**Võ Thanh Ban - Trương Công Đông – Nguyễn Đức Tuấn Thiết kế phần mềm mô phỏng CAN 2017**

**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA ĐIỆN TỬ - VIỄN THÔNG**

**ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH: ĐIỆN TỬ - VIỄN THÔNG**

**ĐỀ TÀI:**

**THIẾT KẾ PHẦN MỀM MÔ PHỎNG CAN**

Sinh viên thực hiện: **NGUYỄN ĐỨC TUẤN – LỚP 12DT3**

**VÕ THANH BAN – LỚP 12DT1**

**TRƯƠNG CÔNG ĐÔNG – LỚP 12DT4**

Người hướng dẫn: **TS. TRẦN THỊ HƯƠNG**

**TS. HUỲNH VIỆT THẮNG**

**TS. HỒ PHƯỚC TIẾN**

**Đà Nẵng, …../2017**

**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA ĐIỆN TỬ VIỄN THÔNG**

**ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH: ĐIỆN TỬ - VIỄN THÔNG**

**ĐỀ TÀI:**

**THIẾT KẾ PHẦN MỀM MÔ PHỎNG CAN**

Sinh viên thực hiện: **NGUYỄN ĐỨC TUẤN – LỚP 12DT3**

**VÕ THANH BAN – LỚP 12DT1**

**TRƯƠNG CÔNG ĐÔNG – LỚP 12DT4**

Người hướng dẫn: **TS. TRẦN THỊ HƯƠNG**

**TS. HUỲNH VIỆT THẮNG**

**TS. HỒ PHƯỚC TIẾN**

**Đà Nẵng, …../2017**

**NHẬN XÉT CỦA NGƯỜI HƯỚNG DẪN**

**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

Sinh viên thực hiện: VÕ THANH BAN

Số thẻ sinh viên: 106120009 Lớp: 12DT1

Sinh viên thực hiện: TRƯƠNG CÔNG ĐÔNG

Số thẻ sinh viên: 106120162 Lớp: 12DT4

Đề tài “Thiết kế phần mềm mô phỏng CAN” thuộc dự án "Thiết kế và thi công hệ thống mô phỏng CAN BUS” do nhóm sinh viên Khoa Điện Tử - Viễn thông tham gia thực hiện dưới hình thức Capstone Project với công ty FPT Software.

Phần mềm mô phỏng CAN được xây dựng với chức năng mô phỏng và hiển thị thông tin của signal và message qua các hộp thoại giao diện nhằm mô tả hoạt động của hệ thống mô phỏng CAN BUS.

Phần mềm có khả năng tương tác với phần cứng thông qua việc điều chỉnh tốc độ Baud từ các cổng CAN, truyền tải thông tin từ cơ sở dữ liệu (message/signal) sau đó xử lí và hiển thị chúng thông qua các khối chức năng Interative Generator, Filter, Hardware và mô tả, hiển thị kết quả trên giao diện hộp thoại Trace. Giúp cho người sử dụng có thể dễ dàng kiểm tra, nhận xét thông tin của các dữ liệu được truyền lên từ thiết bị (CAN Device).

Phần mềm hỗ trợ cho cả 2 nền tảng WinDow và Linux, giao diện được viết bằng ngôn ngữ C++ dùng Qt Framework.

# **NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

# **LỜI NÓI ĐẦU VÀ CẢM ƠN**

Trong thời gian hơn 3 tháng thực hiện đồ án tốt nghiệp dưới hình thức Capstone Project tại FPT Software, chúng em đã được tiếp xúc, làm quen với môi trường làm việc của doanh nghiệp 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.

Chúng em chân thành cảm ơn lãnh đạo Khoa Điện tử - Viễn thông và công ty FPT Software đã tạo điều kiện để em được tham gia và thực hiện đồ án. Em xin chân thành cảm ơn TS. Trần Thị Hương – giảng viên hướng dẫn, KS. Nguyễn Văn Thành, KS. Nguyễn Tấn Huy – FPT Software đã hướng dẫn tận tình, cùng các chị quản lí phòng đào tạo Nguyễn Thị Hải, Nguyễn Thị Hoài An đã quan tâm, giúp đỡ trong suốt quá trình thực hiện để nhóm có thể hoàn thành tốt đồ án này. Em xin chân thành cảm ơn!

# **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 – Lớp 12DT3

Võ Thanh Ban – Lớp 12DT1

Trương Công Đông – Lớp 12DT4

Sinh viên 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 Sinh viên thực hiện Sinh viên thực hiện

Nguyễn Đức Tuấn Trương Công Đông Võ Thanh Ban

# **MỤC LỤC**

[TÓM TẮT](#_Toc482940981)

[NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP](#_Toc482940982)

[LỜI NÓI ĐẦU VÀ CẢM ƠN i](#_Toc482940983)

[LỜI CAM ĐOAN ii](#_Toc482940984)

[MỤC LỤC iii](#_Toc482940985)

[DANH MỤC CÁC BẢNG, HÌNH VẼ vi](#_Toc482940986)

[DANH MỤC CÁC TỪ VIẾT TẮT viii](#_Toc482940987)

Trang

[**MỞ ĐẦU** 1](#_Toc482940988)

[**CHƯƠNG 1: TỔNG QUAN HỆ THỐNG MÔ PHỎNG CAN BUS** 2](#_Toc482940989)

[**1.1.** **Giới thiệu chương** 2](#_Toc482940990)

[**1.2.** **Tổng quan về CAN BUS** 2](#_Toc482940991)

[**1.3.** **Hệ thống mô phỏng CAN BUS** 4](#_Toc482940992)

[**1.4.** **Các thành phần của hệ thống.** 6](#_Toc482940993)

[1.4.1. CAN Software 6](#_Toc482940994)

[1.4.1.1. Phần mềm mô phỏng CAN. 6](#_Toc482940995)

[1.4.1.2. Phần mềm cơ sở dữ liệu CAN. 6](#_Toc482940996)

[1.4.2. CAN Device 7](#_Toc482940997)

[1.4.3. PC Driver và Protocol 7](#_Toc482940998)

[**1.5.** **Kết luận chương** 7](#_Toc482940999)

[**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ MÔI TRƯỜNG PHÁT TRIỂN** 8](#_Toc482941000)

[**2.1.** **Giới thiệu chương.** 8](#_Toc482941001)

[**2.2.** **Cơ sở lý thuyết.** 8](#_Toc482941002)

[2.2.1. Lập trình hướng đối tượng 8](#_Toc482941003)

[2.2.2. Thiết kế chương trình theo hướng đối tượng: 10](#_Toc482941004)

[**2.3.** **Môi trường phát triển** 11](#_Toc482941005)

[2.3.1. Môi trường hệ điều hành Linux. 11](#_Toc482941006)

[2.3.2. Giới thiệu chung về Qt Creater Framework. 14](#_Toc482941007)

[2.3.3. Các ưu điểm của QT Framework. 15](#_Toc482941008)

[2.3.4. Mô hình Model/View trong QT Framework 15](#_Toc482941009)

[2.3.5. Cơ chế Signal và Slot trong QT Framework 17](#_Toc482941010)

[2.3.6. Các lớp QT được sử dụng để xây dựng phần mềm mô phỏng CAN. 17](#_Toc482941011)

[**2.4.** **Cấu hình dữ liệu.** 25](#_Toc482941012)

[2.4.1. Giới thiệu về Json. 25](#_Toc482941013)

[2.4.2. QJson trong QT framework. 25](#_Toc482941014)

[**2.5.** **Kết luận chương.** 26](#_Toc482941015)

[**CHƯƠNG 3:**  **KẾT QUẢ THỰC HIỆN VÀ ĐÁNH GIÁ** 27](#_Toc482941016)

[**3.1.** **Giới thiệu chương.** 27](#_Toc482941017)

[**3.2.** **Tiến trình thực hiện** 27](#_Toc482941018)

[**3.3.** **Bảng phân công công việc.** 27](#_Toc482941019)

[**3.4.** **Giao diện Simulator software.** 28](#_Toc482941020)

[3.4.1. Giao diện các hộp thoại. 28](#_Toc482941021)

[3.4.2. Giao diện khối IG 28](#_Toc482941022)

[3.4.3. Giao diện khối Filter 30](#_Toc482941023)

[3.4.4. Giao diện khối Hardware 32](#_Toc482941024)

[3.4.5. Giao diện khối Trace 33](#_Toc482941025)

[**3.5.** **Quản lý dữ liệu.** 35](#_Toc482941026)

[3.5.1. Luồng dữ liệu hệ thống. 35](#_Toc482941027)

[3.5.2. Luồng dữ liệu Interative Generator Block. 36](#_Toc482941028)

[3.5.3. Luồng dữ liệu Filter Block. 36](#_Toc482941029)

[3.5.4. Luồng dữ liệu Hardware Block. 37](#_Toc482941030)

[3.5.5. Luồng dữ liệu Trace Block 38](#_Toc482941031)

[**3.6.** **Edit file cấu hình theo định dạng json** 38](#_Toc482941032)

[**3.7.** **Hoạt động phần mềm mô phỏng CAN** 40](#_Toc482941033)

[**3.8.** **Kết luận chương.** 44](#_Toc482941034)

[**KẾT LUẬN** 45](#_Toc482941035)

[**HƯỚNG PHÁT TRIỂN ĐỀ TÀI** 46](#_Toc482941036)

[**TÀI LIỆU THAM KHẢO** 47](#_Toc482941037)

[**PHỤ LỤC 1**](#_Toc482941038)

# **DANH MỤC CÁC BẢNG, HÌNH VẼ**

[HÌNH 1.1: Mô hình CAN BUS trong xe oto 2](#_Toc482941492)

[HÌNH 1.2: BUS CAN với 3 node 3](#_Toc482941493)

[HÌNH 1.3: Sơ đồ khối mô tả giao tiếp hệ thống CAN BUS 5](#_Toc482941494)

[HÌNH 1.4: Sơ đồ khối hệ thống CANBUS 5](#_Toc482941495)

[HÌNH 1.5: Các thành phần CAN Software. 6](#_Toc482941496)

[HÌNH 2.1: Môi trường hệ điều hành Linux 12](#_Toc482941497)

[HÌNH 2.2: Kiến trúc Linux 13](#_Toc482941498)

[HÌNH 2.3: Mô tả về QT FrameWork 14](#_Toc482941499)

[HÌNH 2.4: Kiến trúc mô hình MVC 15](#_Toc482941500)

[HÌNH 2.5: Cấu trúc model/view trong QT 16](#_Toc482941501)

[HÌNH 2.6: Cửa sổ giao diện chính sử dụng QWindow và QDialog 18](#_Toc482941502)

[HÌNH 2.7: Trình đơn được hiện ra khi rightclick vào một khối. 18](#_Toc482941503)

[HÌNH 2.8: Các button được khởi tạo khi gọi lớp QPushbutton 19](#_Toc482941504)

[HÌNH 2.9: Hình màu được tạ ra khi gọi lớp QPainter 20](#_Toc482941505)

[HÌNH 2.10: Line được tạo ra khi kết nối các điểm đầu cuối. 20](#_Toc482941506)

[HÌNH 2.11: Tọa độ điểm đầu và cuối của Line trên mặt phẳng trả về lớp QPoint 21](#_Toc482941507)

[HÌNH 2.12: Các lớp dẫn xuất của QLayout 21](#_Toc482941508)

[HÌNH 2.13: QVboxlayout, QHboxLayout, QGridLayout 22](#_Toc482941509)

[HÌNH 2.14: Các cửa sổ con được tạo ra trong cửa sổ chính khi gọi lớp QMdiArea 22](#_Toc482941510)

[HÌNH 2.15: Table được tạo khi gọi lớp QTableWidget, QTableWidgetItem 23](#_Toc482941511)

[HÌNH 2.16: Ngăn xếp các tab trong QTabWidget 24](#_Toc482941512)

[HÌNH 2.17: Giao diện hộp thoại QMessageBox 24](#_Toc482941513)

[HÌNH 2.18: Cấu trúc một đối tượng trong Json 25](#_Toc482941514)

[HÌNH 2.19: Cấu trúc một đối mảng trong Json 25](#_Toc482941515)

[HÌNH 2.20: Cấu trúc một giá trị trong Json 26](#_Toc482941516)

[HÌNH 3.1: Giao diện Measurement Setup phần mềm mô phỏng CAN 28](#_Toc482941517)

[HÌNH 3.2: Giao diện hộp thoại IG 29](#_Toc482941518)

[HÌNH 3.3: Giao diện hộp thoại Filter 32](#_Toc482941519)

[HÌNH 3.4: Giao diện hộp thoại RawData trong Filter 33](#_Toc482941520)

[HÌNH 3.5: Giao diện hộp thoại Filter sau khi chèn ID Range 33](#_Toc482941521)

[HÌNH 3.6: Giao diện hộp thoại Hardware 34](#_Toc482941522)

[HÌNH 3.7: Giao diện hộp thoại CAN với 2 kênh CAN 1 và CAN 2 35](#_Toc482941523)

[HÌNH 3.8: Giao diện hộp thoại Trace 36](#_Toc482941524)

[HÌNH 3.9: Sơ đồ luồng dữ liệu hệ thống từ khối IG đến Hardware 37](#_Toc482941525)

[HÌNH 3.10: Sơ đồ luồng dữ liệu hệ thống từ Hardware đến Trace/View 37](#_Toc482941526)

[HÌNH 3.11: Sơ đồ luồng dữ liệu khối IG 38](#_Toc482941527)

[HÌNH 3.12: Signal được bộ lọc cho phép đi qua 38](#_Toc482941528)

[HÌNH 3.13: Signal bị chặn lại bởi bộ lọc 38](#_Toc482941529)

[HÌNH 3.14: Signal truyền xuống board và được xử lý 39](#_Toc482941530)

[HÌNH 3.15: Signal truyền xuống board nhưng không được xử lý. 39](#_Toc482941531)

[HÌNH 3.16: Signal truyền request to transmit 39](#_Toc482941532)

[HÌNH 3.17: Sơ đồ luồng dữ liệu khối Trace 40](#_Toc482941533)

[HÌNH 3.18: File Json ban đầu 40](#_Toc482941534)

[HÌNH 3.19: File Cấu hình theo định dạng json sau khi edit. 41](#_Toc482941535)

[HÌNH 3.20: Hộp thoại MainSimulator 42](#_Toc482941536)

[HÌNH 3.21: Load file cấu hình với định dạng Json 43](#_Toc482941537)

[HÌNH 3.22: Tương tác với các Message/Signal trong hộp thoại IG 43](#_Toc482941538)

[HÌNH 3.23: Dữ liệu được xử lí trong khối Filter 44](#_Toc482941539)

[HÌNH 3.24: Thiết lập tốc độ Baudrate của cổng CAN 45](#_Toc482941540)

[HÌNH 3.25: Cửa sổ Trace hiển thị thông tin của message và signal 45](#_Toc482941541)

# **DANH MỤC CÁC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **VIẾT TẮT** | **TIẾNG ANH** | **TIẾNG VIỆT** |
| CAN | Controller Area Network | Mạng khu vực điều khiển |
| ECU | Electronic Control Unit | Đơn vị điều khiển điện tử |
| PC | Personal Computer | Máy tính cá nhân |
| CSMA/CD | Carrier Sense Multiple Acess with Colision Delection | Đa truy cập nhận biết sóng mang phát hiện xung đột |
| AMP | Arbitration on Message Priority | Cơ chế ưu tiên xung đột |
| EV | Enviroment Vairable | Biến môi trường |
| GPL | General Public License | Giấy phép công cộng |
| MVC | Moldel View Controler |  |
| DLC | Data Lengh Code | Mã chiều dài dữ liệu |
| SB | Start bit | Bit bắt đầu |

# **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 đồ án 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, nhóm 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.

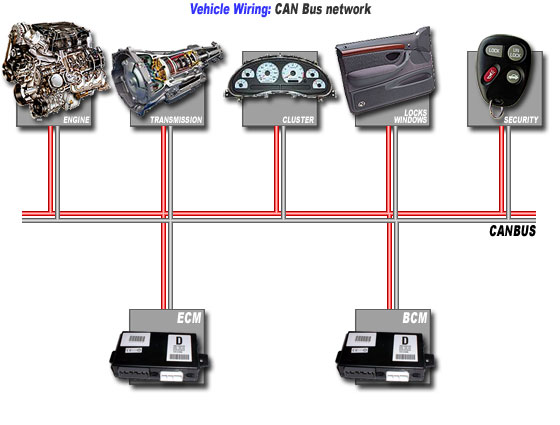
# **CHƯƠNG 1: TỔNG QUAN HỆ THỐNG MÔ PHỎNG CAN BUS**

# **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: Mô hình CAN BUS trong xe oto

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.

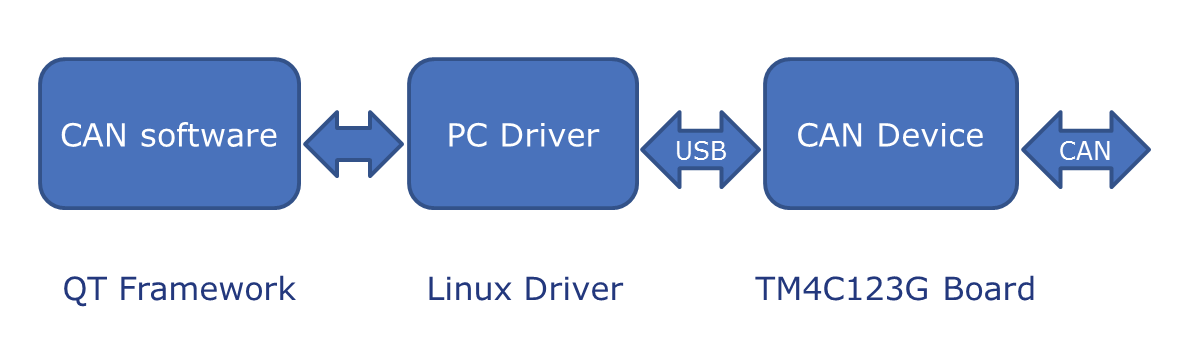
Khi mạng Ethernet phát hiện một vụ xung đột, các nút đang gửi dữ liệu chỉ đơn giản là dừng truyền và đợi một khoảng thời gian ngẫu nhiên trước khi cố gắng để gửi một lần nữa. Tuy nhiên, đối với giao thức CAN có một chút khác biệt đó là các vấn đề xung đột được giải quyết  bằng cách sử dụng các nguyên 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 1.3 : 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 dựa 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 CAN 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 1.4: **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 1.5 : CAN Software gồm: cơ sở dữ liệu CAN và Phần mềm mô phỏng CAN.

# 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 Phần mềm cơ sở dữ liệu CAN. 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.

Phần mềm cơ sở dữ liệu CAN 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 Phần mềm cơ sở dữ liệu CAN là: Tạo mới database và mở 1 database có sẵn.

Hỗ 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 Phần mềm cơ sở dữ liệu CAN 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 gửi lên PC.

CAN Device nhận dữ liệu từ PC thông qua cổng USB và đưa dữ liệu vào hệ thống mạng CAN trên thiết bị. Có 2 nhiệm vụ chính là cấu hình giao tiếp USB và giao tiếp CAN để có thể nhận và gửi các message

Giao tiếp USB nhận và gửi các message để giao tiếp giữa thiết bị và PC

Giao tiếp CAN nhận CAN Frame từ CAN BUS và gửi giữ liệu thu được từ CAN Frame đến cổng USB.

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 baud rate thông qua file descriptor.
* CAN software có thể gửi/nhận message thông qua file descriptor

# **Kết luận chương**

Chương này giới thiệu tổng quan về CAN bus cũng như là hệ thống mô phỏng CAN Bus với các thành phần của nó. Chương 2 sẽ trình bày về sơ sở dữ liệu cũng như là môi trường phát triển để nhóm thực hiện đề tài “Thiết kế phần mềm mô phỏng CAN”.

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ MÔI TRƯỜNG PHÁT TRIỂN**

# **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”.

Chương này sẽ trình bày cơ sở lý thuyết cũng như môi trường phát triển của phần mềm mô phỏng CAN, bao gồm Thiết kế chương trình theo hướng đối tượng, lựa chọn hệ điều hành và môi trường phát triển tích hợp IDE.

# **Cơ sở lý thuyết.**

* + 1. Lập trình hướng đối tượng

Trong lập trình hướng đối tượng, lập trình viên coi các thực thể trong chương trình là các đối tượng sau đó trưu tượng hóa đối tượng thành lớp đối tượng.

Dữ liệu được tổ chức thành các thuộc tính của lớp. Ta chỉ có thể truy cập dữ liệu thông qua đối tượng và các phương thức của đối tượng

**Tính chất:**

* Tính kế thừa: Chúng ta có thể kế thừa các lớp để xây dựng các lớp dẫn xuất, lớp được kế thừa được gọi là lớp cơ sở. Lớp dẫn xuất sẽ có tất cả các thuộc tính và phương thức của lớp cơ sở mà nó kế thừa. Như vậy ta không cần phải xây dựng lại một đối tượng mới hoàn toàn.

Tính kế thừa của lập trình hướng đối tượng cho phép một lớp có thể kế thừa từ các lớp cơ sở. Khi đó lớp dẫn xuất sẽ có thể sử dụng những thuộc tính và phương thức của lớp cơ sở như là của mình.Ngoài ra, lớp dẫn xuất còn có thể bổ sung thêm các thuộc tính phương thức mới giúp giảm được khối lượng công việc trong lập trình hướng đối tượng.

C++ còn hỗ trợ đa kế thừa, nghĩa là một lớp có thể kế thừa từ nhiều lớp khác nhau giúp kế thừa các thuộc tính, phương thức cần thiết từ nhiều lớp.

* Tính đa hình:

Đa hình là tính chất đi kèm với kế thừa, do việc kế thừa, một lớp có thể sử dụng lại các phương thức của lớp khác. Tuy nhiên, nếu cần thiết, lớp dẫn xuất có thể định nghĩa lại phương thức của lớp cơ sở. Đó là sự nạp chồng phương thức trong kế thừa. Nhờ sự nạp chồng phương thức này ta chỉ cần gọi tên phương thức từ đối tượng mà không cần biết đối tượng thuộc lớp nào. Chương trình sẽ tự kiểm tra đối tượng thuộc lớp nào và thực hiện phương thức tương ứng.

* Tính trừu tượng:

Trừu tượng hóa dữ liệu (Data abstraction) liên quan tới việc chỉ cung cấp thông tin cần thiết tới bên ngoài và ẩn chi tiết cơ sở của chúng, ví dụ: để biểu diễn thông tin cần thiết trong chương trình mà không hiển thị chi tiết về chúng. C++ cung cấp đủ các phương thức public tới bên ngoài để thao tác với tính năng của đối tượng và để thao tác dữ liệu đối tượng mà không cần thực sự biết về cách lớp đó đã được triển khai nội tại.

* Tính đóng gói:

Các dữ liệu được đóng gói vào trong đối tượng. Mỗi đối tượng có một phạm vi truy nhập riêng.

Không thể truy nhập đến dữ liệu một cách tự do như lập trình cấu trúc.

Muốn truy nhập vào các thuộc tính được bảo vệ, phải thông qua các đối tượng, nghĩa là phải thông qua các phương thức mà đối tượng đó cung cấp mới có thể truy nhập được dữ liệu đã được bảo vệ.

* **Ưu điểm**:

Không còn nguy cơ dữ liệu bị thay đổi trong chương trình vì dữ liệu đã được đóng gói trong các đối tượng. Việc truy xuất dữ liệu phải thông qua các phương thức được cho phép bởi đối tượng.

Khi thay đổi cấu trúc dữ liệu của đối tượng, không cần thay đổi mã nguồn của các đối tượng khác, mà chỉ cần thay đổi một số thành phần của lớp dẫn xuất. Tránh được việc phải thay đổi quá nhiều mã khi thay đổi cấu trúc dữ liệu.

Có thể sử dụng lại mã nguồn, tiết kiệm đáng kể tài nguyên, thời gian và chi phí. Điều này có được nhờ tính kế thừa.

Phù hợp với các phần mềm lớn, phức tạp.

* **Hạn chế của lập trình hướng đối tượng C++:**

Tính đa kế thừa trong C++ có thể dẫn đến xung đột phương thức nếu 1 lớp kế thwuaf từ 2 lớp khác mà 2 lớp đó lại có các phương thức với các API giống nhau. Lúc đó, đối tượng của lớp dẫn xuất sẽ không biết được đang gọi phương thức của lớp cơ sở nào. Vì vậy lập trình viên cần cẩn trọng không lạm dụng đa kế thừa.

* **Kết luận**: Là ngôn ngữ nữa hướng đối tượng, nữa hướng cấu trúc do kế thừa từ ngôn ngữ C thuần cấu trúc vì vậy C++ có cả đặc tính hướng cấu trúc và hướng đối tượng, tùy thuộc vào trường hợp cụ thể mà ta sẽ sử dụng phương pháp phù hợp.

Như vậy, C++ vừa có thể sử dụng cho lập trình phần cứng, tác động sâu vào từng vùng nhớ, vừa có tính chất hướng đối tượng giúp tiết kiệm thời gian và tài nguyên.

* + 1. Thiết kế chương trình theo hướng đối tượng:
* Có 5 giai đoạn phát triển hệ thống phần mềm theo hướng đối tượng:
* **Phân tích yêu cầu.**

Bằng việc tìm hiểu các trường hợp sử dụng (use case) để nắm bắt các yêu cầu của khách hàng, của vấn đề cần giải quyết. Qua trường hợp sử dụng này, các nhân tố bên ngoài có tham gia vào hệ thống cũng được mô hình hóa bằng các tác nhân. Mỗi trường hợp sử dụng được mô tả bằng văn bản, đặc tả yêu cầu của khách hàng.

* **Phân tích.**

Từ các đặc tả yêu cầu trên, hệ thống sẽ bước đầu được mô hình hóa bởi các khái niệm lớp, đối tượng và các cơ chế để diễn tả hoạt động của hệ thống. Trong giai đoạn phân tích chúng ta chỉ mô tả các lớp trong lĩnh vực của vấn đề cần giải quyết chứ chúng ta không đi sâu vào các chi tiết kỹ thuật.

* **Thiết kế.**

Trong giai đoạn thiết kế, các kết quả của quá trình phân tích được mở rộng thành một giải pháp kỹ thuật. Một số các lớp được thêm vào để cung cấp cơ sở hạ tầng kỹ thuật như lớp giao diện, lớp cơ sở dữ liệu, lớp chức năng…

* **Lập trình.**

Đây còn gọi là bước xây dựng, giai đoạn này sẽ đặc tả chi tiết kết quả của giai đoạn thiết kế. Các lớp của bước thiết kế sẽ được chuyển thành mã nguồn theo một ngôn ngữ lập trình theo hướng đối tượng nào đó.

* **Kiểm tra.**

Trong giai đoạn kiểm tra, có bốn hình thức kiểm tra hệ thống:

- Kiểm tra từng đơn thể (unit testing) được dùng kiểm tra các lớp hoặc các nhóm đơn.

- Kiểm tra tính tích hợp (integration testing), được kết hợp với các thành phần và các lớp để kiểm tra xem chúng hoạt động với nhau có đúng không.

- Kiểm tra hệ thống (system testing) chỉ để kiểm tra xem hệ thống có đáp ứng được chức năng mà người dùng yêu cầu không.

- Kiểm tra tính chấp nhận được(acceptance testing), việc kiểm tra này được thực hiện bởi khách hàng, việc kiểm tra cũng thực hiện giống như kiểm tra hệ thống.

* **Các bước cần thiết để thiết kế chương trình :**

Để thiết kế một chương trình theo hướng đối tượng, chúng ta phải trải qua bốn bước sau, từ đó chúng ta xây dựng được một cây phả hệ mang tính kế thừa và các mối quan hệ giữa các đối tượng:

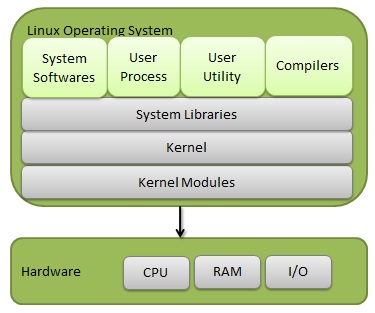
* Xác định các dạng đối tượng (lớp) của bài toán (định dạng các đối tượng).
* Tìm kiếm các đặc tính chung (dữ liệu chung) trong các dạng đối tượng này, những gì chúng cùng nhau chia sẻ.
* Xác định được lớp cơ sở dựa trên cơ sở các đặc tính chung của các dạng đối tượng.
* Từ lớp cơ sở, sử dụng quan hệ tổng quát hóa để đặc tả trong việc đưa ra các lớp dẫn xuất chứa các thành phần, những đặc tính không chung còn lại của dạng đối tượng. Bên cạnh đó, chúng ta còn đưa ra các lớp có quan hệ với các lớp cơ sở và lớp dẫn xuất; các quan hệ này có thể là quan hệ kết hợp, quan hệ tập hợp lại, quan hệ phụ thuộc.
  1. **Môi trường phát triển**

# Môi trường hệ điều hành Linux.

* Giới thiệu hệ điều hành Linux:

Linux là hệ điều hành mô phỏng Unix, được xây dựng trên phần nhân (kernel) và các gói phần mềm mã nguồn mở. Linux được công bố dưới bản quyền của GPL.(General Public Licence).

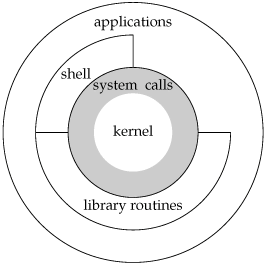
Giống như Unix, Linux gồm 3 thành phần chính: kernel, shell và cấu trúc tệp. Kernel là chương trình nhân, chạy các chương trình và quản lý các thiết bị phần cứng như đĩa và máy in.



HÌNH 2.1: Môi trường hệ điều hành Linux

* **Kernel**: Là trung tâm điều khiển của hệ điều hành Linux, chứa các mã điều khiển hoạt động toàn bộ hệ thống. Kernel được phát triển không ngừng,được phát triển theo dạng module do vậy kích thước thật của kernel rất nhỏ. Nó chịu trách nhiệm cho tất cả hoạt động chính của hệ điều hành, bao gồm nhiều module khác nhau và tương tác trực tiếp với phần cứng. Kernel cung cấp sự trừu tượng bắt buộc để ẩn chi tiết phần cứng mức thấp cho các chương trình hệ thống hoặc ứng dụng.
* **System Libraries**: Là các chương trình hoặc các hàm đặc biệt sử dụng các chương trình ứng dụng hoặc các tiện ích hệ thống truy cập các tính năng của kernel.
* **Các tiện ích**: Các tiên ích được người dùng thường xuyên sử dụng. Nó được dùng cho nhiều thứ như thao tác tập tin, đĩa nén, sao lưu tập tin… tiện ích trong Linux có thể là các lệnh thao tác hay chương trình giao diện đồ họa. Hầu hết các tiện ích trong Linux là sản phẩm của chương trình GNU. Linux có sẵn rất nhiều tiện ích như chương trình biên dịch, chương trình gỡ lỗi, soạn thảo văn bản.
* Kiến trúc Linux:

Kiến trúc Linux được thể hiện như hình dưới:

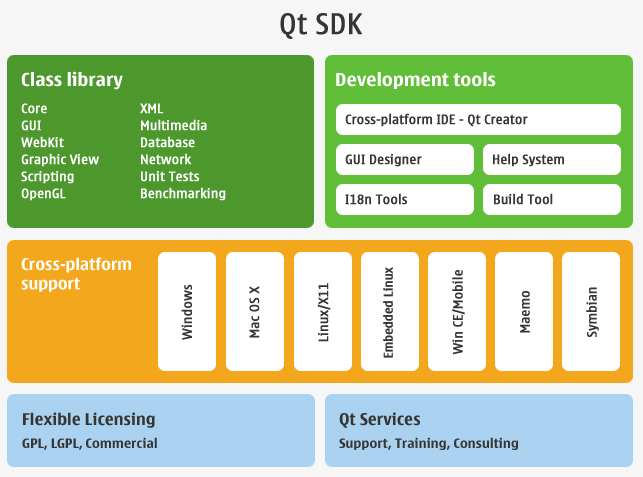


HÌNH 2.2: Kiến trúc Linux

* **System calls**: là giao diện chính giữa kernel và không gian sử dụng, khoảng 300 system call cung cấp các dịch vụ kernel chính.
* **Shell**: Cung cấp các tập lệnh cho người dùng thao tác với kernel để thực hiện công việc. Shell đọc các lệnh từ người dùng và xử lý. Có nhiều loại Shell được sử dụng trong Linux.Shell sử dụng chính trong Linux là GNU Bourne Again Shell, phát triển từ Bourne Shell.
* Lợi ích của Linux:
* Sử dụng Linux bạn sẽ có nhiều lợi điểm. Trong số những hệ điều hành thông dụng, Linux là hệ điều hành mở miễn phí được nhiều người sử dụng nhất hiện nay.
* Tính linh hoạt của Linux được thể hiện ở chỗ nó tương thích được với rất nhiều môi trường. Hiện tại, ngoài Linux dành cho server,máy tính để bàn...nhân Linux (Linux kernel) còn được nhúng vào các thiết bị điều khiển như máy tính palm, robot... Phạm vi ứng dụng của Linux được xem là rất rộng rãi.
* Linux có một cơ cấu phân quyền hết sức rõ ràng. Chỉ có "root"(người dùng tối cao) mới có quyền cài đặt và thay đổi hệ thống. Ngoài ra Linux cũng có cơ chế để một người dùng bình thường có thể chuyển tạm thời chuyển sang quyền "root" để thực hiện một số thao tác. Điều này giúp cho hệ thống có thể chạy ổn định với độ an toàn cao và tránh phải những sai sót dẫn đến đổ vỡ hệ thống.
* Khả năng thích ứng của hệ điều hành giúp bạn chuyển nó từ một nền tảng này sang một nền tảng khác mà nó hoạt động vẫn tốt.

# Giới thiệu chung về Qt Creater Framework.

* Framework: Là một thư viên các lớp được xây dựng hoàn chỉnh, là nền tảng để phát triển các phần mềm ứng dụng. Thay vì tự tay viết tất cả mã cho các lớp, hàm cho toàn bộ dự án phần mềm của mình, các lập trình viên dùng framework để tiết kiệm thời gian và công sức nhưng vẫn đạt hiệu quả mong muốn bằng cách kế thừa các lớp có sẵn từ các thư viện trong framework.



HÌNH 2.3: 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.

# 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.
* Qt hỗ trợ lập trình giao diện chuẩn theo mô hình model/view và hỗ trợ các lớp lập trình giao diện động trong Qt Quick, giúp tạo giao diện chuyên nghiệp và sinh động hơn.
* Với QT chúng ta không cần thiết phải viết giao diện từ những dòng code phức tạp, mà hoàn toàn có thể vẽ giao diện trong tệp có phần mở rộng.ui như vậy việc thiết kế giao diện dễ dàng hơn nhiều. UI của Qt hỗ trợ sẵn các công cụ như QPushButton, QLineEdit, QComboBox, QTreeView, QStackedWidget… giúp việc thiết kế dễ dàng hơn nhiều so với việc phải tự xây dựng tất cả các lớp trên.
* Với những tính năng nổi bật trên, nhóm em quyết định chọn Qt framework làm nền tảng để xây dựng phần mềm mô phỏng CAN.
  + 1. Mô hình Model/View trong QT Framework
* Kiến trúc MVC và kiến trúc model/view trong QT:



HÌNH 2.4: Kiến trúc mô hình MVC

* Mô hình MVC (Model-View-Controller) là một kiến trúc phần mềm, nó giúp cho các nhà phát triển phần mềm tách các ứng dụng của họ ra thành 3 phần có nhiệm vụ riêng biệt và độc lập với các thành phần khác, trong đó:
* Model: chứa các phương thức xử lí và truy xuất dữ liệu.
* View: Đảm bảo việc hiển thị thông tin và tương tác với người dùng.
* Controller: là trung gian giữa Model và View, nhận các yêu cầu từ tương tác người dùng từ view và tác động làm thay đổi dữ liệu trong model.

Như vậy, việc xây dựng phần mềm dùng MVC sẽ dễ dàng, trực quan hơn so với việc xây dựng theo kiểu truyền thống. Ngoài ra, việc chia thành 3 phần riêng biệt giúp cho việc phát triển, nâng cấp, bảo trì và khắc phụ sự cố tiết kiệm thời gian, công sức và chi phí. Việc chia nhỏ như vậy cuãng thuận tiện để phân chia nhiệm vụ và chuyên biệt hóa các phần trong các dự án lớn.

* QT framework sử dụng mô hình model/view thay cho mô hình MVC, trong đó lớp Controller sẽ được tích hợp trong View thông qua Delegate.

HÌNH 2.5: Cấu trúc model/view trong QT

* Lí do lựa chọn mô hình model/view: mô hình này chia phần mềm thành các module riêng biệt, nên sẽ dễ dàng hơn cho việc bảo trì, nâng cấp, phát triển sau này.
* Hoạt động của model/view:
* Ban đầu model sẽ sử dụng phương thức setModelData () của mình để lấy dữ liệu từ tập dữ liệu gốc (dataset), và View sẽ tự cập nhật dữ liệu được lấy từ model thông qua phương thức setModel ().
* Mỗi tương tác chỉnh sửa của người dùng trên giao diện sẽ gọi một đối tượng của 1 lớp Delegate tương ứng, thông qua delegate sẽ làm thay đổi dữ liệu tương ứng trong model, khi đó model sẽ cập nhật hiển thị mới trên View cũng như phát đi một signal để làm thay đổi dữ liệu ở dữ liệu gốc, signal này được kết nối với một slot tương ứng ở lớp quản lí tập dữ liệu gốc, slot này sẽ làm thay đổi dữ liệu trong dataset.
  + 1. Cơ chế Signal và Slot trong QT Framework
* Trong các phần mềm, nơi người dùng có thể tương tác được với giao diện, phải có một cơ chế nào đó để lúc người dùng tương tác( như kích chuột, di chuyển con trỏ chuột,…) với giao diện thì phần mềm sẽ biết được người dùng đang thực hiện thao tác đó. Qt sử dụng cơ chế signal/slot để đảm nhận nhiệm vụ này.
* Signal: khi một sự kiện nào đó xảy ra, một signal sẽ được phát đi, thực ra nó chỉ là một phương thức của một lớp nhưng không có phần thân hàm {}. Các lớp Widget có sẵn trong Qt có rất nhiều signal được định nghĩa sẵn, và chúng ta cũng có thể viết các signal riêng cho các lớp của tự định nghĩa. Signal không có kiểu trả về, kiểu trả về của signal luôn luôn là void.
* Slot:  là một phương thức bình thường của một lớp, các phương thức này sẽ được gọi khi có một signal nào đó được phát đi. Cũng giống như signal, các lớp Widget trong Qt cũng có sẵn rất nhiều slot và chúng ta cũng có thể viết slot cho lớp của riêng chúng ta.
* Connect: Signal và slot được kết nối qua từng đối tượng thông qua phương thức connect.

*Connect (Đối tượng 1, SIGNAL (signal thuộc đối tượng 1), Đối tượng 2, SLOT (slot thuộc đối tượng 2));*

Sau khi kết nối sẽ có một đối tượng phát ra signal và một đối tượng nhận signal đó và thực hiện một phương thức tương ứng, lúc này phương thức đó sẽ là một slot.Kết nối signal và slot là một cách tiện lợi để người dùng gửi đi các signal làm thay đổi dữ liệu tương ứng trên dữ liệu gốc cũng như trên model thay cho delegate.

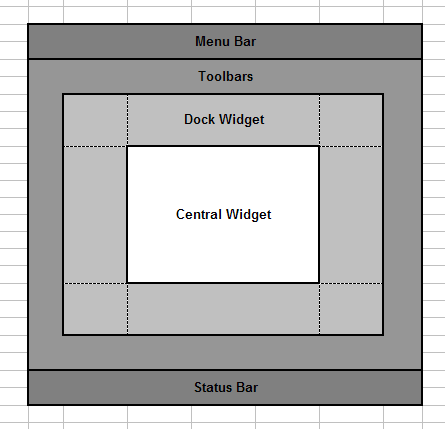
* Để có thể kết nối signal và slot thì hoặc là phương thức slot của đối tượng nhận signal phải có thành phần tham số giống với signal hoặc là slot phải không có tham số.

# Các lớp QT được sử dụng để xây dựng phần mềm mô phỏng CAN.

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

**QMainWindow và QDialog**

* Cửa sổ chính cung cấp một khuôn khỗ 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, QDockWidgets hay QStartusBar.



HÌNH 2.6: 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 2.7: Trình đơn được hiện ra khi rightclick vào một khối.

**QAction**

Trong 1 menu chứa một hoặc nhiều QAction, chính là các hành động để người dùng chọn lựa. Khi kích chuột vào một action thì một signal sẽ được giải phóng, tương ứng với nó, 1 slot sẽ được thực thi.

**QPushbutton**

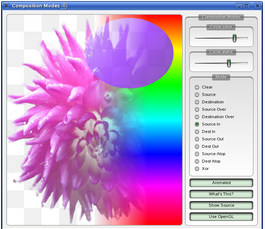
* Tiện ích QPushbutton cung cấp một nút lệnh.
* Click nút để chỉ lệnh cho máy tính thực thi một số hành động hoặc để trả lời câu hỏi.
* Một nút lệnh thường có hình chữ nhật và thường hiển thị một nhãn văn bản mô tả hành động của nó.
* Nút lệnh hiển thị văn bản, và tùy chọn các biểu tượng nhỏ bằng cách 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 2.8 : Các button được khởi tạo khi gọi lớp QPushbutton

**QPainter**

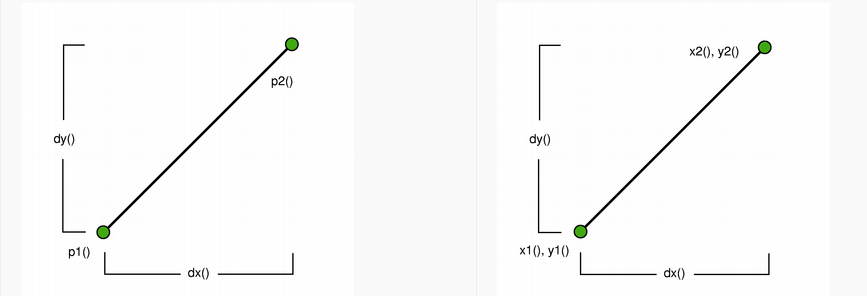
* Lớp QPaint thực hiện việc vẽ ở mực low-level trên các widgets và các devices khác. Nó cung cấp các chức năng tối ưu hóa cao để thực hiện hầu hết các chương trình vẽ GUI.
* Chức năng cốt lõi của lớp này là vẽ nhưng lớp này cung cấp một số chức năng cho phép bạn tùy chỉnh các thiết lập và chất lượng 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 2.9: Hình màu được tạ ra khi gọi lớp QPainter

**QLine:**

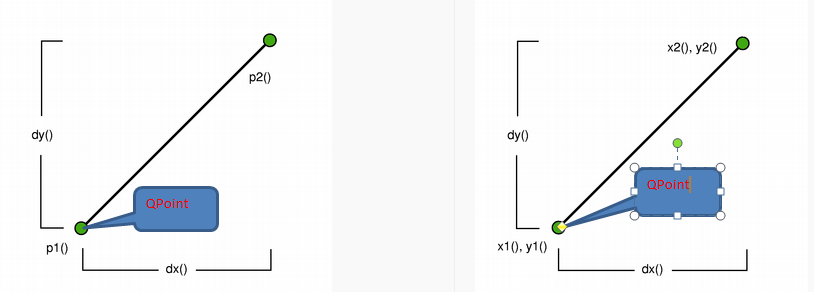
* Lớp QLine cung cấp một vector 2 chiều sử dụng độ chính xác của số nguyên.
* Một QLine mô tả một đường dài hữu hạn (hoặc một đoạn thẳng). Các điểm bắt đầu và kết thúc của đường được xác định bằng cách sử dụng tọa độ.
* Các vị trí của điểm bắt đầu và kết thúc của đường kẻ có thể được lấy bằng các hàm 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 2.10: 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.



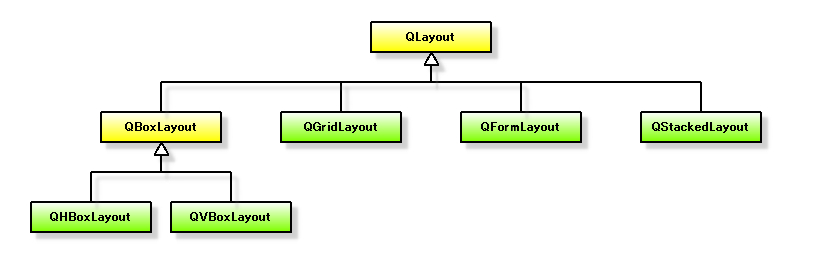
HÌNH 2.11 :Tọa độ điểm đầu và cuối của Line trên mặt phẳng trả về lớp QPoint

**QLayout**

Qt hỗ trợ lớp QLayout cùng các lớp dẫn xuất của nó giúp lập trình viên có thể dễ dang thay đổi định dạng vị trí của các khối chức năng trong các widget, cũng như vị trí của các widget trong các trang.

Layout sẽ khiến giao diện đẹp mắt hơn và thân thiện người dùng hơn. Trong Qt, chúng ta có thể bố trí trang dùng mã hoặc layout bằng giao diện, tức là tự vẽ giao diện layout sau đó chỉnh sửa lại bằng mã C++.

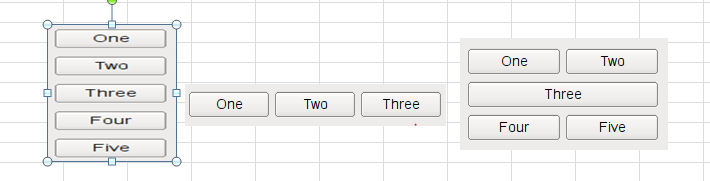
Các lớp dẫn xuất thường dùng của Qt là: **QVBoxlayout, QHBoxLayout, QGridLayout.**

****

HÌNH 2.12:Các lớp dẫn xuất của QLayout

**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 2.13 :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 2.14 :Các cửa sổ con được tạo ra trong cửa sổ chính khi gọi lớp QMdiArea

**QFile:**

* + - Lớp QFile cung cấp một giao diện hỗ trợ việc đọc và ghi vào tệp.
    - Là một thiết bị nhập xuất I/O cho việc đọc và ghi tệp dạng văn bản, dạng nhị phân hoặc dạng tài nguyên khác như hình ảnh, âm thanh. QFile có thể kết hợp với QTextStream hoặc QDataStream để đọc và ghi dữ liệu từ tệp một cách thuận tiện hơn.

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

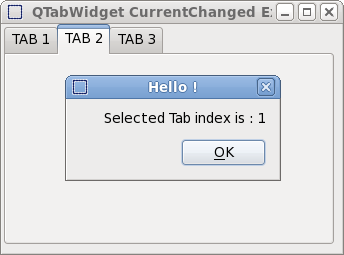
**QStackedWidget:**

Lớp QStackedWidget cung cấp một ngăn xếp các widgets nơi mà chỉ có duy nhất một widget hiển thị trong một thời điểm tùy vào chỉ số index trong hàm setCurrentIndex (index) hay setCurrentWidget (QWidget).

Sử dụng các hàm addWidget(QWidget \*widget ), insertWidget(int index, QWidget \*widget) và removeWidget(QWidget \*widget ) để thêm, chèn và xóa các widget trong QStackedWidget.

**QTabWidget:**

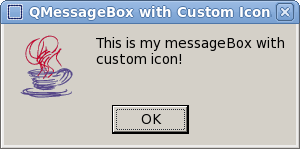
QTabWidget cung cấp một ngăn xếp các tab, trong đó mỗi tab là một widget.



HÌNH 2.16 :Ngăn xếp các tab trong QTabWidget

**QMessageBox:**

QMessageBox cung cấp một hộp thoại nhằm thông báo cho người dùng hoặc yêu cầu người dùng cung cấp một xác nhận, một chọn lựa cho hành động vứa thực hiện. Ví dụ : Khi xóa 1 thông điệp, do sau khi xóa dữ liệu sẽ mất và không thể khôi phục được nên 1 hộp thoại yêu cầu xác nhận có đúng là người dùng muốn xóa hay không sẽ được hiển thị kèm theo cảnh báo. Hoặc khi thực thi một hành động không hợp lê, sẽ có một cảnh báo tới người dùng thông qua QMessageBox.



HÌNH 2.17 :Giao diện hộp thoại QMessageBox

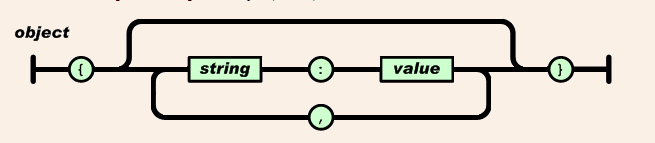
# **Cấu hình dữ liệu.**

# 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 User Interface.

# QJson trong QT framework.

**QjsonObject**

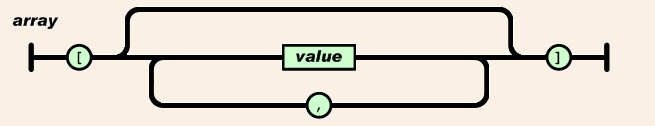


HÌNH 2.18: Cấu trúc một đối tượng trong Json

Một đối tượng là một hỗn độn các cặp tên và giá trị, một đối tượng bắt đầu bởi dấu ngoặc nhọn trái { và kết thúc với dấu ngoặc nhọn phải }. Từng tên được theo sau bởi dấu hai chấm : và các cặp tên/giá trị được tách nhau bởi dấu phẩy.

Qt hỗ trợ lớp QJsonObject cho phép edit các đối tượng một cách dễ dàng.

**QJsonArray**

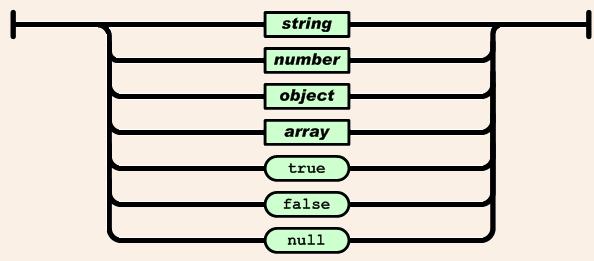


HÌNH 2.19: Cấu trúc một đối mảng trong Json

Một mảng là tập hợp các giá trị đã được sắp xếp. Một mảng bắt đầu bởi dấu mở ngoặc vuông trái [ và kết thúc với dấu ngoặc vuông phải ]. Các giá trị được cách nhau bởi dấu phẩy.

Qt hỗ trợ lớp QJsonArray cho phép edit insert, remove các giá trị từ mảng.

**QJsonValue:**



HÌNH 2.20 Cấu trúc một giá trị trong Json

Một giá trị có thể là một chuỗi string trong những trích dẫn kép hoặc là một số, hoặc true, hoặc false, hoặc Null, hoặc một đối tượng, hoặc một mảng. Những cấu trúc này có thể đã được lồng vào nhau.

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

# **Kết luận chương.**

Trên đây là toàn bộ cơ sở lí thuyết cơ bản phục vụ cho quá trình thiết kế phần mềm mô phỏng CAN. Những kiến thức tham khảo từ các tài liệu tham khảo uy tín là nền tảng vững chắc giúp tư duy đúng hướng và giải quyết đúng vấn đề. Phần mềm được xây dựng dựa trên việc áp dụng chặt chẽ các nguyên tắc cốt lõi trong kỹ thuật lập trình. Chương tiếp theo sẽ đi sâu vào quá trình thiết kế, thi công phần mềm mô phỏng CAN và nhận xét, đánh giá kết quả đạt được.

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

# **Giới thiệu chương.**

Chương này sẽ trình bày tiến trình thực hiện đề tài “Thiết kế phần mềm mô phỏng CAN”. Gồm các bước thiết kế, phân chia công việc và thi công cũng như đánh giá, nhận xét kết quả được đạt của đề tài.

* 1. **Tiến trình thực hiện**

Để thực hiện được đề tài, nhóm đã lên tiến trình thực hiện gồm các giai đoạn:

* Lên ý tưởng và thiết kế giao diện. Gồm có giao diện chính và các hộp thoại con như hộp thoại Interative Generator, hộp thoại Filter, hộp thoại Hardware và hộp thoại Trace.
* Xây dựng mô hình giao tiếp các khối (Parent – Child) và các nhánh trong hệ thống.
* Xây dựng luồng dữ liệu của hệ thống và quản lý, cấu hình dữ liệu.
* Kiểm tra kết quả: Bao gồm kiểm tra phát hiện lỗi và sửa lỗi trong quá trình lập trình.

# **Bảng phân công công việc.**

Dựa trên tiến trình thực hiện, nhóm đã phân công công việc cho các thành viên một cách đồng đều và tương thích nhất.

BẢNG 1.1: Bảng phân chia công việc cho các thành viên

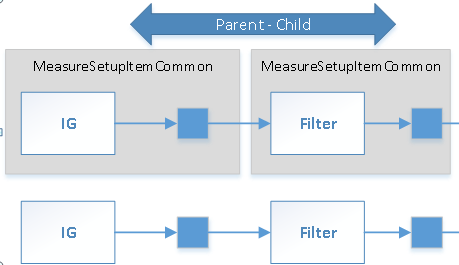
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tên sinh viên  Nhiệm vụ | | Nguyễn Đức Tuấn | Võ Thanh Ban | Trương Công Đông | |
| Thiết kế giao diện | | Giao diện Filter, Trace | Giao diện IG, Hardware | Giao diện chính | |
| Xây dựng mô hình giao tiếp | | Giao tiếp các nhánh | Giao tiếp các khối | Giao tiếp hệ thống | |
| Xây dựng luồng dữ liệu | | Luồng dữ liệu Filter, Trace | Luồng dữ liệu IG, Hardware | Luồng dữ liệu hệ thống | |
| Cấu hình dữ liệu | | Cấu hình bằng Qjson | Cấu hình bằng QJson | Giao tiếp QJson với giao diện chính | |
| Kiểm thử kết quả  (Giai đoạn 1) | Đã thực hiện | | Đã thực hiện | | Đã thực hiện |
| Kiểm thử kết quả  (Giai đoạn 2) | Đang thực hiện | | Đang thực hiện | | Đang thực hiện |

# **Mô hình giao tiếp các thành phần của phần mềm**

* + 1. 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().



HÌNH 3.1 Sơ đồ giao tiếp Parent - Child

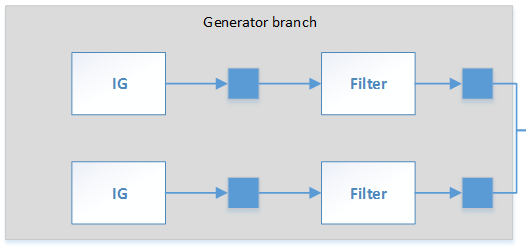
• Trong một khối thường có một widget chính và các widget phụ (join) , widget chính thể hiện thông tin của signal. Wiget phụ chức năng chủ yếu là break, insert hoặc remove.

• 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…

* + 1. Giao tiếp các nhánh

**Nhánh Generator**

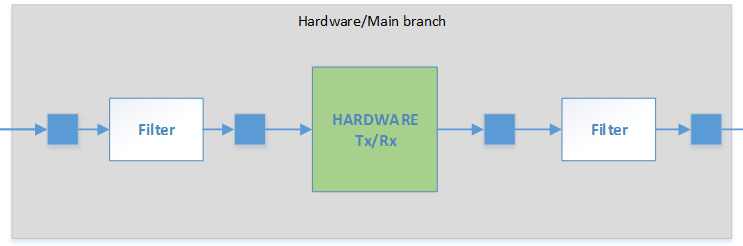
* Nhánh Generator chứa các khối được khai báo trong hàm MeasureSetupItemCommon()
* Khối Parent trong nhánh Generator là khối chứa các Item IG và Join, khối Child chứa các Item Filter và Join.
* Khi khối join phát ra một signal để thực hiện việc break, remove hay insert thì một nhánh theo sau nó sẽ được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



HÌNH 3.2 Sơ đồ nhánh Generator.

**Nhánh Hardware**

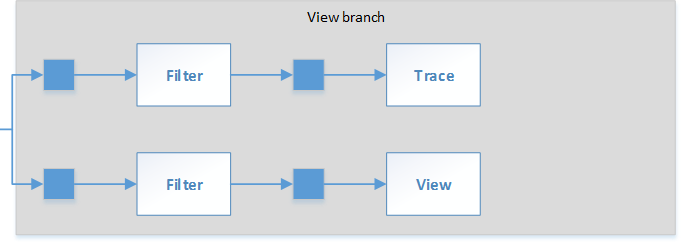
* Nhánh Hardware/Main là nhánh chính của hệ thống với trung tâm là widget Hardware(Tx/Rx).
* Không thể thực hiện các hành động remove hay insert trên Item chính.
* Khi item join trên nhánh chính phát ra tín hiệu break, remove hay insert thì một nhánh lớn(nhánh generator hay nhánh Trace/view) sẽ phải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



HÌNH 3.3 Sơ đồ nhánh Hardware/Main

**Nhánh View**

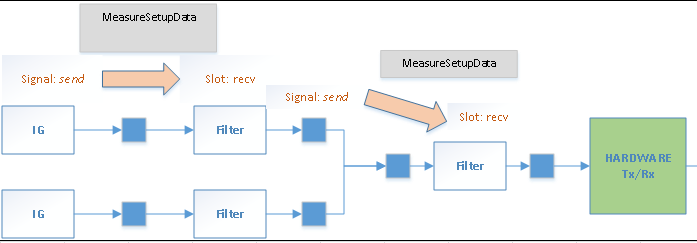
* Nhánh View bao gồm các khối đã được khai báo trong hàm MeasureSetupItemCommon().
* Khối Parent trong nhánh Generator là khối chứa các Item Filter và Join, khối Child chứa các Item Trace và Join.
* Khi khối join phát ra một signal để thực hiện việc break, remove hay insert thì một nhánh theo sau nó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



HÌNH 3.4 Sơ đồ nhánh View

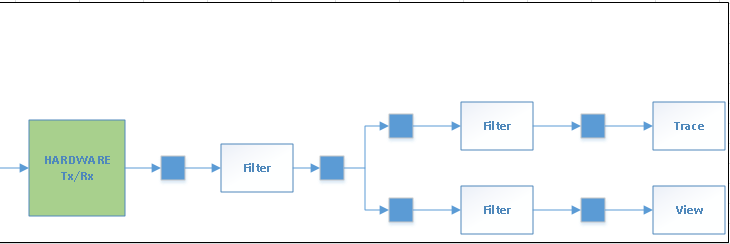
# **Luồng dữ liệu.**

# Luồng dữ liệu hệ thống.



HÌNH 3.5: Sơ đồ luồng dữ liệu hệ thống từ khối IG đến Hardware

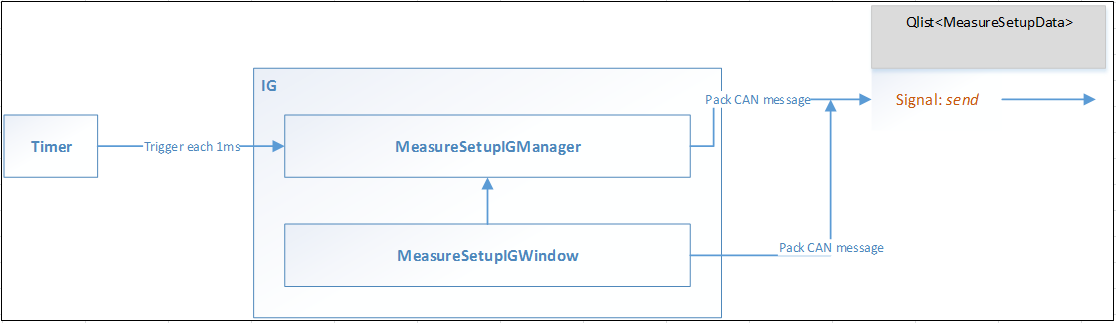
Khi có những signal giả lập được gửi đi từ hệ thống. Để được xử lý tại HARDWARE, các signal sẽ được gửi lên liên tục để người dùng dễ dàng giám sát. Nhưng không phải vì thế mà bất cứ signal nào cũng có thể được truyền đến HARDWARE để được xử lý, nhờ những bộ lọc Filter mà việc gửi các signal sẽ được quản lý, dựa vào ID của dữ liệu gửi 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 3.6: 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 xử 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.

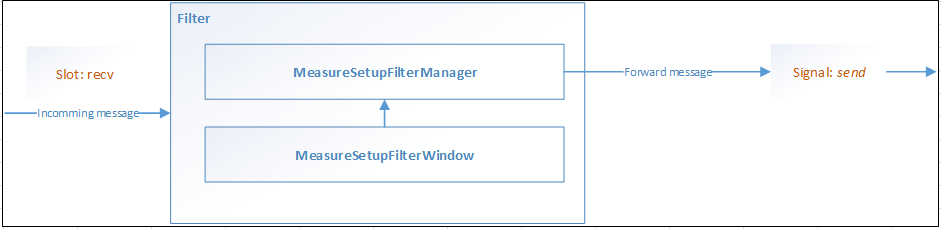


HÌNH 3.7: Sơ đồ luồng dữ liệu khối IG

Những signal khi được gửi 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 xử lý, khi xử lý xong các signal khối IG sẽ đóng gói và tiếp tục gửi đến khối kế tiếp để xử lý.

# Luồng dữ liệu Filter Block.

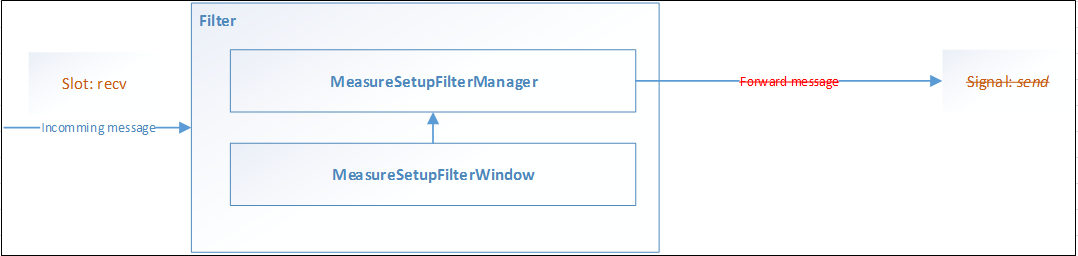
**Pass**



HÌNH 3.8: 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 gửi đi đến trung tâm để xử lý, và nếu những ID này khác nhau chúng sẽ bị chặn lại và không được xử lý.

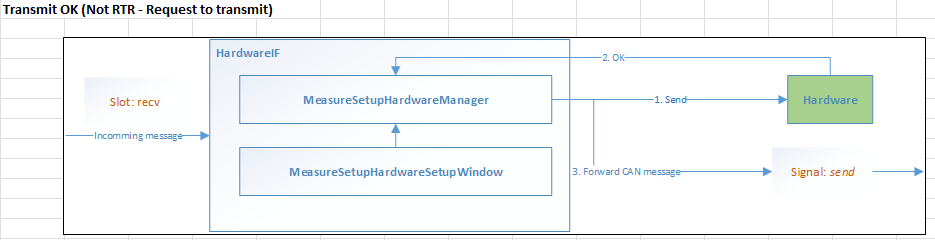
**Ignore**



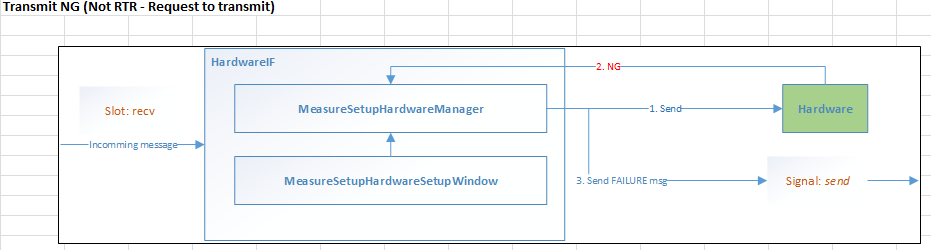
HÌNH 3.9 Signal bị chặn lại bởi bộ lọc

# Luồng dữ liệu Hardware Block.

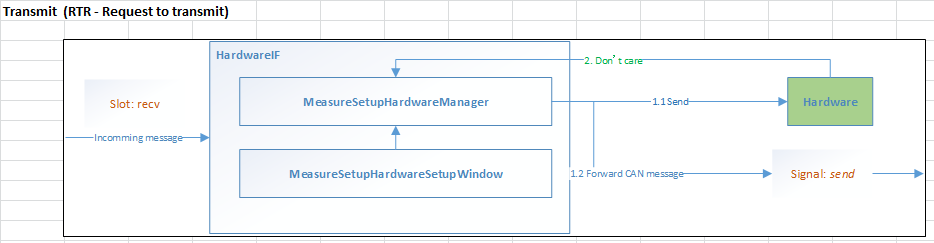
Khi signal gửi đến khối Hardware ở đây sẽ thiết lập việc chọn kênh, tốc độ baudrate… cho cổng CAN, rồi truyền những signal này xuống Board để xử lý, nếu những signal này là hợp lệ board sẽ gửi trả lại Hardware để thực hiện việc tiếp tục gửi những signal này đến các khối tiếp theo, còn nếu những signal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



HÌNH 3.10 Signal truyền xuống board và được xử lý



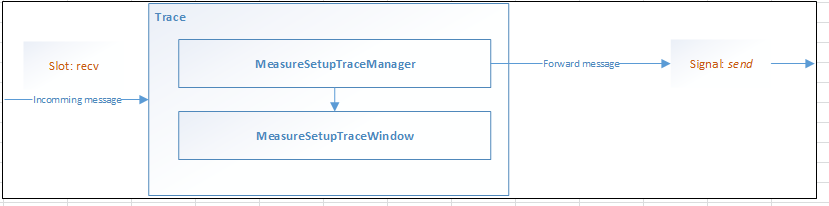
HÌNH 3.11 Signal truyền xuống board nhưng không được xử lý.



HÌNH 3.12 Signal truyền request to transmit

# Luồng dữ liệu Trace Block

Khi signal được gửi từ Hardware đến khối Trace những signal này sẽ được hiển thị cho người dùng, đọc thông tin của message và signal trong message.



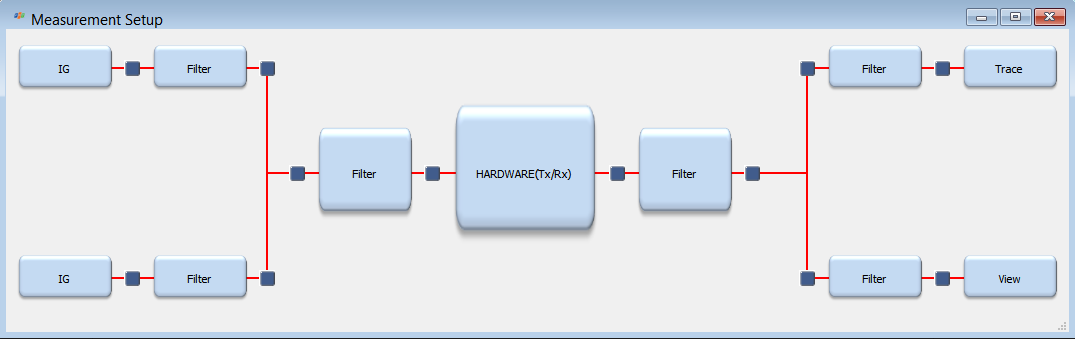
HÌNH 3.13 Sơ đồ luồng dữ liệu khối Trace

# **Giao diện Simulator software.**

**Giao diện chính**

Giao diện Measurement Setup phần mềm mô phỏng CAN được thiết kế dựa trên các lớp QPushButton, QLine, QLayout, QPainter…Giao diện chính thể hiện các khối và các nhánh liên kết với nhau. Bao gồm các chức năng:

* Load dữ liệu từ một file Json để hiển thị giao diện chính của hệ thống
* Liên kết các khối trong hệ thống với nhau.
* Có thể chèn, ngắt kết nối. Xóa một nhánh khi một sự kiện của nút nhấn được kích hoạt.
* Khi click vào các nút nhấn thì mỗi nút nhấn sẽ hiển thị thông tin khác nhau cho người dùng.



HÌNH 3.14 Giao diện Measurement Setup phần mềm mô phỏng CAN

# **Giao diện các hộp thoại.**

Giao diện các hộp thoại trong phần mềm gồm có giao diện khối IG, khối Filter, khối Hardware/Main, khối Trace. Mỗi khối thể hiện một chức năng và xử lí những thông tin khác nhau phù hợp với hệ thống.

# Giao diện khối IG

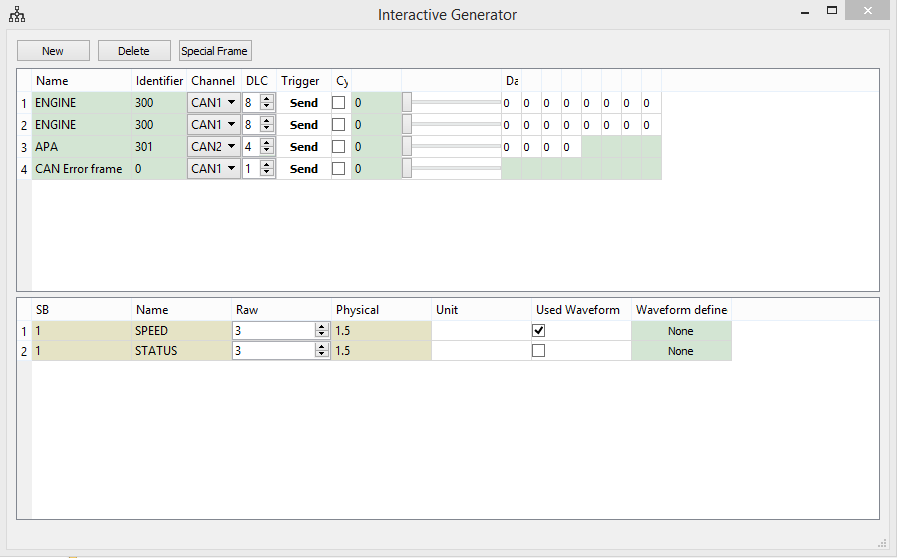
Giao diện khối IG được thiết kế dựa trên các lớp QLabel, QMenu, QModelView, QPushbutton…Tiến trình thực hiện khối IG được thực hiện sau đây:

Mở hộp thoại IG bằng việc click vào button IG trên giao diện chính.

Trong hộp thoại IG được chia nhỏ thành một danh sách transmit và danh sách signal.Trong danh sách transmit bạn có thể chọn từ message và cấu hình chúng. Được gán cho mỗi message là một danh sách signal trong đó các giá trị signal có thể được cấu hình.

Mỗi tin nhắn được gửi bởi khối IG phải được thêm vào danh sách truyền, có các thủ tục sau đây:

* Click vào button “New” và chọn message cần được đưa vào danh sách transmit.
* Sử dụng nút “Special Frame” để chọn CAN Errorframe để gửi thông báo lỗi.
* Click vào các button Delete để thực hiện việc xóa message không cần thiết.



HÌNH 3.15 :Giao diện hộp thoại IG

Các trường của Message:

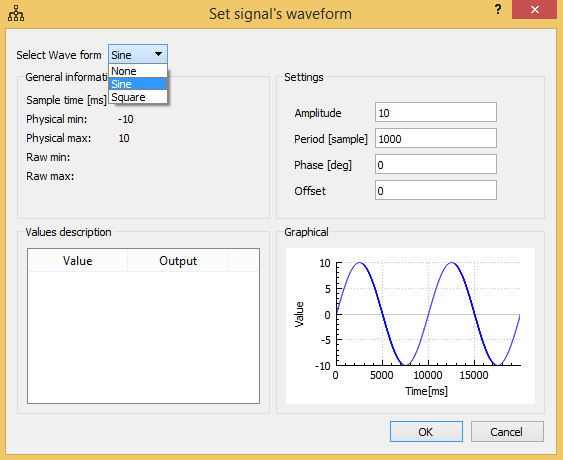
* + Message Name: Hiển thị tên của Message.
  + Message ID: Trường địa chỉ của Message đang được truyền
  + Chanel: Cho biết kênh đang gửi tin nhắn.
  + DLC (Data Length Code): Cho biết số byte dữ liệu trong tin nhắn.
* Trigger: thiết lập điều kiện, message có được kích hoạt hay không? Bằng việc thiết lập chọn gửi hoặc không.
* Send: Thiết lập việc lựa chọn việc message có được gữi.
* Data field: chứa 8 byte dữ liệu, giá trị byte cụ thể (hex/dec) sẽ được hiển thị

Các trường của Signal:

* + SB (Start Bit): Bit đầu tiên của tín hiệu, lấy từ cở sở dữ liệu.
  + Signal Name: Tên của tín hiệu, lấy từ cở sở dữ liệu.
  + Raw Value: Giá trị nguyên của tín hiệu, người dùng có thể cấu hình.
  + Phys Value: Giá trị vật lý của tín hiệu, người dùng có thể cấu hình.

Chuyển đổi giá trị nguyên sang giá trị vật lý và ngược lại, được sử dụng bằng cách sử dụng công thức chuyển đổi: [physical value] = ( [raw value] \* [factor] ) + [offset]

* + Unit: Đơn vị của tín hiệu.
  + Use Waveform: Thiết lập việc có muốn show thông tin ra dưới dạng song.
  + Waveform Define: Mô phỏng message dưới dạng sóng. Khi click vào signal nằm trong khung waveform define thì một hộp thoại dialog thiết lập dạng sóng được hiển thị cho người dùng:



HÌNH 3.16: Hộp thoại thiết lập dạng sóng cho Signal

Các trường của hộp thoại thiết lập dạng sóng:

* Select Wave Form : thiết lập việc chọn dạng sóng.
  + None: không lựa chọn dạng sóng nào.
  + Sine: Thiết lập signal ra ở dạng sóng Sine.
  + Square: Thiết lập signal ra ở dạng sóng vuông.
* General information: Thông tin chung của signal.
  + Sample time [ms]: Thời gian lấy mẫu của signal.
  + Physical (min/max): Giá trị Min/Max của lớp vật lý.
* Settings: Thiết lập các thông số cho Signal.
* Amplitude: Biên độ của signal.
* Period [sample]: Chu kì của signal.
* Phase [deg]: phase của signal.

# Giao diện khối Filter

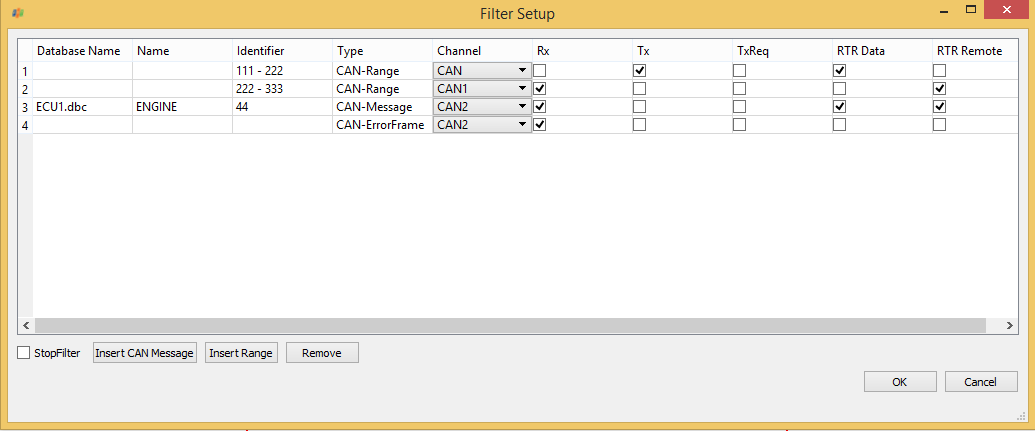
Giao diện khối Filter được thiết kế dựa trên các lớp QLable, QTabwidgetitem, QPushButton, QCheckbox…

Mở hộp thoại Filter bằng việc click chuột vào button Filter trên giao diện chính.

Khối Filter được sử dụng để cố tình giảm bớt khối lượng dữ liệu. Rút ngắn thời gian xử lý. Hơn nữa còn được dùng để cấu hình các message và các node network.

Các Message được lọc có thể được cấu hình, theo đó nó có thể quan sát các khung ErrorFrames.

Trong hộp thoại Filter tất cả các đối tượng được có thể được cấu hình. Hơn nữa loại bộ lọc có thể được chuyển giữa Pass Filter hay Stop Filter bằng các nút chọn liên quan.

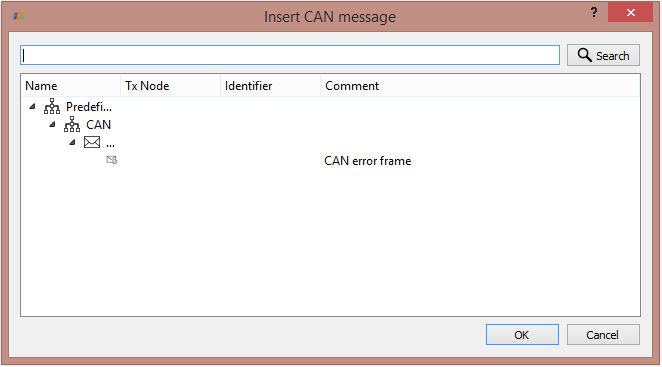


HÌNH 3.17: Giao diện hộp thoại Filter

Các trường của khối Filter:

* Database Name: Tên của Message được gửi lên từ Database.
* Name: Tên của Message( có thể tự thiết lập)
* Identifier: Thiết lập ID cho Message.
* Type: phạm vi kiểu dữ liệu được thiết lập.
* Chanel: Kênh được chọn để xử lý dữ liệu.
* Rx/Tx/TxRq Direction: chọn thuộc tính mong muốn cho việc truyền, nhận, gửi yêu cầu Message.

Khi click vào button [Insert CAN Message], hộp thoại [Dialog] sẽ hiện ra, trong đó có các message được thiết lập sẵn bạn có thể chọn một message bất kì để hiển thị thông tin.

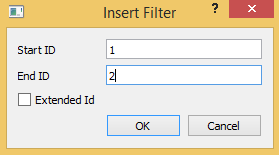


HÌNH 3.18: Giao diện hộp thoại Dialog với các Message được thiết lập trước

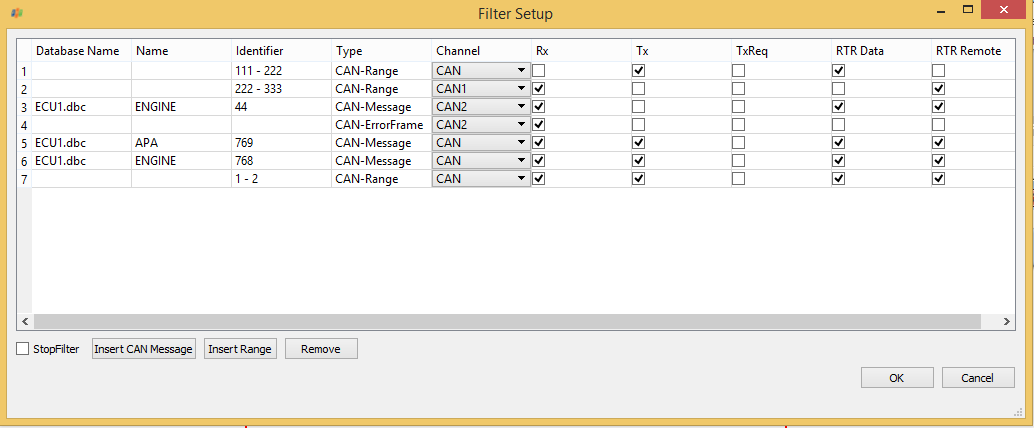
Khi click vào button [Insert Range], hộp thoại [Insert Filter] sẽ hiện ra, trong đó ta có thể nhập giá trị Start ID và End ID (ví dụ: 1 – 2).

* + Giá trị Start ID và End ID phải là số thực.
  + Giá trị Start ID phải nhỏ hơn giá trị End ID.

Khi nhấn Ok bạn sẽ load được dữ liệu với trường ID là 1-2.



HÌNH 3.19: Giao diện hộp thoại Insert Filter



HÌNH 3.20: Giao diện khi insert các Message

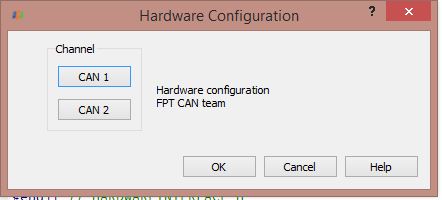
Dữ liệu này sẽ được xóa khi nút “Remove” được kích hoạt.

Click vào các nút lệnh Ok, Cancel, Help để thực hiện các hành động tương ứng.

# Giao diện khối Hardware

Giao diện khối Hardware được thiết kế dựa trên các lớp QPushbutton, Qgroupbox, QMenu…

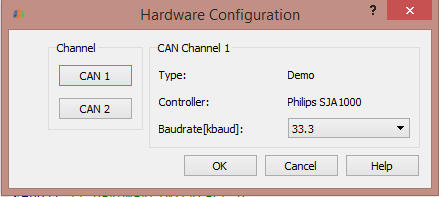
Khối Hardware được sử dụng để thiết lập việc chọn kênh đang mong muốn sử dụng, thiết lập tốc độ Baud cho kênh đang được chọn.



HÌNH 3.21: Giao diện hộp thoại Hardware

Khi nhấn vào nút nhấn CAN 1 (hoặc CAN 2) hộp thoại thiết lập chọn kênh được show cho người dùng thiết lập.

Thiết lập việc chọn Baudrate [kbaud] bằng việc chọn các chế độ sẵn có trong thiết lập (33.3, 50, 83.3, 100, 125, 500…).



HÌNH 3.22: Giao diện hộp thoại CAN với 2 kênh CAN 1 và CAN 2

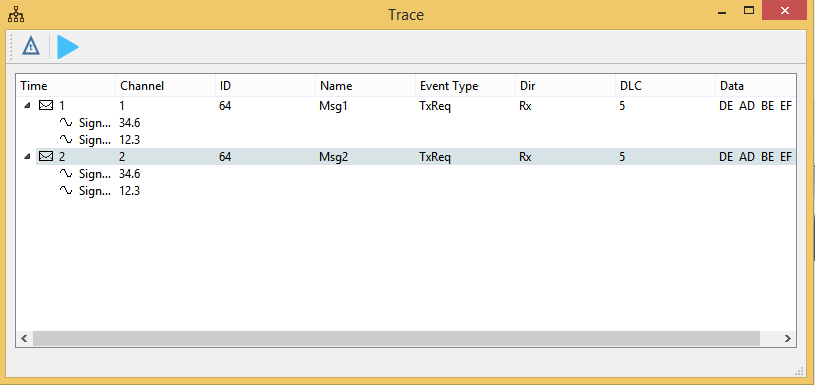
# Giao diện khối Trace

Giao diện khối Trace được thiết kế dựa trên các lớp Qtreeview, Qtablewidgetitem, QMenu…

Những Message được gửi đến khối Trace đều được đánh giá và hiển thị dưới dạng các đường dẫn thông tin trong cửa sổ Trace.

Trong cửa sổ Trace bạn có thể đặt tất cả các tùy chọn liên quan đến việc giải thích dữ liệu và những ảnh hưởng đến chế độ hiển thị hoặc cập nhật hành vi.

Bạn cũng có thể định cấu hình số cột và sắp xếp của chúng cũng như ghi nhãn các cột riêng cho cửa sổ Trace.



HÌNH 3.23: Giao diện hộp thoại Trace

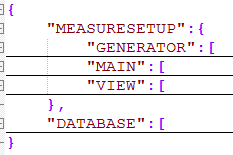
Các trường được thiết lập trong cửa sổ Trace.

* + Time: Thời gian tuyệt đối từ khi bắt đầu đo hoặc tương đối so với sự kiện trước đó.
  + Channel: Số kênh mà thông điệp được gửi/nhận.
  + ID (Identifier): CAN ID của tin Message.
  + Name: Tên biểu tượng của Message.
  + Dir(Direction): Thiết lập việc gửi/nhận Message
    - Tx: Gửi Message.
    - Rx: Nhận Message.
    - TxReq = Gửi yêu cầu.
  + DLC (Data Length Code): Chiều dài của trường dữ liệu theo biểu thức thập phân.
  + Data:  Dữ liệu theo biểu thức thập phân hoặc thập lục phân.

# **Cấu hình dữ liệu theo định dạng json**

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

**File Json thể hiện 3 khối chính của measuresetup.**



HÌNH 3.24: File Json ban đầu

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

**Mesuresetup:**



HÌNH 3.25: File Cấu hình theo định dạng json sau khi edit.

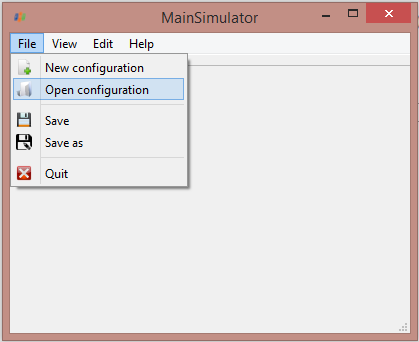
File json mô tả một đối tượng chứa cặp string “MEASURESETUP” và Value là một đối tượng chứa 3 cặp string-value:

* String “GENERATOR”, value là một mảng và mảng này chứa một đối tượng có string “CONTENT” – value là một mảng chứa 2 đối tượng, mỗi đối tượng có 4 cặp string –value, string “MSGS” lại chứa một mảng chứa 1 đối tượng có 3 cặp string-value, string “data” chứa 1 mảng là 1 đối tượng có 3 cặp string-value.
* String “MAIN”, value là một mảng và mảng này chứa 2 đối tượng, một đối tượng chứa 3 cặp string-value, một đối tượng chứa 4 cặp string –value.
* String “VIEW”, value là một mảng và mảng này chứa một đối tượng có 1 cặp string “CONTENT” – value là một mảng chứa 1 đối tượng có 3 cặ string- value.

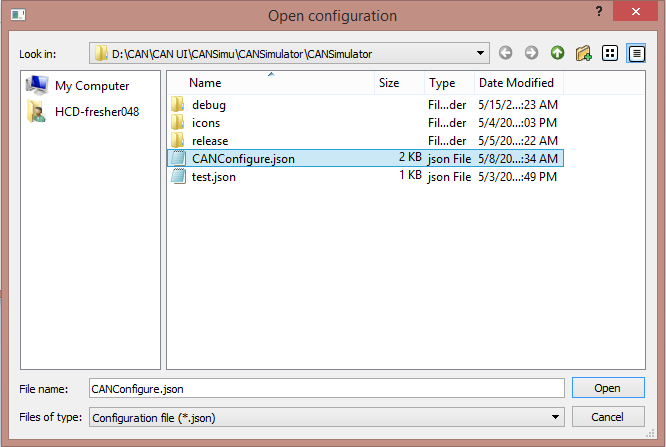
# **Hoạt động phần mềm mô phỏng CAN**

Hoạt động phần mềm mô phỏng CAN theo trình tự sau:

* Để khởi động phần mềm, ta thực hiện click vào File/Open configuration để load file cấu hình với định dạng “json” nhằm hiển thị giao diện Measure setup của hệ thống.

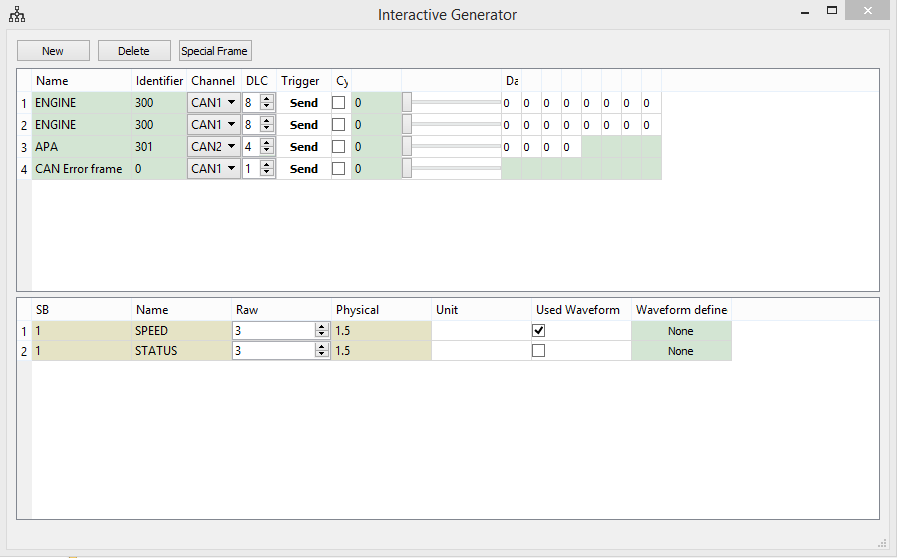


HÌNH 3.26: Hộp thoại MainSimulator



HÌNH 3.27: Load file cấu hình với định dạng Json

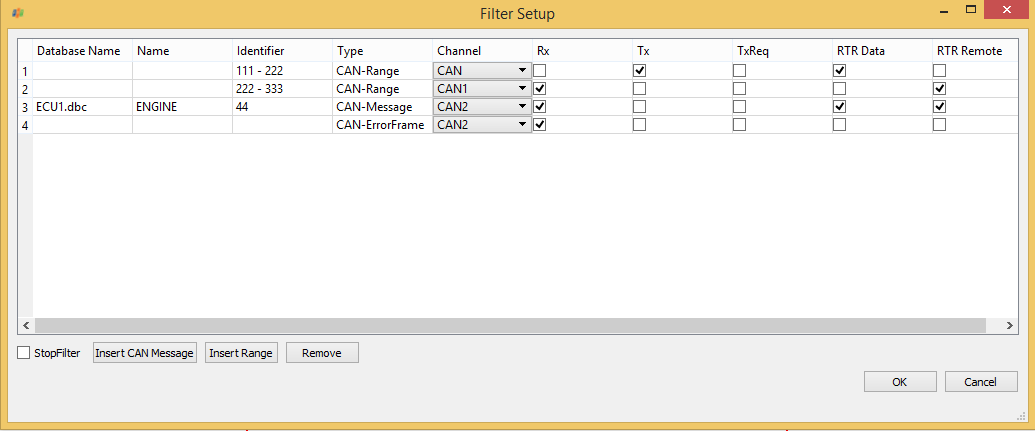
* Phần mềm tương tác với các message và signal từ cơ sở dữ liệu có sẵn (New/Open) thông qua hộp thoại giao diện chức năng IG (interative Generator). Hiển thị các giá trị trường Data, Channel và DLC. Người dùng cũng có thể điều chỉnh các thông số như :Cycle time [ms] của message, Raw Value và Phys Value của signal, đồng thời hiển thị wareform cho phù hợp.



HÌNH 3.28 Tương tác với các Message/Signal trong hộp thoại IG

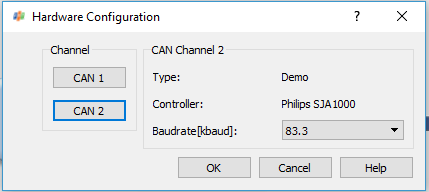
* Dữ liệu (các gói signal) sau khi được xử lý trong hộp thoại IG sẽ được đóng gói và tiếp tục gửi đến khối Filter nhằm giảm bớt kích thước của khung frame dữ liệu, rút ngắn thời gian xử lý thông qua việc thiết lập các thông số Start ID và End ID trong hộp thoại Data Raw. Dữ liệu sẽ được so khớp với ID (vừa thiết lập) của khối lọc, nếu trùng khớp chúng sẽ tiếp tục được gửi đi đến trung tâm để xử lý, những message có ID nằm ngoài giá trị ID thiết lập sẽ bị chặn lại (không được xử lí và đóng gói).

Bên cạnh đó, chúng ta cũng có thể lựa chọn kênh (CAN 1 hoặc CAN 2) và các thuộc tính mong muốn cho việc truyền nhận (Rx, Tx, TxRq).



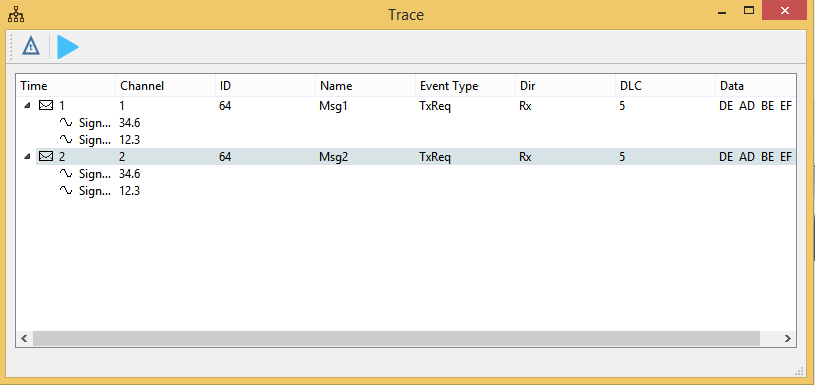
HÌNH 3.29 :Dữ liệu được xử lí trong khối Filter

* Gói signal sau khi được xử lý ở khối IG và Filter sẽ được đóng gói và tiếp tục chuyển đến khối trung tâm (khối Hardware). Tại đây ta có sẽ chọn kênh, thiết lập tốc độ baudrate [kbaud] bằng việc chọn các tốc độ tiêu chuẩn trong giao diện hộp thoại (33.3, 50, 83.3, 100, 125, 500). Hardware sẽ truyền những signal này xuống Board để xử lý, nếu những signal này là hợp lệ board sẽ gửi trả lại khối Hardware để đóng gói và truyền đến khối Trace, còn nếu signal nào gặp lỗi trên đường truyền thì sẽ được chờ để xử lý. Nếu quá thời gian chờ thì Board sẽ gửi phản hồi trở lại Hardware (No) và tiến hành xử lí những signal tiếp theo.



HÌNH 3.30 Thiết lập tốc độ Baudrate của cổng CAN

* Dữ liệu sau khi được phản hồi lại Hardware sẽ được truyền đến khối Trace để mô tả và hiển thị các thông tin như Time, Channel, ID (Identifier), Tx, Rx, DLC,…của message, signal trong message để giúp người sử dụng có thể kiểm tra, dễ dàng nhận xét, đánh giá kết quả sau những thiết lập và xử lý khi luồng dữ liệu khi qua các khối IG, Filter, Hardware.



HÌNH 3.31: Cửa sổ Trace hiển thị thông tin của message và signal

Đây cũng là mục đích chính của nhóm khi xây dựng phần mềm này.

# **Kết luận chương.**

Chương 3 đã trình bày tiến trình thực hiện đề tài “Thiết kế phần mềm mô phỏng CAN”… Gồm các bước phân chia nhiệm vụ, thi công và kết quả thực hiện Giao diện phần mềm của nhóm thực hiện tuy chưa thật thẩm mỹ và bắt mắt những giao diện chính và các hộp thoại đã thực hiện khá đầy đủ các chức năng cần thiết, trực quan và dễ dàng sử dụng. Giai đoạn đầu thực hiện xuất hiện khá nhiều lỗi, tuy nhiên sau quá trình kiểm tra và sửa lỗi, phần mềm mô phỏng CAN đã hoạt động đúng với yêu cầu đề ra.

# **KẾT LUẬN**

Sau hơn 3 tháng thực hiện đề tài “Thiết kế phần mềm mô phỏng CAN” chúng em đã cơ bản hoàn thành được đề tài với đã số các chức năng và yêu cầu đề ra.

* Phần mềm đã đáp ứng được khả năng tương tác với các message/signal từ cơ sở dữ thông qua các khối chức năng Interative Generator, Filter, Hardware và mô tả, hiển thị kết quả trên giao diện hộp thoại Trace. Giúp cho người sử dụng có thể dễ dàng kiểm tra, nhận xét thông tin của các dữ liệu được truyền lên từ thiết bị (CAN Device).
* Tuy chưa có giao diện thực sự nổi bật và bắt mắt, nhưng phần mềm đã có giao diện dễ nhìn, dễ thao tác và xử dung đối với người dùng.
* Phần mềm mô phỏng CAN đóng vai trò là một mắc xích quan trọng giúp tương tác giữa Phần mềm cơ sở dữ liệu CAN, CAN Device và PC Driver trong “Hệ thống mô phỏng CAN BUS”. Nhờ có phần mềm mô phỏng CAN, người dùng có thể gửi những message đúng định dạng (từ phần mềm cơ sở dữ liệu CAN) xuống thiết bị (PC Device), sau đó hiển thị được những message phản hồi thuận tiện cho việc kiểm tra, xác nhận và đánh giá thông tin dữ liệu.

# **HƯỚNG PHÁT TRIỂN ĐỀ TÀI**

Đề tài “Thiết kế phần mềm mô phỏng CAN” tuy đã đáp ứng được những chức năng cần thiết hỗ trợ cho “Hệ thống mô phỏng CAN Bus” nhưng vẫn cần chỉnh sửa và phát triển nhiều hơn nữa hệ thống được hoàn thiện và có thể hoạt động ổn định. Đồng thời, thiết kế giao diện bắt mắt và sinh động hơn (sử dụng QML).

Hỗ trợ thêm các khối chức năng trực quan và các tiện ích tốt hơn. Như thiết kế thêm các khối giao diện Statistic, khối logging, khối Data, View… Thiết kế đa dạng hóa thanh Menu với nhiều chức năng hỗ trợ người dùng hơn (như start, stop, pause, close all windows, restore all windows, layout). Đó cũng là hướng phát triển mà nhóm thực hiện đang và sẽ hướng tới.

# **TÀI LIỆU THAM KHẢO**

**[1]** “Qt Documentation”. [Online]. http://doc.qt.io/qt-5/(Accessed March and April 2017)

**[2]** Texas Instruments, “Introduction to the Controller Area Network (CAN)”, SLOA101B–August 2002–Revised May 2016

**[3]** Mark Mitchell, Jeffrey Oldham, Alex Samuel, “Advanced Linux Programming” Copyright © 2001 by New Riders Publishing, FIRST EDITION: June, 2001

**[4]** Vector, “User Manual CANoe Version 7.5”. © Copyright 2010, Vector Informatik GmbH. Printed in Germany.

**[5]** Nguyễn Mạnh Hùng, “Bài giảng ngôn ngữ lập trình C++”. [Online]. Xem tại:

http://www.e-ptit.edu.vn/hoctap/hoclieu/LTHDT.pdf (Accessed February 2017)

**[6]** “Introduction to Model/View Programming”.[online]. http://doc.qt.io/qt-4.8/model-view-programming.html (Accessed March 2017).

**[7]** “Introduction Signal and Slots”.[online].http://doc.qt.io/qt-5/signalsandslots.html (Accessed March 2017).

**[8]** “Introduction Json”.[Online]. http://www.json.org/ ( Accessed February and March 2017).

**[9]** Lê Thị Mỹ Hạnh, “Lập Trình Hướng Đối Tượng”, Đà Nẵng, 2002.

# **PHỤ LỤC 1**