{Trang trắng này dùng để dán bản Nhận xét của người hướng dẫn, hoặc thay trang  
này bằng Nhận xét của người hướng dẫn}

{Trang trắng này dùng để dán bản Nhận xét của người phản biện, hoặc thay trang này bằng Nhận xét của người phản biện}

# **TÓM TẮT**

Tên đề tài: THIẾT KẾ PHẦN MỀM MÔ PHỎNG CAN

Sinh viên thực hiện: NGUYỄN ĐỨC TUẤN

Số thẻ sinh viên: 106120145 Lớp: 12DT3

# **NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

# **LỜI NÓI ĐẦU VÀ CẢM ƠN**

Trong thời gian hơn 4 tháng thực hiện đồ án tốt nghiệp dưới hình thức Capstone Project tại FPT Software, em đã được tiếp xúc, làm quen với môi trường làm việc tại công ty. Học hỏi được nhiều kiến thức bổ ích, cũng như tích lũy thêm kinh nghiệm để chuẩn bị những hành trang cho bản thân nhằm đáp ứng nhu cầu tuyển dụng của các doanh nghiệp sau khi ra trường.

Em xin chân thành cảm ơn FPT Software và lãnh đạo Khoa Điện tử - Viễn thông đã tạo điều kiện để em được tham gia và thực hiện đồ án . Em xin chân thành cảm ơn giảng viên hướng dẫn - cô Trần Thị Hương và hơn hết là anh Nguyễn Văn Thành – FPT Software đã quan tâm trong suốt quá trình thực hiện, hướng dẫn nhiệt tình để em có thể hoàn thành được đồ án này.

# **LỜI CAM ĐOAN**

Kính gửi: Hội đồng bảo vệ đồ án tốt nghiệp khoa Điện tử - Viễn thông, Trường Đại học Bách Khoa-Đại học Đà Nẵng

Em tên là: Nguyễn Đức Tuấn

Sinh viên lớp 12DT3, Khoa Điện tử -Viễn thông, Trường Đại học Bách Khoa –Đại học Đà Nẵng.

Em xin cam đoan nội dung của Đồ án này không phải là bản sao chép của bất cứ Đồ án hoặc Công trình đã có từ trước. Nếu vi phạm em xin chịu mọi hình thức kỷ luật của Khoa.

Đà Nẵng, tháng 05 năm 2017

Sinh viên thực hiện

Nguyễn Đức Tuấn

# **MỤC LỤC**

[**TÓM TẮT**](#_Toc482643425)

[**NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**](#_Toc482643426)

[**LỜI CẢM ƠN** i](#_Toc482643427)

[**LỜI CAM ĐOAN** ii](#_Toc482643428)

[**MỤC LỤC** iii](#_Toc482643429)

[**DANH MỤC CÁC BẢNG, HÌNH VẼ** vi](#_Toc482643430)

[**DANH MỤC CÁC TỪ VIẾT TẮT** vii](#_Toc482643431)

[**MỞ ĐẦU** 1](#_Toc482643432)

[**CHƯƠNG 1: TỔNG QUAN HỆ THỐNG MÔ PHỎNG CAN BUS** 2](#_Toc482643433)

[**1.1.** **Giới thiệu chương** 2](#_Toc482643434)

[**1.2.** **Tổng quan về CAN BUS** 2](#_Toc482643435)

[**1.3.** **Hệ thống mô phỏng CAN BUS** 4](#_Toc482643436)

[**1.4.** **Các thành phần của hệ thống.** 5](#_Toc482643437)

[1.4.1. CAN Software 6](#_Toc482643438)

[1.4.1.1. Phần mềm mô phỏng CAN. 6](#_Toc482643439)

[1.4.1.2. Phần mềm cơ sở dữ liệu CAN. 7](#_Toc482643440)

[1.4.2. CAN Device 8](#_Toc482643441)

[1.4.3. PC Driver và Protocol 9](#_Toc482643442)

[**1.5.** **Kết luận chương** (nhắc lại phạm vi đề tài lần nữa). 9](#_Toc482643443)

[**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ MÔI TRƯỜNG THỰC HIỆN** 10](#_Toc482643444)

[**2.1.** **Giới thiệu chương.** 10](#_Toc482643445)

[**2.2.** **Công cụ thực hiện.** 10](#_Toc482643446)

[2.2.1. Môi trường hệ điều hành Linux. 10](#_Toc482643447)

[2.2.2. Giới thiệu chung về Qt Creater Framework. 11](#_Toc482643448)

[2.2.3. Các ưu điểm của QT Framework. 12](#_Toc482643449)

[**2.3.** **Phần mềm mô phỏng CAN.** 12](#_Toc482643450)

[2.3.1. Tổng quan phần mềm. 12](#_Toc482643451)

[2.3.2. Tiến trình thực hiện. 13](#_Toc482643452)

[2.3.3. Thiết kế giao diện. 13](#_Toc482643453)

[2.3.4. Giao tiếp các khối (Parent – Child). 18](#_Toc482643454)

[2.3.5. Các nhánh trên giao diện phần mềm. 19](#_Toc482643455)

[2.3.5.1. Nhánh Generator. 19](#_Toc482643456)

[2.3.5.2. Nhánh Hardware/Main. 19](#_Toc482643457)

[2.3.5.3. Nhánh View. 20](#_Toc482643458)

[2.3.6. Quản lý dữ liệu. 21](#_Toc482643459)

[2.3.6.1. Luồng dữ liệu hệ thống. 21](#_Toc482643460)

[2.3.6.2. Luồng dữ liệu Interative Generator Block. 22](#_Toc482643461)

[2.3.6.3. Luồng dữ liệu Filter Block. 22](#_Toc482643462)

[2.3.6.4. Luồng dữ liệu Hardware Block. 23](#_Toc482643463)

[2.3.6.5. Luồng dữ liệu Trace Block 24](#_Toc482643464)

[**2.4.** **Cấu hình dữ liệu.** 24](#_Toc482643465)

[2.4.1. Giới thiệu về Json. 24](#_Toc482643466)

[2.4.2. QJson trong QT framework. 24](#_Toc482643467)

[**2.5.** **Kết luận chương.** 26](#_Toc482643468)

[**CHƯƠNG 3:**  **KẾT QUẢ THỰC HIỆN VÀ ĐÁNH GIÁ** 27](#_Toc482643469)

[**3.1.** **Giới thiệu chương.** 27](#_Toc482643470)

[**3.2. Bảng phân công công việc.** 27](#_Toc482643471)

[3.3. Giao diện Simulator software. 28](#_Toc482643472)

[3.3.1. Giao diện các hộp thoại. 28](#_Toc482643473)

[3.3.2. Giao diện khối IG 28](#_Toc482643474)

[3.3.3. Giao diện khối Filter 29](#_Toc482643475)

[3.3.4. Giao diện khối Hardware 31](#_Toc482643476)

[3.3.5. Giao diện khối Trace 32](#_Toc482643477)

[3.4. Edit fie Cấu hình theo định dạng json 33](#_Toc482643478)

[3.5. Hoạt động phần mềm mô phỏng CAN. (trang 41 - 45) 34](#_Toc482643479)

[3.6. Kết quả nghiệm thu.(trang 44- 50) 38](#_Toc482643480)

[3.7. Kết luận chương. 42](#_Toc482643481)

[**KẾT LUẬN** 43](#_Toc482643482)

[**HƯỚNG PHÁT TRIỂN ĐỀ TÀI** 44](#_Toc482643483)

[**TÀI LIỆU THAM KHẢO** 45](#_Toc482643484)

[**PHỤ LỤC 1** 1](#_Toc482643485)

[**PHỤ LỤC 2** 2](#_Toc482643486)

# **DANH MỤC CÁC BẢNG, HÌNH VẼ**

# **DANH MỤC CÁC TỪ VIẾT TẮT**

# **MỞ ĐẦU**

Cùng với sự phát triển không ngừng của khoa học công nghệ, công nghiệp ôtô cũng đã và đang được hoàn thiện nhằm mục đích đem đến cho người dùng những trải nghiệm ngày càng thoải mái và tiện nghi hơn. Sự ra đời của các công nghệ, thiết bị hiện đại như cảnh báo chệch làn đường, cảnh báo tiền va chạm, hệ thống hỗ trợ phanh và điều khiển động cơ bánh lái…đã mang lại cho người dùng những chuyến hành trình ngày càng thú vị. Tuy nhiên, để áp dụng những công nghệ mới này lên trên một sản phẩm xe ô tô thực tế để đưa ra ngoài thị trường thì từ khâu kiểm thử, vận hành cho đến phát triển các ứng dụng về sau thì đây cũng là một khó khăn đối với các nhà nghiên cứu, cũng như sản xuất ô tô công nghiệp. Việc sử dụng cả hệ thống thực tế chỉ để kiểm thử với một ECU (Electronic Control Unit) mới nhằm hướng tới sự tương thích giữa các ECU với nhau có thể gây ra sự phức tạp cho toàn bộ hệ thống CAN BUS (Controller Area Network).

Chính vì vậy, việc xây dựng nên một phần mềm có thể mô phỏng hoạt động như một hệ thống hoàn chỉnh nhằm thay thế các ECU thực tế là thật sự cần thiết. Nó sẽ giúp cho việc phát triển thêm ứng dụng đơn giản hơn, với độ tin cậy và hiệu quả cao.

Hiểu được tầm quan trọng của nó, nhóm chúng em đã đăng ký tham gia Đồ án tốt nghiệp dưới hình thức Capstone Project cùng FPT Software với đề tài “THIẾT KẾ VÀ THI CÔNG HỆ THỐNG MÔ PHỎNG CAN BUS”. Hệ thống bao gồm: Phần mềm mô phỏng CAN, phần mềm cơ sở dữ liệu CAN, CAN Device và PC Driver.

Phạm vi của em thực hiện trong đề tài là “THIẾT KẾ PHẦN MÊM MÔ PHỎNG CAN”.

Nội dung báo cáo gồm các chương:

Chương 1: Tổng quan hệ thống mô phỏng CAN Bus

Chương 2: Cơ sở lý thuyết và môi trường thực hiện

Chương 3: Kết quả thực hiện và đánh giá

Trong quá trình thực hiện đồ án, em đã cố gắng rất nhiều song không khỏi mắc những sai sót, kính mong quý thầy cô thông cảm và đóng góp ý kiến để đồ án được hoàn thiện hơn.

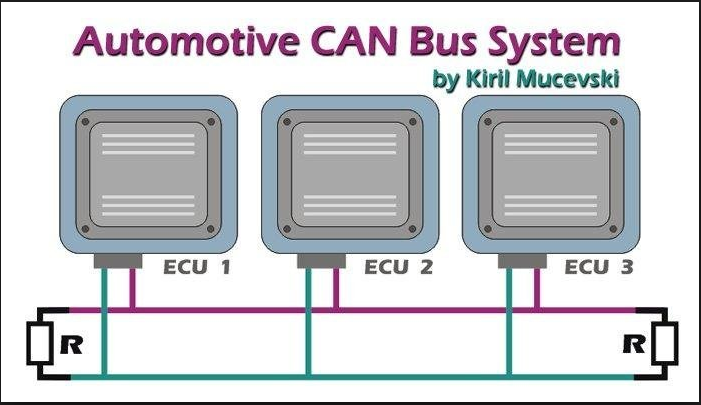
# **CHƯƠNG 1: TỔNG QUAN HỆ THỐNG MÔ PHỎNG CAN BUS**

# **Giới thiệu chương**

Chương này sẽ giới thiệu về đường truyền CAN và hệ thống mô phỏng CAN BUS, bao gồm: Phần mềm mô phỏng CAN, phần mềm cơ sở dữ liệu CAN, CAN Device và PC Driver.

# **Tổng quan về CAN BUS**

Mạng khu vực điều khiển CAN – Controller Area Network (hay còn gọi là CAN BUS) là mạng lưới đường truyền thông tin các bộ điều khiển độc lập dưới dạng đường truyền BUS. Trong đó các thành phần nối với mạng có quyền ngang nhau trong việc truyền và nhận thông tin (multi master), bất kỳ các thành phần đó đều có thể truyền cũng như nhận thông tin mà chúng cần từ các thành phần khác. Vì vậy, đây là một giao thức truyền thông nối tiếp có hỗ trợ hiệu quả phân phối thời gian thực kiểm soát với mức độ bảo mật rất cao.



*Hình 1.1 : Sơ đồ khối mô hình kết nối CAN BUS*

Tiêu chuẩn Bus CAN được phát triển bởi Bosch (một nhà sản xuất thiết bị điện tại Đức) và Intel vào đầu những năm 1980. Sau đó, CAN đã được chuẩn hóa theo tiêu chuẩn ISO-11898 và ISO-11519, thiết lập chính nó như là giao thức chuẩn để giao tiếp kết nối mạng trong ngành công nghiệp ô tô. Trong những ngày đầu của ngành công nghiệp ô tô, bộ điều khiển độc lập cục bộ đã được sử dụng để quản lý cơ cấu truyền động khác nhau và các các hệ thống cơ điện phụ. Bởi kết nối mạng các thiết bị điện tử trong xe theo chuẩn CAN, và được được điều khiển từ một điểm trung tâm, bộ điều khiển động cơ (ECU), do đó làm tăng chức năng, thêm mô đun, và làm cho quá trình chẩn đoán hiệu quả hơn.

Thời gian đầu giao tiếp CAN được phát triển chủ yếu hỗ trợ cho ngành công nghiệp xe, vì vậy nó đã được sử dụng trong xe ô tô chở khách, tàu thuyền, xe tải, và trên nhiều loại xe khác. Ngày nay giao thức CAN  đang được sử dụng trong nhiều lĩnh vực khác nhau trong các ứng dụng với tên gọi mạng điều khiển nhúng (Networked Embedded Control), bao gồm cả tự động hóa công nghiệp, ứng dụng y tế, xây dựng tự động hóa, máy dệt, máy móc sản xuất. CAN cung cấp một giao thức truyền thông hiệu quả giữa các bộ cảm biến, cơ cấu truyền động, điều khiển, và các nút khác trong các ứng dụng thời gian thực, và được biết đến với sự đơn giản, độ tin cậy và hiệu suất cao.



*Hình 1.2 : BUS CAN với 3 node*

Hình trên cho thấy một bus CAN với ba node. Các giao thức CAN được dựa trên giao thức CSMA / CD+AMP (Carrier-Sense Multiple Access:  Đa truy cập có tránh xung đột / Collision Detection Arbitration on Message Priority: Giám sát phát hiện xung đột dựa trên thông điệp ưu tiên), nó tương tự như các giao thức được sử dụng trong mạng Ethernet LAN. Khi mạng Ethernet phát hiện một vụ xung đột, các nút đang gửi dữ liệu chỉ đơn giản là dừng truyền và đợi một khoảng thời gian ngẫu nhiên trước khi cố gắng để gửi một lần nữa. Tuy nhiên, đối với giao thức CAN có một chúc khác biệt đó là các vấn đề xung đột được giải quyết  bằng cách sử dụng các nguyên tắc của trọng tài, nơi chỉ có các nút ưu tiên cao nhất được trao quyền để gửi dữ liệu của nó.

**Một số các tính năng giao thức CAN là:**

* CAN bus là đa chủ. Khi bus dữ liệu rỗi, bất kỳ thiết bị gắn vào bus đều có thể bắt đầu gửi một thông điệp.
* CAN giao thức bus là linh hoạt. Các thiết bị kết nối với bus không có địa chỉ, có nghĩa là tin nhắn không được truyền đi từ một nút này đến một nút khác dựa trên các địa chỉ. Thay vào đó, tất cả các nút trong hệ thống nhận mọi tin nhắn truyền đi trên bus, và nó là đến mỗi nút để quyết định xem đã nhận tin nhắn sẽ được lưu giữ hay loại bỏ. Một tin nhắn duy nhất có thể được dành cho một nút cụ thể hoặc cho nhiều nút, tùy thuộc vào cách hệ thống được thiết kế.
* Một lợi thế của việc không có địa chỉ là khi một thiết bị được thêm vào hay lấy đi khỏi bus, không cần phải thay đổi cấu hình dữ liệu  (tức là, các bus "có thể gắn nóng (hot pluggable)").
* Tốc độ giao tiếp bus CAN là không cố định. Bất kỳ tốc độ truyền thông có thể được thiết lập cho các thiết bị gắn vào một bus.
* Tất cả các thiết bị trên bus có thể phát hiện một lỗi. Các thiết bị đã phát hiện một lỗi ngay lập tức thông báo cho tất cả các thiết bị khác.
* Nhiều thiết bị có thể được kết nối với xe buýt tại cùng một thời điểm, và không có được giới hạn hợp lý về số lượng các thiết bị có thể được kết nối. Trong thực tế, số thiết bị có thể được gắn vào một bus bị hạn chế bởi thời gian trễ của bus và tải điện.

# **Hệ thống mô phỏng CAN BUS**

Hệ thống mô phỏng CAN Bus giao tiếp các ECU trong ô tô sử dụng đường truyền CAN bus, bao gồm CAN Device thực hiện hoạt động truyền nhận thông điệp tương tự như một ECU thực tế,PC Driver có nhiệm vụ giao tiếp giữa Hardware và Software và CAN Simulator là phần mềm mô phỏng, hiển thị định dạng của thông điệp, tín hiệu, đồng thời có thể cấu hình chọn lựa cổng CAN, tốc độ baud…với cửa sổ thống kê hiển thị tương tự như trên phần mềm CANoe Vector đã và đang được các nhà phát triển, sản xuất ô tô trên thế giới sử dụng hiện nay.



*Hình 1.2 : Sơ đồ khối mô tả giao tiếp hệ thống CAN BUS*

Để có thể kiểm tra được ECU có hoạt động đúng yêu cầu thiết kế hay không, ta sẽ xây dựng mô hình kiểm tra như trên , trong đó ý tưởng chính dự trên việc giả lập một hệ thống ô tô hoàn chỉnh trên máy tính và kết nối nó với ECU thực tế để kiểm tra bằng cách truyền các thông điệp yêu cầu ECU thực hiện và nhận lại các thông điệp phản hồi từ ECU, sau đó xử lý nội dung của phản hồi và hiển thị lên màn hình thống kê một cách trực quan với độ chính xác cao.

ECU cần kiểm tra sẽ được kết nối với CAN Device thông qua cổng CAN. CAN Device kết nối với Software, nơi chứa hệ thống giả lập gồm các ECU (Electronic Control Unit) mô phỏng hệ thống thực tế thông qua cổng USB nhờ PC driver .PC driver đóng vai trò cầu nối giao tiếp giữa hardward và software thông qua cơ chế truyền (ghi) và nhận (đọc) tin từ các file descriptor.



**Hình 1.3  Sơ đồ khối hệ thống CAN** BUS

Sau khi kết nối giữa ECU và Software , mô hình hệ thống sẽ gồm nhiều ECU được kết nối với nhau thông qua CAN BUS như là một hệ thống CAN thực tế hoàn chỉnh.

# **Các thành phần của hệ thống.**

Hệ thống mô phỏng CAN Bus bao gồm: Phần mềm mô phỏng CAN, phần mềm cơ sở dữ liệu CAN, CAN Device và PC Driver

# CAN Software

.

*Hình 1.3 CAN Software bao gồm: Database Edit Software và Simulator software.*

# Phần mềm mô phỏng CAN.

Là một phần mềm độc lập, có nhiệm vụ xử lí việc truyền nhận dữ liệu cũng như phân tích và hiển thị nội dung của các thông điệp, mà định dạng của các thông điệp đó được lấy từ D atabase edit software. Bao gồm các chức năng:

* Cấu hình hệ thống, cổng CAN nào đang hoạt động , tốc độ Baud…
* Load được database vào test model.
* Add/remove được sơ đồ khối kết nối đồng thời cập nhật cấu hình và lưu lại.
* Add/remove được các message vào trong IG. (interactive generator).
* Trong IG có thể setting cycle của từng message.
* Trace window hiển thị các thuộc tính của message: ID, length, direction, data, signal value.

# Phần mềm cơ sở dữ liệu CAN.

Database Edit Software là phần mềm độc lập quản lí cơ sở dữ liệu của hệ thống, lưu giữ các thông điệp (message) và các tín hiệu (signal), hỗ trợ các loại format Little Endian và Big Endian , MSB và LSB. Các chức năng chính của Database Software là: Tạo mới database và mở 1 database có sẵn.

Hổ trợ các chức năng:

- Thêm/sửa/xóa các thông điệp và tín hiệu trong các thông điệp.

- Thêm/sửa/xóa các nốt mạng (node network) và các biến môi trường EV (environment variable).

Vai trò chính của Database edit software là tạo dựng một cơ sở dữ liệu gồm các node network và các thông điệp gửi đi cũng như nhận về từ chính node đó để cung cấp dữ liệu cho việc mô phỏng ở simulator software.

# CAN Device

**CAN Device:** Chức năng chính của CAN Device là tạo một node CAN có thể tùy chỉnh được tốc độ của node để có thể kết nối được với một CAN BUS bất kỳ để lấy các CAN\_Frame trong CAN BUS và gởi lên PC.

Để thực hiện các chức năng đó chúng ta sử dụng giao thức USB và CAN. Board DK-TM4C123G hỗ trợ

# PC Driver và Protocol

**PC driver**: PC Driver có nhiệm vụ giao tiếp giữa hardware và software .Khi một device được kết nối với PC, driver nhận nhiệm vụ tạo ra vùng nhớ đệm thông qua các file descriptor. Khi ta muốn truyền một thông điệp từ PC xuống Hardware thì Driver sẽ nhận biết được thông điệp, sau đó ghi vào file descriptor tương ứng. Hardware sẽ đọc dữ liệu từ file descriptor và thực hiện nội dung thông điệp. Thông điệp phản hồi sẽ được truyền theo chiều ngược lại đến Software.

* Nhận biết được thiết bị connect, disconnect thông qua cổng USB, đưa ra thông tin, trạng thái của thiết bị.
* CAN software có thể đóng/ mở thiết bị thông qua device node.
* CAN software có thể cấu hình baurd rate thông qua file descriptor.
* CAN software có thể gửi/nhận message thông qua file descriptor

# **Kết luận chương** (nhắc lại phạm vi đề tài lần nữa).

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ MÔI TRƯỜNG THỰC HIỆN**

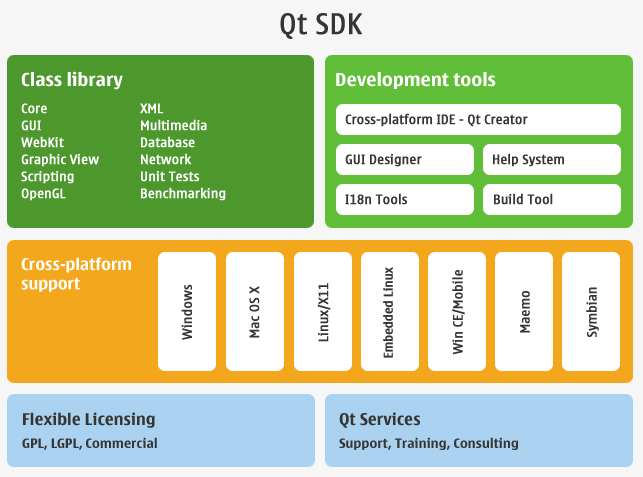
# **Giới thiệu chương.**

Thiết kế phần mềm mô phỏng CAN là nhiệm vụ chính của nhóm trong đề tài “THIẾT KẾ VÀ THI CÔNG HỆ THỐNG MÔ PHỎNG CAN BUS”.

# **Công cụ thực hiện.**

# Môi trường hệ điều hành Linux.

# Giới thiệu chung về Qt Creater Framework.



*Hình 2.1 Mô tả về QT FrameWork*

- Qt là một Application Framework , tạo ra một framework có khả năng thiết kế, tạo ra những phần mềm có thể chạy trên nhiều nền tảng khác ,hỗ trợ đa ngôn ngữ : C++ , Pascal, Python, Ruby….

**-** Hình ảnh trên mô tả đầy đủ về QT FrameWork :

* Cross Flatform: hỗ trợ chạy trên nhiều nền tảng như Windows, Mac , Linux,Maemo, Symbian,…
* Qt modular class library: Các thư viện thành phần cốt lõi cho Framework là Qt Core, QtGUI, Qt Widget, QT QML, Qt Quick, Qt Quick Controls , QT Quick Layouts, QT NetWork, ....
* Development tool : Qt Creator là công cụ dùng để thiết kế giao diện đồ họa GUI.

# 2.2.3. Các ưu điểm của QT Framework.

- Code 1 lần và chạy khắp mọi nơi: code trên Windows, chạy trên Linux, Mac, thiết bị di động, thiết bị nhúng...

- Các thư viện class trực quan, đơn giản và dễ sử dụng.

- Tạo ra code dễ đọc, dễ mở rộng và dễ tái sử dụng.

- Ứng dụng tạo ra có hiệu suất xử lý cao và chạy khá nhanh.

- OpenWebkit cho phép xây dựng và chạy các ứng dụng dựa trên nền web, ví dụ như game viết bằng Html5, Css, JavaScript.

* 1. **Phần mềm mô phỏng CAN.**

# Tổng quan phần mềm.

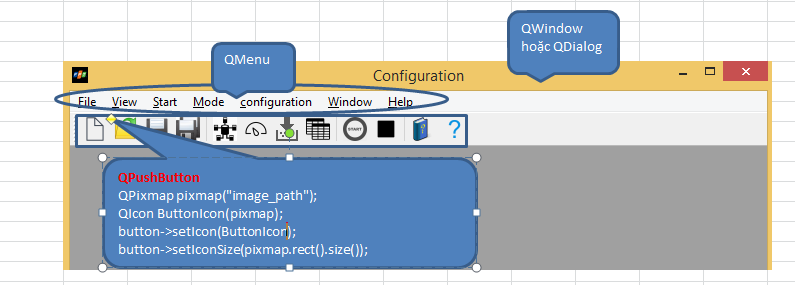
# Tiến trình thực hiện.

# Thiết kế giao diện.

Thiết kế giao diện CAN simulator sử dụng các thư viện:

**QWindow và QDialog**

* Cửa sổ chính cung cấp một khuôn khổ để xây dựng giao diện cho người dùng.
* QWindow, QDialog và các lớp liên quan của nó được sử dụng để quản lý cửa sổ chính.
* QWindow, QDialog có bố cục riêng mà bạn có thể thêm QToolBars, QMenuBar hay QStartusBar.



*Hình 2.2: Cửa sổ giao diện chính sử dụng QWindow và QDialog*

**QMenu**

* + Cung cấp một tiện ích các trình đơn con để sử dụng trong thanh Menu
  + Mỗi một tiện ích trình đơn con là một Menu được lựa chọn.
  + Các Menu được hiển thị trên thanh trình đơn khi người dùng nhấp vào mục tương ứng hoặc nhấn vào phím tắt đã chỉ định.



*Hình 2.2: Trình đơn được hiện ra khi rightclick vào một khối.*

**QPushbutton**

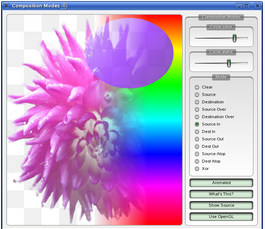
* Tiện ích QPushbutton cung cấp một nút lệnh.
* Click nút để chỉ lệnh cho máy tính thực thi một số hành động hoặc để trả lời câu hỏi.
* Một nút lệnh thường có hình chữ nhật và thường hiển thị một nhãn văn bản mô tả hành động của nó.
* Nút lệnh hiển thị văn bản, và tùy chọn các biểu tượng nhỏ bằng cách sử dụng setText() và setIcon().
* Nút lệnh phát ra Signal khi được kích hoạt bằng chuột hoặc phím tắt. kết nối với Signal này để thực hiện hành động của nút.



*Hình :các button được khởi tạo khi gọi lớp QPushbutton*

**QPainter**

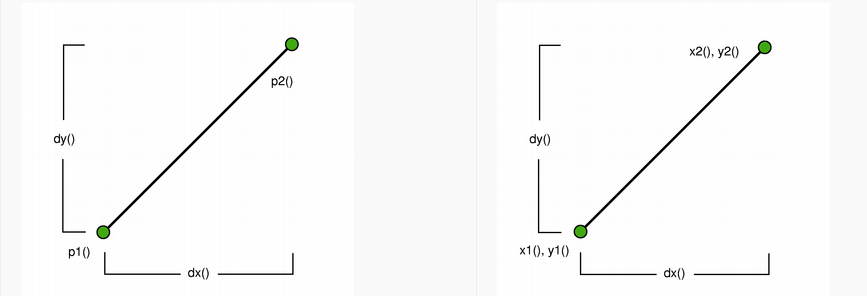
* Lớp QPaint thực hiện việc vẽ ở mực low-level trên các widgets và các devices khác. Nó cung cấp các chức năng tối ưu hóa cao để thực hiện hầu hết các chương trình vẽ GUI.
* Chức năng cốt lõi của lớp này là vẽ nhưng lớp này cung cấp một số chức năng cho phép bạn tùy chỉnh các thiết lập và chất lượng của nó. Ngoài ra bạn còn có thể kiểm soát các hình dạng khác nhau được hợp nhất với nhau như thế nào….
* NOTE: khí Paint deviece là một widget, QPainter chỉ có thể sử dụng bên trong một hàm paintEvent(), hoặc trong một hàm gọi paintEvent().



*Hình 2.2: Hình màu được tạ ra khi gọi lớp QPainter*

**QLine:**

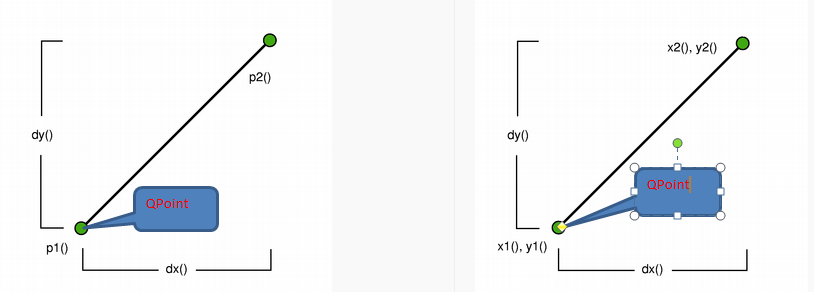
* Lớp QLine cung cấp một vector 2 chiều sử dụng độ chính xác của số nguyên.
* Một QLine mô tả một đường dài hữu hạn (hoặc một đoạn thẳng). Các điểm bắt đầu và kết thúc của đường được xác định bằng cách sử dụng tọa độ.
* Các vị trí của điểm bắt đầu và kết thúc của đường kẻ có thể được lấy bằng các hàm p1 (), x1 (), y1 (), p2 (), x2 () và y2 (). Các hàm dx () và dy () trả về các thành phần nằm ngang và dọc của đường kẻ.
* Các line có thể dịch một đoạn tùy chỉnh bằng cách sử dụng hàm translate().



*Hình : Line được tạo ra khi kết nối các điểm đầu cuối.*

**QPoint**

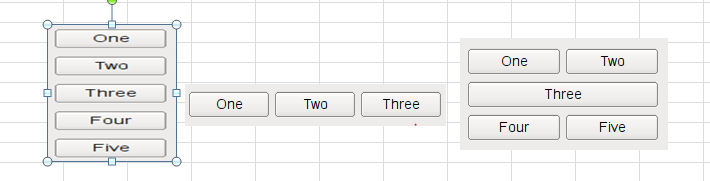
* Lớp QPoint trả về tọa độ của một điểm trên mặt phẳng.
* Một điểm được xác định bởi một tọa độ x và một tọa độ y. các tọa độ có thể được đặt hay thay đổi bằng các hàm setX() và setY(), hoặc các hàm rx() và ry() trả về tham chiếu đến tọa độ.
* Đối tượng Qpoint cũng được sử dụng như một vector.



*Hình :tọa độ điểm đầu và cuối của Line trên mặt phẳng trả về lớp QPoint*

**QVBoxlayout, QHBoxLayout, QGridLayout**

* Các lớp này xắp xếp các widgets theo một mạng lưới hàng ngang hay hàng dọc, và đặt các widgets mà nó quản lý vào ô chính xác.
* Lớp QVBoxLayout xắp xếp các widgets của nó theo hàng dọc.
* Lớp QHboxLayout xắp xếp các widgets của nó theo hàng ngang.
* Lớp QGridLayout tùy chỉnh xắp xếp lại các lớp QVBoxLayout và QHboxLayout bằng cách sử dụng hàm addlayout().
* Khoảng cách giữa các hàng và các cột được tùy chỉnh khi khai báo hàm setColumMinumumWidth().



*Hình :QVboxlayout , QHboxLayout, QGridLayout*

**QMdiArea**

* + - Lớp QMdiArea là tiện ích cung cấp một khu vực trong đó một cửa sổ con được tạo ra trong cưa sổ QMainwindow.
    - Về cơ bản, giống như cửa sổ quản lý, nó kế thừa hầu hết các chức năng của lớp QMainwindow.



*Hình :Các cửa sổ con được tạo ra trong cửa sổ chính khi gọi lớp QMdiArea*

**QTableWidget, QTableWidgetItem.**

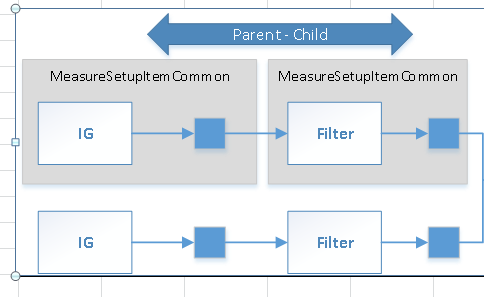
* Lớp QTablewidget cung cấp một khung nhìn dựa trên bảng mục tiêu với một mô hình mặc định.
* Những lớp này cung cấp công cụ xây dựng bảng với số lượng hàng và cột theo yêu cầu.
* Các tiêu đề trong bảng được tạo ra bằng cách cung cấp một danh sách các chuỗi cho các hàm setHorizontalHeaderLabels() and setVerticalHeaderLabels().



*Hình :Table được tạo khi gọi lớp QTableWidget, QTableWidgetItem*

# Giao tiếp các khối (Parent – Child).

* Các khối giao tiếp với nhau theo phương thúc Parent – Child, được sắp xếp theo trình từ từ trái sang phải, khối bên trái sẽ là Parent và khối bên phải là Child.
* Trong mỗi khối Parent-Child đều chứa những widgets cố định và được khai báo thông qua hàm MeasureSetupItemCommon().
* Trong một khối thường có một widget chính và các widget phụ (join) , widget chính thể hiện thông tin của signal. Wiget phụ chức năng chủ yếu là break, insert hoặc remove.
* Việc break, insert hoặc remove sẽ được thực hiện ở widget phụ (join) khi thực hiện các lệnh này thì một khối mới có thể được tạo ra, hoặc ngắt kết nối khối hiện tại…



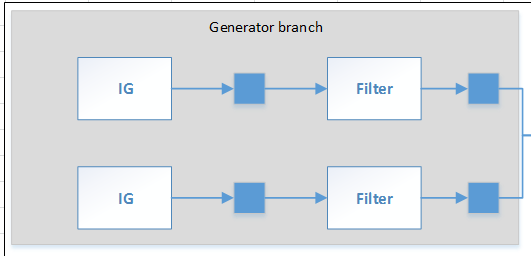
*Hình : Sơ đồ giao tiếp Parent - Child*

# Các nhánh trên giao diện phần mềm.

Giao diện của hệ thống gồm có 3 nhánh chính: Nhánh Generator, nhánh Hardware/Main và nhánh View.

# Nhánh Generator.

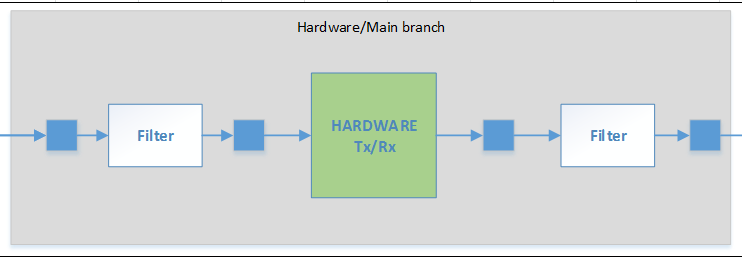
* Nhánh Generator bao gồm các khối được khai báo và thực hiện trong hàm MeasureSetupItemCommon()
* Khối Parent trong nhánh Generator là khối chứa các Item IG và Join, khối Child chứa các Item Filter và Join.
* Khi khối join phát ra một signal để thực hiện việc break, remove hay insert thì một nhánh theo sau nó xẽ được thự thi( có thể là break, remove hay insert nhánh đó).



*Hình :Sơ đồ nhánh Generator.*

# Nhánh Hardware/Main.

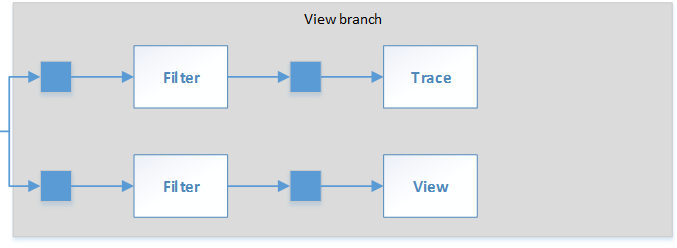
* Nhánh Hardware/Main là nhánh chính của hệ thống với trung tâm là widget Hardware (Tx/Rx).
* Không thể thực hiện các hành động remove hay insert trên Item chính.
* Khi item join trên nhánh chính phát ra tín hiệu break, remove hay insert thì một nhánh lớn(nhánh generator hay nhánh Trace/view) sẽ phải thực thi hành động đó.



*Hình : Sơ đồ nhánh Hardware/Main*

# Nhánh View.

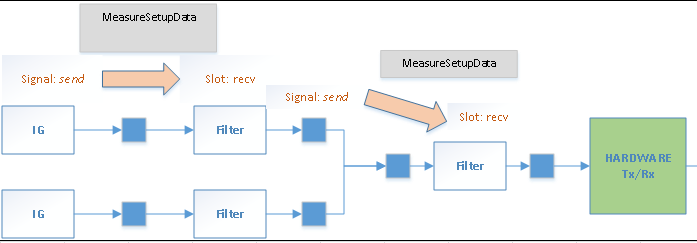
* Nhánh View chứa các khối được khai báo trong hàm MeasureSetupItemCommon()
* Khối Parent trong nhánh Generator là khối chứa các Item Filter và Join, khối Child chứa các Item Trace và Join.
* Khi khối join phát ra một signal để thực hiện việc break, remove hay insert thì một nhánh theo sau nó xẽ được thự thi( có thể là break, remove hay insert nhánh đó).



*Hình : Sơ đồ nhánh Hardware /Main*

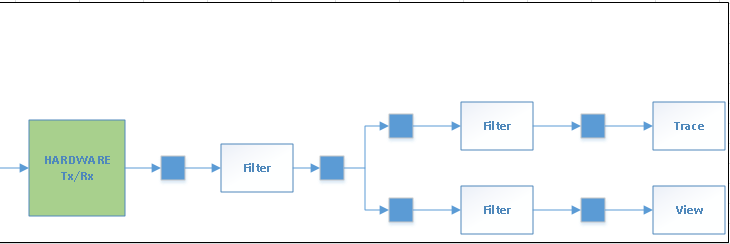
# Quản lý dữ liệu.

# Luồng dữ liệu hệ thống.



*Hình : Sơ đồ luồng dữ liệu hệ thống từ khối IG đến Hardware*

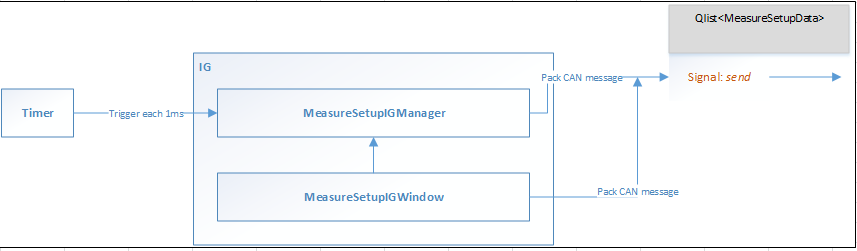
Khi có những signal giả lập được gỡi đi từ hệ thống. để được xử lý tại HARDWARE, các signal sẽ được gỡi lên liên tục để người dùng dễ dàng giám sát. Nhưng không phải vì thế mà bất cứ signal nào cũng có thể được truyền đến HARDWARE để được sử lý, nhờ những bộ lọc Filter mà việc gỡi các signal sẽ được quản lý, dựa vào ID của dữ liệu gỡi đến, và ID của bộ lọc chúng sẽ được so khớp để có thể cho tín hiệu nào đi qua.(ở đây hệ thống đơn giản nên hầu như thiết lập bộ lọc bằng 0 để hầu như các signal đều có thể được cho qua).



*Hình : Sơ đồ luồng dữ liệu hệ thống từ Hardware đến Trace/View*

Signal khi đến được HARDWARE, ở đây sẽ thiết lập việc chọn kênh, tốc độ baudrate cho tín hiệu. khi signal được sử lý xong thì tiếp tục truyền đi, và thông qua việc cho phép truyền signal của các bộ lọc mà signal sẽ được hiển thị ở các khối. tùy vào mục đích của người dùng thì các signal sẽ được hiển thị dưới dạng nào( Trace hoặc View).

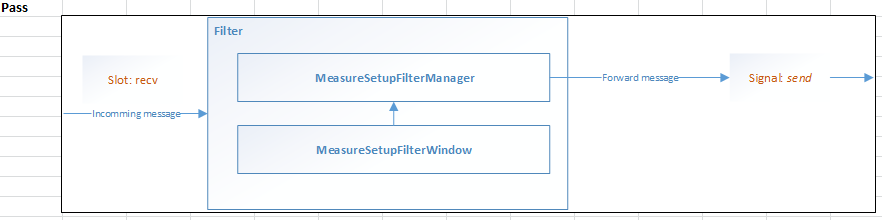
# Luồng dữ liệu Interative Generator Block.



*Hình : Sơ đồ luồng dữ liệu khối IG*

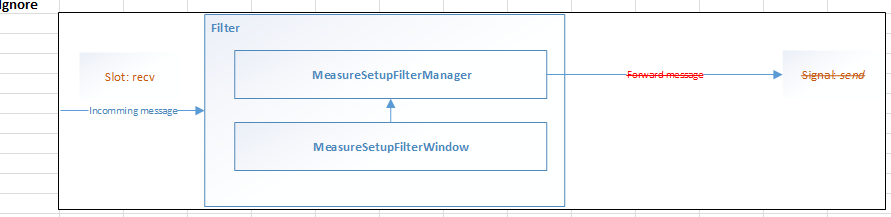
Những signal khi được gỡi lên khối IG thì cứ 1ms khối IG sẽ nhận và xử lý signal một lần, những signal nào nằm ngoài chu kì sẽ không được sử lý, khi xử lý xong các signal khối IG sẽ đóng gói và tiếp tục gỡi đến khối kế tiếp để xử lý.

# Luồng dữ liệu Filter Block.



*Hình : Signal được bộ lọc cho phép đi qua*

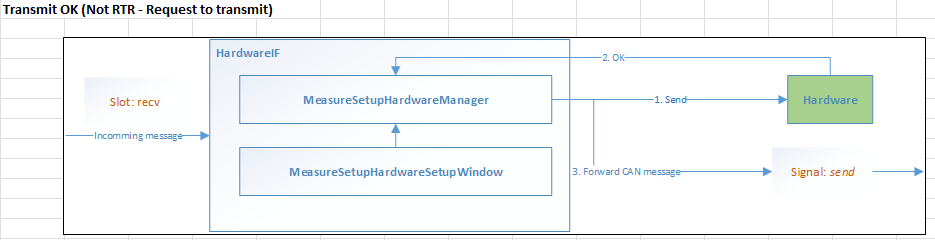
Các gói signal tiếp theo đi qua một bộ lọc, ở đây có tác dụng như việc phân loại signal, những signal có những ID riêng của nó, những ID này sẽ được so khớp với ID của khối lọc, nếu trùng khớp chúng sẽ tiếp tục được gỡi đi đến trung tâm để sử lý, và nếu những ID này khác nhau chúng sẽ bị chặn lại và không được sử lý.



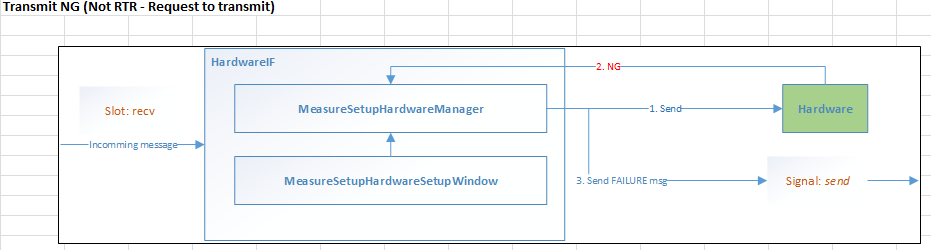
*Hình : Signal bị chặn lại bởi bộ lọc*

# Luồng dữ liệu Hardware Block.

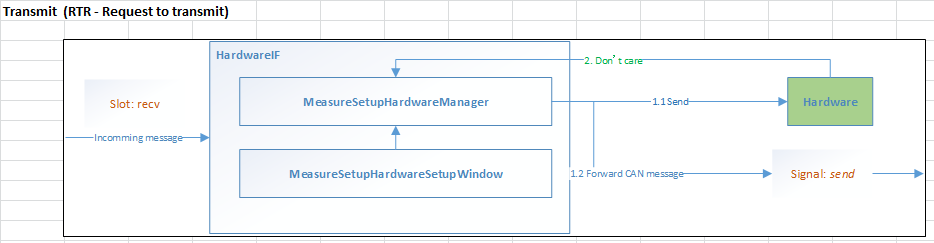
Khi signal gỡi đến khối Hardware ở đây sẽ thiết lập việc chọn kênh, tốc độ baudrate… cho signal, rồi truyền những signal này xuống Board để xử lý, nếu những signal này là hợp lệ board sẽ gỡi trả lại Hardware để thực hiện việc tiếp tục gỡi những signal này đến các khối tiếp theo, còn nếu những signal gặp lỗi trên đường truyền thì sẽ được chờ để xử lý.



*Hình : Signal truyền xuống board và được xử lý*



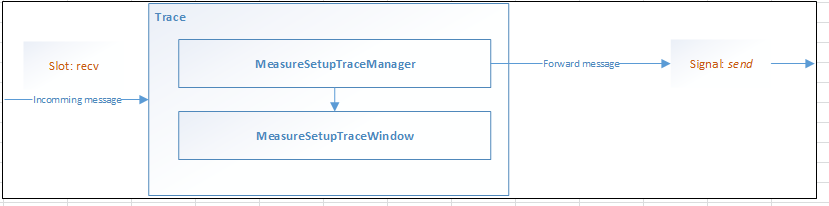
*Hình : Signal truyền xuống board nhưng không được xử lý.*



*Hình : Signal truyền request to transmit*

# Luồng dữ liệu Trace Block

Khi signal được gỡi từ Hardware đến khối Trace những signal này sẽ được hiển thị cho người dùng, đọc thông tin signal ngoài ra còn có thể thêm, xóa bớt các signal…



*Hình : Sơ đồ luồng dữ liệu khối Trace*

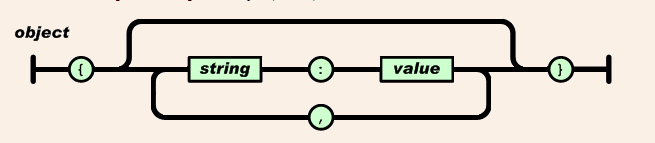
# **Cấu hình dữ liệu.**

# Giới thiệu về Json.

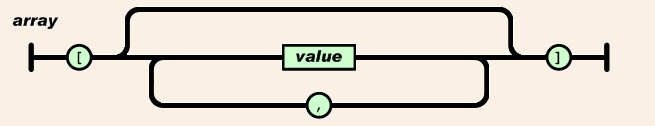
JSON là một định dạng hoán vị dữ liệu nhanh, chúng dễ dàng cho chúng ta đọc và viết , được xây dựng trên cấu trúc Key- Value, chúng có thể là một Object, Array, Value, String, Number để mô tả dữ liệu của UI .

# QJson trong QT framework.

QjsonObject



**QJsonArray**



Json cũng được hỗ trợ trong QT framework cho phép ta tạo file Config theo định dạng Json dễ dàng trong việc edit, save ,..

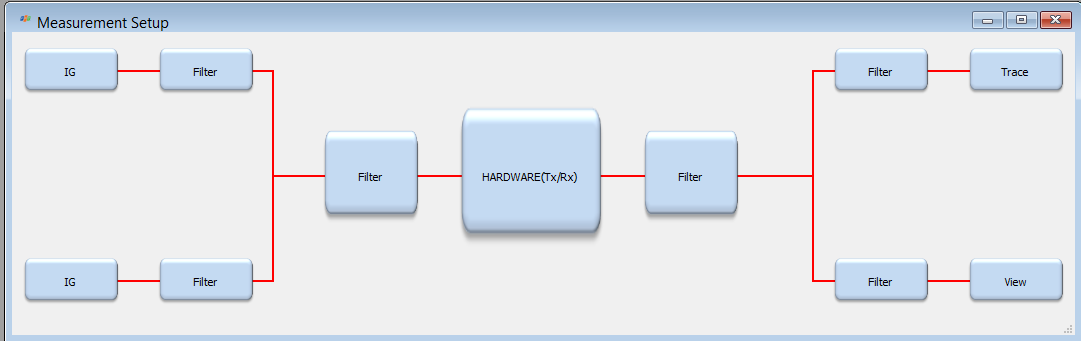
# **Kết luận chương.**

# **CHƯƠNG 3: KẾT QUẢ THỰC HIỆN VÀ ĐÁNH GIÁ**

# **Giới thiệu chương.**

# **Bảng phân công công việc.**

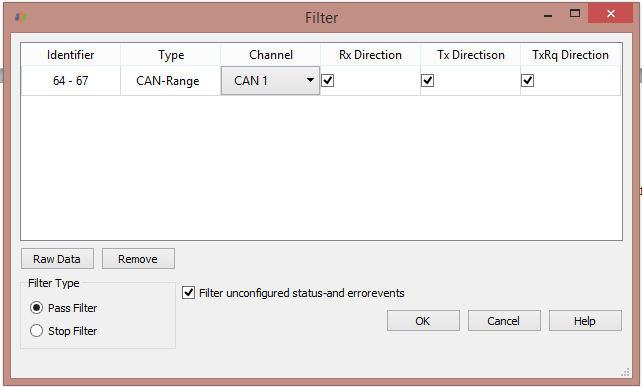
# **Giao diện Simulator software.**

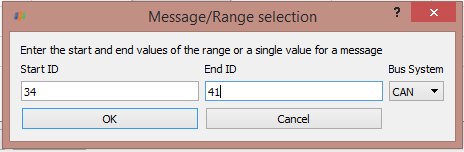


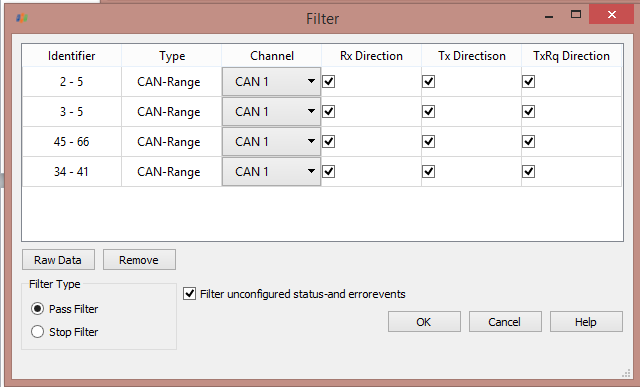
# Giao diện các hộp thoại.

# Giao diện khối IG

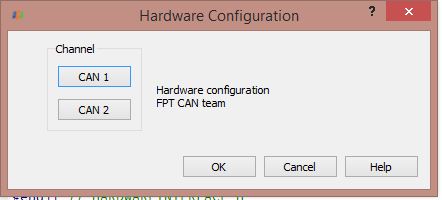
# Giao diện khối Filter

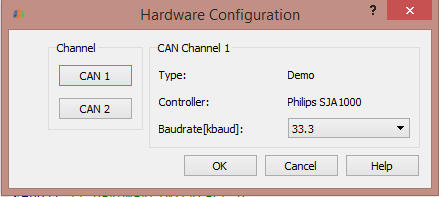




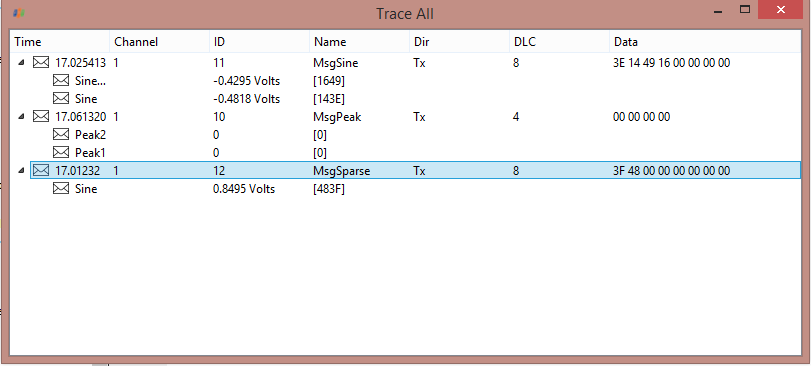


# Giao diện khối Hardware





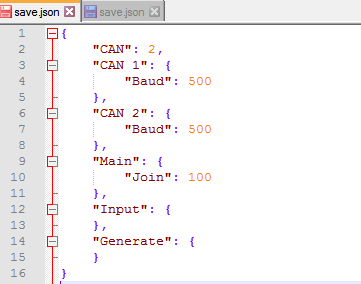
# Giao diện khối Trace



# **Edit file cấu hình theo định dạng json**

Tạo, edit và save được file config theo định dạng json.

**File Json khi mở file simulator ban đầu.**



**File Json được lưu lại sau khi chèn thêm các khối và node.**

****

*Hình 3.2. File Cấu hình theo định dạng json*

# **Hoạt động phần mềm mô phỏng CAN**

# **Kết quả nghiệm thu.**

# **Kết luận chương.**

# **KẾT LUẬN**

# **HƯỚNG PHÁT TRIỂN ĐỀ TÀI**

# **TÀI LIỆU THAM KHẢO**

# **PHỤ LỤC 1**

# **PHỤ LỤC 2**