peripheral và Memory (tương ứng 16-bit dữ liệu ADC)

Use FIFO: Không sử dụng

Ảnh có chứa văn bản, ảnh chụp màn hình, số, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

Hình 5.8. Cấu hình DMA cho ADC2

## Giao tiếp CAN

### Cấu hình CAN

Bộ điều khiển CAN trên STM32 được cấu hình để đảm bảo việc truyền thông ổn định và tuân thủ các yêu cầu kỹ thuật của mạng CANopen.

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

Hình 5.9. Cấu hình CAN trên STM32

Việc cấu hình bao gồm các tham số chính sau:

* Instance: CAN1 là module CAN tích hợp trên STM32 thực hiện chức năng truyền nhận dữ liệu qua busCAN vật lý.
* Prescaler: Giá trị hệ số chia tần số đặt là 9. Prescaler có vai trò điều chỉnh tốc độ bit của CAN. Khi kết hợp với các tham số thời gian khác, cấu hình này cho phép đạt baudrate 500 kbps.
* Chế độ hoạt động: CAN được đặt ở chế độ Normal Mode, đảm bảo truyền thông thực tế với các thiết bị khác trên mạng.

Tham số thời gian:

* SYNC Jump Width (SJW): 1TQ, cho phép đồng bộ lại khi có sai lệch nhỏ.
* Time Segment 1 (BS1): 2TQ, xác định thời gian lấy mẫu bit.
* Time Segment 2 (BS2): 2TQ, giúp kết thúc bit thời gian và hỗ trợ tái đồng bộ.
* Tốc độ truyền: Cấu hình đạt baudrate 1Mbps (1.000.000 bit/s), phù hợp với các ứng dụng cần tốc độ truyền nhanh và thời gian đáp ứng thấp.
* Thời gian cho 1 bit: 1000ns đảm bảo tính chính xác trong giao tiếp.

Cấu hình này đảm bảo hệ thống hoạt động ở tốc độ cao (1Mbps), đáp ứng yêu cầu truyền thông khắt khe của mang CANopen trong hệ thống điều khiển BLDC. Nhờ đó, các lệnh điều khiển và dữ liệu phản hồi được truyền đi nhanh chóng, giúp tối ưu hiệu quả điều khiển và tăng độ tin cậy cho toàn hệ thống.

# XÂY DỰNG PHẦN MỀM ĐIỀU KHIỂN ĐỘNG CƠ BẰNG QT

## Mục tiêu của phần mềm ứng dụng

* Phát triển một ứng dụng giao tiếp với hệ thống động cơ thông qua giao thức CANopen.
* Thực hiện các chức năng cơ bản: gửi lệnh điều khiển động cơ, giám sát trạng thái động cơ và đọc các tham số vận hành.

## Công cụ và công nghệ sử dụng

* Ngôn ngữ lập trình: C++
* Môi trường phát triển: QT Creator

### Giới thiệu về phần mềm QT Creator

QT là một framework đa nền tảng. Một số ứng dụng phổ biến viết từ QT có thể kể đến như KDE, Opera, Google Earth, và Skype. QT lần đầu được giới thiệu vào tháng 5 năm 1995. QT có thể được dùng để phát triển ứng dụng mã nguồn mở lẫn các ứng dụng cho doanh nghiệp. Bộ công cụ phát triển QT rất mạnh mẽ vì nó được cả một cộng đồng mã nguồn mở hỗ trợ. Có đến hàng ngàn các nhà phát triển mã nguồn mở sử dụng QT trên toàn thế giới.

QT được viết bằng C++ và được thiết kế sử dụng trong C++. Tuy nhiên, hiện nay chúng ta đã có thể dùng thư viện với nhiều ngôn ngữ khác như Java hay Python,…

Trên thực tế, QT không phải một thư viện mà tập hợp các thư viện. Chúng rất rộng và thường thì người ta sử dụng thuật ngữ farmework, nghĩa là một khối kiến trúc tập hợp cung cấp nhiều công cụ để lập trình của chúng ta trở nên hữu hiện hơn.



Hình 6.1. Phần mềm QT Creator

### Cấu trúc chương trình giao diện

Giao diện được xây dựng bằng phần mềm QT Creator, nhằm mục đích giám sát dữ liệu thực tế từ động cơ và gửi lệnh điều khiển thông qua giao tiếp CANopen. Giao diện chia thành nhiều khối chức năng riêng biệt, dễ sử dụng và thân thiện với người vận hành.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Nội dung do AI tạo ra có thể không chính xác.

Hình 6.2. Giao diện chính của phần mềm điều khiển trên máy tính

Cấu trúc các khối chức năng bao gồm:

#### Khu vực kết nối cổng COM

Hiển thị danh sách các cổng COM đang có sẵn.

Nút CONNECTED thiết lập giao tiếp với bo STM32.

Nút DISCONNECTED ngắt kết nối.

Nút Refesh để cập nhật lại danh sách cổng COM.

Khi kết nối thành công sẽ hiển thị trang thái “CONNECTED”

#### Data received from STM32

Hiển thị toàn bộ các khung tin CAN được nhận từ động cơ thông qua TPDO.

Dữ liệu được cập nhật liên tục nhờ hàm đọc CAN chạy theo chu kỳ.

Nút READ DATA: Bắt đầu đọc dữ liệu.

Nút CLEAR: xóa nội dung hiển thị.

#### Data sent to STM32

Hiển thị dữ liệu mà người dùng gửi xuống STM32 qua giao thức CAN.

Nút CLEAR: Xóa nội dung vùng này.

#### Khối điều khiển

Nhập giá trị SET POINT mong muốn.

Có 2 lựa chọn điều khiển:

* ANGLE: gửi giá trị vị trí mong muốn.
* SPEED: gửi tốc độ tham chiếu.

Tùy vào nút được nhấn, dữ liệu sẽ được mã hóa và gửi xuống thiết bị thông qua RPDO hoặc SDO tương ứng.

#### Khối dữ liệu thực tế

Hiển thị thông số phản hồi thực từ động cơ:

* **ANGLE:** góc quay thực tế.
* **i\_d, i\_q:** dòng điện thành phần theo trục dq.
* **u\_d, u\_q:** điện áp thành phần theo trục dq

Giá trị này được cập nhật từ TPDO của thiết bị.

### Cơ chế truyền nhận dữ liệu

Gửi dữ liệu: Khi nhấn các nút SEND, ANGLE, SPEED, chương trình tạo khung CAN và gửi qua cổng kết nối với adapter CAN, sử dụng socket hoặc API tương ứng.

Nhận dữ liệu: Chương trình cài đặt QTimer để đọc dữ liệu CAN định kỳ và cập nhật nội dung khối “Dữ liệu nhận” và “Dữ liệu thực tế”.

Dữ liệu truyền/nhận đều được ánh xạ đến các Object Dictionary của thiết bị điều khiển thông qua ID định sẵn.

### Các tab đồ thị giám sát

Khu vực đồ thị nằm phía bên phải của giao diện phần mềm và được chia thành ba tab chính: Graph Voltage, Graph Current và Graph Angle. Mỗi tab có chức năng hiển thị trực quan các tín hiệu điện hoặc cơ từ hệ thống điều khiển động cơ, nhằm hỗ trợ người dùng giám sát thời gian thực và phân tích trạng thái hoạt động của hệ thống

#### Tab Graph Current

Tab Graph Current cho phép giám sát các dòng điện trong hệ FOC:

* Id – dòng điện trục d (liên quan đến điều khiển từ thông).
* Iq – dòng điện trục q (liên quan đến điều khiển momen).

Dữ liệu dòng được lấy từ TPDO của STM32. Người dùng có thể dùng đồ thị:

* Phân tích cách hệ thống điều khiển dòng điện nhằm đáp ứng các thay đổi về momen hoặc tốc độ.
* Đảm bảo rằng dòng điện không vượt quá giới hạn cho phép.
* Theo dõi biến động dòng khi có nhiễu hoặc sự cố từ tải.

#### Tab Graph Angle

Tab Graph Angle có chức năng hiển thị đồ thị góc quay của động cơ BLDC trong thời gian thực. Trong đó:

* Trục hoành X biểu diễn thời gian (s).
* Trục tung Y biểu diễn góc quay (rad).
* Đường màu đỏ (theta\_ref) thể hiện góc đặt (giá trị tham chiếu) được gửi từ phần mềm xuống STM32 thông qua giao thức CANopen, thông qua RPDO.
* Đường màu xanh (theta\_now) thể hiện góc quay thực tế của động cơ được phản hồi từ vi điều khiển lên phần mềm qua TPDO.

Thông qua biểu đồ, người dùng có thể đánh giá:

* Độ chính xác của hệ thống điều khiển trong việc bám theo góc đặt.
* Thời gian đáp ứng của động cơ khi thay đổi góc đặt.
* Hiệu suất của thuật toán FOC cũng như độ trễ hệ thống khi giao tiếp qua CANopen.

Biểu đồ nãy đặc biệt hữu ích trong các tình huống cần theo dõi sự ổn định hoặc phát hiện các hiện tượng sai lệch, như trễ phản hồi, bám sai và dao động dư.

#### 

# KẾT QUẢ THỰC NGHIỆM VÀ HƯỚNG PHÁT TRIỂN

# KẾT LUẬN CHUNG

# TÀI LIỆU THAM KHẢO

1. Mohamed Hassan Ali, Abdelhamid Rabhi, Ahmed El hajjaji and Giuseppe M. Tina. “Real Time Fault Detection in Photovoltaic Systems”, 2016, Turin, ITALY.