ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN

ĐỒ ÁN TỐT NGHIỆP

CAPSTONE PROJECT

NGÀNH: KĨ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HOÁ

ĐỀ TÀI:

ỨNG DỤNG CÔNG NGHỆ TRUYỀN THÔNG ETHERCAT ĐỒNG BỘ HÓA TỐC ĐỘ ĐỘNG CƠ TRONG HỆ THỐNG ĐIỀU KHIỂN THỜI GIAN THỰC

Giảng viên hướng dẫn: **TS. NGUYỄN KHÁNH QUANG**

Cán bộ hướng dẫn: **NGUYỄN ĐĂNG KHOA**

Sinh viên thực hiện:

1. NGUYỄN VĂN QUANG – MSSV: 105200466 – LỚP: 20TDHCLC3

2. TRẦN TUẤN LINH – MSSV: 105200415 – LỚP: 20TDHCLC2

3. LÊ NGỌC HIẾU - MSSV: 105200448 – LỚP: 20TDHCLC3

Đà Nẵng, 6/2025

TÓM TẮT

Tên đề tài : Ứng Dụng Công Nghệ Truyền Thông EtherCAT Đồng Bộ Hóa Tốc Độ Động Cơ Trong Hệ Thống Điều Khiển Thời Gian Thực

Sinh viên thực hiện: Nguyễn Văn Quang……...

Số thẻ SV: 105200466…… Lớp: 20TDHCLC3……

Sinh viên thực hiện: Trần Tuấn Linh…..……...

Số thẻ SV: 105200415…… Lớp: 20TDHCLC2……

Sinh viên thực hiện: Lê Ngọc Hiếu…….……...

Số thẻ SV: 105200448…… Lớp: 20TDHCLC3……

Nội dung tóm tắt của đề tài :

- Mô tả đề tài:

Trong bối cảnh các hệ thống điều khiển hiện đại yêu cầu độ chính xác cao, khả năng phản hồi nhanh và sự đồng bộ trong thời gian thực, đề tài tập trung nghiên cứu và triển khai hệ thống điều khiển tốc độ hai động cơ BLDC bằng giao thức EtherCAT. Hệ thống sử dụng TwinCAT làm master để giao tiếp với EtherCAT Click LAN9252, truyền dữ liệu qua SPI đến vi điều khiển STM32, sau đó điều khiển động cơ bằng phương pháp Field-Oriented Control (FOC) thông qua thư viện SimpleFOC. Hệ thống điều khiển bao gồm các vòng lặp PID lồng nhau: tốc độ – dòng điện, kết hợp với thuật toán FOC nhằm tối ưu hóa độ chính xác và hiệu suất năng lượng.

Trong quá trình thực nghiệm, hệ thống cho thấy khả năng đồng bộ tốc độ động cơ với sai số thấp và thời gian đáp ứng nhanh. Truyền thông EtherCAT, SPI và thuật toán điều khiển được triển khai hiệu quả, đảm bảo khả năng mở rộng cho các ứng dụng công nghiệp. Tuy nhiên, để cải thiện hiện tượng dao động ở tốc độ cao, cần tối ưu thêm thuật toán điều khiển và giới hạn tốc độ động cơ.

Kết quả đề tài không chỉ minh chứng tính khả thi của việc áp dụng giao thức EtherCAT trong điều khiển phân tán mà còn là nền tảng để phát triển các hệ thống điều khiển động cơ thông minh, phục vụ trong robot, máy tự động, và các ứng dụng công nghiệp đòi hỏi tính đồng bộ và thời gian thực.

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

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

TT	Họ tên sinh viên	Số thẻ SV	Lớp	Ngành
1	Nguyễn Văn Quang	105200466	20TDHCLC3	Kĩ thuật Điều khiển và Tự động hóa
2	Trần Tuấn Linh	105200415	20TDHCLC2	Kĩ thuật Điều khiển và Tự động hóa
3	Lê Ngọc Hiếu	105200448	20TDHCLC3	Kĩ thuật Điều khiển và Tự động hóa

1. Tên đề tài đồ án:

Ứng dụng công nghệ truyền thông EtherCAT đồng bộ hóa tốc độ động cơ trong hệ thống điều khiển thời gian thực

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

3. Các số liệu và dữ liệu ban đầu:

4. Nội dung các phần thuyết minh và tính toán:

a. Phần chung:

TT	Tên sinh viên	Nội dung
1	Nguyễn Văn Quang	- Nghiên cứu hệ thống đồng bộ tốc độ động cơ BLDC bằng truyền thông EtherCAT - Nghiên cứu phần mềm TwinCAT và phần mềm lập trình STM32CubeIDE. - Nghiên cứu public hệ thống trên nền tảng internet để truy cập từ xa. - Nghiên cứu kết nối chi tiết thiết bị. - Tìm hiểu nguyên lý hoạt động của các thiết bị điện. - Nghiên cứu quy trình công nghệ của hệ thống. - Thi công đấu nối dây, lắp đặt thiết bị cho hệ thống - Tiến hành chạy thử hệ thống, đánh giá kết quả.
2	Trần Tuấn Linh	
3	Lê Ngọc Hiếu	

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung
1	Nguyễn Văn Quang	- Kết nối các module EtherCAT Click với STM32F446 qua SPI. - Viết và chỉnh sửa tệp mô tả ESC (EtherCAT Slave Controller) bằng EtherCAT SDK. - Cài đặt và cấu hình TwinCAT để làm Master trong hệ thống EtherCAT. - Viết lưu đồ thuật toán và lập trình STM32.
2	Trần Tuấn Linh	- Tìm hiểu kết nối giữa hai STM32F446 qua UART. - Kết nối và kiểm tra hệ thống EtherCAT Master-Slave. - Tổng hợp nội dung, viết báo cáo thuyết minh. - Tính toán các thông số của hệ thống và kiểm nghiệm.
3	Lê Ngọc Hiếu	- Tìm hiểu kết nối giữa hai STM32F446 qua UART. - Thực hiện mô phỏng động cơ trên MotionLab - Tính toán các thiết bị trong hệ thống. - Viết và chỉnh sửa tệp mô tả ESC (EtherCAT Slave Controller) bằng EtherCAT SDK.

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

TT	Họ tên sinh viên	Nội dung
1		

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

7. Ngày giao nhiệm vụ đồ án: 13/3/2025

8. Ngày hoàn thành đồ án: 22/6/2025

Đà Nẵng, ngày 22 tháng 6 năm 2025

Trưởng Bộ môn Tự động hóa **Người hướng dẫn**

TS. Giáp Quang Huy

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

(Phiếu dành cho người hướng dẫn/sinh viên)

Họ tên sinh viên: Nguyễn Văn Quang Số thẻ SV: 105200466
Trần Tuấn Linh 105200415
Lê Ngọc Hiếu 105200448

Tên đề tài ĐATN: Ứng Dụng Công Nghệ Truyền Thông EtherCAT Đồng Bộ Hóa Tốc Độ Động Cơ Trong Hệ Thống Điều Khiển Thời Gian Thực

Họ tên người HD: TS. Nguyễn Khánh Quang

Đơn vị: Trường Đại học Bách khoa – Đại học Đà Nẵng

Tuần	Ngày	Khối lượng		GVHD ký tên
		Đã thực hiện (%)	Tiếp tục thực hiện (%)	
1	13/3/2025	5%	95%	
2	20/3/2025	10%	90%	
3	27/3/2025	15%	85%	
4	3/4/2025	Duyệt lần 1: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
5	10/4/2025	30%	70%	
6	17/4/2025	40%	60%	
7	24/4/2025	50%	50%	
8	3/5/2025	Duyệt lần 2: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
9	8/5/2025	60%	40%	

10	15/5/2025	70%	30%	
11	22/5/2025	75%	25%	
12	29/5/2025	Duyệt lần 3: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
13	5/6/2025	80%	20%	
14	12/6/2025	90%	10%	
15	19/6/2025	95%	5%	
16	21/6/2025	100%	0%	

LỜI NÓI ĐẦU VÀ CẢM ƠN

Sau thời gian học tập và nghiên cứu tại trường Đại học Bách Khoa – Đại học Đà Nẵng. Được sự giảng dạy và chỉ bảo của thầy, cô giáo đến nay nhóm đồ án đã được trang bị đầy đủ những kiến thức và phẩm chất của người làm kỹ thuật. Kết thúc phần học lý thuyết tại trường và thực hiện Đồ án tốt nghiệp nhằm tổng hợp kiến thức và hoàn thành chương trình học.

Trong thời gian qua, dưới sự dẫn dắt và giám sát của thầy Nguyễn Khánh Quang, nhóm đã thực hiện đề tài tốt nghiệp **" *Ứng Dụng Công Nghệ Truyền Thông EtherCAT Đồng Bộ Hóa Tốc Độ Động Cơ trong Hệ Thống Điều Khiển Thời Gian Thực",*** nhóm đã được biết thêm nhiều kiến thức về mạng truyền thông EtherCat. Đây là một cơ hội rất quý báu được tiếp cận được những kiến thức mới là tiền đề giúp chúng tôi hiểu rõ về quy trình thiết kế dự án cho một doanh nghiệp để bổ trợ cho công việc sau này.

Đồ án tốt nghiệp này là dấu mốc quan trọng để kiểm tra nhận thức của sinh viên trong thời gian học tập và những kiến thức đã được giảng dạy ở trong trường. Đồng thời nó còn đánh giá khả năng vận dụng lý thuyết để phân tích tổng hợp và giải quyết các bài toán trong thực tế khi làm một dự án lớn cho công ty. Nhận thức được tầm quan trọng đó, nhóm đã làm việc nghiêm túc vận dụng những kiến thức đã được học trong thời gian ở trường, những đóng góp ý kiến của bạn bè và đặc biệt là sự hướng dẫn của thầy Nguyễn Khánh Quang để hoàn thành tốt đồ án tốt nghiệp.

Mặc dù, bản thân đã cố gắng nhiều để hoàn thành đồ án. Nhưng do trình độ có hạn nên vẫn còn những thiếu sót, nhóm đồ án rất mong được sự chỉ bảo, giúp đỡ của các thầy cô để bài làm này của nhóm được hoàn thiện hơn.

Nhóm chân thành cảm ơn đến tất cả thầy cô khoa Điện nói riêng và trường Đại học Bách Khoa Đà Nẵng nói chung đã tận tình chỉ dạy từ những môn đại cương đến những môn chuyên ngành.

Nhóm chân thành cảm ơn quý Công ty Cổ phần Sybotix Solution đã tạo điều kiện và giúp đỡ nhóm trong thời gian làm đồ án. Cảm ơn các anh kĩ sư đã nhiệt tình hướng dẫn và giải đáp nhiều thắm mắc cho nhóm.

Nhóm đồ án chúng tôi gửi lời chúc sức khỏe và đạt được nhiều thành công trong công việc cũng như trong cuộc sống sau này đến thầy Nguyễn Khánh Quang và toàn thể Thầy, Cô giáo Trường Đại học Bách Khoa Đà Nẵng.

Chúng tôi xin chân thành cảm ơn!

Đà Nẵng, ngày 22 tháng 06 năm 2025

LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT

Nhóm đồ án cam đoan: Đề tài "***Ứng Dụng Công Nghệ Truyền Thông EtherCAT Đồng Bộ Hóa Tốc Độ Động Cơ trong Hệ Thống Điều Khiển Thời Gian Thực***" được thực hiện với sự cố gắng của nhóm, tự nghiên cứu và thí nghiệm dưới sự hướng dẫn của thầy.

Các số liệu và kết quả nghiên cứu trong dự án là trung thực và hoàn toàn không sao chép hoặc sử dụng kết quả của đề tài nghiên cứu nào tương tự. Nếu phát hiện có sự sao chép kết quả nghiên cứu của đề tài khác, nhóm xin hoàn toàn chịu trách nhiệm.

Nhóm sinh viên thực hiện

Nguyễn Văn Quang Trần Tuấn Linh Lê Ngọc Hiếu

MỤC LỤC

DANH SÁCH CÁC BẢNG, HÌNH ẢNH

MỞ ĐẦU

1. Mục đích thực hiện dự án:

Capstone Project là một trong những học phần bắt buộc và rất quan trọng đối với chương trình đào tạo kỹ sư của khoa Điện trường Đại học Bách Khoa – Đại học Đà Nẵng, từ đó sinh viên có thể tự đánh giá và học hỏi thêm được nhiều kiến thức mới. Ngoài ra, sinh viên còn có thể tập được tính làm việc độc lập, nhằm tổng quát kiến thức để ra trường làm việc.

Vì vậy, thầy Nguyễn Khánh Quang bộ môn Tự động hóa- trường Đại học Bách Khoa – Đại học Đà Nẵng tạo điều kiện thuận lợi và tốt nhất cho nhóm thực hiện tốt dự án lần này.

2. Mục tiêu đề tài:

Đề tài này tập trung vào các mục tiêu sau:

- Tìm hiểu giao thức EtherCAT và ứng dụng trong hệ thống.
- Thiết kế mô hình điều khiển đồng bộ hai động cơ BLDC sử dụng TwinCAT làm master, STM32 làm slave thông qua EtherCAT Click.
- Đánh giá hiệu quả điều khiển đồng bộ tốc độ động cơ trong thời gian thực.

3. Phạm vi và đối tượng nghiên cứu:

- Phạm vi nghiên cứu: Giao thức truyền thông EtherCAT trong hệ thống điều khiển thời gian thực.
- Đối tượng nghiên cứu: Hệ thống cấp lệnh và điều khiển đồng bộ hai động cơ BLDC thông qua mạng EtherCAT.

4. Phương pháp nghiên cứu:

- Khảo sát hệ thống điều khiển sử dụng giao tiếp EtherCAT kết hợp SPI.
- Nghiên cứu các tài liệu liên quan được doanh nghiệp cung cấp.
- Phân tích quy trình truyền nhận dữ liệu và điều khiển tốc độ động cơ trong hệ thống.
- Đề xuất và thử nghiệm phương pháp điều khiển, giám sát tốc độ động cơ BLDC đồng bộ.

CHƯƠNG 1: TỔNG QUÁT VỀ HỆ THỐNG ĐỒNG BỘ TỐC ĐỘ ĐỘNG CƠ SỬ DỤNG ETHERCAT

1.1. Giới thiệu chung về đề tài

Đồng bộ hóa tốc độ động cơ là một yêu cầu quan trọng trong các hệ thống điều khiển hiện đại, đặc biệt trong các ứng dụng công nghiệp như dây chuyền sản xuất, robot song song, hoặc các hệ thống cần sự phối hợp nhịp nhàng giữa nhiều cơ cấu chuyển động. Với sự phát triển mạnh mẽ của công nghệ truyền thông thời gian thực, EtherCAT đã trở thành một trong những giao thức truyền thông phổ biến nhờ vào tốc độ cao, độ tin cậy và khả năng mở rộng linh hoạt.

Trong đề tài này, chúng tôi nghiên cứu và thiết kế hệ thống điều khiển đồng bộ tốc độ cho hai động cơ không chổi than (BLDC) sử dụng nền tảng giao tiếp EtherCAT. Hệ thống sử dụng phần mềm TwinCAT chạy trên PC đóng vai trò là master, kết nối tới các slave là vi điều khiển STM32 thông qua module EtherCAT LAN9252. Dữ liệu điều khiển và phản hồi được truyền thời gian thực, từ đó giúp hai động cơ vận hành một cách đồng bộ về tốc độ và vị trí.

1.1.1. Phân loại các hệ thống đồng bộ

- Đồng bộ cứng (Hard Synchronization): Các hệ thống yêu cầu đồng bộ tuyệt đối về tốc độ và vị trí giữa các động cơ, ví dụ như robot song song hoặc hệ thống in đa trục.
- Đồng bộ mềm (Soft Synchronization): Cho phép sai số nhỏ giữa các trục, phù hợp với các hệ thống ít đòi hỏi về chính xác tuyệt đối.
- Đồng bộ có kiểm tra phản hồi: Hệ thống sử dụng cảm biến để thu thập thông tin phản hồi về tốc độ/góc quay nhằm điều chỉnh chính xác tốc độ giữa các động cơ.

1.1.2. Đặc điểm của hệ thống đồng bộ

- Hệ thống đồng bộ tốc độ cần đảm bảo độ trễ truyền dữ liệu và phản hồi nhanh.
- Đòi hỏi thuật toán điều khiển chính xác, thường tích hợp PID, kiểm tra phản hồi encoder.
- Sự ổn định phụ thuộc nhiều vào giao thức truyền thông giữa các thành phần trong hệ thống.

1.2. Nguyên lý hoạt động

1.2.1. Cấu trúc chung của hệ thống

- TwinCAT Master (PC): Gửi tín hiệu điều khiển tốc độ và thông số PID đến các Slave và nhận lại tín hiệu phản hồi từ động cơ.
- EtherCAT Slave (LAN9252 + STM32F446): Nhận dữ liệu từ master, truyền qua SPI đến STM32 điều khiển động cơ, và nhận phản hồi từ STM32 điều khiển để gửi về PC.
- Vi điều khiển điều khiển động cơ (STM32F446): Điều khiển động cơ dựa trên dữ liệu nhận được, đồng thời đọc encoder và truyền dữ liệu phản hồi.
- Giao tiếp SPI: Sử dụng để truyền dữ liệu giữa các tầng vi điều khiển với tốc độ cao và độ chính xác cao.

Hệ thống đảm bảo quá trình truyền và xử lý dữ liệu theo thời gian thực, giúp hai động cơ hoạt động đồng bộ về tốc độ và vị trí thông qua kiểm soát vòng kín.

1.2.2. Logic vận hành của hệ thống

- **Bước 1: Khởi tạo hệ thống:**
  + Máy tính (PC) chạy phần mềm TwinCAT 3 khởi tạo chế độ Master EtherCAT.
  + Các thiết bị slave (LAN9252 + STM32F446) được nhận diện và ánh xạ PDO thông qua TwinCAT.
- **Bước 2: Truyền dữ liệu điều khiển từ Master:**
  + PC gửi chu kỳ dữ liệu điều khiển (tốc độ tham chiếu, thông số PID, enable…) qua EtherCAT đến các slave.
  + Chu kỳ truyền diễn ra với thời gian thực khoảng 1ms.
- **Bước 3: Xử lý dữ liệu tại slave:**
  + LAN9252 nhận gói dữ liệu EtherCAT và chuyển tiếp qua giao tiếp SPI cho STM32F446.
  + STM32F446 phân tích gói tin, tách các giá trị điều khiển, sau đó điều khiển theo thuật toán FOC.
- **Bước 4: Điều khiển động cơ:**

+ STM32F446 xử lý dữ liệu nhận được, áp dụng thuật toán điều khiển (PID) và tạo tín hiệu PWM để điều khiển động cơ BLDC thông qua thư viện SimpleFOC.
+ Đồng thời, đọc encoder để lấy giá trị góc và tốc độ thực tế của động cơ.

- **Bước 5: Gửi dữ liệu phản hồi:**
  + STM32F446 đóng gói dữ liệu và truyền ngược về PC thông qua LAN9252 và EtherCAT.
- **Bước 6: Đồng bộ tốc độ:**
  + **Trường hợp 1: Gửi lệnh xuống hai động cơ cùng lúc**
    - PC (hệ thống TwinCAT) gửi các tham số điều khiển đến cả hai động cơ BLDC thông qua giao tiếp EtherCAT theo chu kì hoạt động của SYNC0
    - Cả hai động cơ sẽ nhận lệnh điều khiển cùng lúc, đảm bảo bắt đầu chuyển động đồng bộ.
  + **Trường hợp 2: Một động cơ quay tay, động cơ kia quay theo**
    - Một động cơ sẽ được điều khiển thủ công bằng tay quay
    - Động cơ còn lại (động cơ theo sau) sẽ nhận tín hiệu từ động cơ dẫn đầu và tự động quay theo với cùng vị trí.

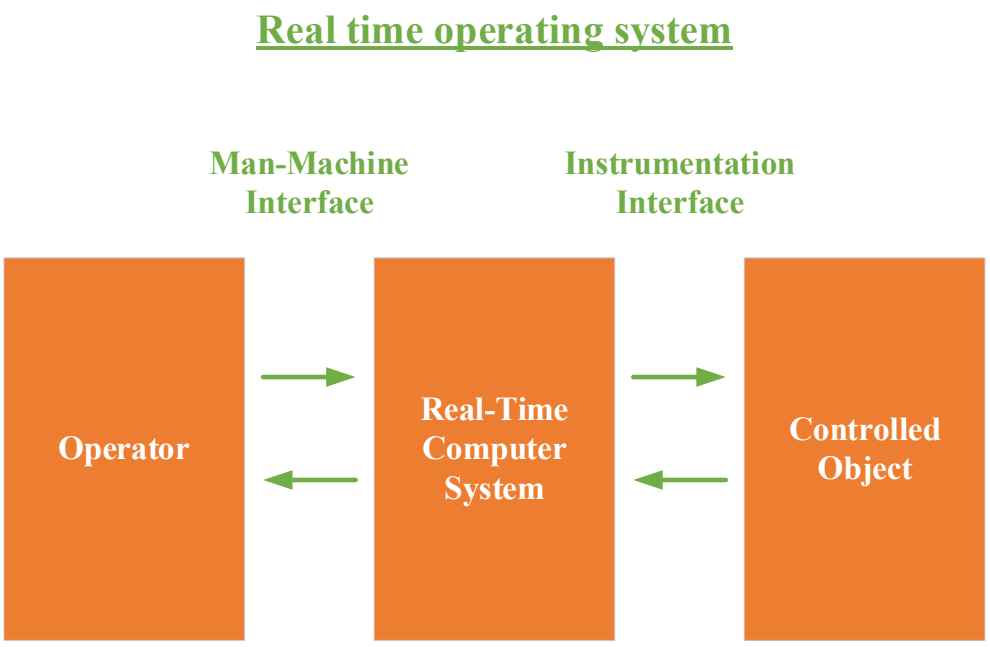
CHƯƠNG 2: LÝ THUYẾT ĐIỀU KHIỂN THỜI GIAN THỰC VÀ CÔNG NGHỆ ETHERCAT

2.1. Giới thiệu về hệ thống điều khiển thời gian thực (RTCS)

Hệ thống điều khiển thời gian thực (Real-Time Control System – RTCS) là một trong những lĩnh vực quan trọng nhất của tự động hóa và điều khiển hiện đại. Trong các hệ thống này, hoạt động xử lý dữ liệu và phản hồi không chỉ yêu cầu đúng về mặt logic mà còn đúng về mặt thời gian. Nói cách khác, một hệ thống thời gian thực phải đảm bảo rằng kết quả đầu ra được tạo ra trong một khoảng thời gian xác định trước, nếu không toàn bộ hệ thống có thể không hoạt động đúng, dẫn đến hậu quả nghiêm trọng.

2.1.1. Khái niệm cơ bản

Một hệ thống thời gian thực có thể được định nghĩa là hệ thống trong đó thời gian xử lý là một ràng buộc thiết yếu, chứ không đơn thuần là yếu tố tối ưu. Ví dụ, trong một hệ thống điều khiển động cơ, nếu tín hiệu điều khiển được gửi trễ so với thời điểm mong muốn, động cơ có thể vận hành không chính xác, gây ra rung lắc, mất cân bằng hoặc hư hỏng thiết bị.



Hình 2.1: Hệ thống thời gian thực

RTCS thường bao gồm:

- Các bộ vi điều khiển hoặc vi xử lý làm nhiệm vụ xử lý dữ liệu.
- Cảm biến (sensor) và bộ truyền động (actuator) để tương tác với môi trường vật lý.
- Các giao thức truyền thông để trao đổi dữ liệu giữa các thành phần hệ thống, đặc biệt khi có nhiều thiết bị cần đồng bộ với nhau.

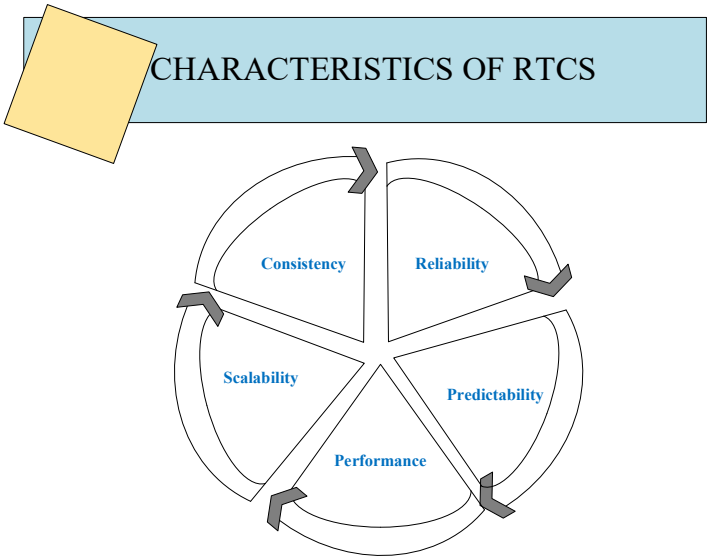
2.1.2. Phân loại hệ thống thời gian thực

Tùy theo mức độ nghiêm ngặt về thời gian đáp ứng, hệ thống RTCS được chia thành hai loại:

- Hệ thống thời gian thực cứng (Hard Real-Time Systems): Đây là loại hệ thống mà mọi tác vụ đều phải được thực hiện trong khoảng thời gian quy định tuyệt đối. Bất kỳ sự vi phạm nào về thời gian cũng được xem là lỗi nghiêm trọng. Các hệ thống điều khiển tên lửa, máy bay, thiết bị y tế cứu sinh, và các hệ thống CNC tốc độ cao là ví dụ điển hình.
- Hệ thống thời gian thực mềm (Soft Real-Time Systems): Trong hệ thống này, các tác vụ vẫn nên được thực hiện trong khoảng thời gian nhất định, nhưng một vài lần vượt quá giới hạn vẫn có thể chấp nhận được. Các hệ thống multimedia, trò chơi điện tử, hoặc điều khiển môi trường thông minh (như HVAC) thường thuộc nhóm này.

2.1.3. Đặc điểm kỹ thuật của hệ thống RTCS

Một hệ thống RTCS hiệu quả cần có các đặc điểm sau:



Hình 2.2: Đặc điểm của hệ thống RTCS

- Tính xác định (Determinism): Hệ thống phải phản hồi với đầu vào trong một khoảng thời gian nhất định đã biết trước.
- Tính tin cậy (Reliability): Hệ thống phải hoạt động ổn định, không bị lỗi hoặc treo trong quá trình chạy.
- Tính sẵn sàng cao (High Availability): Đặc biệt quan trọng trong các ứng dụng công nghiệp liên tục.
- Tính mở rộng (Scalability): Dễ dàng mở rộng số lượng thiết bị hoặc phạm vi điều khiển.
- Tính dự đoán (Predictability): Hệ thống phải đảm bảo rằng các nhiệm vụ hoặc sự kiện sẽ được thực thi trong một khung thời gian xác định trước, đảm bảo tính đúng hạn.

2.1.4. Vai trò của RTCS trong công nghiệp hiện đại

Trong thời đại công nghiệp 4.0, các hệ thống điều khiển thời gian thực đóng vai trò xương sống trong việc tự động hóa các dây chuyền sản xuất. Chúng giúp:

- Tăng hiệu suất sản xuất nhờ khả năng điều khiển chính xác và phản hồi nhanh.
- Đảm bảo chất lượng sản phẩm thông qua kiểm soát chặt chẽ từng công đoạn.
- Giảm thiểu rủi ro và tai nạn, đặc biệt trong môi trường có yêu cầu an toàn cao.
- Tiết kiệm chi phí vận hành bằng cách giảm sai sót, giảm hao phí nguyên vật liệu và năng lượng.

Với xu hướng ngày càng phổ biến của các hệ thống nhúng (embedded systems) và Internet Vạn Vật Công Nghiệp (IIoT), nhu cầu về các hệ thống RTCS có độ chính xác và độ tin cậy cao ngày càng tăng. Tuy nhiên, một trong những thách thức lớn nhất của RTCS chính là khả năng truyền thông giữa các thiết bị trong hệ thống – truyền dữ liệu nhanh, ổn định, và đồng bộ hóa chính xác theo thời gian thực là yếu tố sống còn.

Điều này dẫn đến việc lựa chọn các giao thức truyền thông chuyên biệt như EtherCAT – giao thức nổi bật với khả năng truyền thông tốc độ cao, độ trễ thấp và đồng bộ hóa tốt giữa nhiều thiết bị – trở thành hướng đi chiến lược trong việc xây dựng các hệ thống điều khiển thời gian thực hiện đại.

2.2. Yêu cầu về truyền thông trong RTCS

Trong các hệ thống điều khiển thời gian thực (RTCS), truyền thông giữa các thiết bị đóng vai trò then chốt trong việc đảm bảo hệ thống hoạt động đúng thời gian và đúng chức năng. Dữ liệu cần được trao đổi liên tục và chính xác giữa các thành phần như cảm biến, bộ điều khiển (controller), bộ truyền động (actuator), và hệ thống giám sát trung

tâm. Do đó, hệ thống truyền thông trong RTCS phải đáp ứng được nhiều yêu cầu khắt khe về hiệu năng, độ ổn định và khả năng đồng bộ hóa.

2.2.1. Tính xác định và độ trễ thấp

Một trong những yêu cầu quan trọng nhất là tính xác định (determinism) trong truyền thông. Điều này nghĩa là thời gian truyền dữ liệu từ thiết bị này sang thiết bị khác phải luôn luôn nằm trong một khoảng xác định trước, không được dao động hoặc thay đổi ngẫu nhiên như các mạng truyền thông phi thời gian thực (ví dụ: Ethernet thông thường).

Bên cạnh đó, độ trễ truyền thông phải thấp và ổn định, tức là thời gian từ khi dữ liệu được gửi đến khi được nhận và xử lý phải đủ nhanh để đáp ứng các chu kỳ điều khiển rất ngắn, có thể chỉ vài mili-giây, thậm chí vài chục micro-giây trong các ứng dụng công nghiệp tốc độ cao như:

- Điều khiển tốc độ và vị trí động cơ servo
- Điều khiển robot công nghiệp đa trục
- Hệ thống cân bằng hoặc ổn định quỹ đạo

2.2.2. Băng thông đủ lớn và khả năng mở rộng

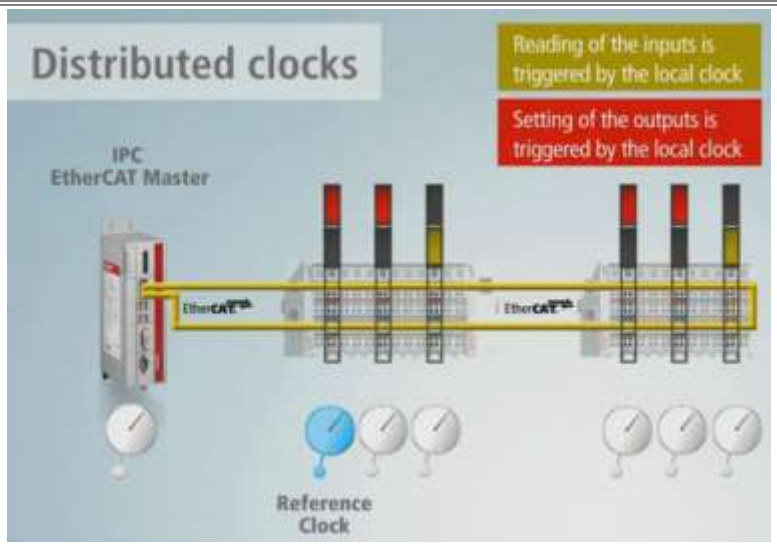
RTCS yêu cầu băng thông truyền thông đủ lớn để có thể truyền đồng thời một lượng lớn dữ liệu từ nhiều thiết bị, đặc biệt khi hệ thống mở rộng về quy mô (nhiều cảm biến, nhiều trục điều khiển). Giao thức truyền thông cần phải hỗ trợ việc truyền dữ liệu theo chu kỳ (cyclic data exchange), cũng như truyền sự kiện bất thường (acyclic/event-driven messaging).

Thêm vào đó, hệ thống truyền thông cũng cần hỗ trợ tốt cho khả năng mở rộng – cho phép dễ dàng thêm thiết bị mới (plug-and-play), không làm ảnh hưởng đến độ trễ hay sự ổn định của hệ thống đang vận hành.

2.2.3. Đồng bộ hóa thời gian

Trong nhiều ứng dụng thời gian thực, việc đồng bộ hóa thời gian giữa các thiết bị là cực kỳ quan trọng. Ví dụ, khi đồng bộ hóa chuyển động của nhiều động cơ hoặc robot, nếu mỗi thiết bị nhận lệnh điều khiển tại những thời điểm khác nhau, kết quả có thể gây lệch pha, va chạm cơ khí, hoặc mất cân bằng hệ thống.

Do đó, hệ thống truyền thông phải hỗ trợ các cơ chế đồng bộ thời gian chính xác, chẳng hạn như sử dụng đồng hồ hệ thống toàn cục (Distributed Clocks) hoặc các tín hiệu đồng bộ như Sync/Sync0 trong EtherCAT.



Hình 2.3: Distributed clocks

Đảm bảo mọi thiết bị cập nhật và thực hiện tác vụ trong cùng một thời điểm chuẩn.

2.2.4. Tính tin cậy và khả năng chịu lỗi

Truyền thông trong RTCS cũng phải có độ tin cậy cao, tức là khả năng duy trì kết nối ổn định trong môi trường công nghiệp khắc nghiệt (nhiễu điện từ, rung động, nhiệt độ...). Các giao thức nên tích hợp sẵn các cơ chế:

- Phát hiện lỗi (CRC, watchdogs)
- Truyền lại gói tin bị mất hoặc hỏng
- Tự động khôi phục kết nối
- Hỗ trợ cấu trúc vòng dự phòng (ring redundancy)

Khả năng này giúp hệ thống tránh được tình trạng treo hoặc dừng hoạt động khi xảy ra lỗi truyền thông – điều không thể chấp nhận được trong điều khiển thời gian thực.

2.2.5. Dễ triển khai và tương thích phần mềm điều khiển

Ngoài các yếu tố kỹ thuật, giao thức truyền thông được chọn cũng nên hỗ trợ:

- Tích hợp dễ dàng vào các hệ điều hành thời gian thực (RTCS) hoặc phần mềm điều khiển như TwinCAT, Codesys, v.v.
- Cung cấp thư viện lập trình, tài liệu, driver đầy đủ cho vi điều khiển hoặc hệ thống nhúng.
- Có tính tương thích cao với phần cứng công nghiệp hiện có, giúp giảm chi phí và thời gian triển khai hệ thống.

Từ các yêu cầu nêu trên, có thể thấy rõ rằng các giao thức truyền thông truyền thống như UART, I2C hay SPI không thể đáp ứng đầy đủ cho hệ thống RTCS có nhiều thiết bị và yêu cầu cao về đồng bộ. Những giao thức mới như EtherCAT, PROFINET, EtherNet/IP đã được phát triển để giải quyết triệt để các thách thức này. Trong đề tài này, EtherCAT được chọn là giao thức truyền thông chính, do đặc tính ưu việt về tốc độ, độ trễ thấp, đồng bộ hóa tốt và dễ tích hợp vào hệ thống điều khiển thời gian thực.

2.3. Giới thiệu công nghệ EtherCAT

EtherCAT (Ethernet for Control Automation Technology) đây là một giao thức truyền thông trong công nghiệp, ban đầu được phát triển bởi hãng Beckhoff Automation - Một thương hiệu chuyên sản xuất các dòng PLC (Bộ điều khiển logic khả trình) cho các hệ thống điều khiển thời gian thực và tự động hóa công nghiệp. EtherCAT ra đời vào năm 2003 bởi Beckhoff Automation, được phát triển dựa trên phiên bản Fieldbus có tên là LightBus vào cuối những năm 1980. Vào năm 2004, thương hiệu này đã trao quyền cho ETG (EtherCAT Technology Group) – Tổ chức chịu trách nhiệm quảng bá tiêu chuẩn này.

Hình 2.4: EtherCAT

Hiện nay EtherCAT được tiêu chuẩn hóa theo IEC 61158. EtherCAT là một mạng mở Ethernet Master/Slave thời gian thực. EtherCAT ban đầu được phát triển bởi hãng Beckhoff Automation - Một thương hiệu chuyên sản xuất các dòng PLC (Bộ điều khiển logic khả trình) cho các hệ thống điều khiển thời gian thực và tự động hóa công nghiệp.

EtherCAT sử dụng giao thức Ethernet để truyền dữ liệu nhanh chóng và đồng bộ giữa các thiết bị trong hệ thống. Đặc điểm chính của EtherCAT là khả năng truyền dữ liệu

qua nhiều thiết bị trong một chuỗi liên kết (daisy-chain) duy nhất, từ đó giảm đáng kể chi phí cáp và thời gian cài đặt. EtherCAT cho phép thiết lập giới hạn hiệu suất theo thời gian thực, khả năng xử lý 1000 I/O phân tán trong 30 µs hoặc 100 trục trong 100 µs khi sử dụng cáp sợi quang hoặc cáp xoắn.

Mạng EtherCAT được thiết kế để hỗ trợ các ứng dụng kiểm soát thời gian thực với độ trễ rất thấp và đồng bộ cao. Giao thức này cho phép các thiết bị kết nối, đồng bộ hóa dữ liệu và hoạt động hiệu quả trong các hệ thống tự động hóa phức tạp. EtherCAT đã trở thành một công nghệ phổ biến trong lĩnh vực tự động hóa công nghiệp và được sử dụng trong các ứng dụng như điều khiển động cơ, hệ thống điều khiển máy móc, robot công nghiệp và nhiều ứng dụng khác trong thời đại 4.0 ngày nay.

2.3.1. Ưu điểm của EtherCAT

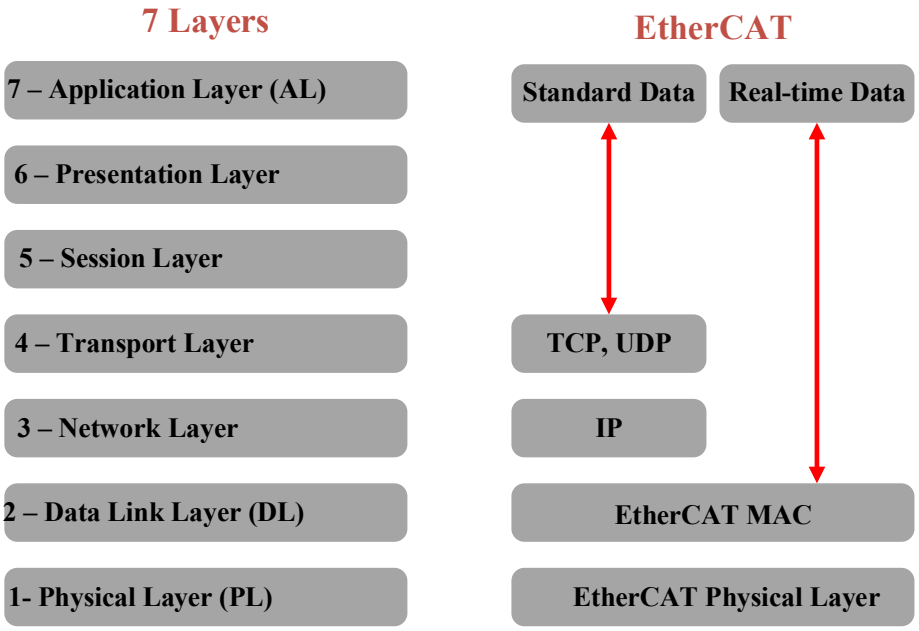
EtherCAT đóng vai trò quan trọng trong công nghiệp tự động hóa hiện nay bởi một lý do duy nhất đó chính là độ chính xác về thời gian. Việc khởi động hay tắt hệ thống máy móc công nghiệp đòi hỏi khắt khe về sự chính xác 100% trong thời gian, thao tác với độ trễ ở mức thấp nhất. EtherCAT là giao thức truyền thông cho phép truyền dữ liệu nhanh chóng và đồng bộ giữa các thiết bị trong hệ thống.

- **Ưu điểm của EtherCAT:**
  + Hiệu suất cao: cấu trúc của giao thức truyền thông công nghiệp EtherCAT nhanh nhất với tốc độ 200 Mbps (chế độ song công Duplex = 100 Mbps x 2).
  + Độ chính xác cao về yếu tố thời gian với độ trễ chỉ 1 µs.
  + Cấu trúc liên kết linh hoạt: Mô hình EtherCAT có thể được thiết lập theo vòng, thẳng, cây, sao không giới hạn.
  + Khả năng mở rộng mạng gần như vô hạn.
  + Thao tác dễ dàng: EtherCAT Master sẽ tự động gán địa chỉ node và gửi thông báo đồng bộ thời gian cho từng thiết bị trong mạng.
  + Cấu hình đơn giản: EtherCAT không sử dụng địa chỉ IP hoặc địa chỉ MAC để hoạt động.
  + Giá cả phải chăng: EtherCAT có mức giá triển khai tương tự hoặc thấp hơn so với mạng Fieldbus thông thường.

2.3.2. Các lớp vật lý của EtherCAT

EtherCAT sử dụng cùng một lớp liên kết vật lý và dữ liệu của Ethernet, được thể hiện bên dưới:

Mô hình tham chiếu OSI có 7 lớp, mỗi lớp đại diện cho một chức năng truyền thông:



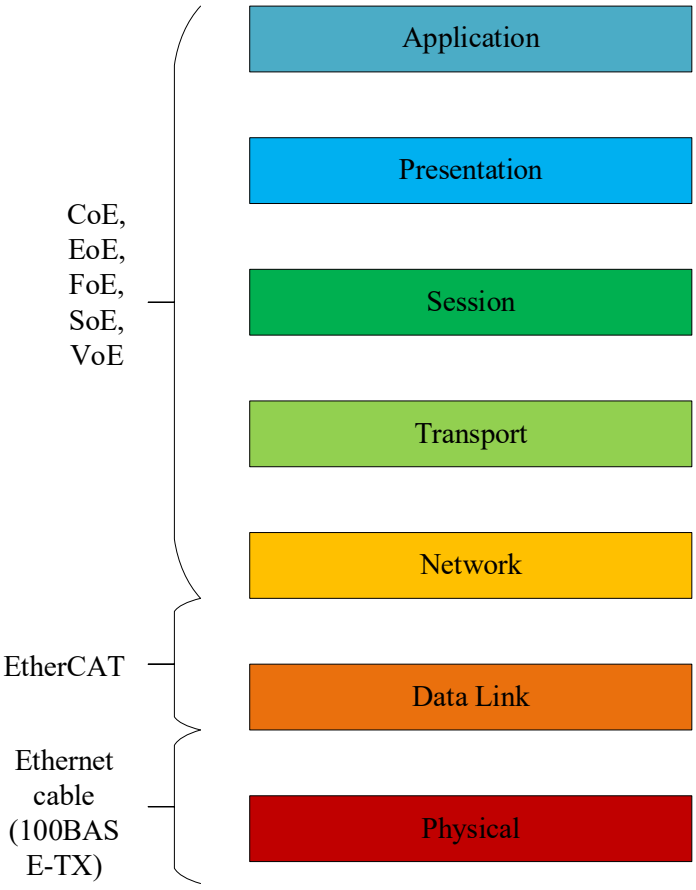
Hình 2.5: Lớp liên kết vật lý và dữ liệu của Ethernet

- Lớp 7 - Application Layer (AL): Tương tác giữa người dùng và mạng, cung cấp giao diện cho ứng dụng.
- Lớp 6 - Presentation Layer: Định dạng dữ liệu (nén, mã hóa).
- Lớp 5 - Session Layer: Quản lý phiên kết nối.
- Lớp 4 - Transport Layer: Đảm bảo truyền dữ liệu an toàn (TCP, UDP).
- Lớp 3 - Network Layer: Định tuyến gói tin (IP).
- Lớp 2 - Data Link Layer: Quản lý truyền tín hiệu qua mối trường vật lý.
- Lớp 1 - Physical Layer (PL): Truyền dữ liệu qua cáp hoặc tín hiệu vô tuyến.

Tối ưu hóa của EtherCAT:

EtherCAT tối ưu hóa quy trình truyền thông bằng cách:

- Bỏ qua các lớp 3 đến 6 trong mô hình OSI:
  + Thay vì sử dụng giao thức IP (Internet Protocol) và giao thức truyền tải (TCP/UDP), EtherCAT truyền dữ liệu thẳng từ Data Link Layer (lớp 2) lên Application Layer (lớp 7).
  + Giảm thiểu độ trễ trên mạng, giúp truyền thông thời gian thực hiệu quả hơn.
- Khai thác hiệu quả MAC (Media Access Control):
  + EtherCAT sử dụng các cách truy cập kênh truyền vật lý tối ưu hóa để truyền gói tin nhanh chóng.



Hình 2.6: Lớp liên kết vật lý và dữ liệu của EtherCAT

- Tính linh hoạt với các dữ liệu:
  + Dữ liệu thời gian thực (“Real-Time Data”) truyền thẳng từ EtherCAT MAC đến Application Layer.
  + Dữ liệu tiêu chuẩn (“Standard Data”) có thể truyền qua giao thức TCP/UDP khi cần.

Lớp vật lý EtherCAT (EtherCAT Physical Layer):

- Ethernet 100BASE-TX:
  + Sử dụng cáp xoắn đôi (twisted pair, CAT5).
  + Tốc độ truyền: 100 Mbps.
  + Khoảng cách: Tối đa 100m giữa hai thiết bị.
- Ethernet 100BASE-FX:
  + Sử dụng cáp quang (fiber optic).
  + Chị hoàn hảo cho môi trường nhiễu nhiễu điện từ.
  + Khoảng cách: Vài km tùy thuộc loại cáp.

Lớp liên kết dữ liệu (Data Link Layer):

- EtherCAT dựa trên lớp liên kết dữ liệu của Ethernet nhưng được tối ưu hóa để tăng hiệu suất:
- Loại bỏ các lớp cao hơn như lớp Mạng (IP) và lớp Truyền tải (TCP/UDP) nhằm giảm độ trễ.
- Đây là nơi dữ liệu được mã hóa thành các gói và truyền đi trong mạng.
- Sử dụng cơ chế "on the fly" để xử lý dữ liệu trực tiếp khi gói tin đi qua các Slave mà không cần dừng lại.
- Tập trung vào việc đảm bảo thời gian chu kỳ chính xác và hiệu quả truyền tải.

2.3.3. Nguyên lý hoạt động của EtherCAT

Chuỗi liên kết (daisy-chain): Các thiết bị EtherCAT được kết nối thành một chuỗi liên kết vật lý thông qua cáp Ethernet. Mỗi thiết bị slave trong chuỗi nhận dữ liệu từ thiết bị trước đó và chuyển tiếp dữ liệu đến thiết bị kế tiếp trong chuỗi. Điều này cho phép truyền dữ liệu qua các thiết bị một cách nhanh chóng và hiệu quả.

Hình 2.7: Kết nối thành chuỗi của các thiết bị EtherCAT

Frame EtherCAT: Dữ liệu trong mạng EtherCAT được gói gọn trong các frame EtherCAT. Mỗi frame chứa các trường dữ liệu như tiêu đề frame, dữ liệu I/O, dữ liệu điều khiển và kiểm soát, và kiểm tra dư.

EtherCAT Master: Là một thiết bị có khả năng điều khiển toàn bộ mạng EtherCAT. EtherCAT Master tạo ra các frame EtherCAT và gửi chúng xuống chuỗi liên kết thiết bị. Master cũng có khả năng thu thập dữ liệu từ các thiết bị slave và điều khiển chúng theo yêu cầu.

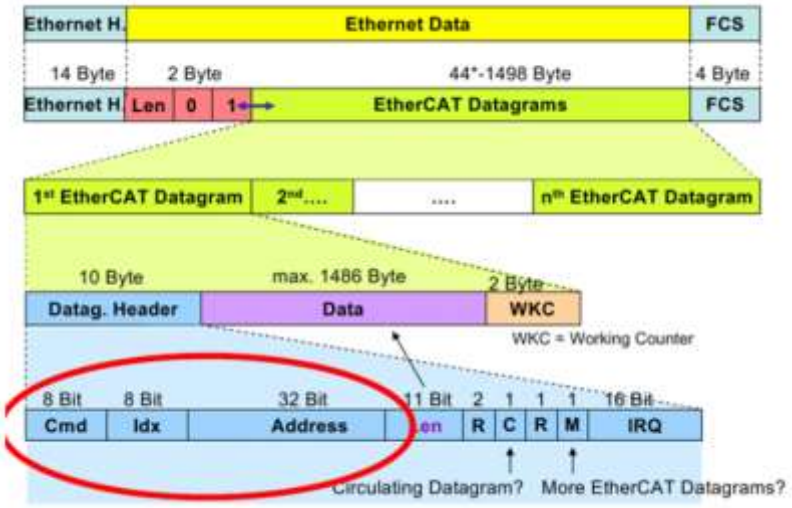
Thiết bị EtherCAT Slave: Là các thiết bị trong mạng EtherCAT, chẳng hạn như cảm biến, bộ điều khiển, động cơ, hay bất kỳ thiết bị nào khác. Mỗi slave được gắn với một địa chỉ duy nhất và có khả năng truyền và nhận dữ liệu trong các frame EtherCAT.

Truyền dữ liệu theo thời gian thực: Dữ liệu trong mạng EtherCAT được truyền qua các frame EtherCAT theo thời gian thực. Master gửi frame xuống chuỗi liên kết và các slave nhận và xử lý frame theo thời gian đồng bộ.

Kiểm soát và đồng bộ: EtherCAT Master kiểm soát toàn bộ quá trình truyền thông và đồng bộ hóa dữ liệu trong mạng dựa trên việc gửi các lệnh và yêu cầu đến các EtherCAT Slave phản hồi theo yêu cầu. Điều này cho phép master kiểm soát và tương tác với các thiết bị slave theo yêu cầu của ứng dụng.

Đồng bộ hoá thời gian thực (Distributed Clock): Ethercat sử dụng cơ chế đồng hồ phân tán để đồng bộ tất cả các thiết bị trong mạng với độ chính xác ở mức 1ns. Master phát tín hiệu đồng bộ và mỗi slave tự điều chỉnh dựa trên tín hiệu này. Tối ưu hoá thời gian thực nhờ: không chờ tín hiệu phải hồi từ từng thiết bị, mà master gửi dữ liệu theo chi kỳ định sẵn

2.3.4. Khung truyền EtherCAT



Hình 2.8: Khung truyền dữ liệu EtherCAT

EtherCAT là một giao thức truyền thông dựa trên Ethernet, được thiết kế đặc biệt cho các hệ thống thời gian thực yêu cầu độ trễ thấp và đồng bộ hóa cao. Khung truyền EtherCAT được xây dựng dựa trên định dạng Ethernet tiêu chuẩn, nhưng được tối ưu hóa để đạt hiệu suất cao.

Như hình minh họa, khung truyền EtherCAT bao gồm ba phần chính:

- Ethernet Header
- EtherCAT Data
- Ethetnet

Ethernet Header:

- EtherType (2 bytes): Loại giao thức. EtherCAT sử dụng giá trị 0x88A4.
- Phần đầu của khung Ethernet chứa thông tin điều khiển và định danh cần thiết để truyền dữ liệu qua mạng.
- Destination MAC Address (6 bytes): Địa chỉ MAC của thiết bị nhận (thường là broadcast).
- Source MAC Address (6 bytes): Địa chỉ MAC của thiết bị gửi.

EtherCAT Data:

Hình 2.9: Cấu trúc của EtherCAT Data

EtherCAT Header:

- Length (11 bit): Độ dài dữ liệu EtherCAT.
- Res. (1 bit): Reserved bit (bit dự phòng, không sử dụng).
- Type (4 bit): Loại dữ liệu EtherCAT.

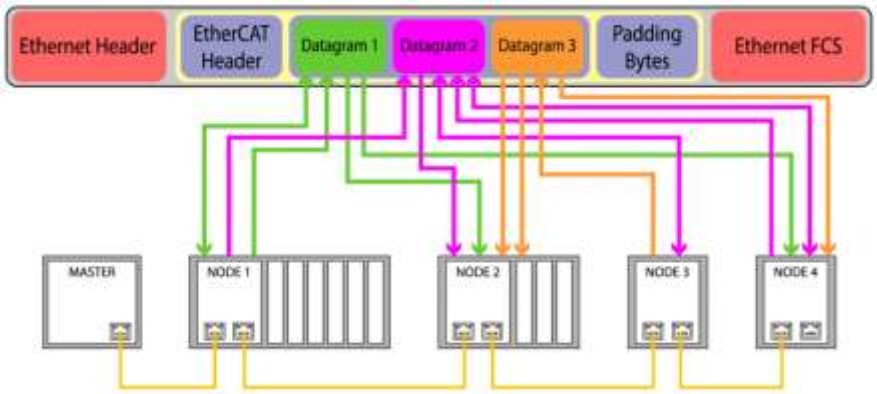
Datagram:

Hình 2.10: Cấu trúc của Datagram

- Datagram Header: Phần đầu của datagram chứa thông tin điều khiển cần thiết để xác định cách xử lý và định danh gói dữ liệu. Thông tin này có thể bao gồm địa chỉ, loại dữ liệu, và các thông tin điều khiển khác.
- Data: Phần chính của datagram, chứa dữ liệu thực tế được truyền tải giữa các thiết bị trong mạng EtherCAT. Dữ liệu này có thể là thông tin từ cảm biến, lệnh điều khiển, hoặc bất kỳ thông tin nào cần thiết cho hoạt động của thiết bị.
- Working Counter: Một bộ đếm cho biết số lần datagram đã được xử lý hoặc số lần truyền tải. Thông tin này có thể được sử dụng để theo dõi trạng thái của datagram và đảm bảo rằng nó đã được xử lý đúng cách.
- Ethernet FCS (Frame Check Sequence): Phần này chứa thông tin kiểm tra lỗi, thường được gọi là kiểm tra CRC (Cyclic Redundancy Check).

Nó được sử dụng để xác minh tính toàn vẹn của dữ liệu trong khung Ethernet. Nếu dữ liệu bị lỗi trong quá trình truyền tải, phần FCS sẽ giúp phát hiện lỗi đó.

Dữ liệu Quá trình:



Hình 2.11: Cấu trúc mẫu của một khung truyền

Trong mỗi chu kỳ, master EtherCAT sẽ tập hợp dữ liệu vào khung truyền (frame) và gửi nó đến các node trong mạng. Các node đọc dữ liệu được gửi đến chúng và ghi dữ liệu phản hồi mới vào khung truyền khi nó di chuyển xuống luồng. Khi khung truyền đến cuối mạng và quay trở lại master, master sẽ đọc dữ liệu mới và xây dựng khung truyền tiếp theo. Dữ liệu được trao đổi giữa master và các slave này được gọi là dữ liệu quá trình tuần hoàn (cyclic process data).

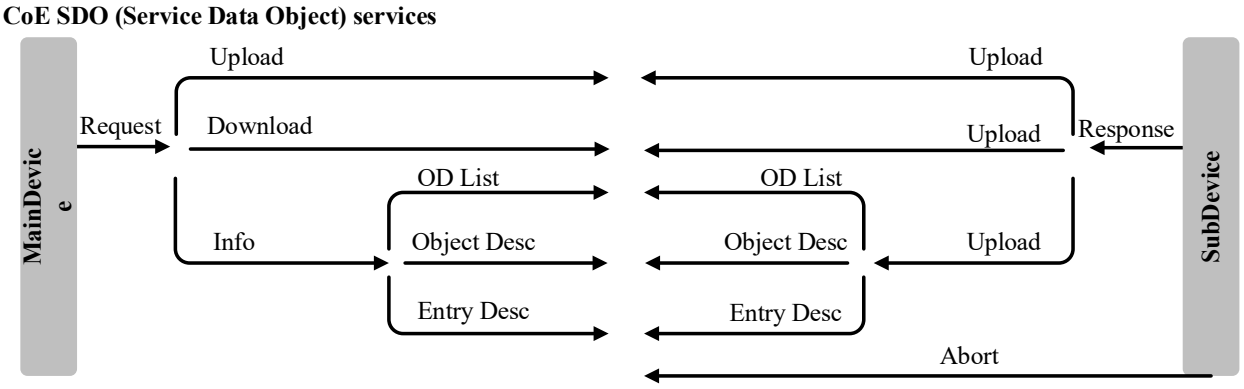
Các thiết bị slave sử dụng Bộ Quản lý Bộ nhớ Fieldbus (FMMU) chịu trách nhiệm đọc và ghi dữ liệu của chúng vào đúng vị trí trong khung truyền. Điều này được gọi là ánh xạ dữ liệu (data mapping) và rất quan trọng để đảm bảo thiết bị master EtherCAT nhận được một hình ảnh quá trình hoàn chỉnh và được sắp xếp trong mỗi chu kỳ. Dữ liệu quá trình tuần hoàn có thể được tập hợp theo nhu cầu của dự án bằng cách kích hoạt các đối tượng dữ liệu quá trình tùy chọn. Việc tổ chức dữ liệu quá trình theo cách này cho phép EtherCAT đạt được hiệu suất nhanh và đáng tin cậy.

Hồ sơ giao tiếp và giao thức hộp thư:

Ngoài dữ liệu quá trình tuần hoàn, EtherCAT còn hỗ trợ giao tiếp phi tuần hoàn, bao gồm các giao thức hộp thư khác nhau. Các hồ sơ giao tiếp EtherCAT được xây dựng nhằm hỗ trợ nhiều thiết bị hiện trường và các lớp ứng dụng hơn. Theo đặc đi EtherCAT, các thiết bị slave không bắt buộc phải hỗ trợ tất cả các hồ sơ giao tiếp; thay vào đó, chúng có thể hỗ trợ một hoặc nhiều hồ sơ phù hợp nhất với trường hợp sử dụng của mình. Hơn nữa, trái ngược với dữ liệu quá trình tuần hoàn, việc truyền dữ liệu giao thức hộp thư không được đảm bảo tuân thủ bất kỳ ràng buộc thời gian thực nào.

CAN Application Protocol over EtherCAT (CoE):

CANopen® là một giao thức giao tiếp phổ biến cho các hệ thống nhúng, thường được sử dụng trong các ứng dụng tự động hóa. Hồ sơ giao tiếp EtherCAT gọi là CoE cho phép triển khai Giao thức Ứng dụng CAN trên mạng EtherCAT. Điều này bao gồm hỗ trợ từ điển đối tượng (Object Dictionary) CANopen, ánh xạ các Đối tượng Dữ liệu Quá trình (PDO), các thông báo lỗi khẩn cấp của CoE, chẩn đoán, và các Đối tượng Dữ liệu Dịch vụ (SDO).



Hình 2.12: Luồng Dịch Vụ CoE SDO trong EtherCAT

Hơn nữa, nhiều hồ sơ CANopen tiêu chuẩn, chẳng hạn như hồ sơ CiA 402 cho các bộ truyền động và hồ sơ CiA 406 cho các bộ mã hóa, có thể được tái sử dụng trong giao tiếp EtherCAT. CoE thường được sử dụng khi các tham số cụ thể của dự án được gửi đến thiết bị slave trong quá trình khởi động theo cấu hình mạng. Các tham số của thiết bị slave cũng có thể được kiểm tra và chỉnh sửa khi mạng đang ở trạng thái hoạt động thông qua CoE.

Object Dictionary:

Các thiết bị EtherCAT chứa rất nhiều mục dữ liệu (đối tượng) có thể cần truy cập để cấu hình và trong quá trình vận hành thông thường. Nhà sản xuất thiết bị tạo ra các đối tượng này. Các đối tượng thay đổi tùy theo thiết bị và thường có nhiều điểm chung giữa các thiết bị.

Object Dictionary là danh sách các dữ liệu (đối tượng) có thể truy cập trong một thiết bị EtherCAT. Object Dictionary cung cấp các kiểu dữ liệu và khả năng truy cập của các đối tượng, chẳng hạn như chỉ đọc (read-only), đọc/ghi (read/write), địa chỉ (Index) và Subindex giúp định vị và truy cập từng mục dữ liệu, v.v. RMC hỗ trợ giao thức CANopen over EtherCAT (CoE), được hầu hết các thiết bị EtherCAT hỗ trợ. CoE định nghĩa cách định địa chỉ (Index) và kiểu dữ liệu của các đối tượng. Tuy nhiên, RMC

không hỗ trợ trực tiếp SERCOS over EtherCAT (SoE), vì SoE sử dụng một phương pháp định địa chỉ khác.

Cấu trúc Object Dictionary:

Bảng 2.1: Phân vùng chỉ số đối tượng (Object Dictionary)

Object Index Range	Description
0x1000 – 0x1FFF	Khu vực giao tiếp CoE (được cấu hình bởi SDK)
0x2000 – 0x5FFF	Khu vực cụ thể của nhà sản xuất
0x6000 – 0x9FFF	Khu vực hồ sơ

a. Khu vực giao tiếp (0x1000 – 0x1FFF)

Những đối tượng này cung cấp thông tin về danh tính thiết bị, phiên bản phần mềm và phần cứng, phân bổ SyncManager, cấu hình ánh xạ PDO cùng với các cài đặt khác. Những đối tượng này được tạo tự động bởi SDK sử dụng thông tin được cung cấp bởi các hàm API.

Bảng 2.2: Từ điển đối tượng chi tiết

Index	Tên	Chức năng/API liên quan
0x1000	Loại thiết bị (Device Type)	EC_API_SLV_setDeviceType
0x1001	Thanh ghi lỗi (Error Register)	EC_API_SLV_setErrorRegister
0x1008	Tên thiết bị (Device Name)	EC_API_SLV_setProductName
0x1009	Phiên bản phần cứng (Hardware Version)	EC_API_SLV_setHwVersion
0x100A	Phiên bản phần mềm (Software Version)	EC_API_SLV_setSwVersion
0x1018	Nhận diện thiết bị (Identity Object)	Bao gồm Vendor ID, Product Code, Revision Number, Serial Number
0x1600 - 0x17FF	RxPDO đầu vào	Tự động tạo dựa trên cấu hình PDO
0x1A00 - 0x1BFF	TxPDO đầu ra	Tự động tạo dựa trên cấu hình PDO
0x1C00	Loại quản lý đồng bộ (Sync Manager Types)	Tạo tự động bởi SDK
0x1C10 - 0x1C13	Gán PDO cho Sync Manager	Tạo tự động bởi SDK
0x1C30 - 0x1C33	Đồng bộ hóa Sync Manager	Tạo tự động bởi SDK

b. Khu vực cụ thể của nhà sản xuất (0x2000-0x5FFF)

Các đối tượng dành riêng nhà sản xuất (Vendor-specific objects). Những đối tượng này được định nghĩa bởi nhà sản xuất thiết bị. Do đó, chúng sẽ khác nhau giữa các thiết bị.

c. Khu vực hồ sơ (0x6000-0xFFFF)

Những đối tượng này được định nghĩa bởi một hồ sơ ứng dụng chuẩn. Nếu thiết bị sử dụng một hồ sơ chuẩn, chẳng hạn như CiA-402 cho bộ điều khiển động cơ, hoặc CiA-408 cho van, thì các Index này sẽ được điền và sẽ tương tự nhau giữa các thiết bị loại này.

Bảng 2.3: Phân vùng chỉ số dữ liệu (PDO Mapping Ranges)

Index	Tên
0x6000 – 0x6FFF	Tx-mappable, read-only
0x7000 – 0x7FFF	Rx-mappable, read-writeable
0x8000 – 0x8FFF	Read-writeable, usually not mappable
0x9000 – 0x9FFF	Read-only, usually not mappable
0xA000 – 0xAFFF	Dành cho các thông tin chẩn đoán thiết bị
0xF000 – 0xFFFF	Bao gồm các đối tượng đặc thù của thiết bị

Ứng dụng trong EtherCAT

- PDO Mapping:
  - PDO (Process Data Object): Làm cầu nối truyền dữ liệu trong thời gian thực giữa các thiết bị.
  - Object Dictionary được sử dụng để chọn các đối tượng cần ánh xạ vào PDO để truyền dữ liệu nhanh chóng mà không cần giao tiếp SDO.
- SDO Communication:
  - SDO (Service Data Object): Được sử dụng để đọc hoặc ghi các đối tượng từ Object Dictionary.
  - Phù hợp cho cấu hình hoặc các yêu cầu giao tiếp không thời gian thực.
- Giám sát thời gian thực: Các giá trị trong Object Dictionary có thể được giám sát qua EtherCAT Diagnostics, trong các tab:
  - CoE Object Dictionary: Hiển thị thông tin cấu hình của các đối tượng.
  - Process Data: Hiển thị giá trị dữ liệu thời gian thực.

Ethernet over EtherCAT (EoE):

Một hồ sơ giao tiếp mạnh mẽ khác là Ethernet qua EtherCAT. Với EoE, lưu lượng dữ liệu Ethernet có thể được truyền trong mạng EtherCAT. Các thiết bị switchport có thể được sử dụng trong phân đoạn EtherCAT để kết nối với các thiết bị Ethernet như máy

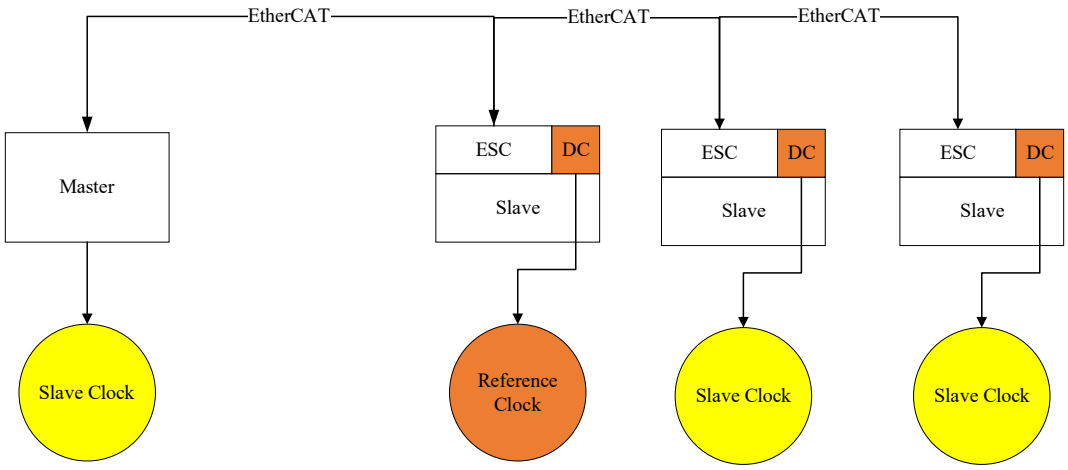
tính và bộ định tuyến. Thiết bị switchport chịu trách nhiệm chèn các phân đoạn TCP/IP vào luồng dữ liệu EtherCAT. EoE thường được sử dụng để truy cập máy chủ web của thiết bị slave nhằm cấu hình và chẩn đoán thiết bị. Theo cách này, các thiết bị xuất hiện như thể chúng được kết nối trực tiếp với một mạng cục bộ không phải EtherCAT.

File Access over EtherCAT (FoE):

Truy cập tệp qua EtherCAT, hay FoE, là một giao thức hộp thư phổ biến khác được sử dụng trong giao tiếp EtherCAT. FoE thường được sử dụng để cập nhật firmware cho các thiết bị trong mạng EtherCAT.

2.3.5. Đồng bộ hóa - Đồng hồ phân tán (DC)

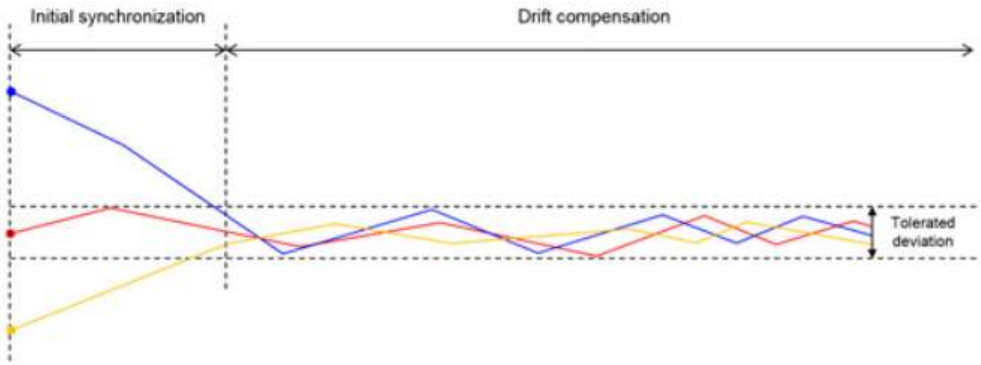
Các thiết bị phụ cần được đồng bộ hóa thời gian để thực hiện điều khiển chuyển động đồng bộ hoặc các hành động đồng bộ khác trong mạng. Một cách có thể thực hiện điều này trong trường hợp của EtherCAT là sử dụng đồng hồ phân tán.



Hình 2.12: Đồng bộ phân tán

Ý tưởng cơ bản là sử dụng một đồng hồ tham chiếu và truyền thông tin qua khung EtherCAT để tất cả các thiết bị slave trong mạng biết chính xác thời gian trên đồng hồ tham chiếu đó và điều chỉnh đồng hồ cục bộ của mình để khớp với thời gian tham chiếu. Master truyền một thông điệp phát quảng bá. Thời gian đến của khung thông điệp này được ghi lại bởi mỗi thiết bị slave, một lần khi khung đi từ master và một lần khi khung quay lại master (EtherCAT là full duplex).

Thông tin đã ghi lại sau đó được đọc bởi master, độ trễ đến từng slave được tính toán tương ứng và độ trễ này được truyền trong các khung EtherCAT kế tiếp để chỉ ra cho từng slave cách điều chỉnh đồng hồ cục bộ của mình để đồng bộ với thời gian tham chiếu. Khung đặc biệt này được gửi định kỳ để bù trừ trôi hoặc nhiễu trong mạng. Trong Hình 2.13, giai đoạn bù trừ trôi được thể hiện bằng các đường dao động.



Hình 2.13: Đồng Bộ Hóa và Bù Trôi Sai Số trong Điều Khiển Thời Gian Thực

2.3.6. Đặc điểm nổi bật của EtherCAT

EtherCAT sở hữu nhiều ưu điểm nổi bật khiến nó trở thành một trong những giao thức truyền thông công nghiệp phổ biến nhất hiện nay:

- Tốc độ truyền cực nhanh: Sử dụng chuẩn Ethernet vật lý 100BASE-TX (100 Mbps), nhưng hiệu suất thực tế đạt được rất cao do cơ chế xử lý dữ liệu "on-the-fly".
- Độ trễ cực thấp: EtherCAT có thể đạt độ trễ dưới 100 µs cho cả hệ thống nhiều thiết bị – điều cực kỳ quan trọng trong hệ thống điều khiển thời gian thực.
- Đồng bộ thời gian chính xác: Hỗ trợ cơ chế Distributed Clocks, giúp đồng bộ tất cả các slave trong hệ thống với sai số dưới 1 µs. Đây là yếu tố then chốt trong việc đồng bộ hóa tốc độ và vị trí giữa nhiều động cơ.
- Chi phí phần cứng thấp: Vì sử dụng chuẩn Ethernet vật lý thông thường, EtherCAT cho phép sử dụng cáp RJ45 tiêu chuẩn, không cần thiết bị chuyển đổi phức tạp như một số giao thức khác.
- Khả năng mở rộng tốt: Một mạng EtherCAT có thể kết nối lên đến 65.535 thiết bị, với cấu trúc dạng chuỗi (line), vòng (ring), sao (star), hoặc kết hợp tùy nhu cầu.
- Tự động phát hiện và cấu hình thiết bị: Nhờ vào cơ chế hot-plug, các thiết bị có thể được gắn thêm hoặc thay thế mà không cần cấu hình lại toàn bộ hệ thống.
- Tương thích cao: Hệ thống có thể tích hợp dễ dàng với phần mềm điều khiển công nghiệp như TwinCAT (Beckhoff), Codesys, hoặc các nền tảng lập trình PLC khác.

2.3.7. Ứng dụng của EtherCAT trong công nghiệp

EtherCAT được ứng dụng rộng rãi trong nhiều lĩnh vực:

- Điều khiển chuyển động (motion control): Đặc biệt phù hợp với các hệ thống yêu cầu đồng bộ nhiều trục động cơ với độ chính xác cao (robot, dây chuyền lắp ráp).
- Máy CNC, in 3D, máy đóng gói: Nơi cần điều khiển nhiều động cơ servo và thiết bị I/O trong thời gian thực.
- Ô tô & hàng không: Ứng dụng trong hệ thống test bench, hệ thống mô phỏng HIL (Hardware-in-the-loop).
- Tự động hóa nhà máy thông minh: Trong các giải pháp SCADA, MES, và các hệ thống IIoT (Industrial IoT).

Trong đề tài này, EtherCAT được lựa chọn làm nền tảng truyền thông chính giữa máy tính điều khiển trung tâm (chạy phần mềm TwinCAT) và các thiết bị slave sử dụng chip LAN9252. Với khả năng truyền dữ liệu nhanh, chính xác và đồng bộ hóa hiệu quả, EtherCAT cho phép hệ thống điều khiển và đồng bộ tốc độ hai động cơ trong thời gian thực, đáp ứng yêu cầu kỹ thuật khắt khe của hệ thống điều khiển hiện đại.

2.4. Ưu điểm của EtherCAT so với các giao thức khác (CAN, Modbus, v.v.)

Trên thị trường hiện nay tồn tại nhiều giao thức truyền thông công nghiệp được sử dụng trong các hệ thống điều khiển tự động, bao gồm Modbus, CAN, PROFIBUS, PROFINET, EtherNet/IP, SERCOS, và các biến thể khác nhau của Ethernet công nghiệp. Mỗi giao thức có ưu điểm riêng, phù hợp với từng môi trường và ứng dụng cụ thể. Tuy nhiên, trong bối cảnh yêu cầu ngày càng cao về tốc độ, tính xác định và khả năng đồng bộ trong các hệ thống thời gian thực, EtherCAT đã chứng minh được nhiều lợi thế vượt trội so với các giao thức truyền thống.

Dưới đây là một số ưu điểm nổi bật của EtherCAT khi so sánh với các giao thức công nghiệp phổ biến khác:

2.4.1. Tốc độ và độ trễ

- **EtherCAT:**
  Xử lý frame dữ liệu theo cơ chế "on-the-fly", không cần lưu và chờ xử lý tại từng node. Điều này giúp giảm tối đa độ trễ trong toàn mạng. Một chu kỳ điều khiển có thể chỉ mất vài µs đến vài trăm µs cho toàn bộ hệ thống hàng chục đến hàng trăm thiết bị.
- **So sánh:**
  Các giao thức như Modbus RTU hay CANopen có tốc độ truyền thấp (Modbus: ~115.2 kbps, CAN: ~1 Mbps) và truyền dữ liệu tuần tự, dẫn đến độ trễ lớn hơn

và khó mở rộng hệ thống. PROFIBUS và PROFINET có hiệu suất khá cao, tuy nhiên vẫn không đạt được độ trễ thấp như EtherCAT khi số lượng node tăng.

2.4.2. Tính xác định và đồng bộ hóa

- **EtherCAT:**
  Hỗ trợ cơ chế Distributed Clocks với độ lệch thời gian giữa các slave chỉ còn <1 µs, cực kỳ phù hợp cho ứng dụng đồng bộ hóa động cơ nhiều trục hoặc điều khiển robot.
- **So sánh:**

  Một số giao thức như PROFINET IRT hay SERCOS III cũng hỗ trợ đồng bộ hóa, tuy nhiên cần phần cứng phức tạp và cấu hình phức tạp hơn. Các giao thức thông thường như Modbus TCP hay EtherNet/IP không có cơ chế đồng hồ phân tán, do đó không phù hợp cho ứng dụng điều khiển real-time có đồng bộ cao.

2.4.3. Băng thông và hiệu quả sử dụng

- **EtherCAT:**
  Sử dụng băng thông hiệu quả gần như tối đa nhờ việc gói tin duy nhất đi qua tất cả slave. Ngoài ra, EtherCAT cho phép truy cập I/O trực tiếp tại mức bit, rất tiết kiệm dữ liệu truyền và tài nguyên xử lý.
- **So sánh:**

  Các giao thức như Modbus TCP hoặc EtherNet/IP sử dụng mô hình client-server, yêu cầu nhiều gói tin và thao tác riêng lẻ cho mỗi thiết bị, dẫn đến lãng phí băng thông khi số thiết bị tăng lên.

2.4.4. Đơn giản về phần cứng và chi phí triển khai

- **EtherCAT:**
  Có thể sử dụng hạ tầng cáp Ethernet tiêu chuẩn (RJ45/Cat5e), không cần switch chuyên dụng. Đặc biệt, chip EtherCAT slave có thể tích hợp trực tiếp vào hệ thống nhúng, giảm chi phí so với các giải pháp có phần cứng trung gian.
- **So sánh:**

  Các giao thức như PROFINET IRT hoặc SERCOS yêu cầu phần cứng điều khiển riêng biệt hoặc FPGA đắt tiền. CAN hay PROFIBUS dùng chuẩn vật lý khác biệt, cần thiết bị chuyển đổi chuyên dụng và tốn chi phí hơn trong bảo trì.

2.4.5. Khả năng mở rộng và tự động phát hiện thiết bị

- **EtherCAT:**
  Hỗ trợ lên đến 65.535 thiết bị trên một mạng duy nhất. Có khả năng tự động phát hiện thiết bị, hỗ trợ hot - plug, thuận tiện trong việc nâng cấp hoặc bảo trì hệ thống mà không cần dừng toàn bộ hệ thống.
- **So sánh:**
  Các giao thức như CANopen hoặc Modbus có giới hạn số thiết bị thấp (CAN: ~127 node), khả năng phát hiện thiết bị và thay thế phần cứng hạn chế.

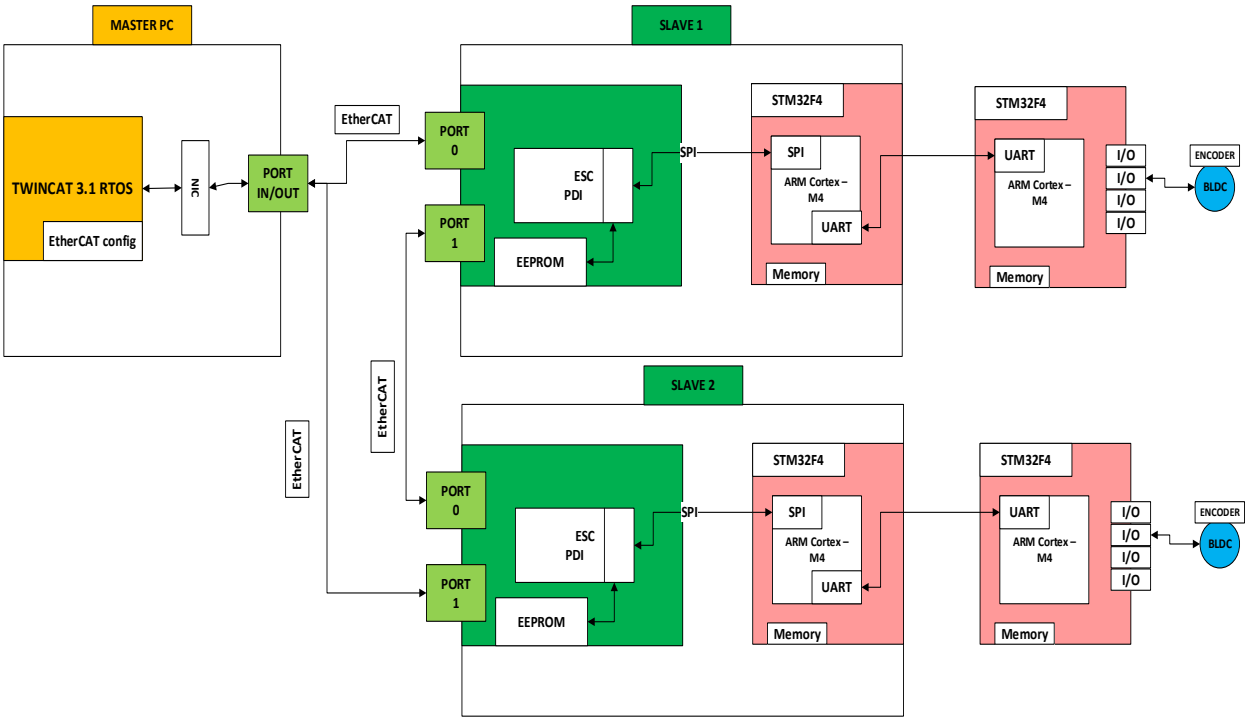
2.4.6. Tính mở và cộng đồng phát triển

- **EtherCAT:**
  Dù được phát triển bởi Beckhoff, EtherCAT là một tiêu chuẩn mở (IEC 61158) và được quản lý bởi tổ chức phi lợi nhuận EtherCAT Technology Group (ETG). Các công ty, tổ chức, và cá nhân có thể đăng ký sử dụng, phát triển thiết bị tương thích một cách minh bạch và dễ dàng.
- **So sánh:**
  Một số giao thức khác có thể yêu cầu giấy phép riêng hoặc thiếu cộng đồng hỗ trợ rộng lớn như EtherCAT.

CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG ĐỒNG BỘ TỐC ĐỘ ĐỘNG CƠ

3.1. Tổng quan về thiết kế hệ thống EtherCAT

Trong các hệ thống điều khiển hiện đại, đặc biệt là trong lĩnh vực tự động hóa và robot công nghiệp, yêu cầu về việc đồng bộ hóa tốc độ và vị trí giữa nhiều động cơ là vô cùng quan trọng. Việc này không chỉ giúp đảm bảo hiệu suất vận hành ổn định mà còn làm tăng độ chính xác trong các ứng dụng như băng tải đồng bộ, cánh tay robot, hoặc hệ thống truyền động nhiều trục.



Hình 3.1: Tổng quan về hệ thống đồng bộ động cơ

Đề tài này hướng đến việc xây dựng một hệ thống điều khiển thời gian thực, trong đó hai động cơ được đồng bộ tốc độ và vị trí thông qua nền tảng giao tiếp EtherCAT. EtherCAT là giao thức truyền thông Ethernet công nghiệp thời gian thực, với độ trễ thấp, khả năng mở rộng cao và hiệu suất truyền dữ liệu vượt trội. Hệ thống sử dụng PC làm EtherCAT Master, kết nối đến hai vi điều khiển STM32 đóng vai trò EtherCAT Slave thông qua module LAN9252, và thực hiện thuật toán FOC để điều khiển hai động cơ BLDC.

Tổng thể hệ thống được phân thành ba lớp chính:

3.1.1. Lớp điều khiển trung tâm (PC - EtherCAT Master):

- Máy tính cá nhân chạy hệ điều hành Windows cài phần mềm TwinCAT 3 của hãng Beckhoff.
- TwinCAT đóng vai trò EtherCAT Master, chịu trách nhiệm tạo chu kỳ truyền và nhận dữ liệu EtherCAT với tần số cao (~1ms hoặc nhỏ hơn).
- Trong chu kỳ hoạt động, TwinCAT gửi tín hiệu điều khiển như tốc độ đặt, vị trí đặt, … đến các thiết bị slave, đồng thời thu thập dữ liệu phản hồi như tốc độ thực tế, vị trí góc của động cơ để xử lý và giám sát.
- Các thuật toán đồng bộ tốc độ và vị trí giữa hai động cơ có thể được xử lý một phần tại TwinCAT hoặc tại slave tùy vào cấu trúc hệ thống và yêu cầu thời gian thực.

3.1.2. Lớp EtherCAT Slave (LAN9252 + STM32F446):

- Sử dụng hai board EtherCAT Click tích hợp chip LAN9252 – chip chuyên dụng cho giao tiếp EtherCAT.
- LAN9252 không có bộ xử lý, do đó cần vi điều khiển hỗ trợ để tạo thành một thiết bị slave hoàn chỉnh.
- Cả hai EtherCAT Click kết nối với STM32F446 thông qua giao tiếp SPI.
- Hai vi điều khiển STM32 này chịu trách nhiệm:
  + Nhận dữ liệu EtherCAT từ PC thông qua LAN9252.Trích xuất và xử lý dữ liệu điều khiển như tốc độ đặt, vị trí đặt.
  + Truyền lệnh điều khiển đến vi điều khiển điều khiển động cơ.
  + Nhận phản hồi từ STM32F446 như tốc độ thực tế, vị trí góc, và truyền ngược lại về cho PC.
  + Cần đảm bảo việc đồng bộ dữ liệu trong quá trình truyền để duy trì tính thời gian thực.

3.1.3. Lớp điều khiển động cơ (STM32F446):

- STM32F446 đóng vai trò điều khiển trực tiếp động cơ BLDC thông qua thư viện SimpleFOC.
- Nhận lệnh điều khiển TwinCAT thông qua EtherCAT Click như tốc độ mục tiêu, vị trí, sau đó xử lý và điều khiển động cơ theo lệnh này.

- Đọc tín hiệu phản hồi từ encoder tích hợp trên động cơ để đo tốc độ và vị trí thực tế.
- Thực hiện các thuật toán điều khiển như PID, điều chế PWM để ổn định tốc độ động cơ.
- Có thể thực hiện thuật toán đồng bộ hóa với động cơ còn lại bằng cách so sánh tốc độ và vị trí của hai động cơ (tùy cấu hình hệ thống).
- Truyền lại dữ liệu phản hồi về STM32F446 để gửi về PC.

Nguyên lý hoạt động tổng quát:

- TwinCAT (EtherCAT Master) trên máy tính gửi dữ liệu điều khiển (tốc độ hoặc vi trí mục tiêu) đến các thiết bị slave thông qua giao thức EtherCAT.
- Chip LAN9252 trên các EtherCAT Click nhận dữ liệu và chuyển tiếp đến vi điều khiển STM32F446 qua giao tiếp SPI.
- STM32F446 thực hiện điều khiển động cơ (BLDC) bằng thuật toán FOC, đồng thời đọc tín hiệu phản hồi từ encoder để tính toán tốc độ và vị trí.
- Dữ liệu phản hồi được truyền ngược về theo chiều: STM32F446 → LAN9252 → TwinCAT để cập nhật và giám sát.
- Việc đồng bộ tốc độ giữa hai động cơ được đảm bảo nhờ cơ chế điều khiển phối hợp giữa master và các vi điều khiển slave, đảm bảo hoạt động thời gian thực và chính xác.

3.2. Yêu cầu kỹ thuật của hệ thống:

- Truyền dữ liệu EtherCAT với chu kỳ nhỏ hơn hoặc bằng 1ms.
- Đồng bộ tốc độ hai động cơ BLDC với sai số nhỏ hơn 5%.
- Hệ thống hoạt động ổn định trong thời gian dài.
- Tốc độ truyền SPI tối thiểu 10MHz.
- SPI giữa STM32 và LAN9252 truyền dữ liệu ổn định với độ trễ thấp (<1ms).
- Encoder độ phân giải cao (ít nhất 1000 xung/vòng).
- Sử dụng thuật toán FOC hoặc các thuật toán điều khiển nâng cao hơn nếu cần thiết.

3.3. Lựa chọn thiết bị cho hệ thống:

3.3.1. Giao tiếp EtherCAT – EtherCat Click

– **Giới thiệu:**

EtherCAT Click là một bo mạch nhỏ gọn tích hợp chip LAN9252 của Microchip, cho phép kết nối EtherCAT qua giao tiếp SPI với vi điều khiển. LAN9252 tích hợp 2 cổng Ethernet PHY hỗ trợ tốc độ 100Mbps và tính năng HP Auto-MDIX, cho phép sử dụng cả cáp LAN thường hoặc cáp chéo. Bo mạch này phù hợp cho các ứng dụng công nghiệp như điều khiển tự động, hệ thống van, nguồn điện, v.v.

Hình 3.2: EtherCAT Click (LAN9252)

– **Thông số kỹ thuật:**

Bảng 3.1: Thông số kỹ thuật của EtherCAT Click

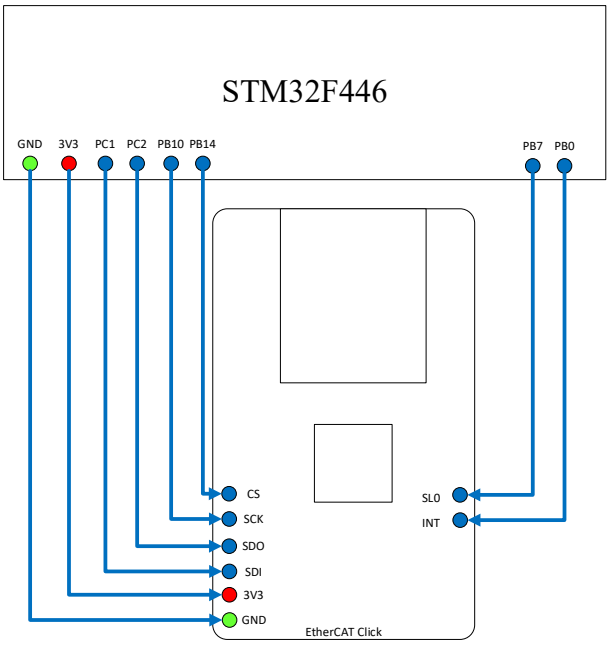
Thông số	Mô tả
Chip điều khiển	LAN9252
Cổng Ethernet	2 cổng Ethernet 100BASE-TX full-duplex
Tốc độ truyền	100 Mbps
Hỗ trợ Auto-MDIX	Có (cho phép sử dụng cáp LAN trực tiếp hoặc chéo)
Giao tiếp với MCU	SPI/SQI

– **Cấu hình EtherCAT Click:**

+ **Kết nối phần cứng:**

- Kết nối EtherCAT Click với mạng EtherCAT qua cổng RJ45.
- Kết nối chân SPI của EtherCAT Click với STM32:
- MISO (PA7): Dữ liệu từ Click về STM32.
- MOSI (PA6): Dữ liệu từ STM32 đến Click.
- SCLK (PA5): Đồng hồ từ STM32.
- CS (Chip Select): Chọn module EtherCAT Click để giao tiếp SPI.
- INT (PB8): Chân ngắt
- 3.3V: Nguồn 3.3VGND: Chân GND
- Nguồn cấp: Cấp nguồn cho EtherCAT Click (3.3V)

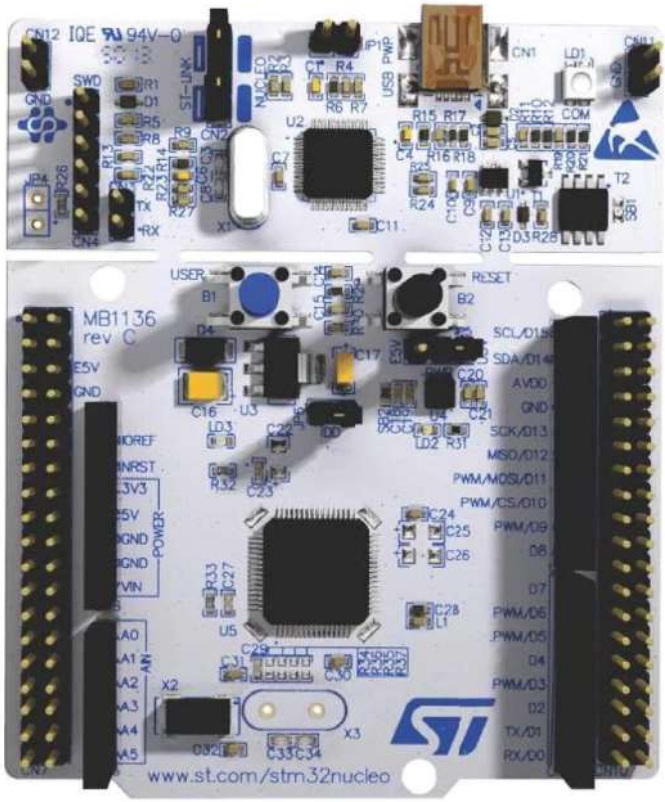
– **Sơ đồ nối dây:**



Hình 3.3: Sơ đồ kết nối EtherCAT Click

3.3.2. Vi điều khiển Nucleo-64 STM32F446RE

Vi điều khiển STM32F446 sử dụng lõi Arm® Cortex®-M4 32-bit RISC hiệu năng cao, hoạt động ở tần số tối đa 180 MHz. Lõi Cortex-M4 có bộ tính toán dấu phẩy động (FPU) hỗ trợ xử lý dữ liệu đơn chính xác, tích hợp tập lệnh DSP và đơn vị bảo vệ bộ nhớ (MPU) tăng cường bảo mật ứng dụng. Các chip này có bộ nhớ nhúng tốc độ cao (Flash tối đa 512 Kbytes, SRAM tối đa 128 Kbytes), SRAM dự phòng 4 Kbytes, và nhiều I/O và ngoại vi được kết nối qua các bus APB, AHB và ma trận bus đa AHB 32-bit. Được trang bị ba ADC 12-bit, hai DAC, RTC tiết kiệm năng lượng, mười hai timer 16-bit (bao gồm hai timer PWM điều khiển động cơ), hai timer 32-bit, cùng các giao tiếp chuẩn và nâng cao.



Hình 3.4: Vi điều khiển STM32F446RE

– **Thông số kỹ thuật:**

Bảng 3.2: Thông số kĩ thuật STM32F446

Thông số	Mô tả
Bộ xử lý lõi	ARM® Cortex®-M4
Kích thước lõi	Lõi đơn 32-Bit
Tốc độ	180MHz
Kết nối	CANbus, EBI/EMI, Ethernet, I2C, IrDA, LINbus, SPI, UART/USART, USB OTG
Thiết bị ngoại vi	Phát hiện/Đặt lại khi mất điện, DMA, I2S, LCD, POR, PWM, WDT
Số lượng I/O	114

3.3.3. Simple FOC

SimpleFOC Shield là một bo mạch mở rộng được thiết kế để 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 3.5: MKS SimpleFOC Shield V2.0.4

- **Đặc điểm:**
  + Khả năng điều khiển động cơ:
    - Điều khiển động cơ BLDC hoặc động cơ gimbal.
    - Hỗ trợ điều khiển với cảm biến (encoder, hall sensor) hoặc không cảm biến.
  + Tương thích phần cứng:
    - Hoạt động với nhiều loại vi điều khiển (STM32, Arduino, ESP32, Raspberry Pi).
    - Tương thích với chuẩn chân Arduino Uno R3.
  + Điện áp và dòng tải:
    - Điện áp đầu vào từ 12V đến 24V (có thể hơn tùy phiên bản).
    - Dòng điện đầu ra tối đa ~10A (tùy thuộc vào tản nhiệt và động cơ).
  + Tích hợp linh kiện:
    - Driver cầu H (L6234).
    - Khe cắm cho cảm biến Hall, encoder.
- **Nối dây cấp nguồn:**

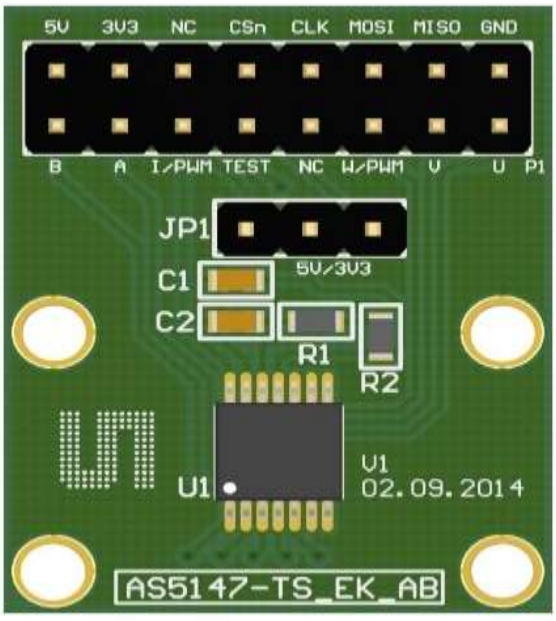
+ Kết nối cực dương (+) của nguồn điện với terminal dương (có ký hiệu "V" hoặc "+") ở phía trái dưới của board.
+ Kết nối cực âm (-) của nguồn điện với terminal âm (có ký hiệu "GND" hoặc "-").
+ Đảm bảo nguồn cung cấp điện áp trong khoảng 8V-12V, phù hợp với yêu cầu của board.

– **Nối đầu ra với động cơ BLDC:**

+ Các đầu ra động cơ nằm ở khu vực bên phải, gồm 3 chân ký hiệu U, V, W:
+ Nối dây của các pha động cơ BLDC với các chân tương ứng U, V, W.
+ Nếu động cơ không quay đúng hướng hoặc không hoạt động ổn định, bạn có thể thử hoán đổi hai dây bất kỳ trong ba pha.

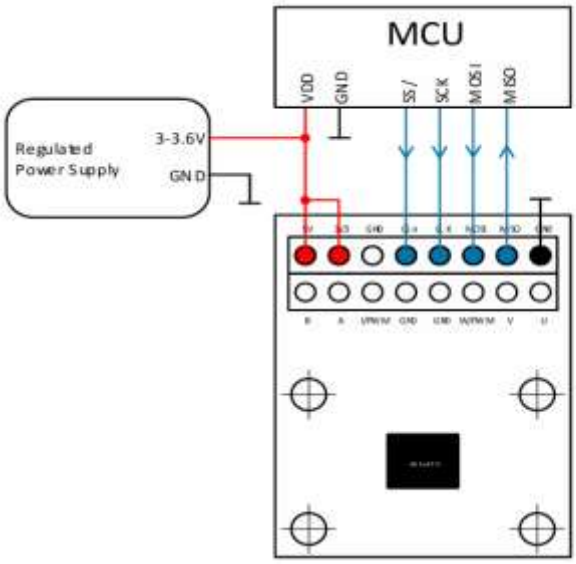
3.3.4. Cảm biến vị trí AS5147U

AS5047U là cảm biến từ Hall CMOS đo góc từ trường, xử lý tín hiệu qua AFE, ADC và CORDIC để tính toán góc và tự động điều chỉnh hệ số khuếch đại (AGC). Thiết bị có độ phân giải 14-bit nội bộ (SPI) và đầu ra ABI lập trình từ 10 đến 14-bit. AS5047U hỗ trợ bù góc động bằng thuật toán dự đoán để giảm sai số trễ, tích hợp bộ lọc giảm nhiễu ở tốc độ thấp, và hoạt động ổn định đến 28.000 vòng/phút. Cảm biến cung cấp đầu ra linh hoạt qua SPI, ABI, UVW hoặc PWM, với các thiết lập được lập trình qua SPI và lưu trữ không thay đổi sau khi cấp nguồn.



Hình 3.6: Encoder AS5147U

Sơ đồ nối dây:



Hình 3.7: Sơ đồ kết nối AS5047U

- **Nguồn:**
  + Chân VDD của mạch điều khiển được nối trực tiếp với VDD của nguồn (3-3.6V).
  + GND của mạch điều khiển và MCU đều nối chung với GND của nguồn.
- **Giao tiếp SPI:**
  + Các chân SPI của MCU được nối tương ứng với các chân SPI của mạch điều khiển:
    - SS → Chân lựa chọn slave.
    - SCK → Tín hiệu clock.
    - MOSI → Tín hiệu dữ liệu từ MCU đến mạch điều khiển.
    - MISO → Tín hiệu dữ liệu từ mạch điều khiển đến MCU.
- **Các tín hiệu PWM:** Mạch điều khiển sử dụng các tín hiệu PWM (Pulse Width Modulation) từ MCU để điều khiển tốc độ và hướng của động cơ BLDC.

3.3.5. 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.

- **Thông số kĩ thuật:**

Bảng 3.3: Thông số kĩ thuật đông cơ gimbal 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 %

– **Đặc tính cơ – điện:**

Bảng 3.4: 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 lý 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.

3.4. Triển khai Slave

Slave EtherCAT bao gồm hai thành phần chính linh kiện phần cứng:

- Bộ điều khiển phụ EtherCAT (ESC) xử lý giao tiếp EtherCAT.
- Khối xây dựng chuyển động xử lý ESC và thông tin mà cần phải được trao đổi.

Các thành phần phần mềm được triển khai trong phần mềm phụ bao gồm:

- Phần mềm EtherCAT slave với đồng hồ phân tán.
- Giao diện ngoại vi nối tiếp (SPI) được sử dụng làm giao tiếp giữa STM32 với ESC.

3.4.1. Giao diện ngoại vi nối tiếp (SPI)

SPI là một giao thức truyền thông nối tiếp. Giao thức này có tính chất đồng bộ và được sử dụng chủ yếu trong các hệ thống nhúng cho việc truyền thông tốc độ cao ở khoảng cách ngắn. Trong dự án này, SPI được sử dụng làm giao diện truyền thông giữa bộ điều khiển EtherCAT Slave Control (ESC) và STM32.

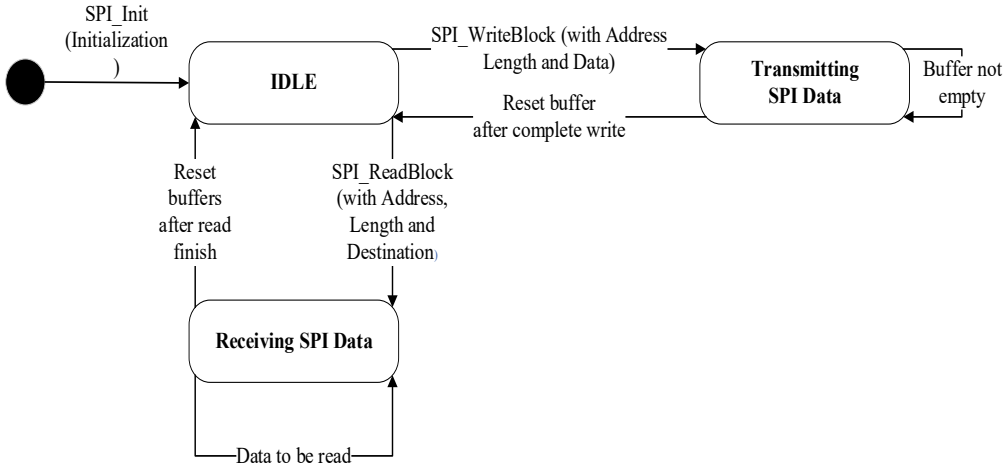
SPI là một giao thức truyền thông full duplex với bốn dây và sử dụng hệ thống master-slave. Trong dự án này, master là bo mạch STM32, và slave là ESC. Đối với cả việc đọc và ghi, bên khởi tạo quá trình truyền thông luôn là master.

Bảng 3.5: Kết nối giao tiếp SPI

LK	Xung nhịp xuất ra từ master.
MOSI	Master Out Slave In - Dữ liệu xuất ra từ master tới slave.
MISO	Master In Slave Out - Dữ liệu nhập vào master từ slave.
EN(CS)	Dây kích hoạt (hoặc Slave Select) từ master để kích hoạt và duy trì liên lạc.

Thiết kế phần mềm của giao diện SPI được thể hiện trong Hình 3.8. SPI được khởi tạo ở trạng thái IDLE. Mọi tương tác giữa STM32 và ESC đều đưa SPI vào chế độ ghi hoặc chế độ đọc và các bộ đệm tương ứng của nó được xử lý để tạo ra giao tiếp thành công.

Đối với giao tiếp SPI, STM32 là SPI master và ESC là SPI slave. Như đã chỉ ra trong thiết kế, thiết bị ngoại vi SPI được khởi tạo (cùng với các chân GPIO được sử dụng cho giao tiếp SPI) và được giữ ở trạng thái Init. Từ đây, thiết bị ngoại vi SPI được gọi, để ghi hoặc đọc dữ liệu vào/ra ESC.



Hình 3.8: Trạng thái ghi/đọc SPI.

Đồng bộ hóa đồng hồ phân tán chúng tôi sử dụng đồng hồ phân tán để đồng bộ hóa tất cả các đồng hồ trong mạng, do đó cho phép chúng tôi thực hiện các tác vụ hoặc hành động được đồng bộ hóa. Sử dụng cấu hình, chúng tôi ghi các thiết lập DC vào ESC. Một tín hiệu ”SYNC” tuần hoàn được tạo ra phù hợp với các thiết lập. Tín hiệu này ngắt DSP và tác vụ ứng dụng cần thiết được thực hiện. Trong dự án của chúng tôi, chúng tôi sử dụng tín hiệu này để đọc/ghi dữ liệu từ/vào ESC.

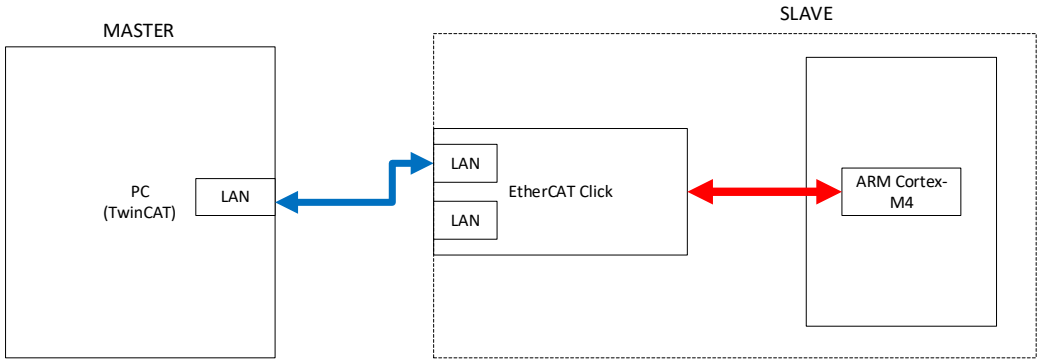
3.4.2. Phầm mềm EtherCAT SDK

Chúng ta thấy rằng slave bao gồm hai thành phần phần cứng - ESC và STM32. Chúng ta sẽ xem sơ qua phần mềm nào được sử dụng trong ESC.

Phần mềm ESC mà chúng tôi sử dụng trong thiết lập của chúng tôi là EtherCAT SDK từ Công ty RT_Lab , Thụy Điển. Nó là công cụ trợ giúp chúng ta tạo ra các file mô tả cho slave nhằm lập trình cho STM32 cũng như sử dụng trong phầm mềm TwinCAT.

Hình 3.9: Công ty RT-LABS

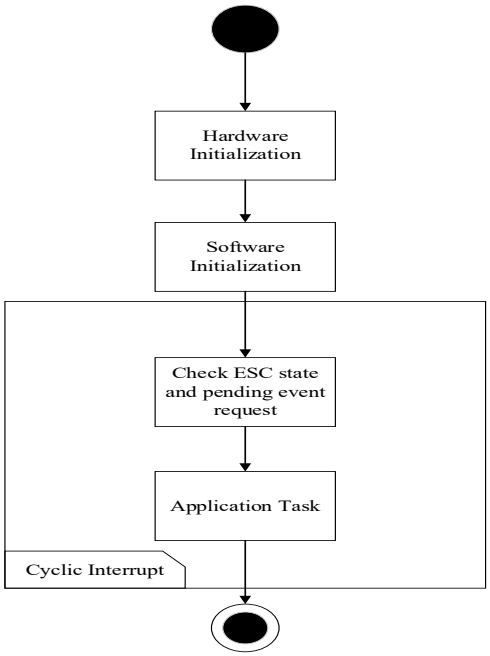
Beckhoff cung cấp các hướng dẫn chi tiết mô tả cách chúng tôi có thể cấu hình ESC để phản hồi theo một cách nhất định và cũng mô tả cách chúng tôi có thể tương tác với ESC. Chúng tôi sử dụng EtherCAT để tương tác với ESC như minh họa trong Hình 3.10.



Hình 3.10: Thiết bị Master và STM32 truy cập ESC thông qua EtherCAT và SPI.

Phần mềm Slave EtherCAT được viết bằng C. Có nhiều nhà cung cấp cung cấp khuôn khổ cho phần mềm Slave. Chúng tôi đã chọn phần mềm EtherCAT Simple Open làm khuôn khổ vì nó là mã nguồn mở và có thể được sử dụng lại mà không gặp rắc rối về cấp phép.

Ý tưởng cốt lõi là khi khởi động phần mềm này trước tiên khởi tạo các thiết bị ngoại vi cần thiết để tương tác với ESC và sau đó khởi tạo ESC thành trạng thái "INIT". Sau đó, theo chu kỳ, trạng thái của ESC (máy trạng thái EtherCAT) được giám sát và nếu có bất kỳ sự kiện nào đang chờ xử lý, hành động tương ứng sẽ được thực hiện. Sơ đồ luồng được hiển thị trong Hình 3.11.



Hình 3.11: Quá trình khởi tạo và vận hành của hệ thống

Các chức năng khác nhau được phân loại thành các tệp khác nhau được thêm vào trong STM32CubeIDE:

Bảng 3.6: Tệp mã nguồn liên quan đến việc triển khai EtherCAT

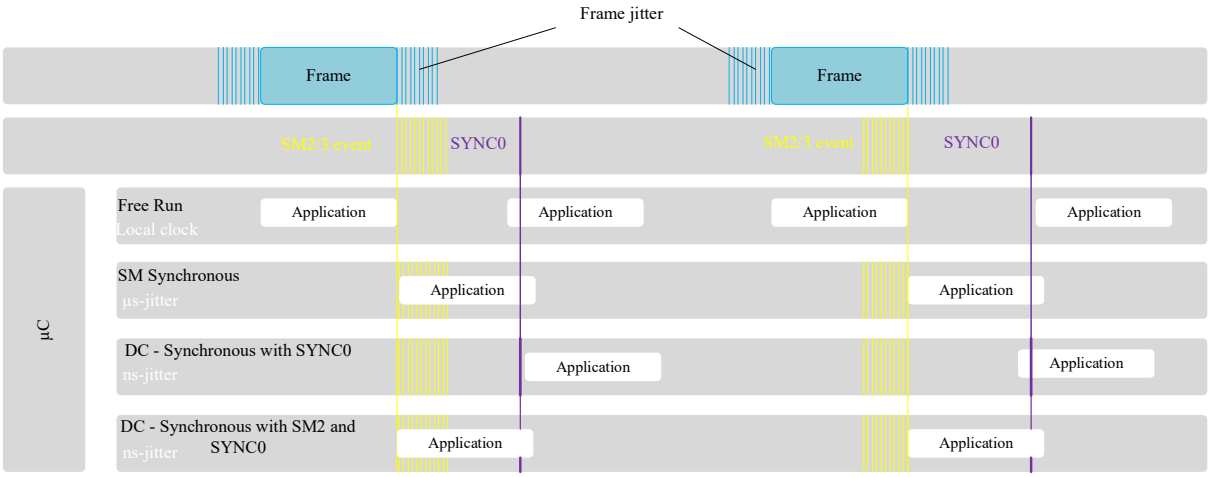
Tên tệp	Chức năng
Objectlist.c	Một danh sách các cấu trúc bao gồm mô tả ESC
MAIN.c	Tệp chính của dự án, cũng bao gồm các hàm phục vụ ngắt bộ đếm thời gian
esc.c	Các hàm xử lý liên kết dữ liệu và giao tiếp hộp thư.
esc.h	Tệp tiêu đề cho các hàm trong esc.c, bao gồm các hàm đóng gói và mở gói dữ liệu
esc.coe.c	Module CoE
esc.coe.h	Tệp tiêu đề cho các hàm trong esc.coe.c.
esc.foe.c	Module FoE
esc.foe.h	Tệp tiêu đề cho các hàm trong esc.foe.c.
ecat_options.h	Tệp tiêu đề chứa các định nghĩa macro về kích thước SM và hộp thư.
utypes.h	Chứa các cấu trúc đầu vào/đầu ra giữ một bản sao trong STM32, của dữ liệu đầu vào/đầu ra tại ESC.
middleSOES.c	Các hàm hỗ trợ mở gói thủ công dữ liệu được truyền từ ESC đến STM32

Tín hiệu đồng bộ đồng hồ có thể tận dụng đồng hồ chính xác của ESC để tạo ra các tín hiệu tuần hoàn cho STM32. Những tín hiệu này được phát sinh từ hệ thống DC trong ESC. Trước khi tạo tín hiệu, đồng hồ cần được đồng bộ hóa với tham chiếu để đảm bảo độ chính xác. Như đã đề cập trong chương thiết kế, quá trình đồng bộ hóa DC diễn ra trong trạng thái SAFE-OP. Sau khi chuyển sang trạng thái OP, các tín hiệu đồng bộ sẽ được tạo ra tuần hoàn.

SM-Synchronous: Ứng dụng được đồng bộ hóa với sự kiện SM2 (SM3), được tạo ra khi dữ liệu quy trình được ghi vào SM2 (đọc từ SM3). Sự kiện được ánh xạ tới IRQ toàn cục hoặc được thăm dò từ reg0x0220. Phần tử ESI Device: DC không khả dụng nếu chỉ

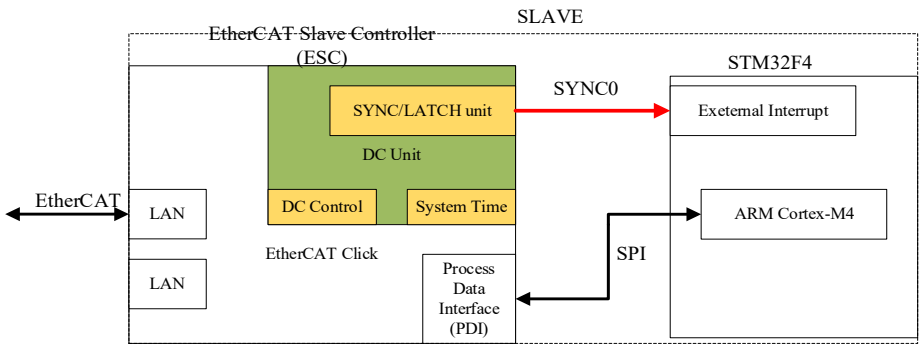
hỗ trợ SM-Synchronous. Nếu cả SM-Synchronous và DC-Synchronous đều được hỗ trợ, thì nó được chỉ định trong ESI bằng Dc:AssignActivate = "#x0000"

DC-Synchronous: Ứng dụng được đồng bộ hóa bằng tín hiệu ngắt dựa trên DC (SYNC0, SYNC1; ns-accuracy), được tạo ra bởi đơn vị SYNC/LATCH. Trong số nhiều chế độ DC khác, hai chế độ sau đây hiển thị các khái niệm cơ bản:SYNC0: kích hoạt quá trình xử lý hoàn chỉnh chu kỳ cục bộ (xem Thời gian chu kỳ tối thiểu) SM2 và SYNC0 (độ chính xác đồng bộ hóa thậm chí cao hơn của sự kiện đầu ra): sự kiện SM2 kích hoạt việc đọc dữ liệu đầu ra từ SM2, xử lý, ghi giá trị vào trình điều khiển phần cứng; sau đó sự kiện SYNC0 được sử dụng để kích hoạt trình điều khiển đầu ra. Phần tử ESI Dc:AssignActivate = "#x0300" để tạo sự kiện SYNC0



Hình 3.11: Chế độ đồng bộ hóa thời gian trong EtherCAT

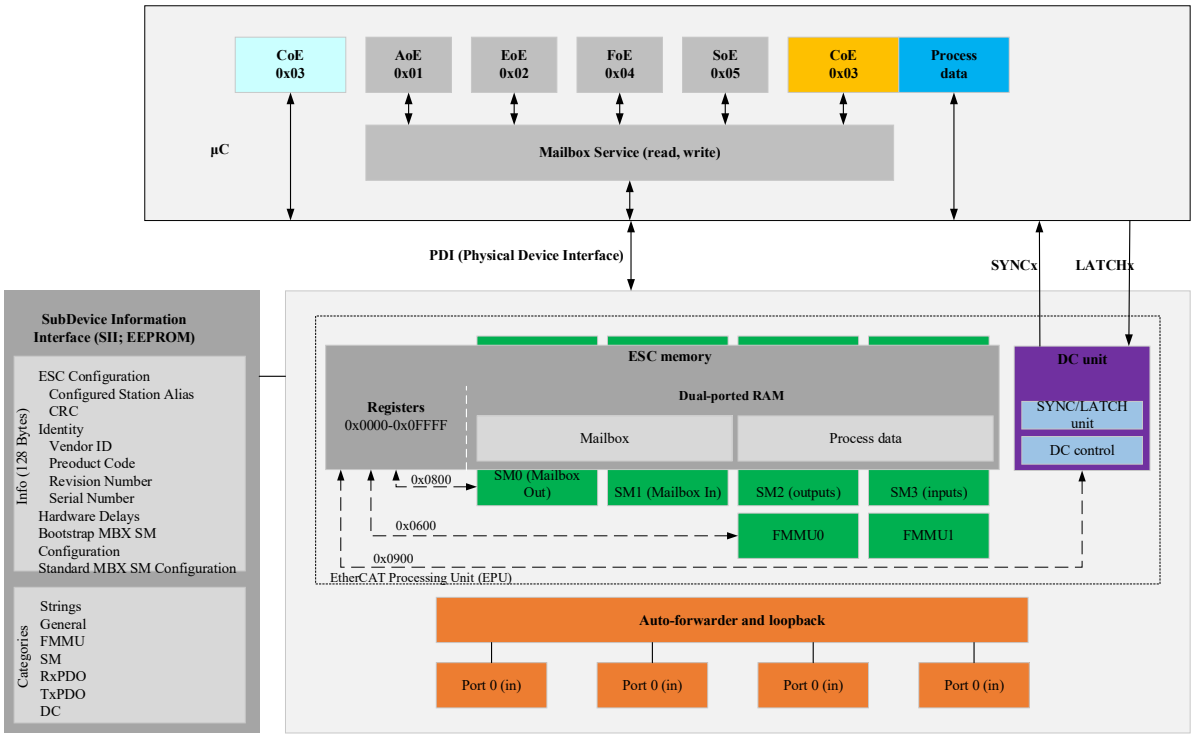
Có hai tín hiệu chính có thể sử dụng từ ESC là SYNC0 và SYNC1. Tùy thuộc vào hướng truyền tín hiệu, chúng được gọi là SYNC0/SYNC1 (ESC đến STM32) hoặc LATCH0/LATCH1 (STM32 đến ESC).Trong dự án này, chúng tôi sử dụng một tín hiệu SYNC0 như thể hiện trong Hình 3.13. Ngắt này là được sử dụng để kích hoạt tác vụ ứng dụng.



Hình 3.12: Tín hiệu SYNC tuần hoàn từ ESC đến STM32. Tín hiệu SYNC được tạo ra bằng cách sử dụng đồng hồ DC chính xác của ESC.

3.4.3. Tệp mô tả ESI

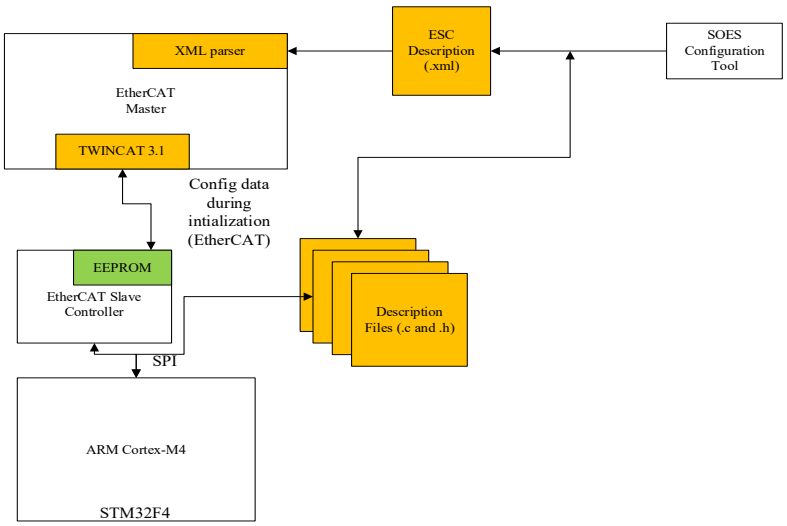
ESC xử lý các khung EtherCAT khi đang chạy trong phần cứng và triển khai các chức năng của lớp liên kết dữ liệu (DLL). Các chức năng này bao gồm SyncManagers (SM),Fieldbus Memory Management Units (FMMU) và DC unit. Thông thường, EtherCAT SubDevices triển khai các chức năng của lớp ứng dụng (AL) như ESM, xử lý tham số và xử lý dữ liệu quy trình trên µC – các SubDevice như vậy được gọi là "thiết bị phức hợp". Chỉ dành cho các thiết bị I/O rất đơn giản không có tham số và không cần xử lý lỗi AL, trình điều khiển phần cứng I/O mới được kết nối trực tiếp với giao diện I/O kỹ thuật số của ESC – các SubDevice như vậy được gọi là "thiết bị đơn giản". Sơ đồ khối kết hợp các thành phần của cấu trúc phần cứng của SubDevice, các thực thể chức năng của ESC, cấu trúc phần mềm và các thành phần giao thức.



Hình 3.13: Kiến trúc của EtherCAT Slave Controller (ESC)

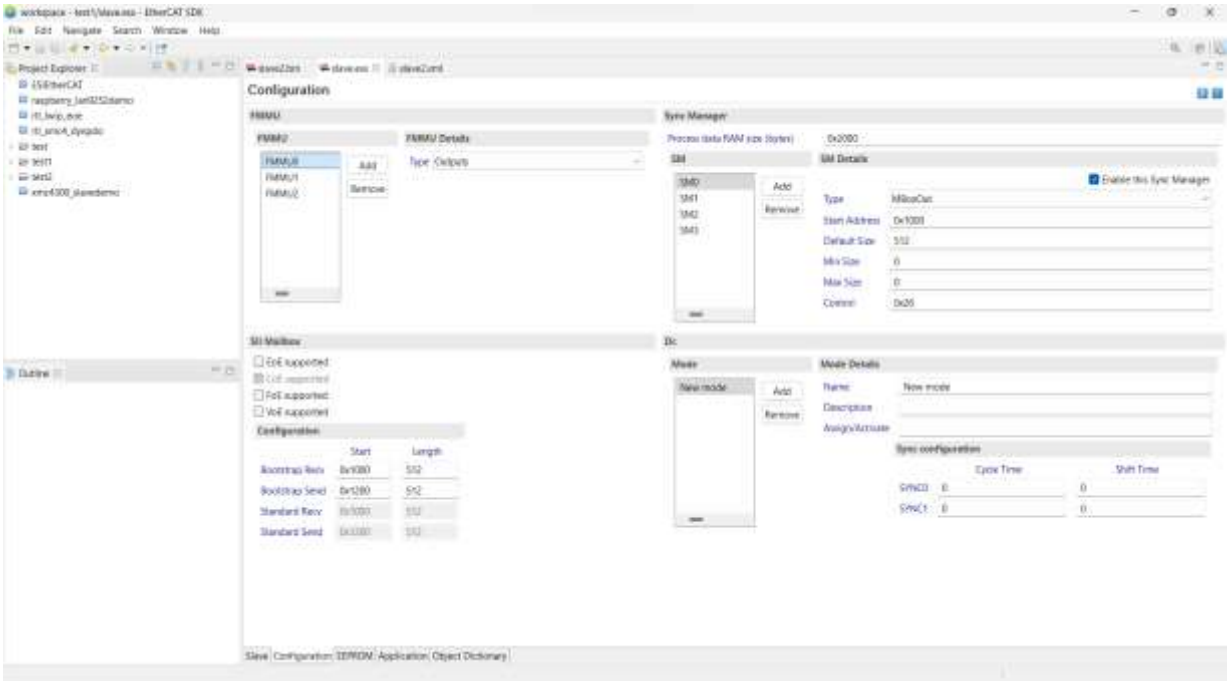
Như đã nêu, phần mềm để viết tệp mô tả ESI được phát triển bởi công ty Beckhoff và mã nguồn không thể truy cập. Cũng đã đề cập rằng master giao tiếp với ESC thông qua EtherCAT, trong khi STM32 giao tiếp với ESC qua SPI. Điều này cho thấy ESC đóng vai trò là điểm giao tiếp giữa master và STM32. Để đảm bảo sự tương tác an toàn và thống nhất giữa cả ba thành phần phần cứng, chúng ta cần một mô tả thống nhất về ESC ở master, STM32 và trong chính ESC. Cụ thể, mô tả ESC được chứa trong master, một bản sao của mô tả này sẽ được truyền đến ESC qua EtherCAT (mặc dù có những phương

thức khác nhưng không được bàn đến ở đây), và một mô tả tương ứng sẽ được viết trong STM32, phải khớp với mô tả trong master và ESC. Điều này được thể hiện trong Hình 3.14. Nếu không có một mô tả thống nhất giữa các thiết bị, ESC sẽ không thể hoạt động chính xác.

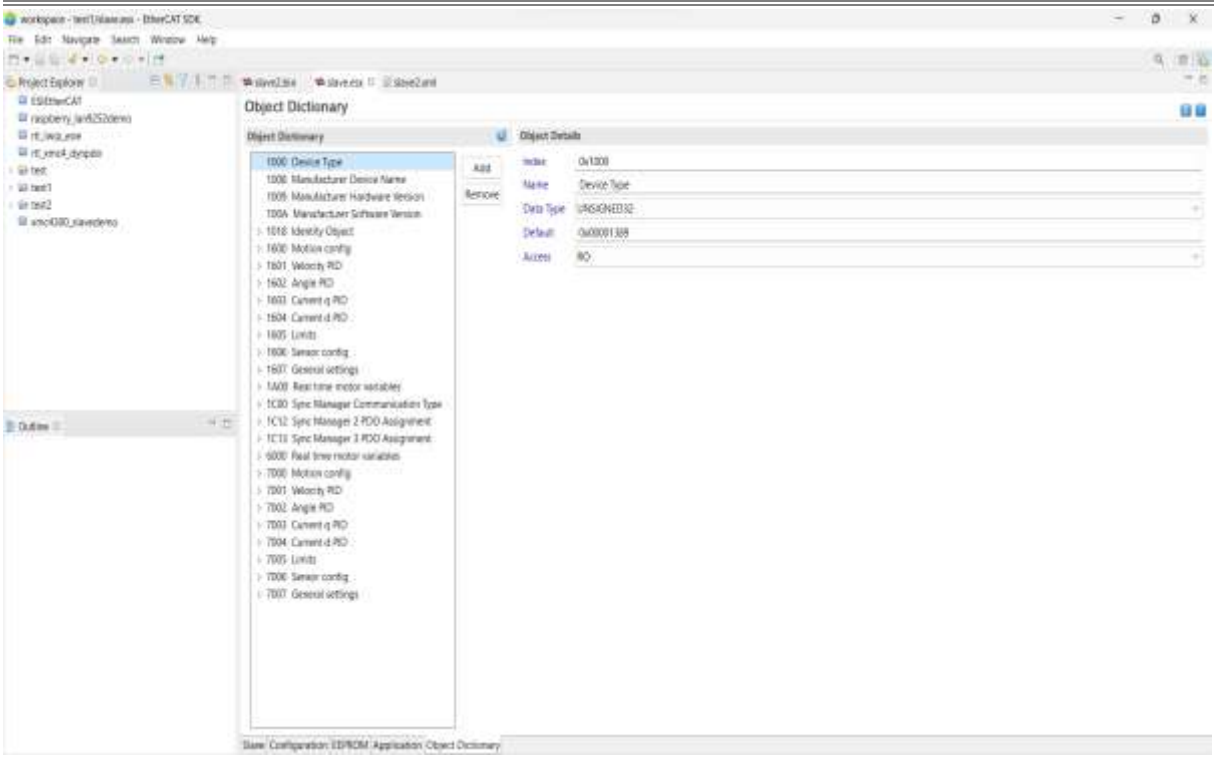


Hình 3.14: Các tệp mô tả ESC (.xml và .c) được lập trình cho thiết bị chính, ESC và thiết bị phụ để có khả năng giao tiếp EtherCAT thống nhất.

Để tránh rắc rối khi phải viết tệp cấu hình cho master và sao chép lại mã C trong STM32, chúng tôi sử dụng công cụ cấu hình SOES, như được minh họa trong Hình 3.15. Trình bày các tùy chọn cấu hình và từ điển đối tượng, một phương pháp mô tả theo chuẩn CoE. Tệp mô tả dành cho master được định dạng bằng XML.



Hình 3.15: Công cụ cấu hình SOES được sử dụng để tạo tệp mô tả ESC.



Hình 3.16: Công cụ cấu hình SOES hiển thị các thiết lập có thể cấu hình và cách mô tả từ điển đối tượng của CANopen.

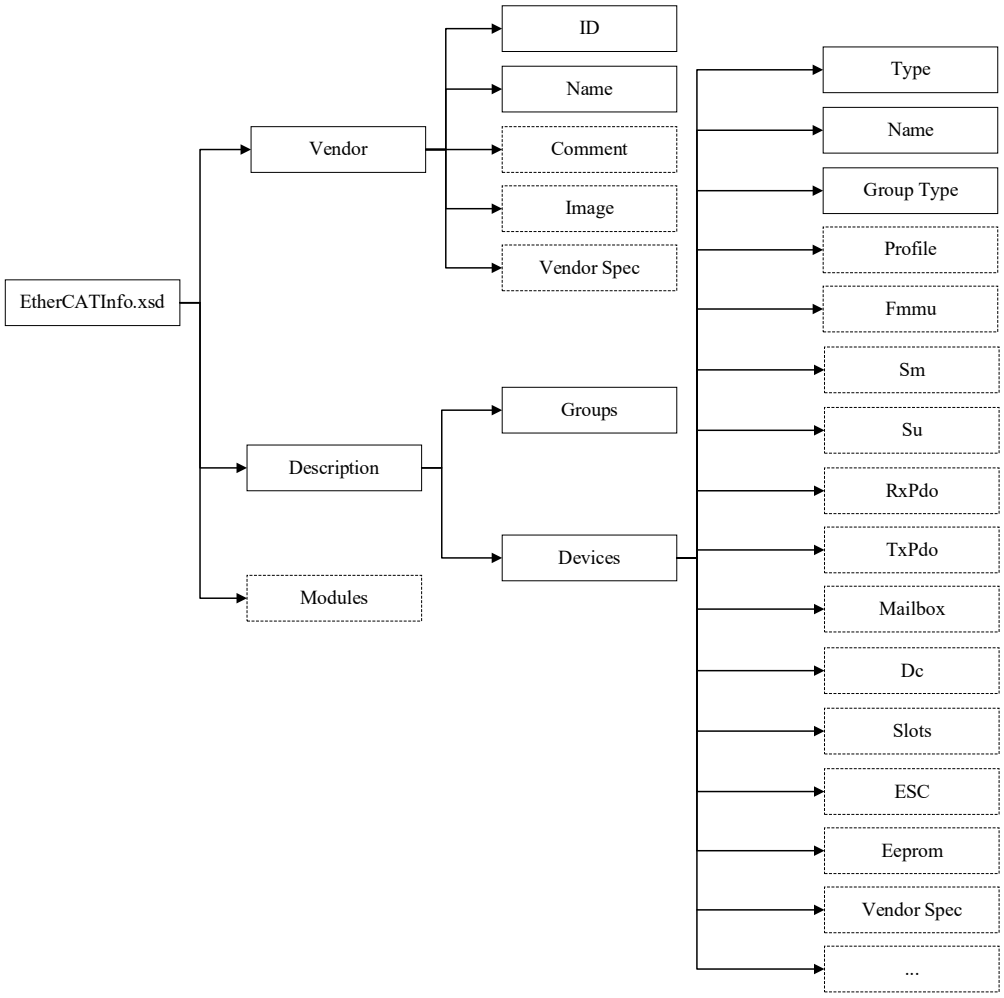
Sau khi cấu hình trong phần mềm SDK EtherCat thì chúng tôi sẽ thu được file XML mô tả Slave được sử dụng để eeprom vào master và các file .c và .h là danh sách các cấu trúc chứa thông tin giống như XML.

3.5. Triển khai Master

Trong dự án này, chúng tôi đã sử dụng một máy tính thông dụng chạy hệ điều hành Windows làm bộ điều khiển chính. TwinCAT 3.1 là một phần mềm đặc thù của Beckhoff, biến bất kỳ máy tính chạy Windows nào thành hệ thống điều khiển thời gian thực (Real-Time Control PC). Các ràng buộc thời gian thực cứng (Hard Real-Time) hoặc mềm (Soft Real-Time) phụ thuộc vào khả năng tương thích của thẻ mạng (NIC - Network Interface Card).

3.5.1. Cấu hình

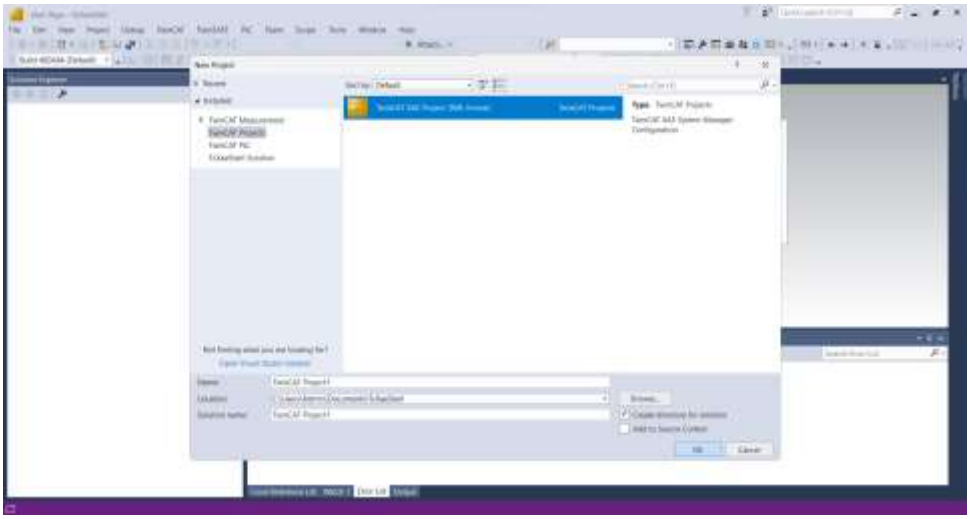
Làm thế nào để một master nhận ra một slave? Đối với điều này, chúng ta cần cung cấp EtherCAT Slave và thông tin tùy chọn trong tệp cấu hình. Điều này được cung cấp cho TwinCAT trong một sơ đồ XML như đã đề cập phía trên để TwinCAT phân tích cú pháp. Thông tin đã phân tích cú pháp sau đó được lưu trữ trong chủ và được sử dụng để xác định slave trong mạng trong quá trình khởi tạo. Thông tin (ESI) cho máy chủ (trong trường hợp này là TwinCAT)



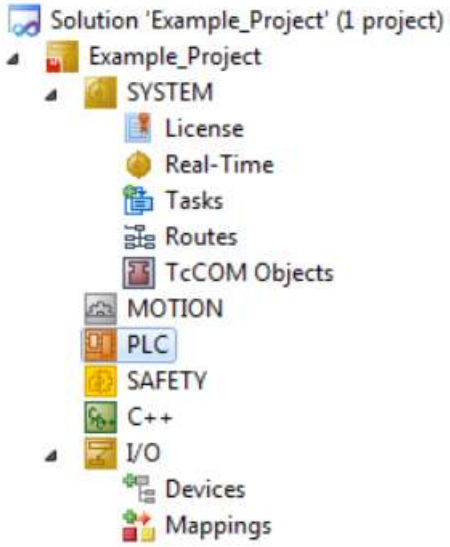
Hình 3.17: Thông tin EtherCAT Slave chứa thông tin bắt buộc và tùy chọn được chỉ định về slave. Thông tin này được đưa vào TwinCAT để phân tích cú pháp.

Trước tiên, hãy tạo một dự án mới thông qua 

(hoặc trong "File"→"New"→ "Project..."). Trong hộp thoại sau, thực hiện các mục tương ứng theo yêu cầu



Sau đó, dự án mới có sẵn trong trình khám phá thư mục dự án:



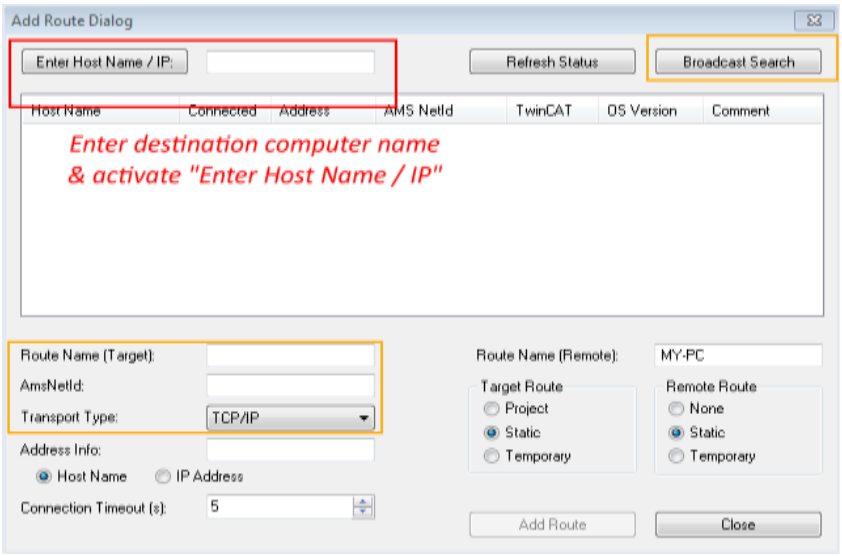
Nói chung, TwinCAT có thể được sử dụng ở chế độ cục bộ hoặc từ xa. Khi hệ thống TwinCAT bao gồm giao diện người dùng (tiêu chuẩn) được cài đặt trên PLC tương ứng, TwinCAT có thể được sử dụng ở chế độ cục bộ và do đó bước tiếp theo là "Chèn thiết bị".

Nếu mục đích là giải quyết môi trường thời gian chạy TwinCAT được cài đặt trên PLC làm môi trường phát triển từ xa từ một hệ thống khác, hệ thống mục tiêu phải được biết trước. Thông qua biểu tượng trên thanh menu:

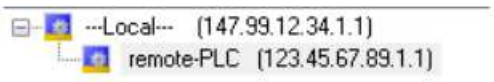
mở cửa sổ sau:

Sử dụng "Tìm kiếm (Ethernet)..." để vào hệ thống mục tiêu. Do đó, một hộp thoại tiếp theo sẽ mở ra:

- nhập tên máy tính đã biết sau "Nhập tên máy chủ / IP:" (như màu đỏ)
- thực hiện "Tìm kiếm phát sóng" (nếu không biết tên máy tính chính xác)
- nhập IP máy tính đã biết hoặc AmsNetID.



Khi hệ thống mục tiêu đã được nhập, nó có sẵn để lựa chọn như sau (có thể phải nhập mật khẩu):



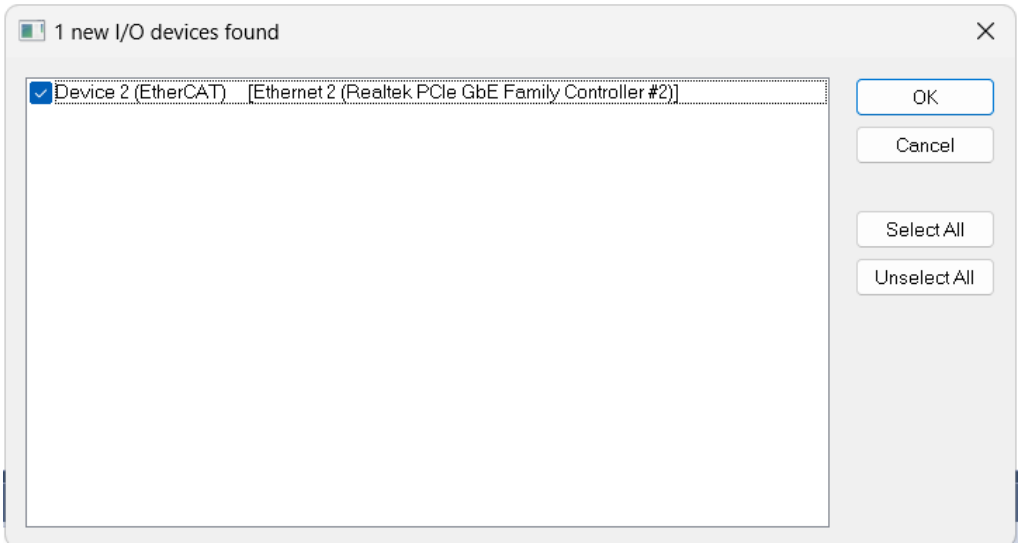
Sau khi xác nhận với "OK", hệ thống mục tiêu có thể được truy cập thông qua Visual Studio shell.

Thêm thiết bị

Trong trình khám phá thư mục dự án của giao diện người dùng Visual Studio shell ở bên trái, chọn "Thiết bị" trong phần tử "I / O", sau đó nhấp chuột phải để mở menu ngữ cảnh và chọn "Quét" hoặc bắt đầu hành động thông qua  thanh menu. Trước tiên, Trình quản lý hệ thống TwinCAT có thể phải được đặt thành "Chế độ cấu hình" thông qua  hoặc thông qua menu "TwinCAT" → "Khởi động lại TwinCAT (Chế độ cấu hình)".

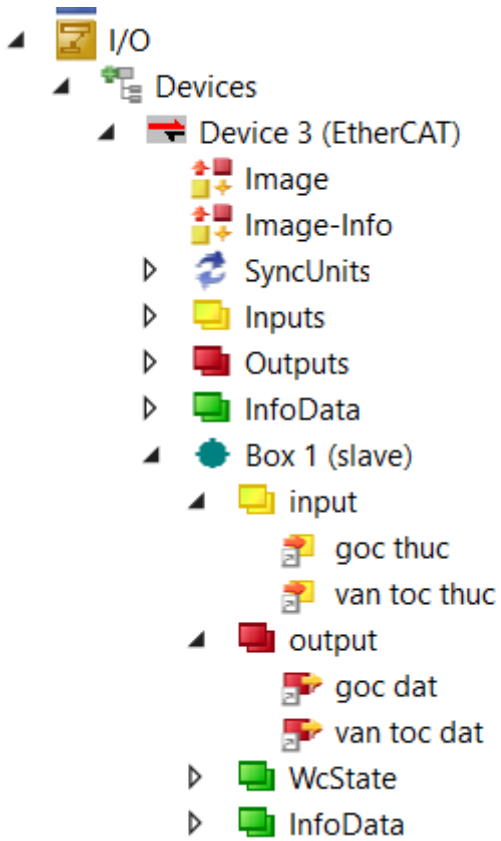
Chọn "Quét"

Xác nhận thông báo cảnh báo sau đó và chọn "EtherCAT" trong hộp thoại:

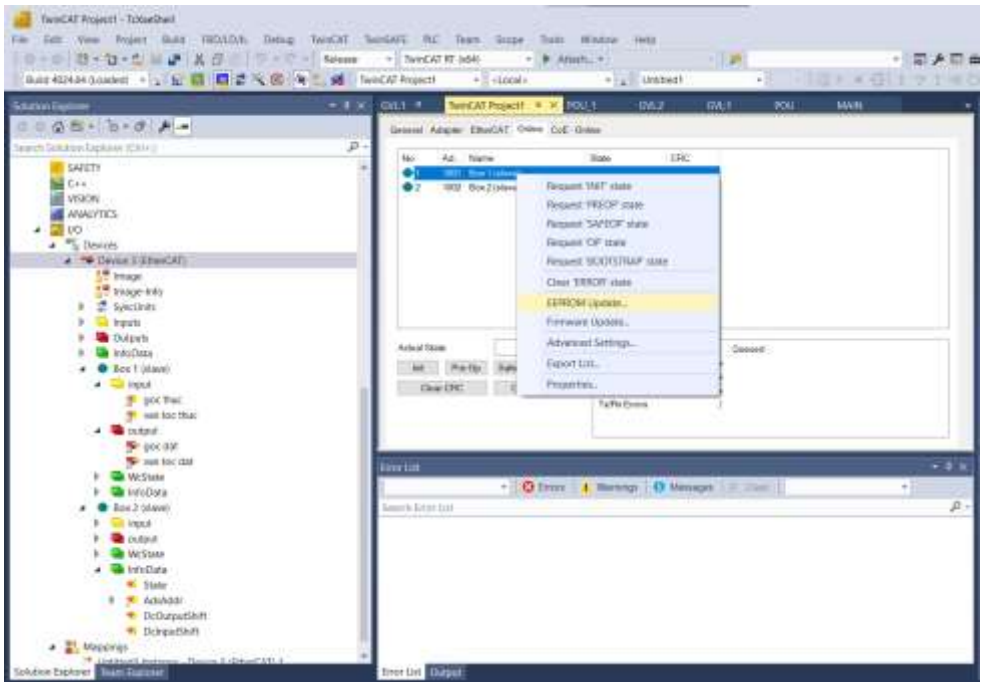


Xác nhận thông báo "Tìm hộp mới", để xác định các thiết bị đầu cuối được kết nối với thiết bị. "Free Run" cho phép thao tác các giá trị đầu vào và đầu ra trong "Chế độ cấu hình" và cũng cần được xác nhận.

Dựa trên cấu hình mẫu được mô tả ở đầu phần này, kết quả như sau:



Với thiết bị Slave được sử dụng trong hệ thống ta cần thực hiện thêm bước là eeprom để TwinCat có thể nhận biết đầu vào đầu ra của Slave.



Lập trình chương trình PLC

TwinCAT PLC Control là môi trường phát triển để tạo bộ điều khiển trong các môi trường lập trình khác nhau. TwinCAT PLC Control hỗ trợ tất cả các ngôn ngữ được mô tả trong tiêu chuẩn IEC 61131-3. Có hai ngôn ngữ dạng văn bản và ba ngôn ngữ dạng đồ họa.

Các ngôn ngữ dạng văn bản

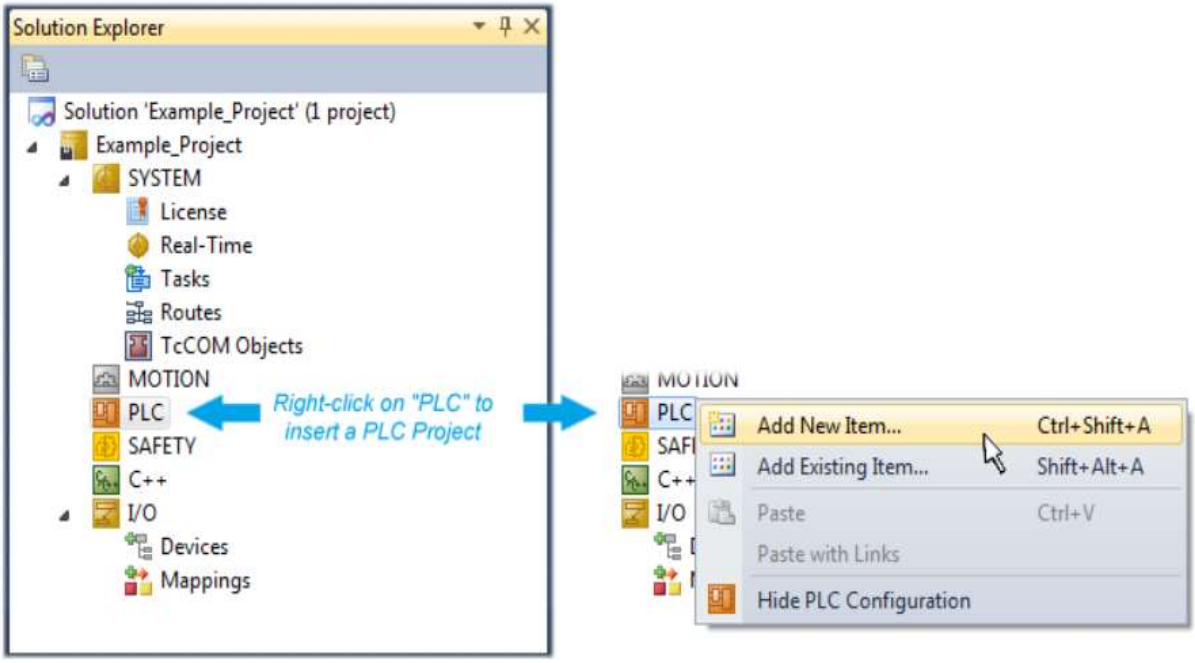
- Instruction List (IL)
- Structured Text (ST)

Các ngôn ngữ dạng đồ họa

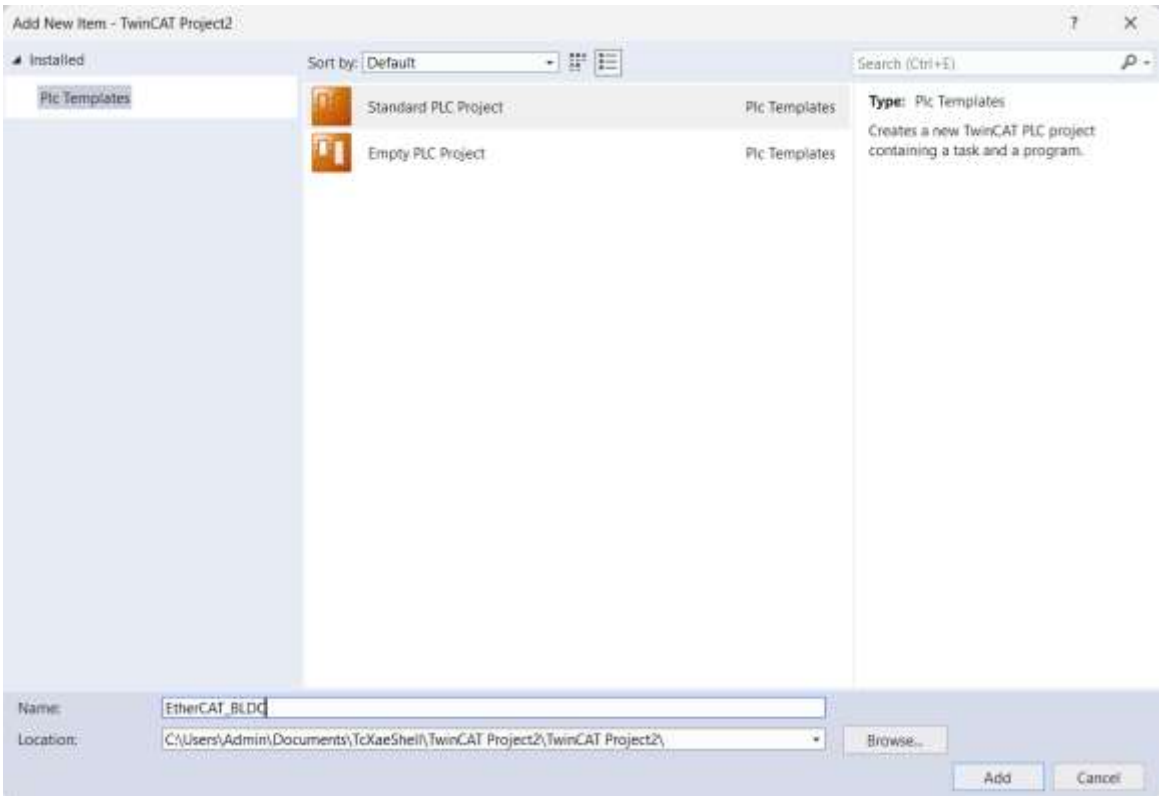
- Function Block Diagram (FBD)
- Ladder Diagram (LD)
- Continuous Function Chart Editor (CFC)
- Sequential Function Chart (SFC)

Phần sau đây sẽ đề cập đến Structured Text (ST).

Để tạo một môi trường lập trình, một dự án con PLC được thêm vào dự án mẫu thông qua menu ngữ cảnh của mục "PLC" trong trình khám phá thư mục dự án bằng cách chọn “Add New Item….”.

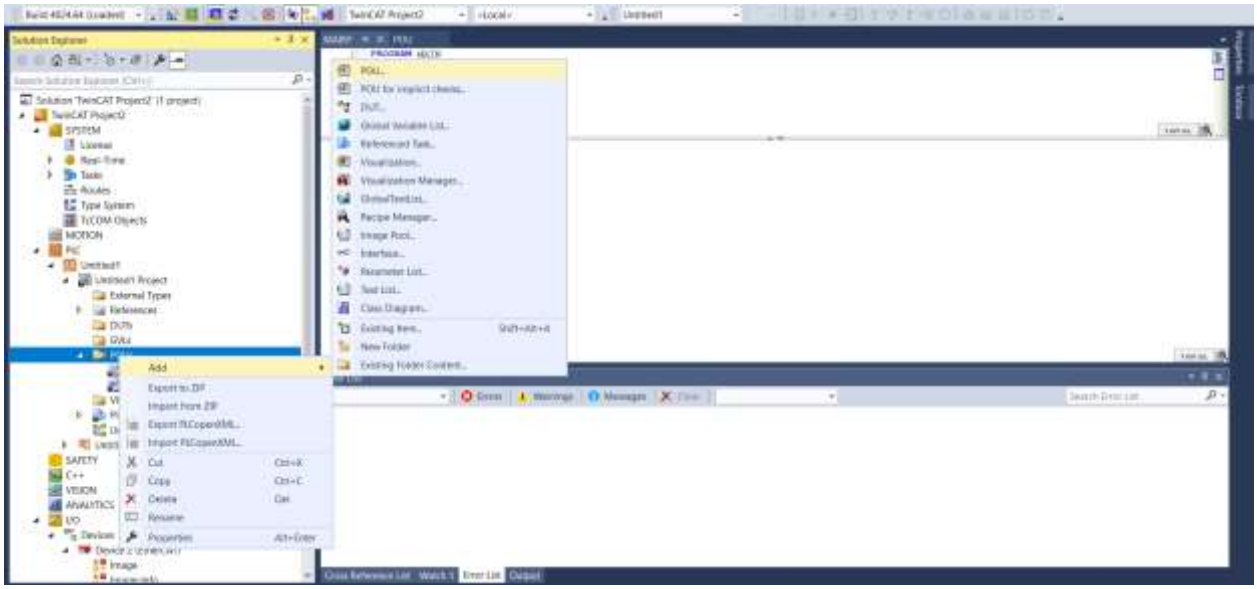


Trong hộp thoại mở ra, hãy chọn "Standard PLC project" và nhập "EtherCAT_BLDC" làm tên dự án.

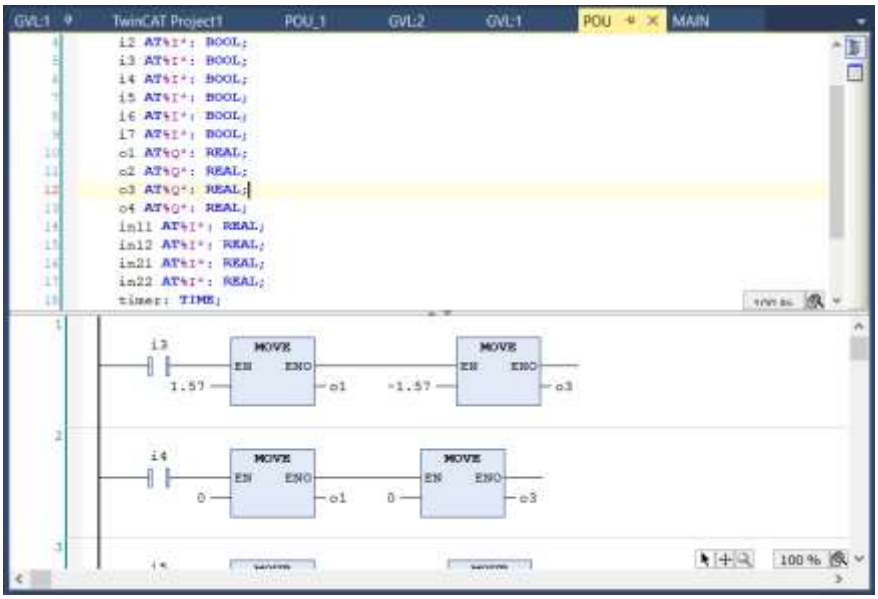


Chương trình "MAIN", đã tồn tại bằng cách chọn "Standard PLC project", có thể được mở bằng cách nhấp đúp vào "EtherCAT_BLDC_project" trong "POU". Giao diện người dùng sau đây được hiển thị cho một dự án ban đầu:

Để viết tiếp chương trình bằng ngôn ngữ LD ta chọn vào POUs và lựa chọn "Add" và tiếp theo chọn vào POU.



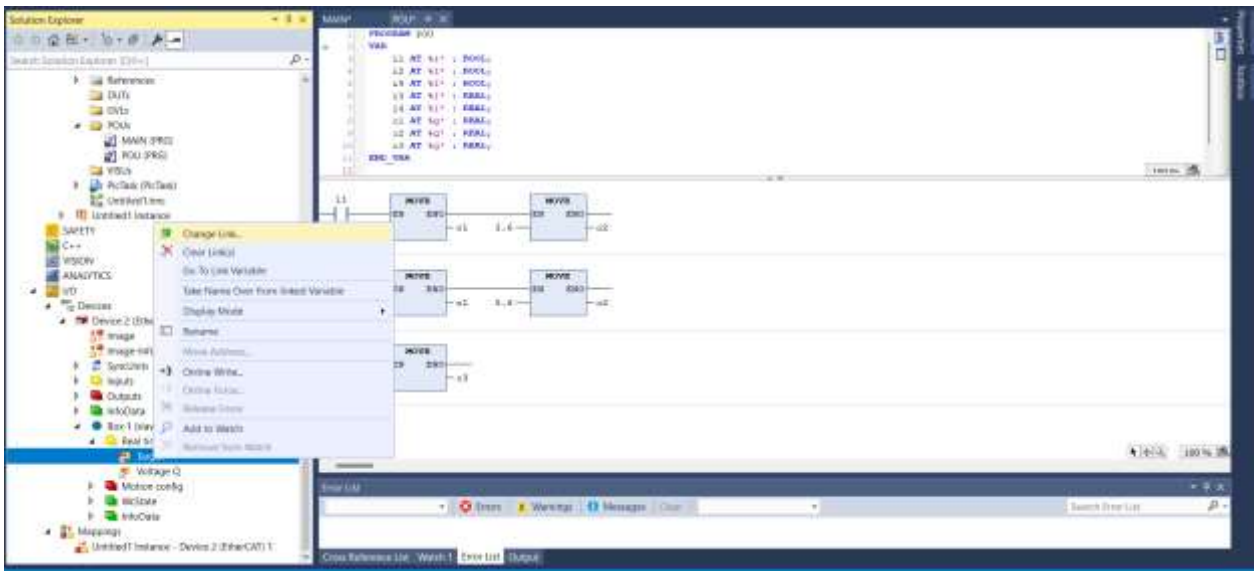
Trong POU chúng ta thực hiện khai báo các biến đầu vào và đầu ra theo cú pháp " Tên biến AT %I* : "Kiểu dữ liệu". và viết code LD ở bên dưới.

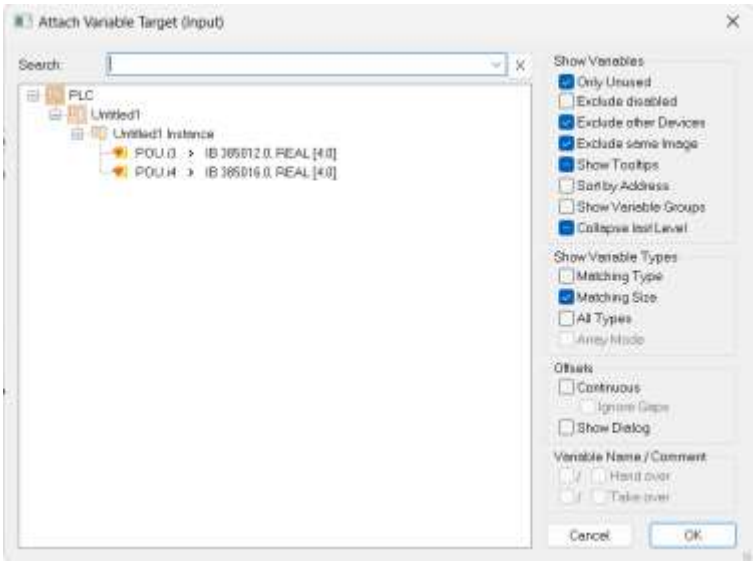


Chương trình điều khiển bây giờ được tạo dưới dạng thư mục dự án, tiếp theo là quá trình biên dịch:

Gán biến

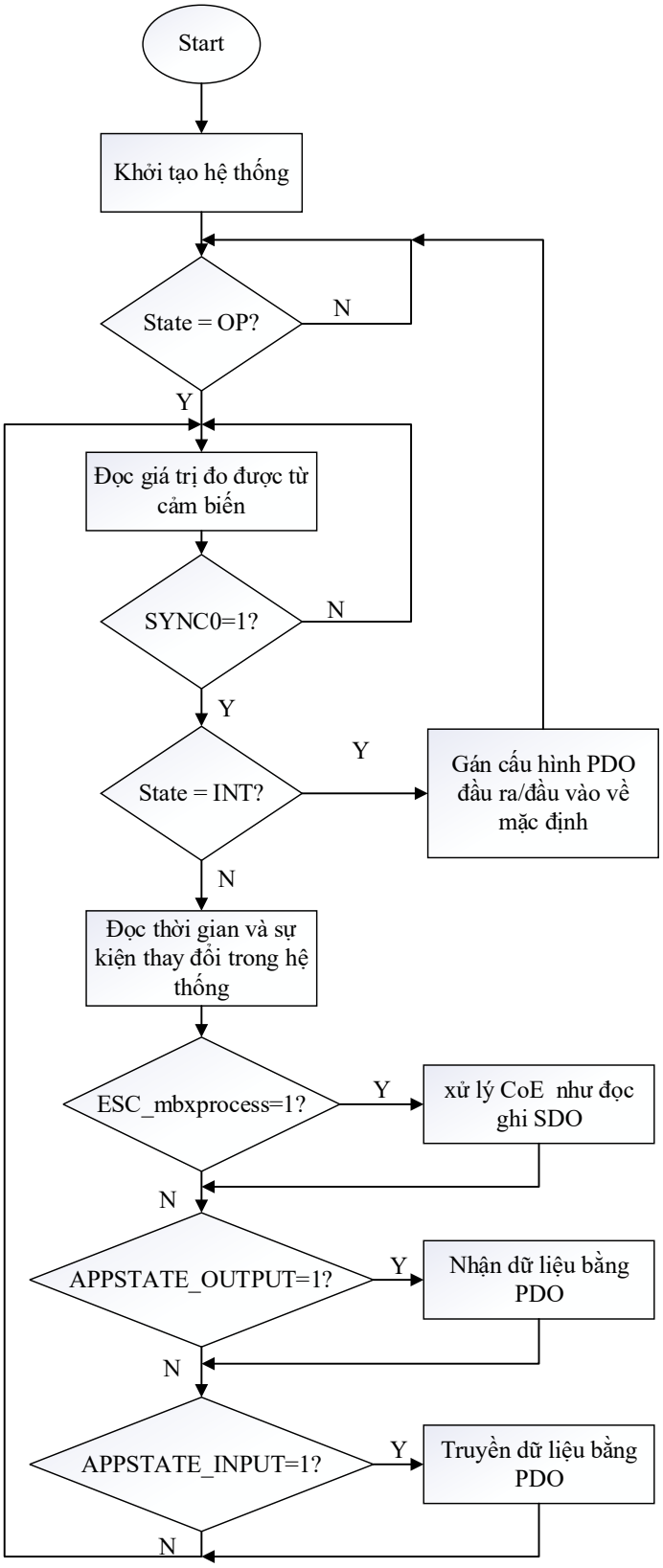
Thông qua menu của một phiên bản - các biến trong ngữ cảnh "PLC", sử dụng "Modify Link…" tùy chọn để mở cửa sổ để chọn đối tượng quy trình (PDO) phù hợp để liên kết và thực hiện link các biến phù hợp với yêu cầu của chương trinhg PLC.



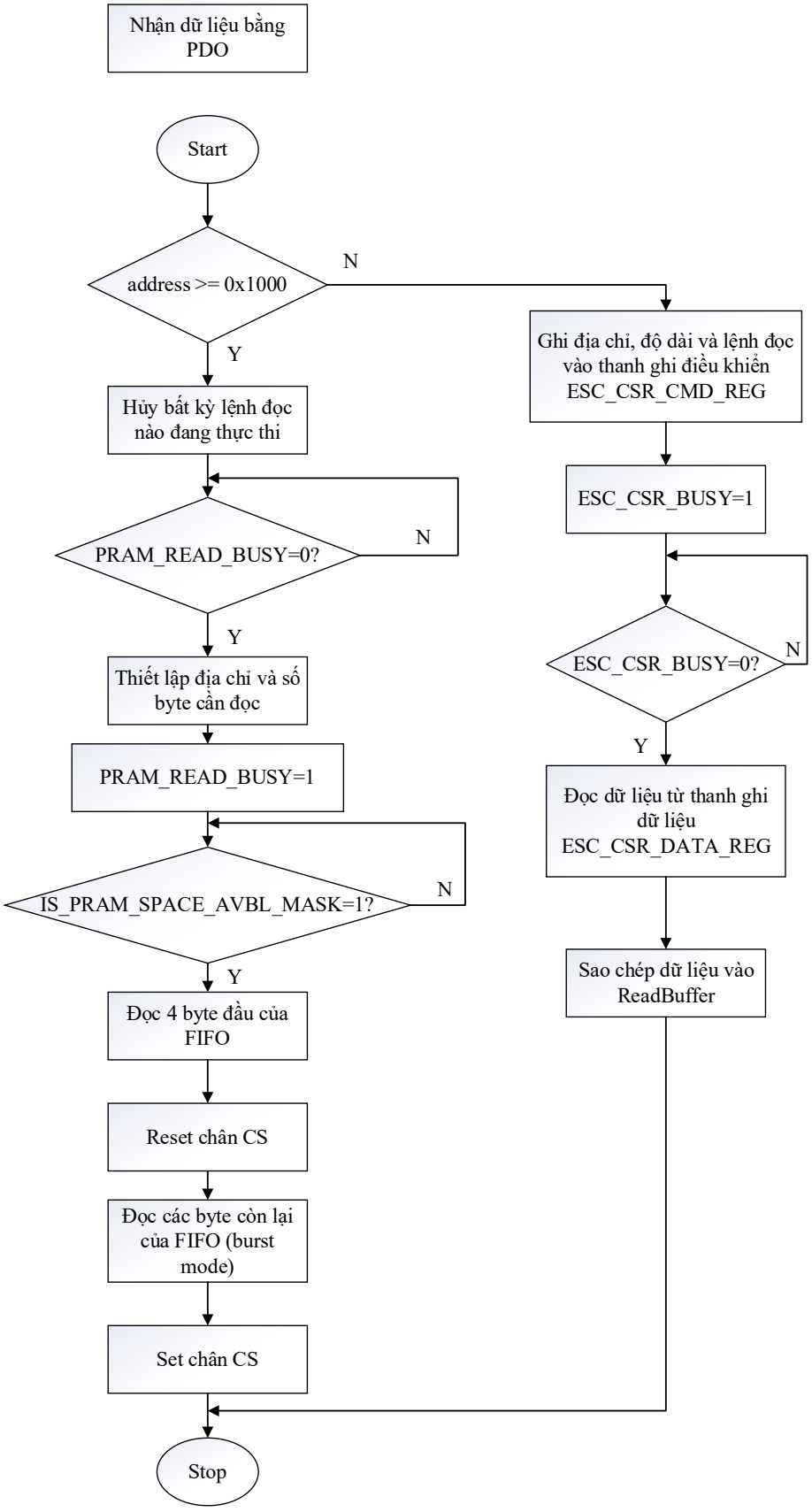


3.6. Triển khai code trên STM32CubeIDE

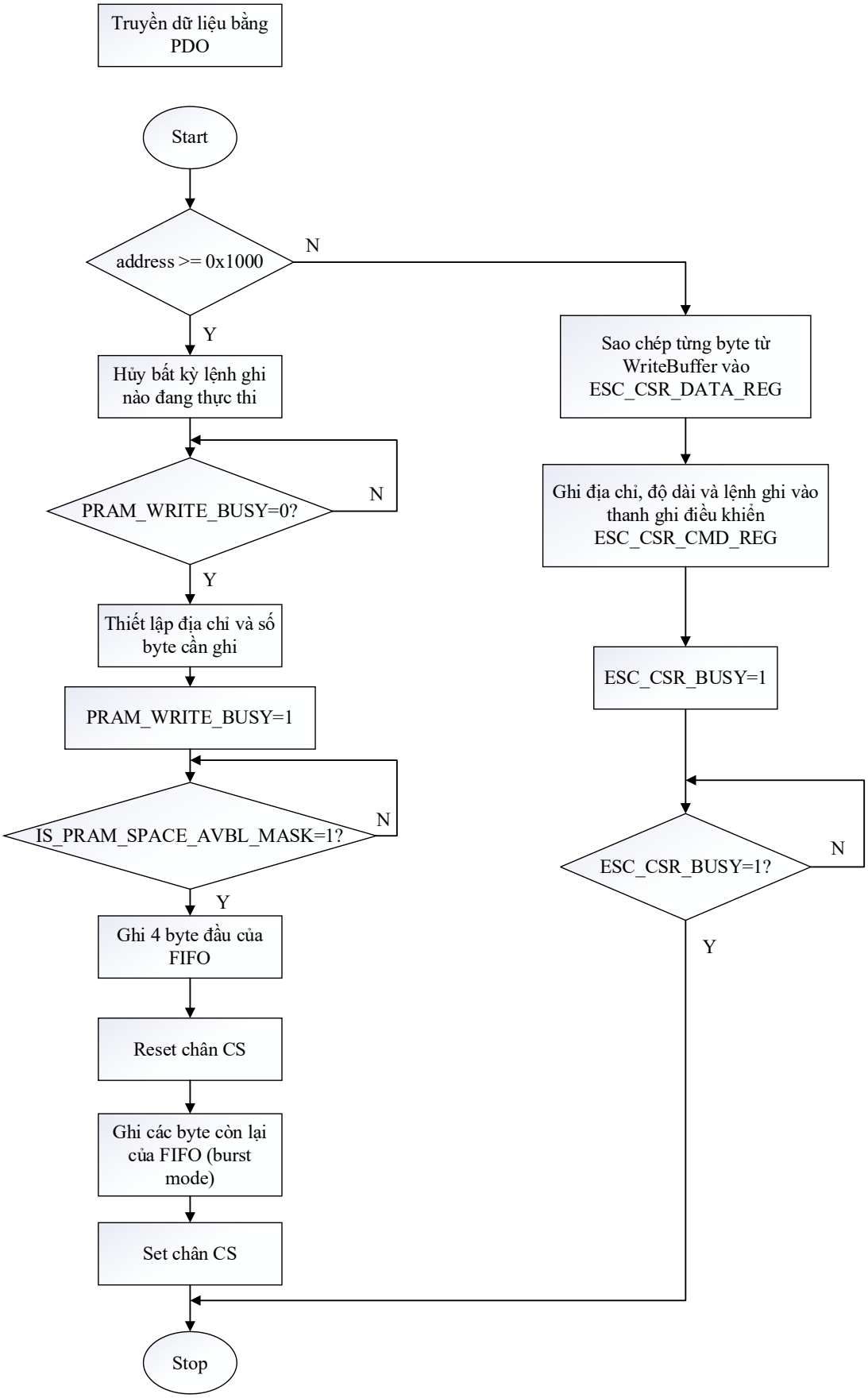
3.6.1. Lưu đồ thuật toán



Hình 3.18: Lưu đồ thuật toán truyền nhận dữ liệu của hệ thống



Hình 3.19: Lưu đồ thuật toán nhận dữ liệu PDO



Hình 3.20: Lưu đồ truyền dữ liệu PDO

3.6.2. Cấu hình Timer 7 (TIM7)

Trong dự án, Timer 7 (TIM7) của vi điều khiển STM32F446 được sử dụng như một bộ định thời chu kỳ (periodic timer) nhằm đảm bảo việc cập nhật dữ liệu đo được từ cảm biến/mạch đo về trạng thái động cơ một cách định kỳ và chính xác thời gian. Dữ liệu này sau đó được đưa vào Object Dictionary dưới dạng PDO (Process Data Object) để truyền lên phần mềm điều khiển trung tâm TwinCAT thông qua giao thức EtherCAT.

Hình 3.21: Cấu hình Timer 7

Trong hệ thống điều khiển sử dụng vi điều khiển STM32, Timer 7 (TIM7) được cấu hình để tạo ngắt định kỳ nhằm phục vụ cho các tác vụ kiểm tra và cập nhật dữ liệu đo lường. TIM7 thuộc bus APB1, và trong cấu hình hệ thống hiện tại, tần số clock của APB1 là 45 MHz. Do bộ chia APB1 được đặt lớn hơn 1 (cụ thể là 4), tần số cấp cho các timer thuộc APB1 (bao gồm TIM7) được nhân đôi, tức là 90 MHz.

Thông số cấu hình cụ thể của Timer 7 như sau:

- Prescaler: 1800 – 1 → Tín hiệu clock nội bộ sau chia là: 90 MHz / 1800 = 50 kHz
- Auto-reload register (ARR): 100 – 1 → Timer sẽ đếm từ 0 đến 99, tức là 100 chu kỳ
- Thời gian tràn (ngắt): 100 / 50,000 = 2 ms

Chức năng chính:

Timer 7 tạo ra một ngắt định kỳ, và trong hàm xử lý ngắt, chương trình sẽ gọi hàm xử lý dữ liệu trung gian để cập nhật dữ liệu như tốc độ, góc quay vào cấu trúc dữ liệu, cấu trúc này sau đó sẽ được cập nhật vào PDO và truyền đi qua EtherCAT đến TwinCAT

Việc sử dụng Timer 7 giúp hệ thống tách biệt rõ ràng giữa logic xử lý và thời gian lấy mẫu, đảm bảo tính ổn định, định kỳ chính xác trong hệ thống điều khiển thời gian thực.

3.6.3. Cấu hình SPI – Giao tiếp với EtherCAT Click

– **Chức năng:**

Cầu nối truyền thông chính giữa EtherCAT Click và STM32, giúp đảm bảo dữ liệu điều khiển và trạng thái hoạt động được trao đổi nhanh chóng, chính xác, đồng thời hỗ trợ đồng bộ hóa thời gian thực trong hệ thống.

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 và chân giao tiếp: SPIx_MOSI, MISO, SCK, và chân CS điều khiển riêng bằng GPIO.

Hình 3.22: Cấu hình SPI2

CHƯƠNG 4: TRIỂN KHAI HỆ THỐNG VÀ ĐÁNH GIÁ KẾT QUẢ

4.1. Thi công hệ thống

4.1.1. Lắp đặt và bố trí thiết bị

- **Mục tiêu:** Đảm bảo các thiết bị được bố trí khoa học, tối ưu không gian và thuận tiện trong quá trình vận hành và kiểm tra.
- **Thiết bị chính trong hệ thống:**
  + EtherCAT Click: Đặt ở vị trí trung tâm để tối ưu kết nối SPI với các vi điều khiển STM32.
  + Các STM32F446: Được lắp đặt trong hộp nhựa cách điện với khoảng cách hợp lý nhằm giảm chiều dài dây kết nối SPI.
  + Driver SimpleFOC: Đặt lắp đặt trên STM32F446 gần động cơ BLDC để tối thiểu hóa tổn hao năng lượng trong mạch động.
  + Nguồn cấp: Sử dụng nguồn 8V để cấp điện cho các bộ điều khiển và động cơ BLDC.
  + Nguyên tắc bố trí thiết bị:
    - Thiết bị được cố định chắc chắn để tránh rung trong quá trình vận hành.
    - Dây tín hiệu được bọc chống nhiễu và đi riêng biệt với dây nguồn.

4.1.2. Kết nối mạch động lực và điều khiển

- **Kết nối mạch động lực:**
  + Dây nguồn từ nguồn 8V được nối trực tiếp đến các driver SimpleFOC.
  + Các cổng PWM từ driver SimpleFOC được nối với các cuộn dây của động cơ BLDC.
  + Sử dụng cầu chì bảo vệ để đảm bảo an toàn khi vận hành.
- **Kết nối mạch điều khiển:**
  + Giao tiếp SPI: EtherCAT Click được nối với các vi điều khiển STM32F446 thông qua SPI.
  + Encoder: Các tín hiệu từ encoder của động cơ BLDC được nối trực tiếp đến STM32F446 để đọc dữ liệu về tốc độ và vị trí.

4.1.3. Cài đặt các thông số ban đầu

- **Trên TwinCAT:**
  + Tạo cấu hình PDO (Process Data Objects) để truyền dữ liệu điều khiển từ TwinCAT xuống EtherCAT Click.
  + Cấu hình EEPROM cho EtherCAT Click để xác định các tham số SPI và chế độ hoạt động.
  + Tải xuống chương trình cho hoạt động demo của hệ thống.
- **Trên SimpleFOC:**
  + Thiết lập PID cho vòng điều khiển tốc độ và vị trí của động cơ BLDC.
  + Cấu hình dòng điện tối đa để bảo vệ động cơ.
- **Trên STM32:**
  + Sử dụng CubeMX để cấu hình SPI và các giao tiếp cần thiết.
  + Tối ưu sử dụng DMA để giảm tải cho CPU.

4.2. Thi công mô hình đồ án

- **Lắp ráp mô hình:**
  + Hai động cơ BLDC được gắn trên giá đỡ đảm bảo chắc chắn.
  + Sử dụng mũi tên để kiểm tra khả năng đồng bộ hóa vị trí giữa hai động cơ.
  + Các cảm biến encoder được gắn chính xác vào trục động cơ để thu thập dữ liệu vị trí và tốc độ.
- **Kiểm tra ban đầu:**
  + Kiểm tra tính ổn định của nguồn cấp và hoạt động của các driver động cơ.
  + Kiểm tra phản hồi encoder xác minh hoạt động chính xác.
  + Kiểm tra kết nối tín hiệu SPI và EtherCAT.

4.3. Vận hành hệ thống và đánh giá kết quả

4.3.1. Kết quả mô phỏng

- **Trường hợp truyền tín hiệu từ master xuống 2 slave:**
  + **Nhận xét:**

Hình ảnh chụp từ oscillo cho thấy hai tín hiệu phản hồi từ hai slave (màu xanh - Slave 1, màu vàng - Slave 2) khi nhận dữ liệu điều khiển từ TwinCAT thông qua EtherCAT. Có thể quan sát rằng tín hiệu màu vàng (Slave 2) một khoảng thời gian ngắn.

Hình 4.1: Độ lệch khi truyền dữ liệu từ TwinCat xuống 2 slave

Dựa trên thang đo thời gian 10 µs/div, độ trễ ước tính giữa hai tín hiệu là khoảng 2–3 µs. Với độ trễ chỉ vài micro giây giữa hai slave, hệ thống vẫn đảm bảo tính đồng bộ hóa cao, hoàn toàn phù hợp cho các ứng dụng điều khiển thời gian thực yêu cầu chính xác về thời gian như điều khiển động cơ đồng bộ.

– **Trường hợp truyền tín hiệu từ slave này sang slave kia:**

Hình 4.2: Độ lệch khi truyền dữ liệu từ slave này sang slave kia

+ **Nhận xét:**

Hình ảnh từ máy hiện sóng cho thấy hai tín hiệu phản hồi từ hai slave (xanh - Slave 1, vàng - Slave 2) khi slave 2 truyền dữ liệu đến slave 1 thông qua hệ thống EtherCAT nội bộ. Tín hiệu màu vàng (Slave 2) thay đổi trạng thái trước tín hiệu màu xanh (Slave 1), với độ trễ ước tính khoảng 1ms, dựa trên thang đo thời gian 5 ms/div. Điều này cho thấy quá trình truyền dữ liệu từ Slave 2 sang Slave 1 vẫn diễn ra tương đối nhanh, tuy nhiên có độ trễ nhất định do liên quan đến cơ chế xử lý nội bộ hoặc đồng bộ dữ liệu giữa các slave thông qua SPI và stack SOES. Mức trễ này vẫn nằm trong giới hạn chấp nhận được cho nhiều ứng dụng điều khiển thời gian thực.

– **Đánh giá chung:**

Thông qua các ảnh chụp từ máy hiện sóng, hệ thống đã được kiểm tra độ trễ trong hai tình huống truyền dữ liệu đặc trưng:

1. Trường hợp 1 – TwinCAT truyền dữ liệu đồng thời đến hai slave: Tín hiệu phản hồi từ hai slave (xanh – Slave 1, vàng – Slave 2) cho thấy độ trễ giữa hai phản hồi gần như không đáng kể, chỉ vào khoảng 2–3 µs. Điều này phản ánh tính chất hoạt động song song, đồng bộ cao của giao thức EtherCAT trong việc phân phối dữ liệu từ master đến các slave. Đây là một đặc điểm quan trọng của EtherCAT, cho phép điều khiển đa trục hoặc các hệ thống phân tán hoạt động đồng thời và tức thời.
2. Trường hợp 2 – Một slave truyền dữ liệu sang slave còn lại: Tín hiệu phản hồi ghi nhận độ trễ khoảng 1 ms, cho thấy có sự trễ đáng kể hơn khi truyền dữ liệu giữa các slave. Nguyên nhân có thể đến từ quy trình xử lý nội bộ tại slave, thời gian xử lý SPI, hoặc thứ tự cập nhật dữ liệu trong chu trình EtherCAT (process data exchange). Mặc dù vậy, độ trễ này vẫn ở mức chấp nhận được đối với phần lớn các hệ thống điều khiển không yêu cầu tốc độ phản hồi dưới millisecond.

– **Tổng kết:**

Hệ thống truyền thông EtherCAT thể hiện khả năng truyền dữ liệu đồng thời, độ trễ cực thấp từ master đến các slave, phù hợp với yêu cầu thời gian thực cao. Truyền dữ liệu giữa các slave có độ trễ cao hơn, nhưng vẫn đảm bảo tính ổn định và đáng tin cậy trong hệ thống. Với thiết kế và cấu hình hiện tại, hệ thống đủ điều kiện ứng dụng vào các bài toán điều khiển tốc độ, vị trí động cơ yêu cầu đồng bộ và phản hồi nhanh.

4.3.2. Phân tích và đánh giá kết quả

- **Ưu điểm:**
  + Hệ thống đảm bảo thời gian thực với độ trễ thấp và độ chính xác cao.
  + Động cơ hoạt động ổn định ngay cả khi thay đổi tải đột ngột.
  + Giao tiếp SPI giữa các STM32 và EtherCAT Click hoạt động mượt mà.
- **Hạn chế:**
  + Tốc độ xử lý có thể bị giới hạn khi hệ thống phải xử lý thêm nhiều động cơ hơn.
  + Hiệu suất PID chưa tối ưu khi động cơ hoạt động ở chế độ tải lớn.

4.3.3. Kinh nghiệm đạt được và hướng phát triển

- **Kinh nghiệm đạt được:**
  + Hiểu rõ quy trình thiết lập và vận hành giao thức EtherCAT.
  + Nắm vững kỹ thuật điều khiển động cơ BLDC bằng SimpleFOC và STM32.
  + Thành thạo việc tích hợp các thành phần phần cứng và phần mềm trong hệ thống điều khiển thời gian thực.
- **Hướng phát triển:**
  + Tăng số lượng động cơ được điều khiển đồng bộ hóa.
  + Tích hợp các thuật toán điều khiển tiên tiến, như điều khiển dự đoán hoặc AI, để cải thiện hiệu suất.
  + Tối ưu giao tiếp để giảm độ trễ hơn nữa và nâng cao khả năng mở rộng hệ thống.

KẾT LUẬN

Những kết quả đạt được:

- **Hoàn thành yêu cầu thiết kế:**
  + Nhóm đã hoàn thiện thiết kế và triển khai hệ thống điều khiển đồng bộ hóa tốc độ động cơ bằng công nghệ truyền thông EtherCAT, đáp ứng yêu cầu kỹ thuật ban đầu.
  + Hệ thống đã đảm bảo tính thời gian thực với độ trễ thấp và độ chính xác cao, đặc biệt trong việc đồng bộ hóa tốc độ và vị trí giữa hai động cơ BLDC.
- **Tích hợp phần mềm và phần cứng:**
  + Thành công cấu hình và tích hợp TwinCAT, EtherCAT Click, và các vi điều khiển STM32 để tạo thành hệ thống hoạt động mượt mà.
  + Chương trình điều khiển được xây dựng trên STM32 đã điều khiển chính xác động cơ thông qua driver SimpleFOC.
- **Thử nghiệm và đánh giá:**
  + Mô hình đã được kiểm nghiệm thành công, đáp ứng đúng các nguyên tắc điều khiển và đảm bảo khả năng đồng bộ hóa trong điều kiện vận hành thực tế.

Hạn chế trong đề tài:

- Hệ thống hiện chỉ hỗ trợ đồng bộ hóa hai động cơ và chưa được mở rộng để điều khiển nhiều động cơ hơn.
- Thuật toán điều khiển hiện tại chưa tối ưu khi động cơ hoạt động ở tải lớn hoặc môi trường thay đổi phức tạp.

Hướng đề xuất phát triển:

- **Mở rộng hệ thống:**
  + Nâng cấp hệ thống để điều khiển đồng bộ hóa nhiều động cơ hơn.
  + Tích hợp thêm các cảm biến khác như cảm biến dòng điện, nhiệt độ để giám sát toàn diện hơn.
- **Cải tiến thuật toán điều khiển:**
  + Áp dụng thuật toán bù PID hoặc AI để tăng hiệu suất và giảm độ trễ.
- **Cảnh báo và xử lý lỗi:**

+ Tích hợp chức năng phân tích dữ liệu để phát hiện và dự báo lỗi trong hệ thống.
+ Phát triển tính năng thông báo qua email hoặc ứng dụng di động để người quản lý nhận cảnh báo và báo cáo tức thời.

– **Nâng cấp phần mềm:**

+ Tạo giao diện giám sát thời gian thực trực quan hơn trên TwinCAT hoặc Website điều khiển.
+ Tăng cường khả năng cấu hình hệ thống trực tiếp từ phần mềm điều khiển.

TÀI LIỆU THAM KHẢO

[1] Cunfeng Kang, Yan Pang, Chunmin Ma and Chenmei Li, *Design of EtherCAT Slave Module,* p. 5, 2011.

[2] Junyan Qi, Le Wang, *Networked motion control system design based on EtherCAT,* p. 3, 2009.

[3] Vinh Quang Nguyen, Jae Wook Jeon, *EtherCAT Network Latency Analysis,* p. 5, 2016.

[4] Sridevi G, Anantha Saligram, Nattarasu V, *Establishing EtherCAT Communication between Industrial PC and Variable Frequency Drive,* p. 7, 2018.

[5] Mengtao Huang, Ping Li, *Design of Electronic Shaft Synchronization Control System Based on EtherCAT Bus,* p. 4, 2018.

[6] BECKHOFF, "EtherCAT Distributed Clocks," [Online]. Available: https://byvn.net/5WwZ.

[7] BECKHOFF, "EtherCAT Distributed Clocks - default settings," [Online]. Available: https://byvn.net/svrv.

[8] DENSOWAVE, "Object Dictionary List," [Online]. Available: https://byvn.net/T6zn.

[9] MESIDAS, "EtherCAT là gì? Giao thức truyền thông công nghiệp EtherCAT," [Online]. Available: https://byvn.net/h0ON.

[10] M. T. Inc, *EtherCAT Software Framework User's Guide,* p. 30, 2020.

[11] M. Technology, *2/3-Port EtherCAT Slave Controller with Integrated Ethernet PHYs,* p. 329, 2015.

[12] M. T. Inc., "EtherCAT LAN925x Library," [Online]. Available: https://byvn.net/QTEr.

[13] STM, *Datasheet - STM32F446xC/E,* p. 198, 2021.

[14] E. T. Group, *The Ethernet Fieldbus,* p. 86, 2009.

[15] E. T. Group, *EtherCAT Device Protocol,* p. 1, 2020.