**ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH**

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

**BỘ MÔN VIỄN THÔNG**

---------------o0o---------------

****

**BÁO CÁO**

**THỰC TẬP TỐT NGHIỆP**

**GVHD: TS.Phạm Quang Thái**

**SVTH: Đặng Lâm Tùng**

**MSSV: 1713856**

**TP. HỒ CHÍ MINH, THÁNG 09 NĂM 2020**

***LỜI CẢM ƠN***

Để hoàn thành báo cáo thực tập này trước hết em xin gửi đến công ty RBVH (*Robert Bosch Engineering and Business Solutions Vietnam Company Limited*) đã tạo điều kiện để thực tập lời cảm ơn chân thành.  
Đặc biệt, em xin gởi đến tất cả các thành viên trong đại gia đình 2WPs (2WPs team), những người đã tận tình hướng dẫn, giúp đỡ em hoàn thành báo cáo thực tập này lời cảm ơn sâu sắc nhất.  
Cảm ơn anh Đỗ Phi Thịnh, anh Đàm Quang Phục, anh Vũ Đức Huy vì sự giúp đỡ và những bài học, những phương pháp tiếp cận và cách giải quyết vấn đề đáng học hỏi,...

Cảm ơn anh Vũ Hải Đăng, anh Nguyễn Văn Sỹ, anh Nguyễn Thành Luân, anh Nguyễn Quang Trưởng, anh Bùi Nguyễn Trung Sơn, anh Nguyễn Cao Cường, anh Trần Anh Kiệt,… vì những sự giúp đỡ nhiệt tình.

Cảm ơn anh Nguyễn Minh Thái vì sự giúp đỡ và những đề xuất, những lời khuyên kịp thời.

Đồng thời nhà trường, bộ môn Viễn Thông, thầy Phạm Quang Thái: thầy hướng dẫn thực tập tốt nghiệp, đã giới thiệu cho em có cơ hội được thưc tập tại công ty, cho em bước ra công việc thực tế để áp dụng những kiến thức mà các thầy cô giáo đã giảng dạy. Qua công việc thực tập này em nhận ra nhiều điều mới mẻ và bổ ích trong thực tiễn để giúp ích cho công việc sau này của bản thân.  
Vì kiến thức bản thân còn hạn chế, trong quá trình thực tập, hoàn thiện báo cáo thực tập này em không tránh khỏi những sai sót, kính mong nhận được những ý kiến đóng góp từ thầy cũng như các anh /chị trong công ty.

“Trân trọng cảm ơn!”

*Tp. Hồ Chí Minh, ngày 18 tháng 09 năm 2020.*

Sinh Viên

**Đặng Lâm Tùng**

**MỤC LỤC**

[1. GIỚI THIỆU 1](#_Toc17818602)

[1.1 Giới thiệu về công ty 1](#_Toc17818603)

[1.2 Nhiệm vụ được giao thực tập 1](#_Toc17818604)

[1.3 Thời gian và lịch trình thực tập 2](#_Toc17818605)

[2. NỘI DUNG THỰC TẬP 4](#_Toc17818606)

[2.1 Nội dung 1: Đọc tài liệu, tìm hiểu về hệ thống chống bó cứng phanh ABS (Anti-lock Braking System) và hiểu chi tiết cách hoạt động của nó 8](#_Toc17818607)

[2.2 Nội dung 2: Hàn dây cáp của bộ wheel speed sensor 12](#_Toc17818608)

[2.3 Nội dung 3: Viết Tool Generate test case 13](#_Toc17818609)

[2.4 Nội dung 4: Xây dựng hệ thống kiểm sát thiết bị trong Lab 15](#_Toc17818610)

[2.5 Nội dung 5: Tìm hiểu về giao thức CAN 19](#_Toc17818610)

[2.6 Nội dung 6: Thực hiện giao thức CAN trên STM32 29](#_Toc17818611)

[2.7 Nội dung 7: Vẽ mạch test giao thức CAN cho IMU 37](#_Toc17818612)

[3. TỔNG KẾT CÔNG VIỆC THỰC TẬP 43](#_Toc17818613)

[3.1 Kết quả công việc thực tập 43](#_Toc17818614)

[3.2 Kinh nghiệm học được sau khi thực tập 43](#_Toc17818615)

3.3 Kế hoạch nghề nghiệp trong tương lai …………………….……….………………..44

[4. TÀI LIỆU THAM KHẢO 45](#_Toc17818616)

**DANH SÁCH HÌNH MINH HỌA**

[Hình 2.1.1 Hệ thống ABS xe 2 bánh và mô hình mô phỏng thời gian giảm tốc độ của hệ thống có ABS và không có ABS 10](#_Toc17819958)

[Hình 2.1.2: ABS enhanced 11](#_Toc17819959)

[Hình 2.1.3: Cảm biến tốc độ bánh xe (Wheel-Speed Sensor (WSS)) 12](#_Toc17819960)

Hình 1.1.4: Vị trí của ABS thủy lực và cảm biến tốc độ bánh xe trong xe gắn máy……………………………………………………………………………..12

[Hình 2.2.1: Cảm biến Wheel Speed Sensor 13](#_Toc17819961)

[Hình 2.3.1: Ví dụ file CSV 14](#_Toc17819962)

[Hình 2.3.2: Ví dụ file txt 14](#_Toc17819963)

[Hình 2.3.3: Code tool Generate testcase 15](#_Toc17819964)

[Hình 2.4.1: *Quy trình checkin – checkout…………………………………………..*16](#_Toc17819966)

[Hình 2.4.2: *Giao diện app…………………………………………………………….*17](#_Toc17819967)

[Hình 2.4.3: Module RFID RDM6300 …………………………………………17](#_Toc17819968)

[Hình 2.4.4: Sơ đồ chân…………………………………………………………18](#_Toc17819969)

[Hình 2.4.5: Đọc QR code với camera………………………………………….18](#_Toc17819970)

[Hình 2.5.1. Minh họa kết nối các thành phần điều khiển trong xe ô tô khi chưa sử dụng chuẩn CAN ……………………………...............................................20](#_Toc17819971)

[Hình 2.5.2. Minh họa kết nối các thành phần điều khiển trong xe ô tô khi sử dụng chuẩn CAN……………………………………………………………….20](#_Toc17819965)

[Hình 2.5.3: Đường dây CAN (tài liệu của Texas Instrument)…………….……23](#_Toc17819966)

Hình 2.5.4: Frame truyền cơ bản của CAN…………………………………….24

[Hình 2.5.5: Trường mở đầu của CAN……………………………………….....24](#_Toc17819968)

Hình 2.5.6: Trường điều khiển của CAN………………………………………25

Hình 2.5.7: Trường data và CRC của CAN……………………………………26

Hình 2.5.8: Các lỗi có thể xảy ra trong giao tiếp CAN…………………...……27

[Hình 2.5.9: Các mode lỗi của CAN…………………………………………….28](#_Toc17819965)

Hình 2.6.1: Thanh ghi quản lý clock của CAN………………………………...29

Hình 2.6.2: Mô hình kết nối CAN của STM32………………………………...30

Hình 2.6.3: Tính toán bit timing với STM32…………………………………..31

Hình 2.6.4: Mô tả bit timing của CAN……………………………………...…32

Hình 2.6.5: Giá trị thanh ghi bit timing với STM32………………………..….32

Hình 2.6.6: Setup clock STM32……………………………………………..…33

Hình 2.6.7: Cấu hình ngoại vi CAN với STM32………………………………33

Hình 2.6.2: Mô hình kết nối CAN của STM32………………………...………34

Hình 2.6.8: Các biến cần thiết cho ngoại vi CAN STM32 ……………………34

Hình 2.6.9: Header của message CAN ……………………………………..…34

Hình 2.6.10: Config filter cho ngoại vi CAN……………………………….…35

Hình 2.6.11: Bật ngoại vi CAN……………………………………………..…35

Hình 2.6.12: Hàm truyền CAN message …………...…………………………35

Hình 2.6.13: Tín hiệu trên dây CAN\_H so với GND …………………………35

Hình 2.6.14: Bit 0 trên dây CAN\_H ………………………………………..…36

Hình 2.6.15: Bit 0 trên dây CAN\_L………………………………………...…36

Hình 2.7.1: Cảm biến MM5.10 ………………………………………………37

Hình 2.7.2: Thông số data output của MM5.10…………………………….…38

Hình 2.7.3: Thông số resolution của MM5.10…………………………..….…39

Hình 2.7.4: Thông số resolution của MPU9250…………………………….…40

Hình 2.7.5: CAN ID của các message của MM5.10………………………..…40

Hình 2.7.6: Schematic của mạch mô phỏng ………………………………..…41

Hình 2.7.7: Schematic của mạch mô phỏng ………………………………..…41

Hình 2.7.6: Mạch thực tế………………………………..……………………..42

**DANH SÁCH BẢNG SỐ LIỆU**

Bảng 1: Các dòng phân khúc của sản phẩm ABS và cấu tạo tương ứng……………………………………………………………………6

# GIỚI THIỆU

## Giới thiệu về công ty

Năm 1886, Robert Bosch thành lập “Phân xưởng Cơ khí Chính xác và Kỹ thuật Điện” tại Stuttgart, nước Đức. Đây chính là sự ra đời của tập đoàn Bosch hiện nay.

Bosch bắt đầu hoạt động tại Việt Nam từ năm 1994, và đến hiện tại có trụ sở chính tại TP. Hồ Chí Minh, hai văn phòng chi nhánh ở Hà Nội và Đà Nẵng cùng nhà máy Gasoline Systems sản xuất dây đai truyền lực biến đổi liên tục (CVT) tại tỉnh Đồng Nai. Ngoài ra, công ty cũng vận hành một trung tâm Nghiên cứu và Phát triển Công nghệ và các giải pháp doanh nghiệp (Công ty TNHH Robert Bosch Engineering & Business Solutions Việt Nam) và Trung tâm Nghiên cứu & Phát triển Công nghệ Ô tô tại TP. Hồ Chí Minh.

Công ty **Robert Bosch Engineering and Business Solutions Vietnam (RBVH)** là công ty con được đầu tư 100% vốn từ Robert Bosch GmbH, một trong những nhà cung cấp hàng đầu thế giới về công nghệ và dịch vụ. Thành lập từ năm 2010 tại TP.HCM, RBVH là trung tâm phát triển phần mềm đầu tiên của Bosch tại khu vực Đông Nam Á. RBVH cung cấp giải pháp cho doanh nghiệp trong các lĩnh vực:

* **Thiết kế cơ khí & mô phỏng**
* **Phát triển phần mềm nhúng & phần cứng**
* **Giải pháp CNTT cho doanh nghiệp**

## Nhiệm vụ được giao thực tập

* Nội dung 1: Đọc tài liệu, tìm hiểu về hệ thống chống bó cứng phanh - ABS (Anti-lock Braking System) và hiểu chi tiết cách hoạt động của nó.
* Nội dung 2: Hàn dây cáp của bộ wheel speed sensor
* Nội dung 3: Xây dựng hệ thống kiểm sát thiết bị trong Lab
* Nội dung 4: Tìm hiểu giao thức CAN
* Nội dung 5: Thực hiện giao thức CAN trên STM32
* Nôi dung 6: Thiết kế mạch mô phỏng giao tiếp CAN trên STM32

## Thời gian và lịch trình thực tập

* Thời gian thực tập: 3 tháng (từ ngày 17/07/2020 đến ngày 22/09/2020)
* Lịch trình thực tập:
  + Tuần 1: Tham gia OnBoarding Training (Etown 2 Training Room 3+4, 8F)
  + Tuần 2: Đọc tài liệu, hiểu thêm về các hệ thống ABS, ESP, TCS. Tham gia học tập và hoàn thiện các chứng chỉ yêu cầu bắt buộc trong công ty, vd: WBT Training, Labcar (Laboratory access right), ESD,ISP, OSS…
  + Tuần 3: Nghiên cứu tài liệu về hệ thống ABS cho xe 2 bánh. Tham gia các buổi training về các chủ đề liên quan
  + Tuần 4: Viết Tool Generate test case, tham gia các buổi training về các chủ đề liên quan
  + Tuần 5: Review Tool Generate test case, tham gia các buổi training về các chủ đề liên quan
  + Tuần 6: Thiết kế hệ thống kiểm soát thiết bị trong lab
  + Tuần 7: Review thiết kế hệ thống kiểm soát thiết bị trong lab Tuần 8: Thực hiện giao thức CAN trên STM32, tham gia các buổi training về các chủ đề liên quan
  + Tuần 9: Thiết kế mạch mô phỏng giao tiếp CAN trên STM32

, tham gia các buổi training về các chủ đề liên quan

* + Tuần 10: Chưa diễn ra (Hiện chưa có Task được giao)
  + Tuần 11: Chưa diễn ra (Hiện chưa có Task được giao)
  + Tuần 12: Chưa diễn ra (Hiện chưa có Task được giao)

# NỘI DUNG THỰC TẬP

## Nội dung 1: Đọc tài liệu, tìm hiểu về hệ thống chống bó cứng phanh ABS (Anti-lock Braking System) và hiểu chi tiết cách hoạt động của nó

Đọc tài liệu tổng quan về hệ thống ABS (Anti-lock Braking System) và hệ thống ABS cho xe 2 bánh, 3 bánh (2WP: Two-wheelers and powersports) nói riêng.

Tìm hiểu chức năng, cách thức hoạt động của các thành phần trên hệ thống ABS cho xe 2 bánh:

Hệ thống phanh ABS có các bộ phận chính sau đây:

* ECU điều khiển trượt: Bộ phận này xác định mức trượt giữa bánh xe và mặt đường dựa vào các tín hiệu từ các cảm biến và điều khiển bộ chấp hành của phanh. Gần đây, một số kiểu xe có ECU điều khiển trượt lắp trong bộ chấp hành của phanh.
* Công tắc phanh: báo cho ECU biết khi nào người lái đạp phanh và dừng đạp phanh
* Bộ chấp hành của phanh: Bộ chấp hành của phanh điều khiển áp suất thuỷ lực của các xilanh ở bánh xe bằng tín hiệu ra của ECU điều khiển trượt.
* Cảm biến tốc độ: Cảm biến tốc độ phát hiện tốc độ của từng bánh xe và truyền tín hiệu đến ECU điều khiển trượt.

Ngoài ra, trên táp lô điều khiển còn có:

* Đèn báo táp-lô: Đèn báo của ABS, khi ECU phát hiện thấy sự trục trặc ở ABS hoặc hệ thống hỗ trợ phanh, đèn này bật sáng để báo cho người lái. Đèn báo hệ thống phanh, khi đèn này sáng lên đồng thời với đèn báo của ABS, nó báo cho người lái biết rằng có trục trặc ở hệ thống ABS và EBD. Công tắc đèn phanh: Công tắc này phát hiện bàn đạp phanh đã được đạp xuống và truyền tín hiệu đến ECU điều khiển trượt. ABS sử dụng tín hiệu của công tắc đèn phanh. Tuy nhiên dù không có tín hiệu công tắc đèn phanh vì công tắc đèn phanh bị hỏng, việc điều khiển ABS vẫn được thực hiện khi các lốp bị bó cứng. Trong trường hợp này, việc điều khiển bắt đầu khi hệ số trượt đã trở nên cao hơn (các bánh xe có xu hướng khoá cứng) so với khi công tắc đèn phanh hoạt động bình thường.
* Cảm biến giảm tốc: Chỉ có ở một số loại xe. Cảm biến giảm tốc cảm nhận mức giảm tốc của xe và truyền tín hiệu đến ECU điều khiển trượt. Bộ ECU đánh giá chính xác các điều kiện của mặt đường bằng các tín hiệu này và sẽ thực hiện các biện pháp điều khiển thích hợp.
* **Nguyên lý hoạt động:**

Đây là một hệ thống sử dụng các cảm biến điện tử để nhận biết một hoặc nhiều bánh bị bó cứng trong quá trình phanh của xe. Khi hệ thống phát hiện tốc độ của bánh xe bằng 0 trong khi xe còn đang di chuyển (bị trượt) nó sẽ hiểu là bánh xe đó bị bó cứng. Vậy để ngăn chặn tình trạng bó cứng phanh xảy ra hệ thống sẽ điều chỉnh áp lực phanh tới má phanh tương ứng bằng cách đóng hoặc mở van liên tục trên đường dẫn thủy lưc sao cho phù hợp

* Lúc đó dầu phanh sẽ được hệ thống tình toán đẩy vào trong bộ điều khiền thủy lực, và được ép lại tại đây để nâng cao áp suất trước khi đưa dầu đến các bộ phận phanh trong mỗi bánh xe.
* Trong khi phân tích những dữ liệu do bộ cảm biến vân tốc bánh xe gửi về, nếu Bộ Điều Khiển ABS nhận ra một bánh xe nào đó sắp bị khóa cứng, thì hệ thống điều khiển ABS sẽ đưa tín hiệu xuống để đóng Valve không cho dầu đổ xuống đó nữa. Và mở Valve khi cần thiết cho dầu phanh lưu thông trở lại. Quá trình này lặp đi lặp lại nhiều lần bảo đảm cho bánh xe lăn đều trong khi giảm tốc, tránh tình trạng bánh bị khóa cứng.
* Khi đạp phanh, áp suất của dầu tăng lên làm cho má phanh kẹp lại, khi trường hợp xe bị trượt, van EV đóng lại van AV mở ra, Pump hút bớt dầu lên trên chứa vào trong Acc (Accomulate) chứa tạm thời. Áp suất trên má phanh giảm đến khi nào xe không bị trượt. Dầu được bơm lên lại chu trình, van EV mở, van AV đóng.

Công thức xác định độ trượt:

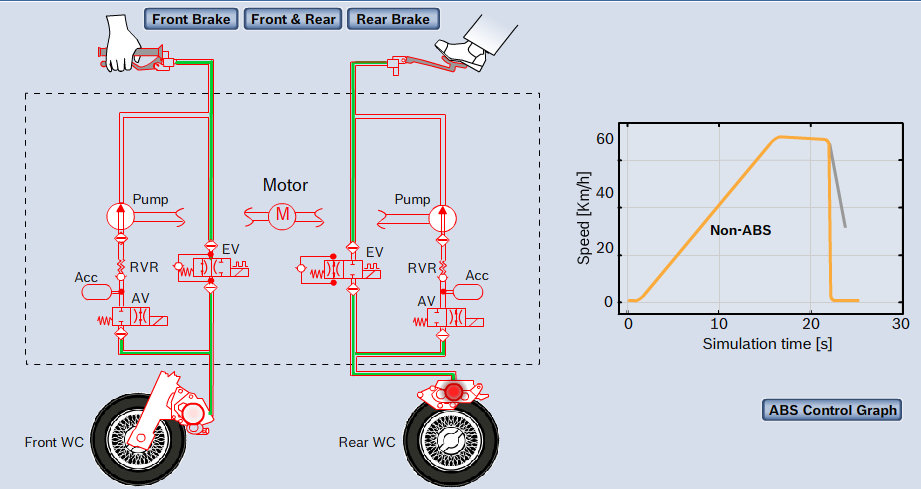
\*100%

Với:

* là tốc độ của xe
* là tốc độ bánh xe

Khi xe đang chạy mà thì xe bị trượt hay .

Hình 2.1.1 phía bên phải so sánh thời gian giảm tốc độ khi có ABS sẽ giảm chậm hơn so với không có ABS, giúp người lái xe an toàn hơn.



Hình 2.1.1 Hệ thống ABS xe 2 bánh và mô hình mô phỏng thời gian giảm tốc độ của hệ thống có ABS và không có ABS

Tìm hiểu các dòng phân khúc của sản phẩm ABS và cấu tạo tương ứng của nó.

* Đối với dòng ABSMP xe gắn máy (Anti-lock Braking System Motocycle Plus) sẽ có 2 WSS, 4 Valve và 1 Pressure Sensor.
* Đối với dòng ABSME xe gắn máy (Anti-lock Braking System Motocycle Enhanced) sẽ có 2 WSS, 8 Valve và 4 Pressure Sensor.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cấu tạo  (Component) | Xe Ô Tô | | Xe Gắn Máy | |
| ABS  (Anti-lock Braking System) | ESP (Electronics Stability Program) | ABSMP | ABSME |
| Cảm biến tốc độ bánh xe (WSS) | 4 | 4 | 2 | 2 |
| Van (Valves) | 8 | 12 | 4 | 8 |
| Cảm biến áp suất (Pressure Sensor) | Not Used | 1 | 1 | 4 |

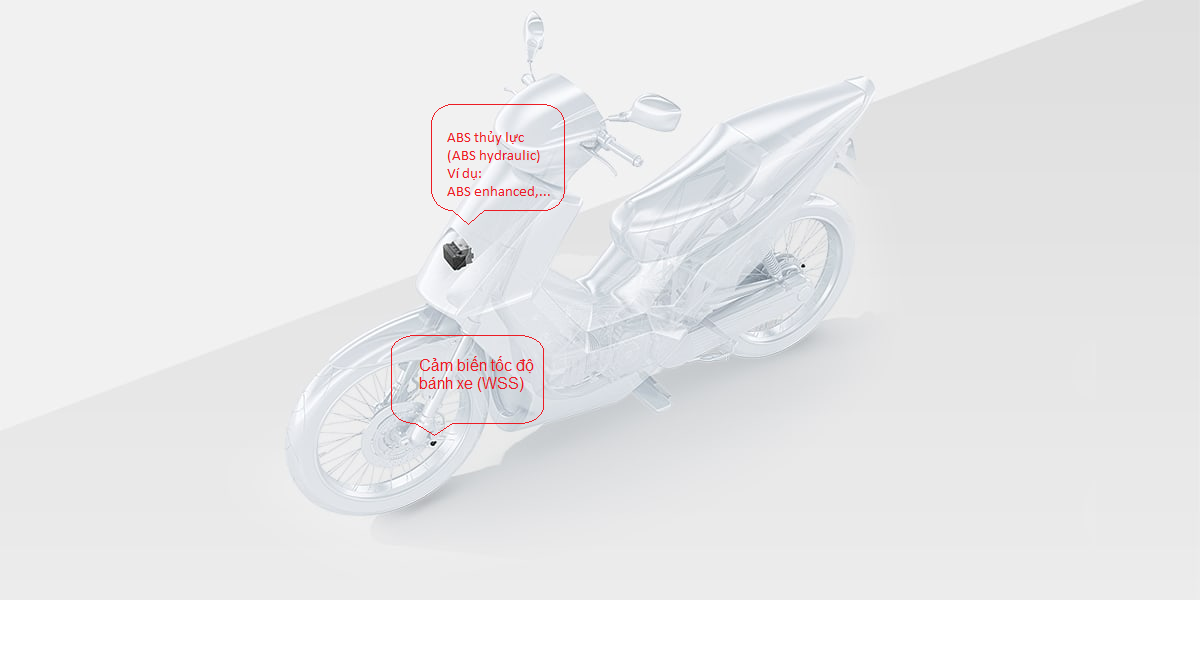
*Bảng 1: Các dòng phân khúc của sản phẩm ABS và cấu tạo tương ứng.*



Hình 2.1.2: *ABS enhanced*



Hình 2.1.3: *Cảm biến tốc độ bánh xe (Wheel-Speed Sensor (WSS))*

Hình 2.1.4: *Vị trí của ABS thủy lực và cảm biến tốc độ bánh xe trong xe gắn máy*

## Nội dung 2: Hàn dây cáp của bộ wheel speed sensor

Hàn dây cáp cho wheel speed sensor để thực hiện test bộ ABS test.

Trên thị trường có nhiều loại wheel speed sensor, trong đó có phần lớn là cảm biến sử dụng công nghệ cảm biến Hall, data ra là data analog hoặc là tín hiệu CAN

Trong quá trình thực tập, em được sử dụng cảm biến Wheel speed sensor sử dụng CAN

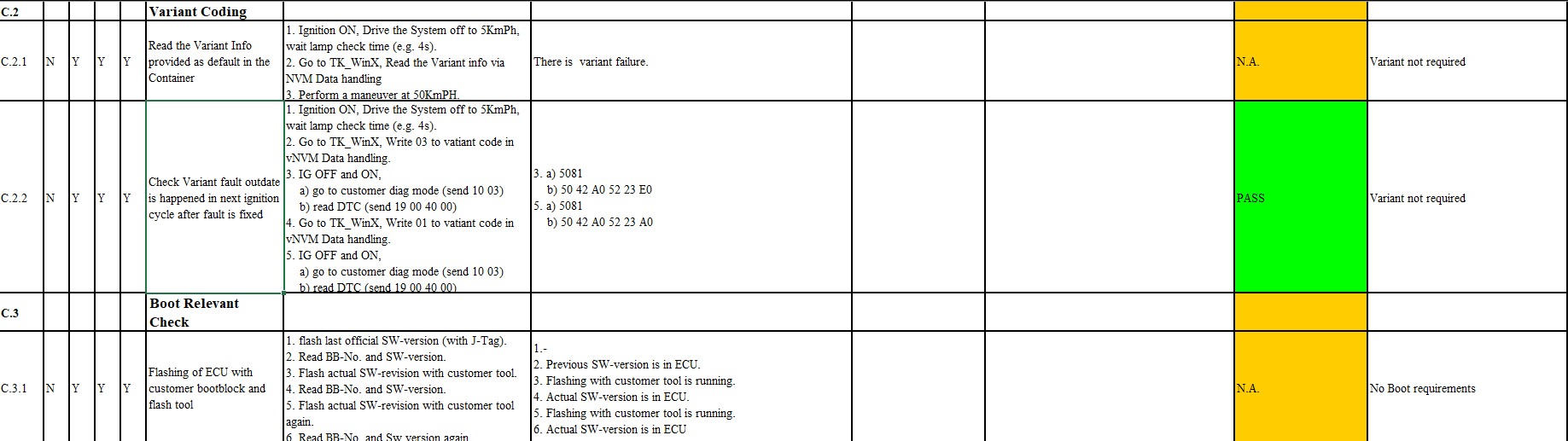


Hình 2.2.1: *Wheel speed sensor dùng của xe Piaggo*

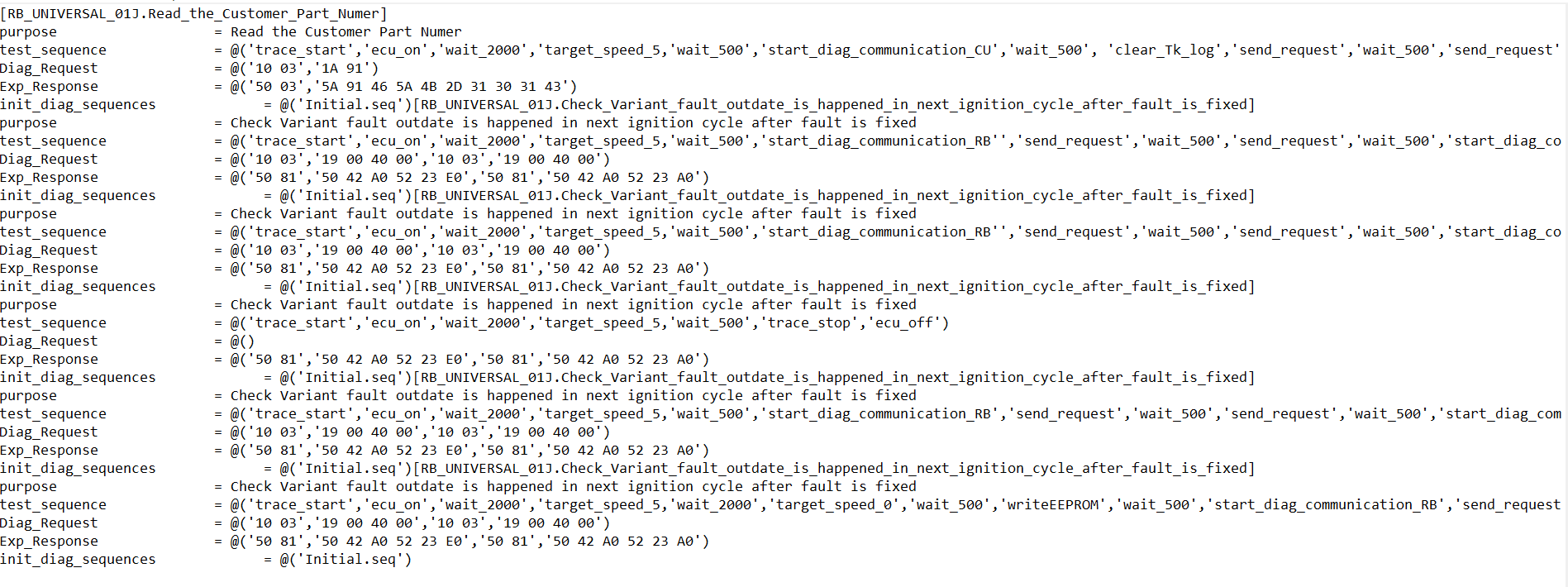
Công việc cần thực hiện: Cắt bỏ vỏ nhựa sensor và hàn 2 dây tín hiệu CAN high, CAN low vào ABS

## Nội dung 3: Viết Tool Testcase generate

Để quản lý việc build software trong công ty, công cụ được sử dụng là Jenkins, công cụ này (nguồn mở) có chức năng tự động xử lý các công việc building software và version control cho code trong công ty, đồng thời tích hợp thêm cả chức năng tự xử lý việc testing sử dụng các tool nguồn đóng của công ty, các test này được lưu vào trong một file CSV do khách hàng yêu cầu hoặc do Bosch test, để thực hiện test tự động cần yêu cầu có một file .txt lưu các từ khóa test để đưa vào Jenkin. Công việc của em là sinh ra test case từ file CSV này.

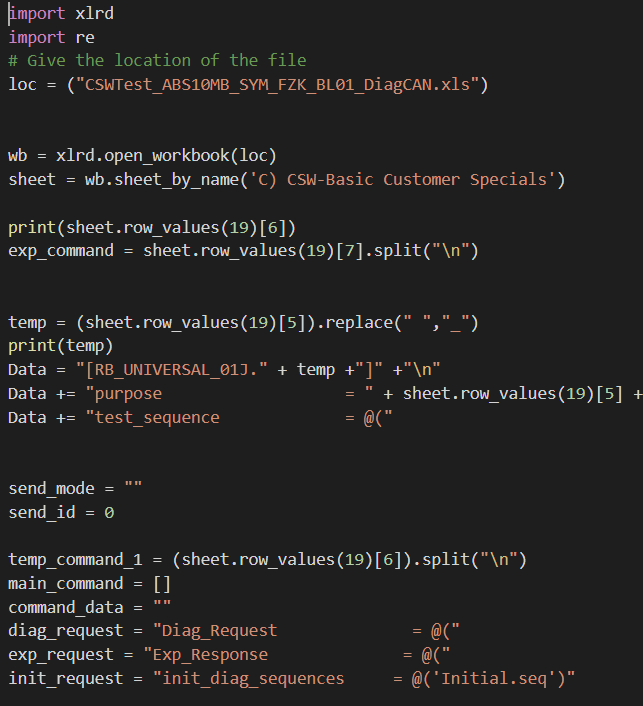


Hình 2.3.1: Ví dụ file CSV



Hình 2.3.2: Ví dụ file txt

Ngôn ngữ sử dụng cho tool này là Python. Để truy cập vào file csv, em sử dụng thư viện xlrd [3] để đọc file và thư viện regex[4] của python để tìm các từ khóa đúng:



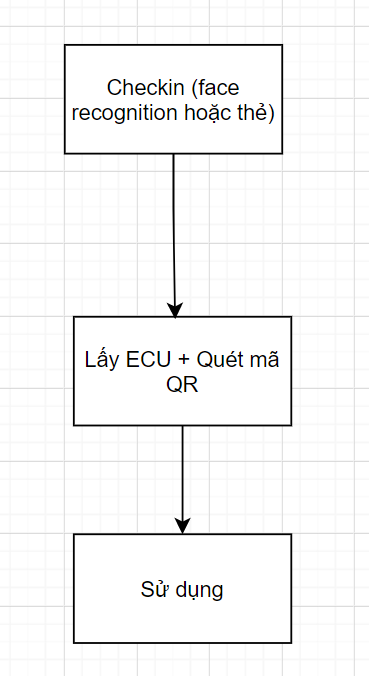
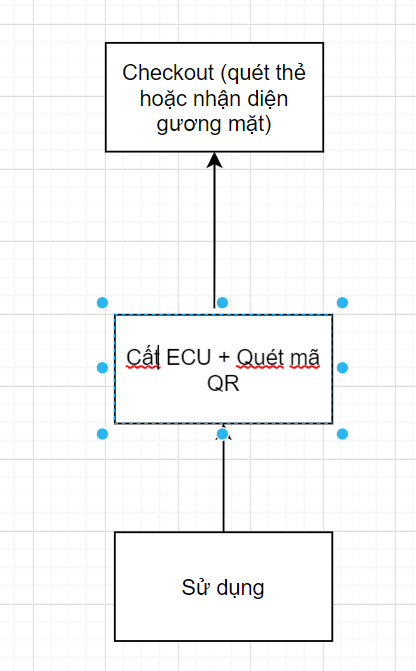
Hình 2.3.3: Một phần code

## Nội dung 4: Thiết kế hệ thống giám sát mượn thiết bị trong Lab

Do quá trình sử dụng thiết bị trong lab không tốt, có một số thiết bị (ECU) đã bị hư hỏng, do đó em được giao thiết kế một hệ thông giám sát việc mượn thiết bị trong lab.

Ý tưởng ban đầu là sử dụng nội dung đồ án của em: Hệ thống nhận diện gương mặt kết hợp với một hệ thống đọc mã vạch sử dụng xử lý ảnh cùng với hệ thống rfid để quét thẻ.

Sơ đồ quy trình:

Hình 2.4.1: *Quy trình checkin – checkout*

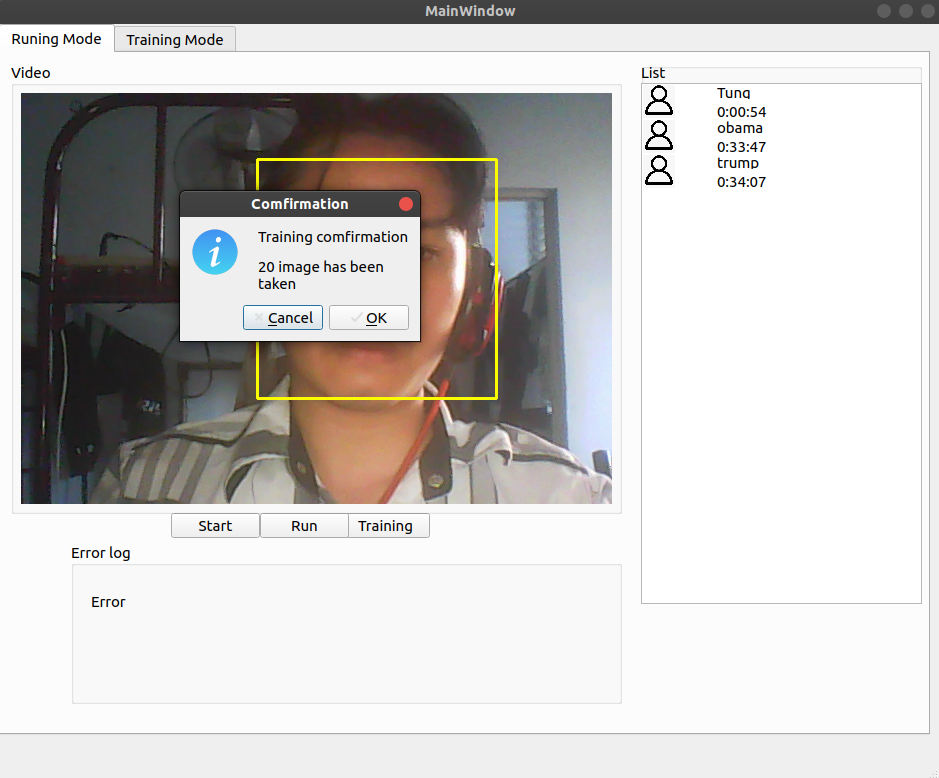
Thiết bị sử dụng:

+ 1 máy tính của công ty

+ Webcam Logitech

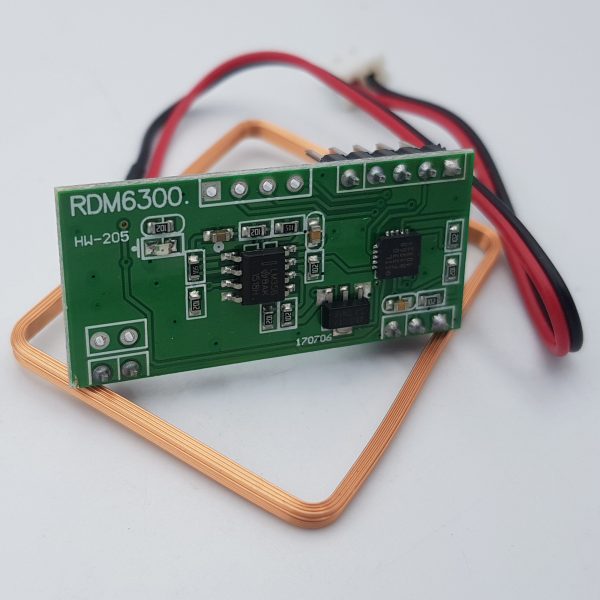
+ Module đọc thẻ RFID có kết nối UART

Phần mềm được dựa trên đồ án về nhận diện gương mặt của em



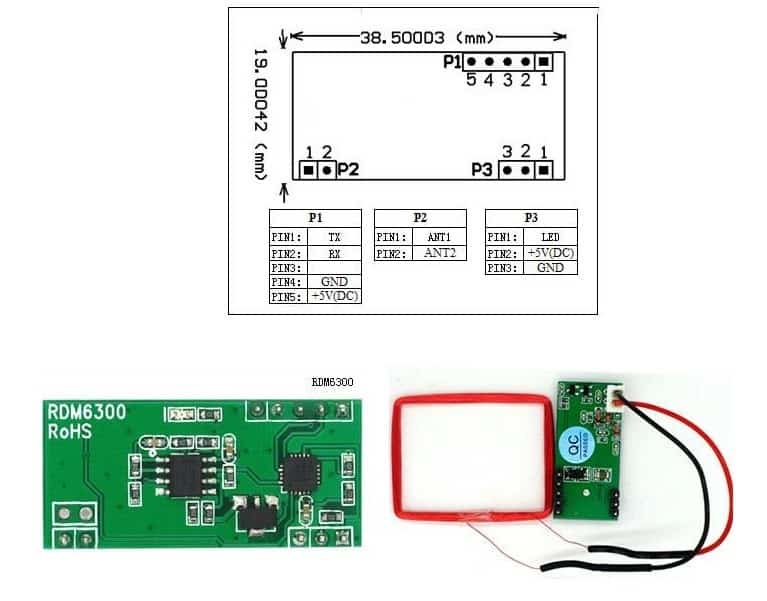
Hình 2.4.2: *Giao diện app*

Để đọc RFID, em sử dụng module RFID RDM6300 125kHz :



Hình 2.4.3: Module RFID RDM6300

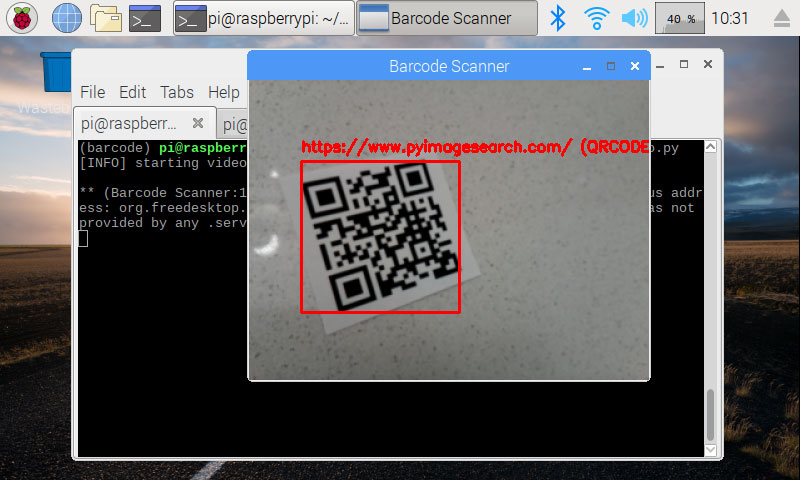
Sơ đồ chân:



Hình 2.4.4: Sơ đồ chân

Module này có output là tín hiệu UART có baudrate 9600, rất dễ dàng sử dụng, có thể cắm trực tiếp vào máy tính

Để nhận diện và đọc barcode, em sử dụng thư viện zbar python:



Hình 2.4.5: Đọc QR code với camera

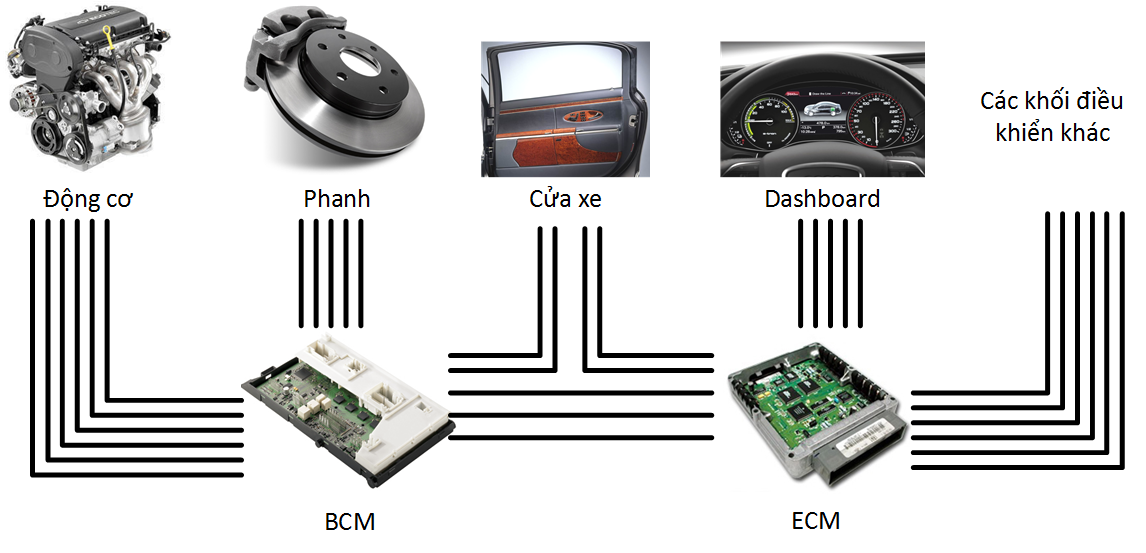
Đây là thư viện đọc mã QR và barcode nguồn mở, rất dễ sử dụng và độ chính xác rất cao.

Tất cả hệ thống được gói gọn thành một bộ PC. Tuy nhiên do vấn đề lắp đặt PC và camera trong lab khá phức tạp, đồng thời ECU của công ty cũng cần có quyết định của cấp trên, nên task này chưa được đưa vào ứng dụng mà chỉ có demo.

## . Nội dung 5: Tìm hiểu giao thức CAN

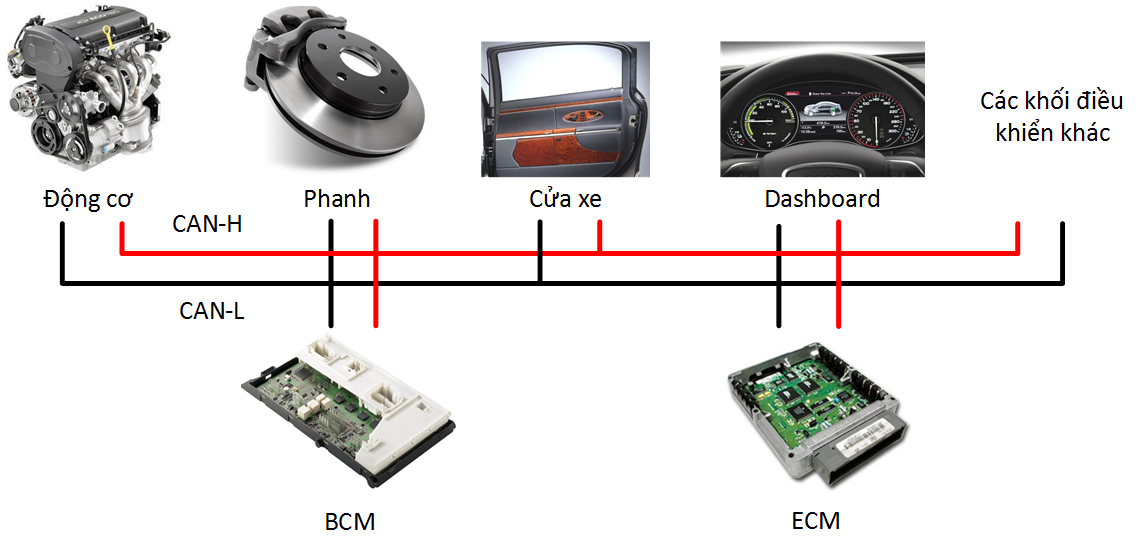
### NGUYÊN NHÂN RA ĐỜI [5]

Chuẩn CAN ra đời với mục đích ban đầu là phục vụ cho ngành công nghiệp xe ô tô. Trước khi sử dụng chuẩn CAN, việc kết nối giữa các khối (module) điều khiển trong xe hơi rất phức tạp và yêu cầu nhiều dây nối. Khi ngành công nghiệp xe ô tô phát triển, các khối điều khiển điện tử với nhiều chức năng phức tạp (điều khiển thân xe, điều khiển cửa, điều khiển động cơ, thu thập dữ liệu các cảm biến, định vị xe, ...) trong xe ngày càng nhiều làm việc bố trí kết nối càng trở nên phức tạp, tốn kém và bảo trì khó khăn.



Hình 2.5.1. Minh họa kết nối các thành phần điều khiển trong xe ô tô khi chưa sử dụng chuẩn CAN

CAN ra đời đã giải quyết các vấn đề tồn tại. Các module chỉ cần 2 dây để kết nối với nhau. Việc thêm hay bớt module trong bus CAN dễ dàng. Việc truyền dữ liệu có độ tin cậy cao, tốc độ nhanh.



Hình 2.5.2. Minh họa kết nối các thành phần điều khiển trong xe ô tô khi sử dụng chuẩn CAN

### 2. TÓM LƯỢC LỊCH SỬ PHÁT TRIỂN

Lịch sử phát triển của CAN được tóm lược như sau:

· 1983: Bắt đầu dự án phát triển mạng trên xe hơi trong nội bộ hãng Bosch.

· 1986: Chính thức giới thiệu giao thức CAN.

· 1987: Những chíp điều khiển CAN đầu tiên xuất hiện ở nhà sản xuất linh kiện bán dẫn Intel và Philips.

· 1991: Bosch xuất bản thông số kỹ thuật CAN 2.0.

· 1992: Thành lập nhóm các nhà sử dụng và sản xuất CAN quốc tế: Hội CAN tự động hóa (CiA). Hội CiA xuất bản giao thức Lớp ứng dụng CAN (CAN Application Layer, CAL). Những chiếc xe Mercedes-Benz đầu tiên được trang bị CAN xuất hiện.

· 1993: Xuất bản tiêu chuẩn ISO 11898.

· 1994: CiA tổ chức Hội nghị CAN quốc tế lần thứ nhất (iCC). Allen-Bradley giới thiệu giao thức DeviceNet.

· 1995: Xuất bản Tiêu chuẩn ISO 11898 sửa đổi (định dạng khung mở rộng). CiA xuất bản giao thức CANopen.

### THUỘC TÍNH & ƯU ĐIỂM CỦA BUS CAN

+ Đơn giản, chi phí thấp: Như đã trình bày ở trên, bus CAN chỉ có 2 dây giúp kết nối các module điều khiển với nhau dễ dàng hơn khi so sánh với cách làm truyền thống. Kèm theo đó là nhiều lợi ích về việc dễ lắp đặt và dễ sửa chữa, bảo trì khi có sự cố.

+ Tạo ra một giao thức chung để nhiều nhà cung cấp khác nhau có thể phát triển các module điều khiển tương thích với nhau

+ Tính ưu tiên của thông điệp (Prioritization of messages): mỗi thông điệp được truyền ra từ một nút (node) hay trạm (station) trên bus CAN đều có mức ưu tiên. Khi nhiều thông điệp được truyền ra bus cùng lúc thì thông điệp có mức ưu tiên cao nhất sẽ được truyền. Cá thông điệp có mức ưu tiên thấp hơn sẽ tạm dừng và được truyền lại khi bú rảnh. Việc xác định mức ưu tiên của thông điệp dựa trên cấu tạo (cấu trúc) thông điệp và cơ chế phân xử quy định trong chuẩn chuẩn CAN.

+ Cấu hình linh hoạt: Cho phép thiết lập cấu hình thời gian bit, thời gian đồng bộ, độ dài dữ liệu truyền, dữ liệu nhận, ... (tham khảo thêm datasheet của một ngoại vi CAN trong vi điều khiển bất kỳ để có thông tin cụ thể)

+ Nhận dữ liệu đa điểm với sự đồng bộ thời gian: một thông điệp có thể được nhận bởi nhiều node khác nhau trong bus cùng lúc. Tất cả các node trên bus đều có thể thấy thông điệp đang truyền trên bus, tùy vào cấu hình ở mỗi node mà node sẽ quyết định có chấp nhận thông điệp này hay không.

+ Nhiều master (Multimaster)

+ Phát hiện và báo hiệu lỗi: Mỗi thông điệp có kèm theo mã CRC (Cyclic Redundancy Code) để thực hiện kiểm tra lỗi. Nếu lỗi xuất hiện, node nhận sẽ bỏ qua thông điệp lỗi và truyền khung báo lỗi (error frame) lên bus CAN. Mỗi node trong bus có bộ đếm quản lý lỗi truyền nhận riêng để xác định trạng thái lỗi của chính nó. Nếu lỗi xuất hiện quá nhiều, một node có thể tự động ngắt khỏi bus. Ngoài ra còn một số dạng lỗi khác có thể được phát hiện với chuẩn CAN.

+ Tự động truyền lại các thông điệp bị lỗi khi bus rảnh: Một thông điệp được truyền ra bus nếu bị lỗi thì sẽ không mất đi mà node truyền thông điệp này sẽ giữ nó lại và tự động phát lại thông điệp này khi bus CAN rảnh cho đến khi thành công. Điều này giúp đảm bảo tính toàn vẹn dữ liệu trong bus

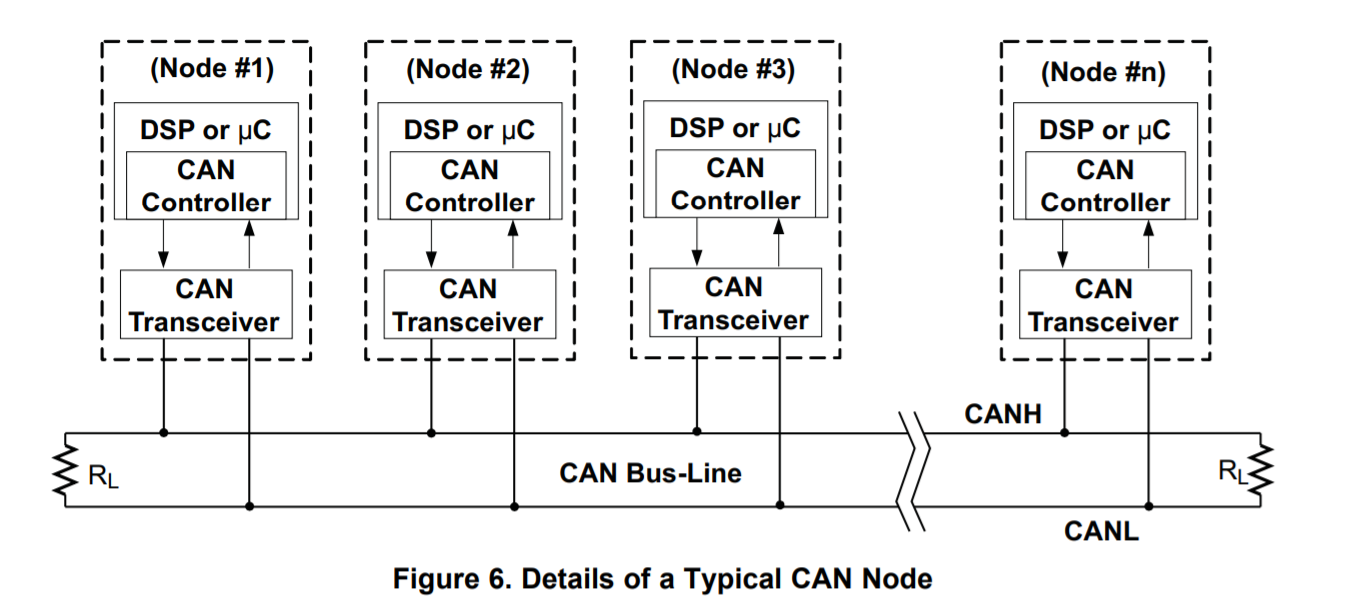
### Chuẩn vật lý của giao tiếp CAN:

+ Giao tiếp CAN là chuẩn giao tiếp đồng bộ sử dụng mã NRZ, có sử dụng Bit Stuffing.

+ Đường dây truyền CAN là đường dây vi sai, trong đó bit 0 tương ứng với 2 dây này có điện áp ngược nhau và bit 1 là 2 dây có cùng mức điện áp 0.

+ Để quản lý giao tiếp CAN, cần có một CAN Controller và một CAN tranceiver quản lý việc truyền nhận dữ liệu.

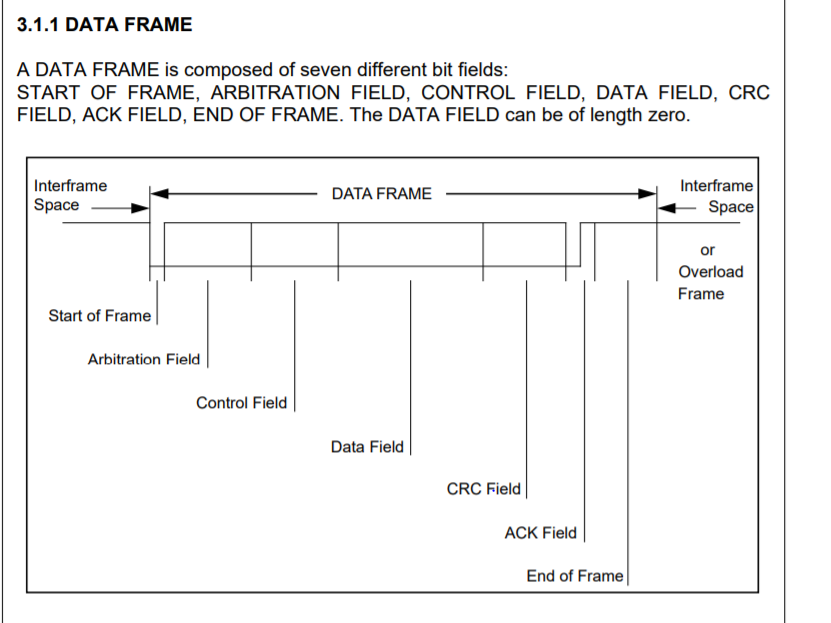
+ Vấn đề cân bằng trở kháng trên đường dây CAN cũng rất quan trọng, ở mỗi đầu terminal của CAN bus cần có một trở 120 Ohm.



Hình 2.5.3: Đường dây CAN (tài liệu của Texas Instrument)[6]

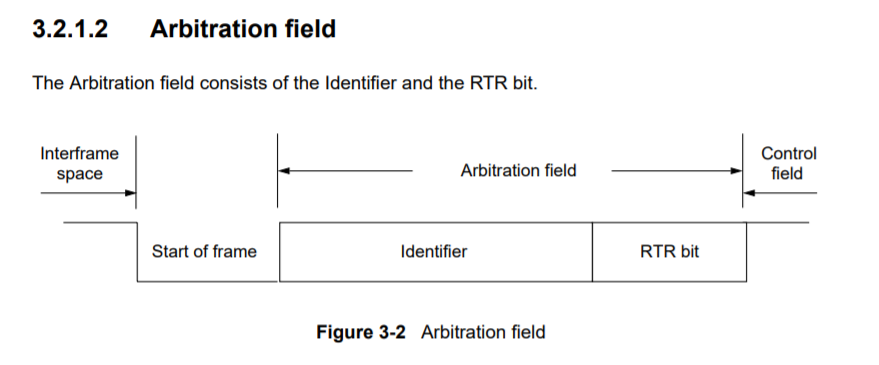
### Data frame của giao tiếp CAN: [7]

CAN có nhiều loại data frame, chúng được cấu thành từ các trường cơ bản như sau:



Hình 2.5.4: Frame truyền cơ bản của CAN

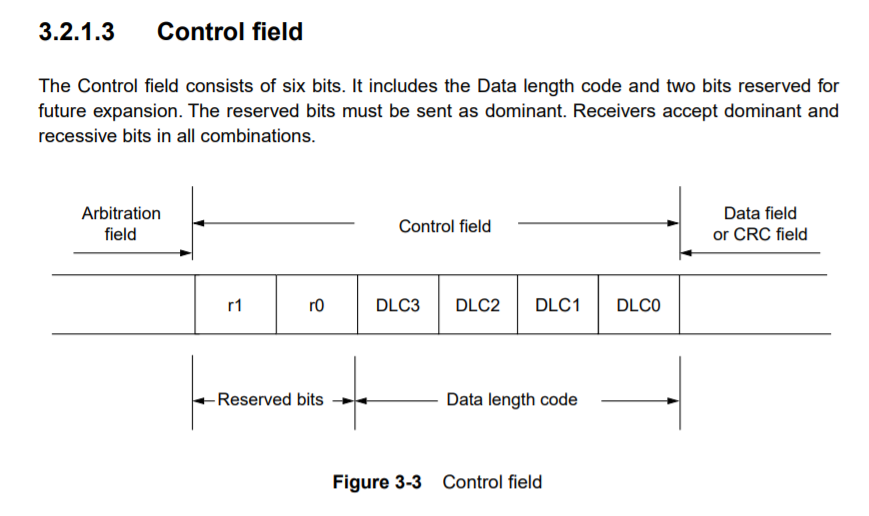
Trường mở đầu:



Hình 2.5.5: Trường mở đầu của CAN

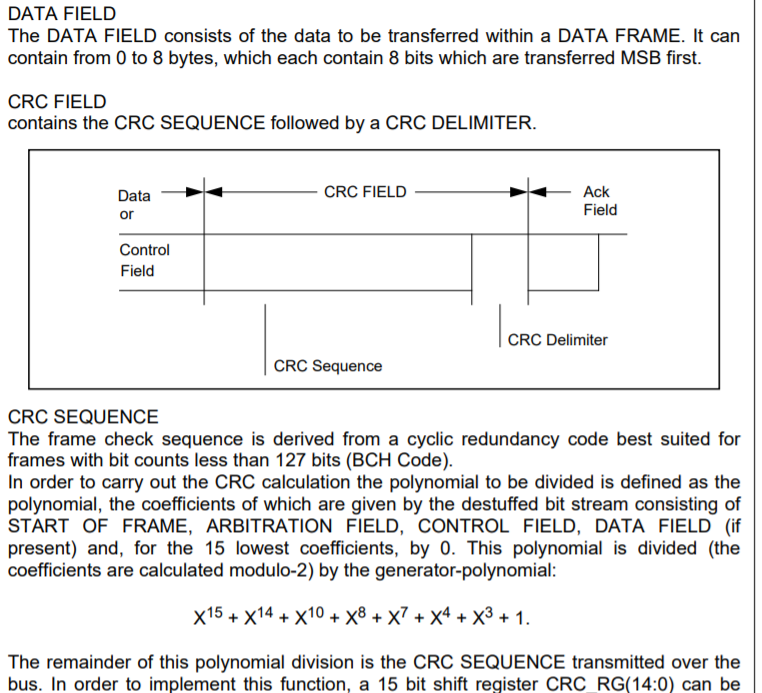
Trường này chứa phần CAN ID của từng messager, phần ID này thường có 11 bit cho CAN thường và 29 bit cho CAN extended.

Sau đó là trường điều khiển, phần này được sử dụng để thông báo về ý nghĩa của từng message, data length của phần data phía sau:



Hình 2.5.6: Trường điều khiển của CAN

Trường data: Trường này là data của messager CAN được truyền:



Hình 2.5.7: Trường data và CRC của CAN

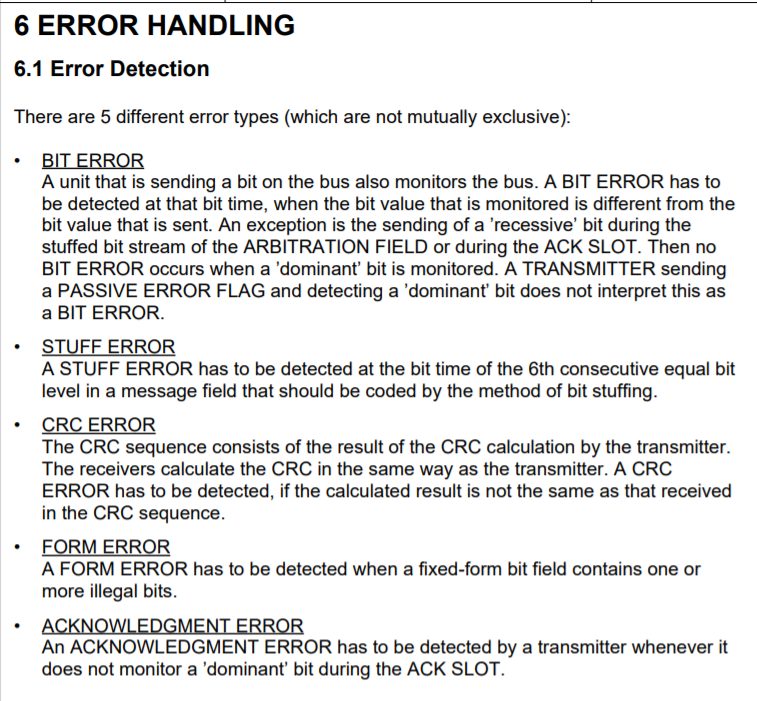
Một CAN messager có tối đa là 8bytes data, nếu muốn nhiều hơn thì ta cần tách messager

Ngoài ra còn có trường CRC, đây là kết quả tính CRC của message được truyền.

Cuối cùng của mỗi frame là một trường ACK, trường này để xác nhận tín hiệu đã được truyền hay chưa

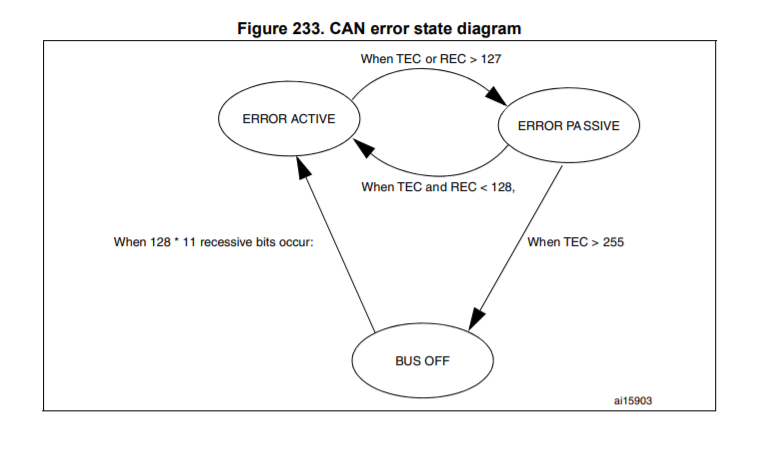
### Cơ chế Retransmit và Error Handler:

Có 5 loại lỗi của CAN bus (trích CAN bus specification)



Hình 2.5.8: Các lỗi có thể xảy ra trong giao tiếp CAN

Để Hander các lỗi này, CAN có diagram quản lý như sau:



Hình 2.5.9: Các mode lỗi của CAN

Một message khi được truyền mà không được ACK lại, số lượng lỗi sẽ được counter Transmit Error Count hoặc Receive Error count sẽ được cộng thêm vào, còn nếu truyền thành công counter sẽ trừ xuống. Nếu số count này lớn hơn 127 thì bus sẽ chuyển qua Error Passive, tức là chưa hỏng hoàn toàn, nếu như counter > 255 thì bus vào trạng thái Bus off tức hư hoàn toàn

### Độ ưu tiên của CAN ID:

Mỗi CAN messag có một CAN ID riêng, và với mỗi CAN Controller khi truyền thì sẽ so sánh mỗi giá trị của CAN ID bằng toán tử AND, trong đó bit 0 có giá trị cao hơn bit 1, CAN ID nào có giá trị cao hơn sẽ được ưu tiên truyền trước

## .Nội dung 6: Tìm hiểu giao thức CAN trên STM32

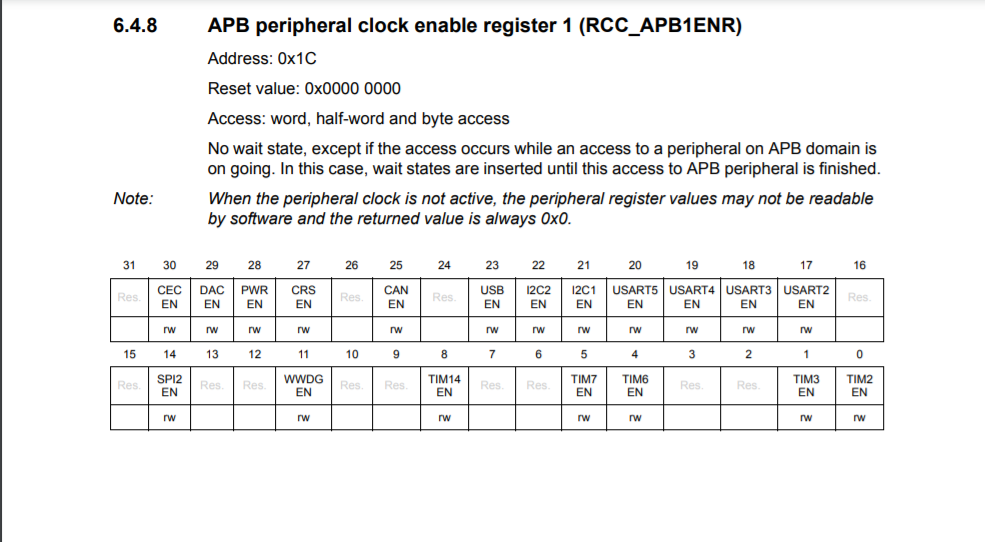
Phần này được tham khảo theo tài liệu STM32F1 reference manual [8]

Các dòng vi điều khiển STM32 của hãng STMicroelectronics hầu như đều được trang bị một CAN controller ở bên trong, CAN Controller này có tên mã là bxCAN, để sử dụng ta chỉ cần setup ngoại vi cho vi điều khiển và sử dụng thêm CAN transceiver để truyền nhận data.

Trong đề tài này, em sử dụng STM32F042 và STM32F103 để truyền data CAN, và CAN transceiver là TJA1050 của NXP để thực hiện

Để bật ngoại vi CAN, ta cần quan tâm những vấn đề sau:

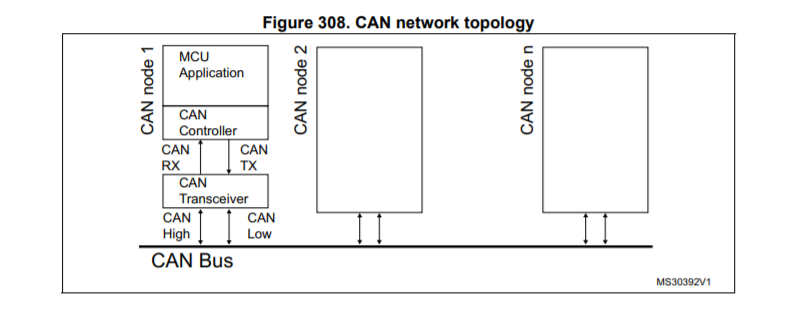
Thanh ghi RCC\_APB1ENR quản lý clock của CAN trên STM32F042



Hình 2.6.1: Thanh ghi quản lý clock của CAN

Để set up CAN bus, ta cần tính toán giá trị thanh ghi của ngoại vi bxCAN bus:

Ngoại vi này đóng vai trò quản lý và điều khiển các giao thức CAN trong mạng CAN

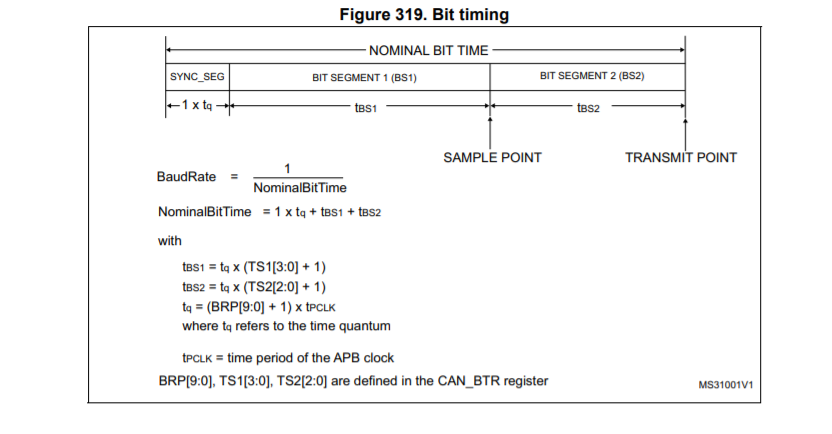


Hình 2.6.2: Mô hình kết nối CAN của STM32

Ở lớp Vật lý của ngoại vi CAN, ta có nhắc đến vấn đề ngoại vi CAN sử dụng mã NRZ, tuy nhiên để đọc được data cho đúng baudrate, ta cần phải cấu hình thời gian lấy mẫu sao cho đúng tùy thuộc vào tần số clock cấp cho ngoại vi CAN:

Với các chuẩn CAN phổ biến, thời gian lấy mẫu chiếm khoảng 50% - 90% thời gian của một bit được truyền:

Cơ cấu tính toán bit timing của CAN:

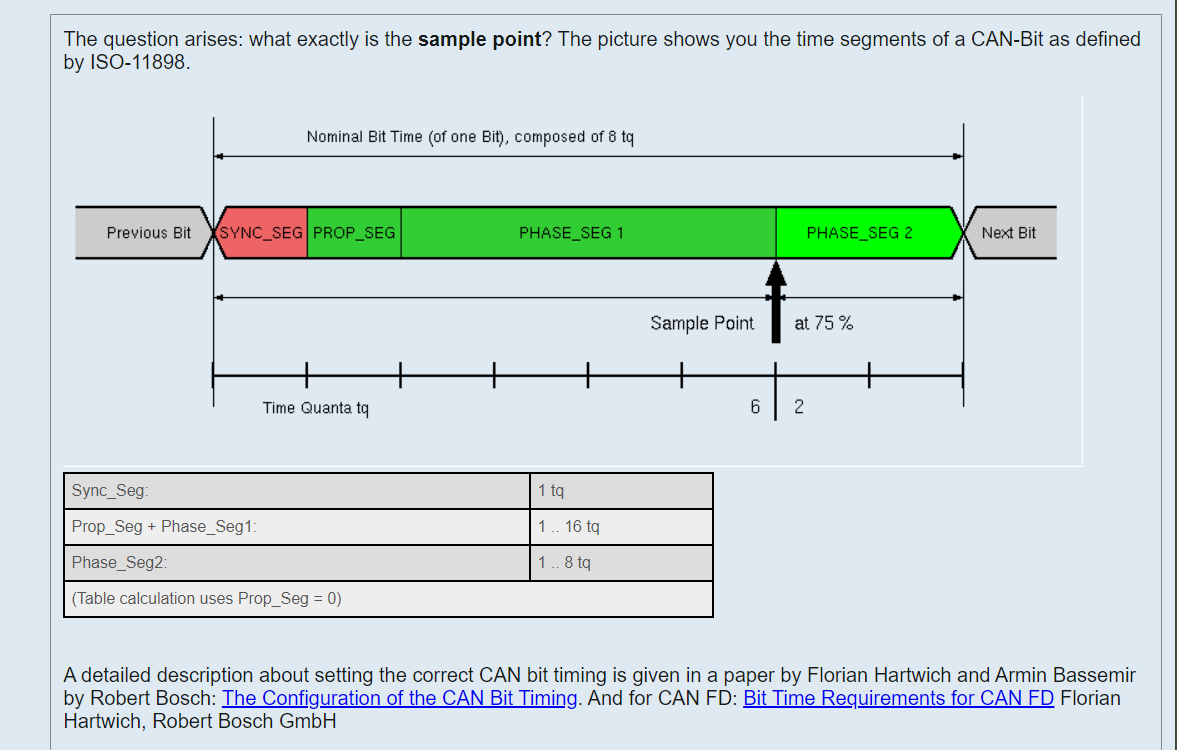


Hình 2.6.3: Tính toán bit timing với STM32

Trong đó tq là thời gian lượng tử hóa, có giá trị bằng tần số từ APB1/cho số prescaler được setup.

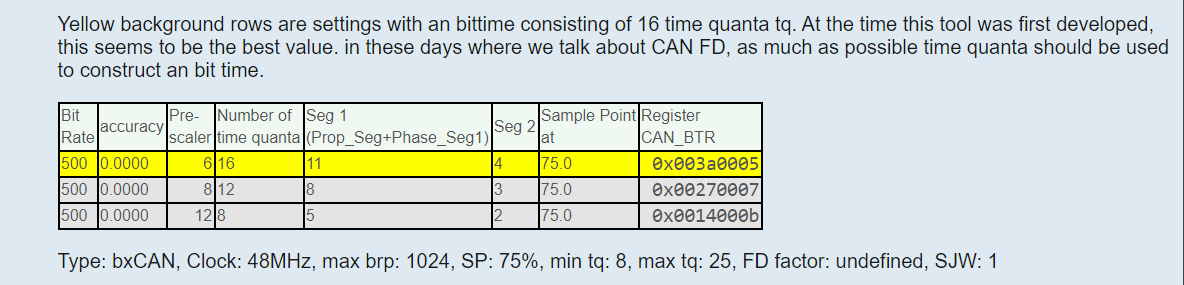
Ta cần chọn điểm sample point nằm trong khoảng từ tbs1/tbs2 = 75 – 90%

Sử dụng công cụ tính bit timing[9] để tính các thông số cho ngoại vi CAN trên STM32:



Hình 2.6.4: Mô tả bit timing của CAN

Giá trị tính ở time sample = 75%, fclk = 48Mhz, baudrate = 500kbit/s:



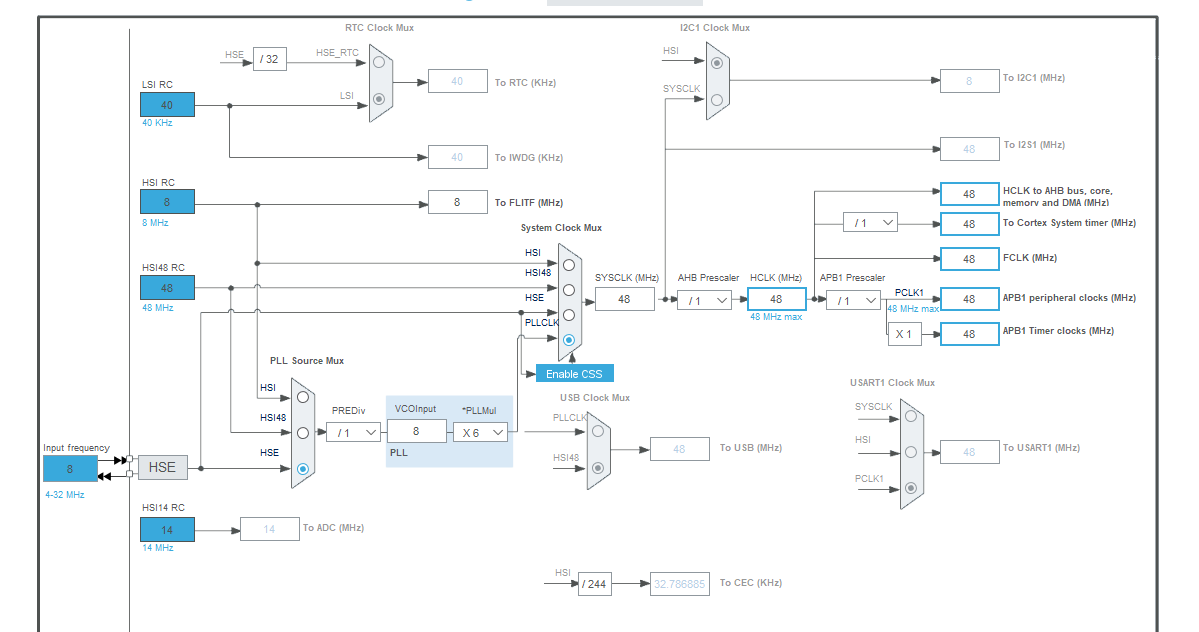
Hình 2.6.5: Giá trị thanh ghi bit timing với STM32

Chọn giá trị thứ 2 trong bảng

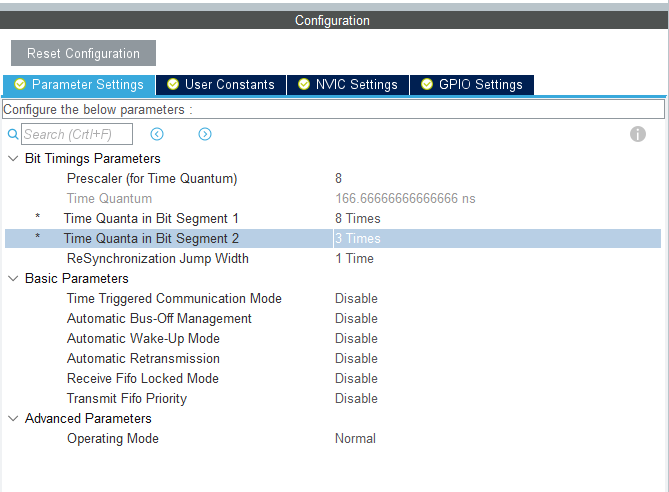
Như vậy tq = 1 / (48\*10^6/8) = 166.66666… ns

Bit time sẽ là: bt = tq \_+ 8\*tq + 3\*tq = 1999.9999… ns xấp xỉ 2us => baudrate = 1/bt = 500kbit/s

Setup trong STM32IDE:



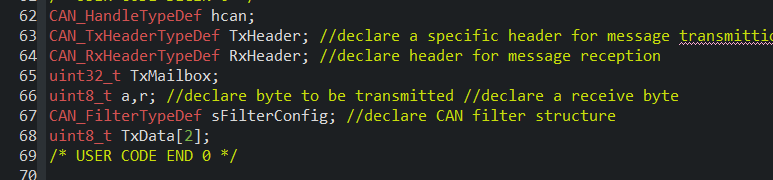
Hình 2.6.6: Setup clock STM32



Hình 2.6.7: Cấu hình ngoại vi CAN với STM32

Code giao tiếp CAN:

Khai báo các biến cần thiết:



Hình 2.6.8: Các biến cần thiết cho ngoại vi CAN STM32

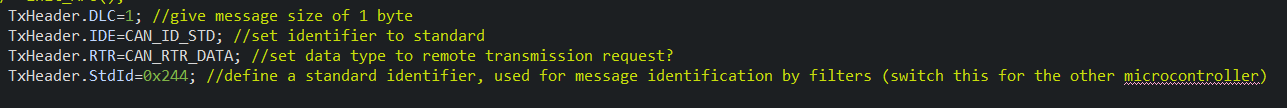
Setup các thành phần của CAN message:

+ Chiều dài data

+ Set Mode của Identifier

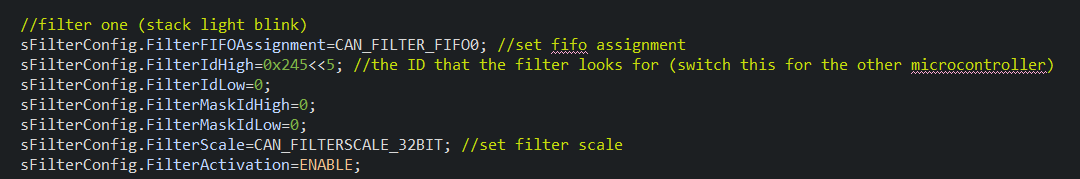
+ Set data có retransmit không

+ Set CAN ID của message



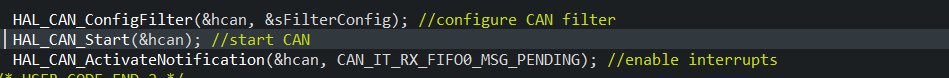
Hình 2.6.9: Header của message CAN

Setup filter lọc cho CAN controller:



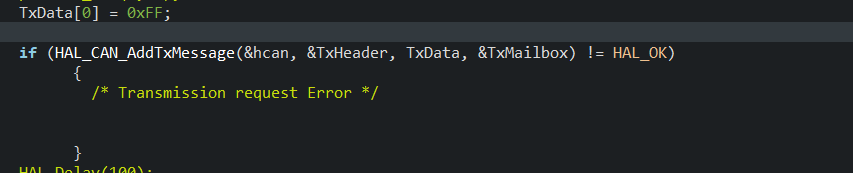
Hình 2.6.10: Config filter cho ngoại vi CAN

Setup filter và bật ngoại vi CAN ở mode interrupt khi có message nhận được:



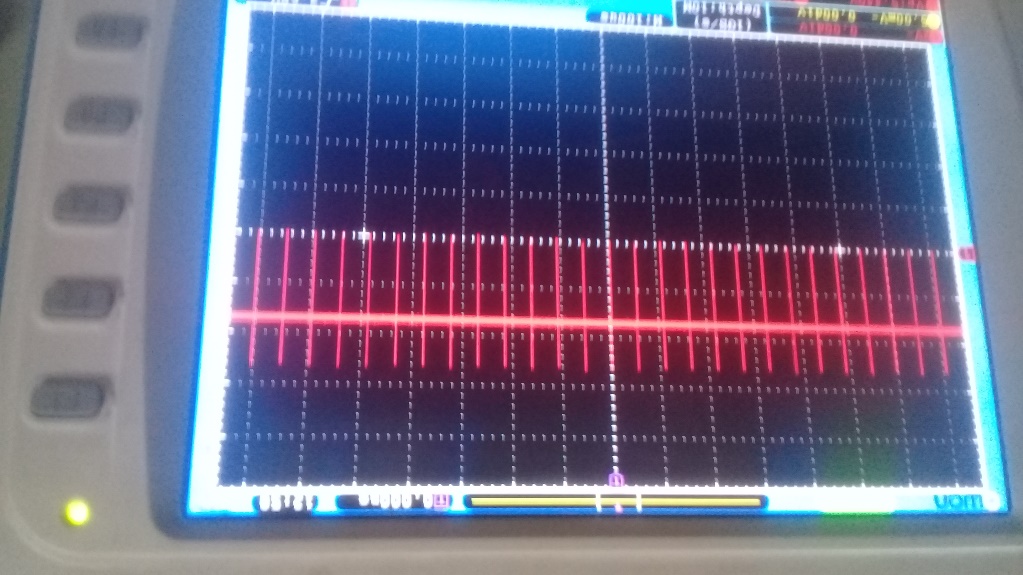
Hình 2.6.11: Bật ngoại vi CAN

Truyền data vào mail box để gửi:



Hình 2.6.12: Hàm truyền CAN message

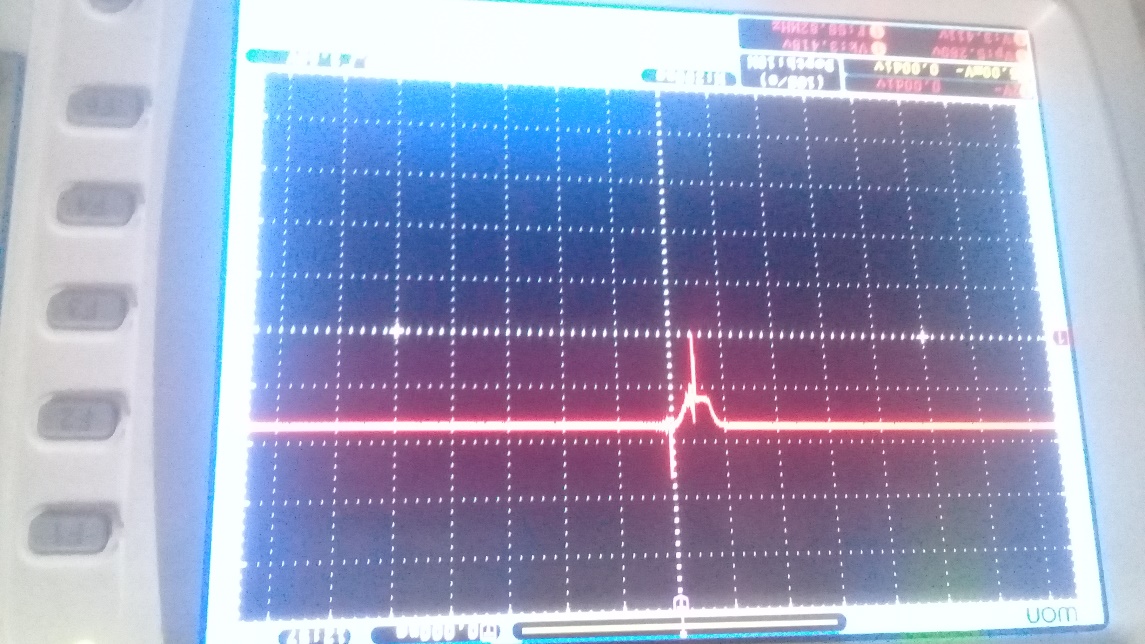
Kết quả: Đo trên Oscilloscope



Hình 2.6.13: Tín hiệu trên dây CAN\_H so với GND



Hình 2.6.14: Bit 0 trên dây CAN\_H



Hình 2.6.15: Bit 0 trên dây CAN\_L

Kết quả: Hiện tại đã giao tiếp được giữa 2 board STM32 bằng giao tiếp CAN, nhưng hiện tại trên công ty chưa thể mượn được mạch CANanalyzer và Lab để làm thí nghiệm nên chưa chắc chắn là tín hiệu này có thể đưa vào ABS

## Nội dung 7: Vẽ mạch test CAN cho IMU

Trong một bộ ABS, cảm biến tốc độ bánh xe chỉ là một trong nhiều cảm biến có thể sử dụng của bộ ABS. Một cảm biến khác đó là cảm biến Inerial Measurement Unit, cảm biến này có chức năng đo đạc gia tốc, vận tốc góc của xe.

Với Task này, em được giao cho việc tạo ra một IMU có tính năng gần tương tự như IMU có trên thị trường để test:

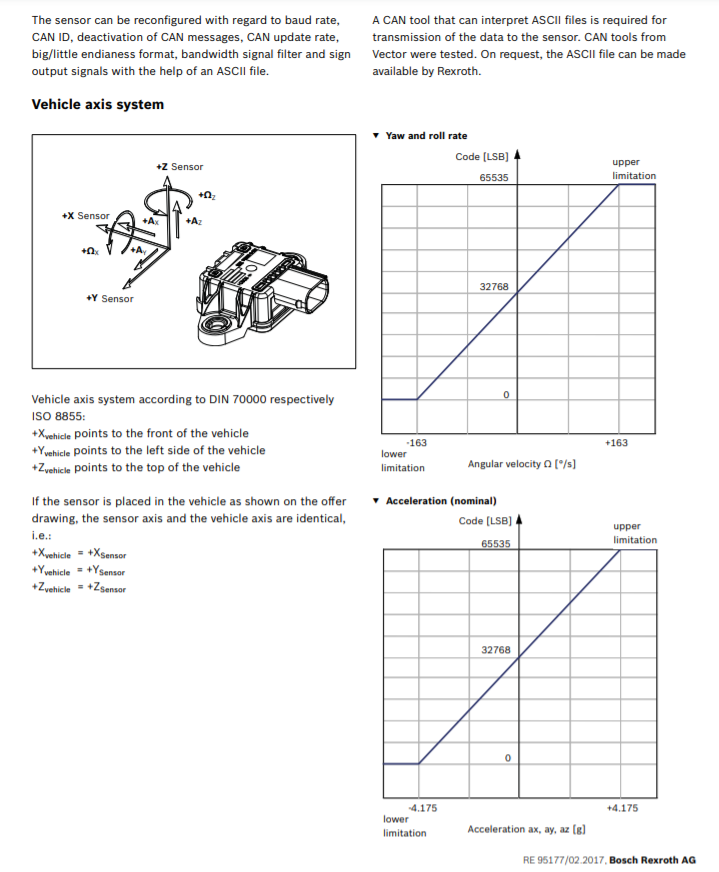
Loại IMU thực hiện mô phỏng:[10]

Hình 2.7.1: Cảm biến MM5.10

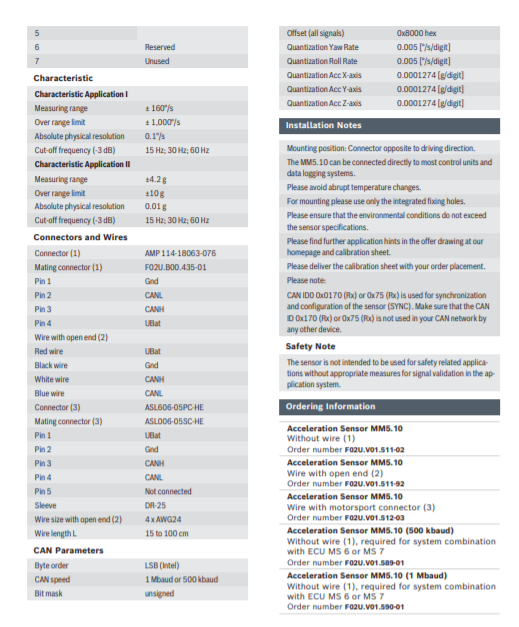
MM5.10 là loại IMU cho xe máy do Bosch sản xuất, là loại IMU có 5 trục, 3 trục gia tốc và 2 trục góc quay (Gyro)

Tầm đo của Accelerometer là từ -4.175g cho đến 4.175g

Tầm đo của Gyroscope là từ -163 độ/s đến 163 độ/s



Hình 2.7.2: Thông số data output của MM5.10



Hình 2.7.3: Thông số resolution của MM5.10

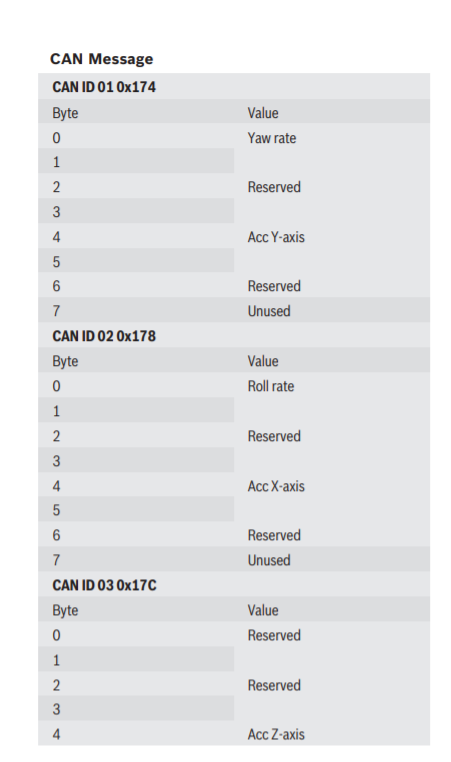
Cảm biến này có độ nhạy là 0.01g và 0.1 độ/s

So sánh với cảm biến MPU9250 rất phổ biến trên thị trường:[11]



Hình 2.7.4: Thông số resolution của MPU9250

Em nhận thấy cảm biến MPU9250 thậm chí có thông số tốt hơn cảm biến được sử dụng trên xe, do MPU9250 mới hơn, tuy nhiên chắc chắn MPU9250 không thể thay thế được MM5.10 do MM5.10 đạt chuẩn automotive.

Tổ chức message của cảm biến như sau: Gia tốc có 16bit data, chia làm 2 byte trong CAN message, tương tự như thế là tốc độ góc của Gyroscope: [12]

Hình 2.7.5: CAN ID của các message của MM5.10

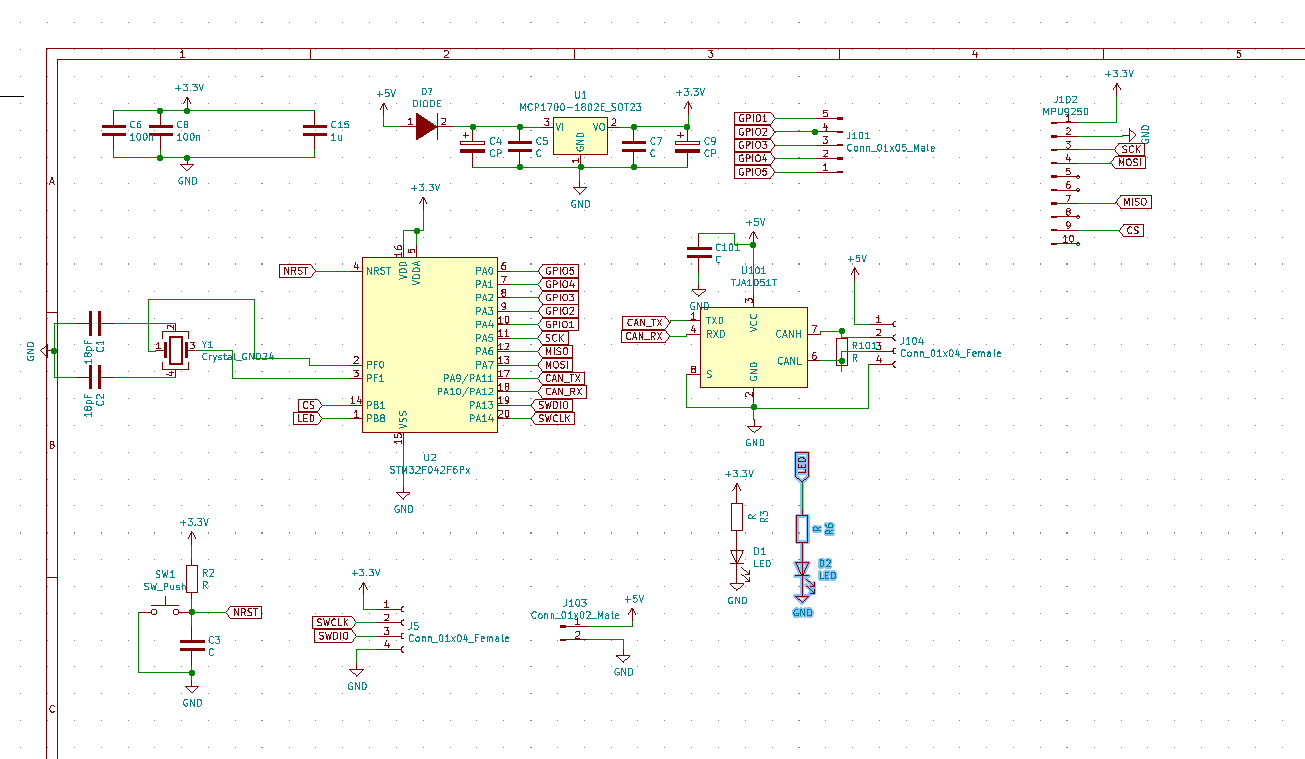
Như vậy phần cứng em thực hiện mô phỏng gồm:

+ MCU: STM32F042F6P6

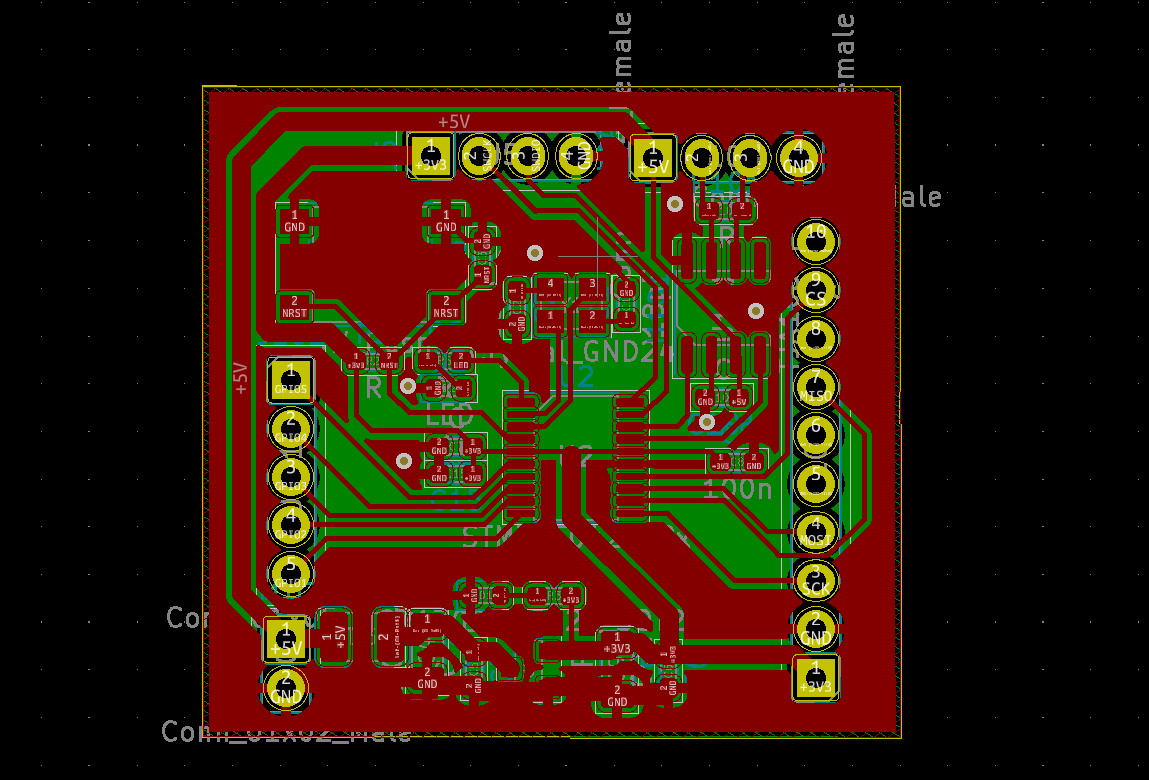
+ IMU: MPU9250 module

+ CAN transceiver: TJA1050:

Schematic mạch:

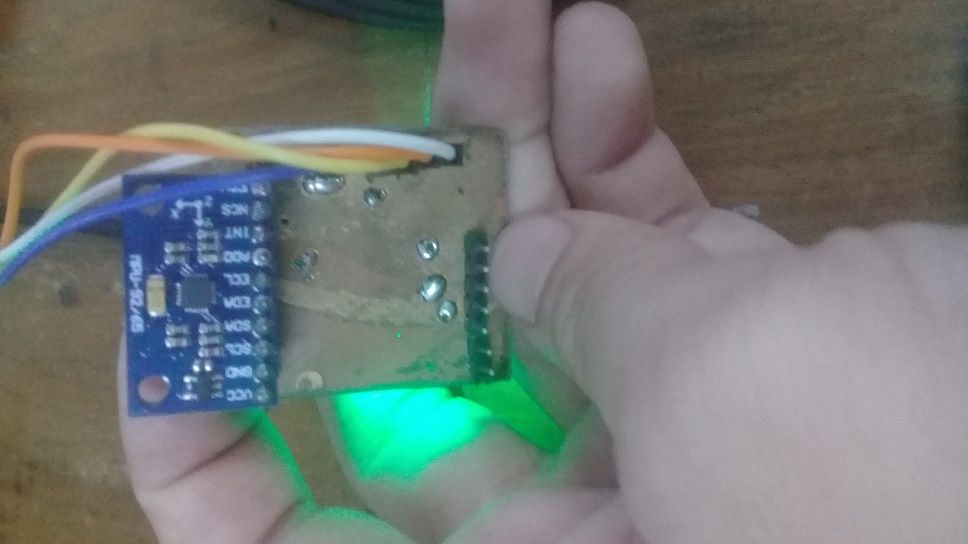
Mạch sử dụng kết nối SPI để kết nối giữa STM và IMU

Hình 2.7.6: Schematic của mạch mô phỏng

 Layout mạch:

Hình 2.7.7: Layout của mạch mô phỏng

Mạch thực tế:



Hình 2.7.8: Mạch thực tế

**Kết quả:** mạch đọc được tín hiệu từ cảm biến, thực hiện mô phỏng được các tín hiệu CAN signal như MM5.10, tuy nhiên do chưa được sử dụng lab nên chưa thể test với ABS được.

# TỔNG KẾT CÔNG VIỆC THỰC TẬP

## Kết quả công việc thực tập

Sau gần 9 tuần thực tập tại công ty, mặc dù đôi khi vẫn có gặp khó khăn (trong cách tiếp cận vấn đề mới, một số công cụ sử dụng, …) nhưng với sự giúp đỡ rất nhiệt tình của các thành viên trong team, là động lực to lớn để bản thân hoàn thành tốt nhất có thể những công việc được giao đúng hạn ở mục 1.2.

## Kinh nghiệm học được sau khi thực tập

Sau khoảng thời gian thực tập tại công ty bản thân đã học hỏi được rất nhiều kinh nghiệm từ các anh/chị đi trước. Những bài học, kinh nghiệm rất hữu ích cho công việc sau này.

* Kĩ năng mềm: kĩ năng làm việc nhóm, kĩ năng thuyết trình, kĩ năng ứng xử trong công ty, kĩ năng giải quyết vấn đề, v.v.
* Hiểu thêm về các hệ thống trong xe: ABS, ESP, TCS, v.v.
* Rèn luyện thêm kĩ năng lập trình, kĩ năng tiếng Anh,v.v.
* Hiểu thêm về các quy tắc sử dụng phòng thí nghiệm, internet, v.v; từ các chứng chỉ ESD (ElectroStatic Discharge), LABCAR (laboratory car), WBT (Web base training),v.v.
* Hiểu thêm cấu trúc, tổ chức quản lý của một nhóm (team).

## Kế hoạch nghề nghiệp trong tương lai:

Qua khoảng thời gian 3 tháng thực tập tại công ty, em đã có định hướng kế hoạch nghề nghiệp cho bản thân trong tương lai:

**Mục tiêu ngắn hạn:**

* Hoàn thành xong chương trình Đại Học và cố gắng học hỏi, trau dồi thêm nhiều kiến thức từ trên giảng đường.
* Rèn luyện thêm kỹ năng tiếng Anh.
* Rèn luyện thêm các kỹ năng mềm: kỹ năng thuyết trình trước đám đông, kỹ năng giao tiếp, ứng xử trong công ty, kỹ năng làm việc nhóm, kỹ năng giải quyết vấn đề,v.v.
* Học hỏi thêm nhiều kinh nghiệm, kiến thức trong quá trình thực tập

**Mục tiêu dài hạn:**

* Trong khoảng thời gian thực tập bản thân đã được trải nghiệm môi trường làm việc thực tế ở công ty và cảm nhận được RBVH chính là môi trường rất tốt để bản thân tham gia trau dồi học hỏi, làm việc và cống hiến sau này (nếu có cơ hội trong tương lai).
* Dự định sau tốt nghiệp Đại Học sẽ tham gia phỏng vấn vào công ty RBVH (team 2WPs) trong một vị trí mới.

# 

# TÀI LIỆU THAM KHẢO

1. <https://en.wikipedia.org/wiki/Anti-lock_braking_system>
2. <https://www.bosch-mobility-solutions.com/en/products-and-services/two-wheeler-and-powersports/riding-safety-systems/motorcycle-abs/>
3. <https://xlrd.readthedocs.io/en/latest/>
4. <https://docs.python.org/3/library/re.html>
5. <https://www.semiconvn.com/home/hoc-thiet-ke-vi-mach/bai-hc-vi-mch/12350-controller-area-network-can-hay-can-bus-la-gi.html>
6. <https://www.ti.com/lit/an/slla270/slla270.pdf?ts=1600764162061&ref_url=https%253A%252F%252Fwww.google.com%252F>
7. <http://esd.cs.ucr.edu/webres/can20.pdf>
8. <https://www.st.com/resource/en/reference_manual/cd00171190-stm32f101xx-stm32f102xx-stm32f103xx-stm32f105xx-and-stm32f107xx-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf>
9. <http://www.bittiming.can-wiki.info/>

[10]<https://dc-br.resource.bosch.com/media/us/products_13/product_groups_1/mobile_hydraulics_4/pdfs_6/re95177.pdf>

[11] <https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf>

[12] http://www.bosch-motorsport.de/content/downloads/Raceparts/Resources/pdf/Data%20Sheet\_68922379\_Acceleration\_Sensor\_MM5.10-R.pdf