Mục lục

[CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI 2](#_Toc481073725)

[1.1. Tổng quan về hệ thống 2](#_Toc481073726)

[1.2. Nội dung thực hiện. 2](#_Toc481073727)

[1.2.1. Mô hình giao tiếp hệ thống. 3](#_Toc481073728)

[1.2.2. CAN Device và PC driver. 4](#_Toc481073729)

[1.2.3. CAN Simulator Software. 5](#_Toc481073730)

[1.2.3.1. Database Edit Software 5](#_Toc481073731)

[1.2.3.2. Simulator software. 5](#_Toc481073732)

[CHƯƠNG 2: PHẦN MỀM CƠ SỠ DỮ LIỆU CAN 7](#_Toc481073733)

[2.1. Giới thiệu chung về QT Framework 7](#_Toc481073726)

[2.2. Các lớp trong QT được sử dụng trong Can Database. 7](#_Toc481073727)

[2.3. Ứng dụng các công cụ trên vào thiết kế phần mềm Can Database. 9](#_Toc481073728)

[CHƯƠNG 3: KẾT QUẢ THỰC HIỆN VÀ ĐÁNH GIÁ. 12](#_Toc481073734)

[3.1. Phần giao diện Can Database 12](#_Toc481073726)

[3.1.1. Kết quả. 12](#_Toc481073727)

[3.1.2. Đánh giá. 13](#_Toc481073728)

[3.2. Phần edit cơ sở dữ liệu. 13](#_Toc481073729)

[3.2.1. Kết quả. 13](#_Toc481073730)

[3.2.2. Đánh giá 15](#_Toc481073731)

[TÀI LIỆU THAM KHẢO 15](#_Toc481073735)

# CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

Cùng với sự phát triển không ngừng của khoa học công nghệ, công nghiệp ô tô 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.

# Tổng quan về hệ thống

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” và được hướng dẫn để thực hiện một hệ thống Automotive CAN Bus giao tiếp các ECU trong ô tô sử dụng đường truyền CAN bus, bao gồm CAN Device, PC Driver và CAN Simulator Software.



Hình 1.1. Hệ thống Automotive CAN Bus thực tế.

# Nội dung thực hiện.

Hoàn thành một hệ thống mô phỏng CAN Bus gồm các chức năng cơ bản, giao tiếp các ECU trong ô tô sử dụng đường truyền CAN bus, bao gồm CAN simulator device thực hiện hoạt động truyền nhận thông điệp tương tự như một ECU thực tế, Simulator softwave là phần mềm mô phỏng, hiện 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 hiện 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.

# Mô hình giao tiếp hệ thống.

PC driver

CAN device

CAN SimulatorSoftware

CAN

USB

Hình 1.2. Hệ thống gồm 3 phần chính là CAN Simulator Software, PC driver và CAN device.

* 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**

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.

**1.2.2. CAN Device và PC driver.**

* **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 dung giao thức USB và CAN. Board DK-TM4C123G hỗ trợ đầy đủ các yêu cầu trên và chúng ta sẽ tìm hiểu kỹ về giao thức CAN, USB trong phần này.

* **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

**1.2.3. CAN Simulator Software.**

Sử dụng database

Simulator software

Database Edit Software

Gọi để mở

**Hình 2.3 Software PC**

Hình 1.4. Software PC gồm: Database Edit Software và Simulator software.

# 1.2.3.1. Database Edit Software

* 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.

- Thêm/ sửa/xóa các 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.

# 1.2.3.2. Simulator software.

* 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 database edit software.
* Chức năng:
* Cấu hình hệ thống, cấu hình được 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.
* Trong IG (interactive generator) có thể sửa được dạng sóng của các signal: hình sin, răng cưa, hình chữ nhật,…
* Trong IG có thể setting cycle của từng thông điệp.
* Trong lúc đang simulate có thể edit nội dung message ở dạng raw data hoặc physical value.
* Trace window hiển thị các thuộc tính của message: ID, length, direction, data, signal value.

# 

# CHƯƠNG 2. PHẦN MỀM CƠ SỞ DỮ LIỆU CAN

**2.1. Giới thiệu chung về QT Framework :**

* QT Framework là một nền tảng xây dựng các ứng dụng chạy trên nhiều hệ điều hành khác nhau.
* Mục tiêu của các nhà phát triển nên Qt chính là tạo ra một nền tảng có khả năng thiết kế những phần mềm có thể chạy trên nhiều nền tảng phần mềm lẫn phần cứng khác nhau mà không phải thay đổi nhiều về code.
* Qt hổ trợ các nền tảng sau: Windows,Linux, OS X, iOS, Android, WinRT, …
* Qt hổ trợ ngôn ngữ C/C++/Java/Python/…
* Qt hổ trợ lập trình giao diện chuẩn theo mô hình model/view và hổ trợ các lớp lập trình giao diện động trong Qt Quick, giúp tạo giao diện chuyên nghiệp và sinh động hơn.

**2.2. Các lớp trong QT được sử dụng trong CAN Database:**

* Kiến trúc MVC và kiến trúc model/view trong QT:
* Mô hình MVC (Model-View-Controller) là một kiến trúc phần mền, nó giúp cho các nhà phát triển phần mền 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.
* 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 2.1. Cấu trúc model/view trong QT

* **Lí do lựa chọn mô hình model/view:** mô hình này chia phần mền 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.
* Signal và Slot trong QT:
* 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.

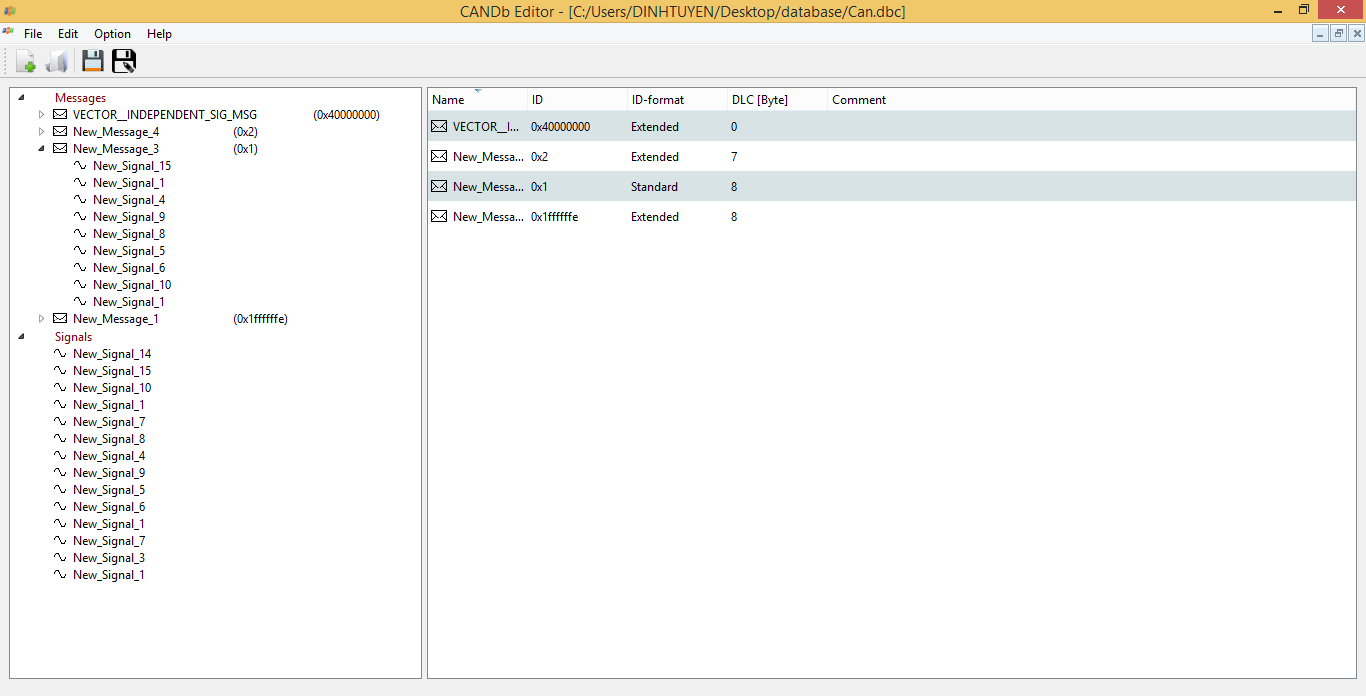
Connect (Đối tượng 1, SIGNAL (signal thuộc đối tượng 1), Đối tượng 2, SLOT (slot thuộc đối tượng 2));

Sau khi kết nối sẽ có một đối tượng phát ra signal và một đối tượng nhận signal đó và thực hiện một phương thwucs 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.

**2.3. Ứng dụng các công cụ trên vào thiết kế phần mền cơ sở dữ liệu CAN database software.**

* Phần mềm sẽ gồm 2 giao diện chính :
* Tại màn hình chính:



Hình 2.2 Giao diện chính phần mền cơ sở dữ liệu CAN bus

Giao diện chính của CAN database gồm 2 phần chính:

* Một danh sách dạng cây quản lí tên của các message và signal:

Trong phần này, ta sẽ sử dụng mô hình model/view gồm các thành phần:

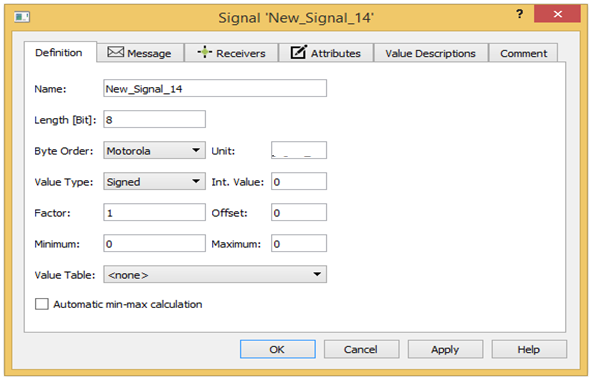
Model: kế thừa lớp QAbstractItemModel để tạo QMainDBTreeViewModel, trong đó chứa các hàm xử lí dữ liệu từ datasource.

View: sử dụng QTreeView để hiển thị dữ liệu ở trong model

* Phần hiển thị thông tin chi tiết gồm:
* Các bảng chứa thông tin chi tiết của các message(hiển thị khi nhấn chuột vào message ở QTreeView)
* Các bảng chứa thông tin chi tiết của các signal(hiển thị khi nhấn chuột vào signal ở QTreeView)
* Các bảng chứa thông tin chi tiết của các signal trong message (hiển thị khi nhấn chuột vào từng message , signal hoặc signal trong message ở QTreeView)

Cả 3 bảng trên ta đều sử dụng lớp QTableView phần View, Model của các bảng sẽ được thực hiện bằng cách kế thừa từ lớp QAbstractItemModel để tạo ra các TableModel tương ứng với 3 view ở trên. Trong mỗi bảng sẽ có các lớp Delegate tương ứng được kế thừa từ lớp QAbstractItemDelegate, khi người dùng tương tác với View, View sẽ thông qua delegate để chỉnh sửa được dữ liệu.

* Màn hình chỉnh sửa : message editor và signal editor



Hình 2.3 Giao diện signal editor

Giao diện chỉnh sửa sẽ có nhiều trang(tab), mỗi trang sẽ chứa nhiều thông số khác nhau , lúc nhấn Ok hoặc Apply thì thông tin tất cả các thông tin trong các trang chỉnh sửa sẽ được lưu lại .

* Đối với giao diện này ta sẽ sử dụng lớp QTabWidget trong QT, hổ trợ các phương thức addTab (QWidget \* page, const QString & label) để thêm các trang mới.
* Đối với nội dung trong từng trang ta sẽ sử dụng các lớp QLabel để đặt tên của thuộc tính, và sử dụng QComboBox, QLineEdit để hiển thị giá trị tương ứng với giá trị là các bảng lựa chọn hay text
* Để có được định dạng và sắp xếp hợp lí, ta sẽ sử dụng QLayout và các lớp con QHBoxLayout, QVBoxLayout để layout cho giao diện.

Khi người dùng thao tác chỉnh sửa xong và nhấn một trong các nút Ok, Apply, Cancel hay Help thì các signal tương ứng sẽ được tạo ra và các lớp Model và DBMainManager sẽ nhận những signal này thông qua các slot, từ đó dữ liệu sẽ được thay đổi tùy thuộc vào việc xử lí trong các slot.

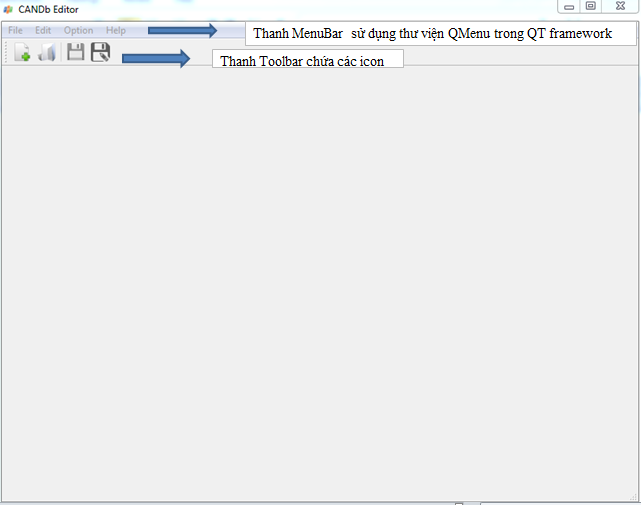
# 

# CHƯƠNG 3: KẾT QUẢ THỰC HIỆN VÀ ĐÁNH GIÁ

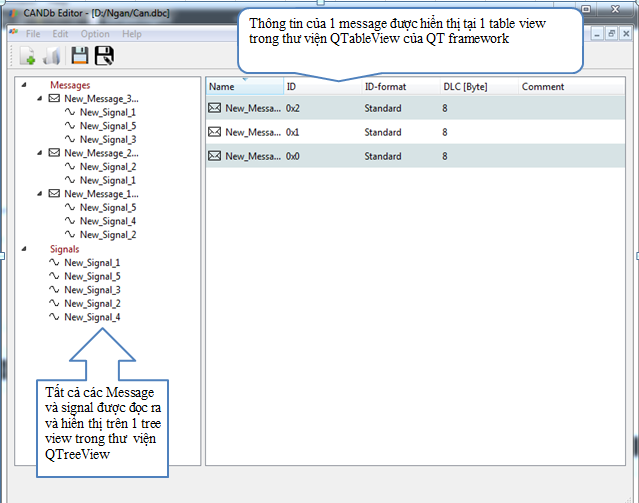
**3.1. Phần giao diện Can Database**

**3.1.1. Kết quả**

* Giao diện chính khi chạy chương trình

Hình 3.1: Giao diện chính CAN database

* Sau khi load database
* Tất cả các message và signal cùng thông tin trong file.dbc được đọc ra

Hình 3.2. Biểu diễn thông tin Message và Signal

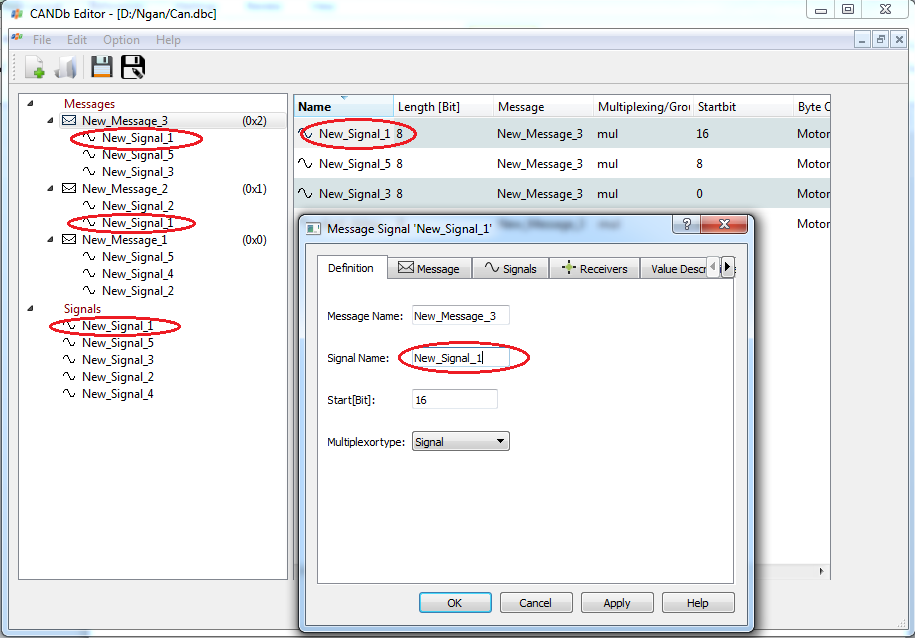
**3.1.2. Đánh giá**

* Giao diện chính vẫn còn 1 vài chức năng chưa hoàn thiện
* Hiện tại chỉ thực hiện được các chức năng như mở file, tạo file và lưu file, chức năng setting trong Options vẫn chưa được thực thi.
* Khi load database, giữ liệu đã đọc được chính xác với file .dbc có sẵn, các thông tin về message, signal đã được lưu trữ đúng vị trí

**3.2. Phần edit cơ sở dữ liệu**

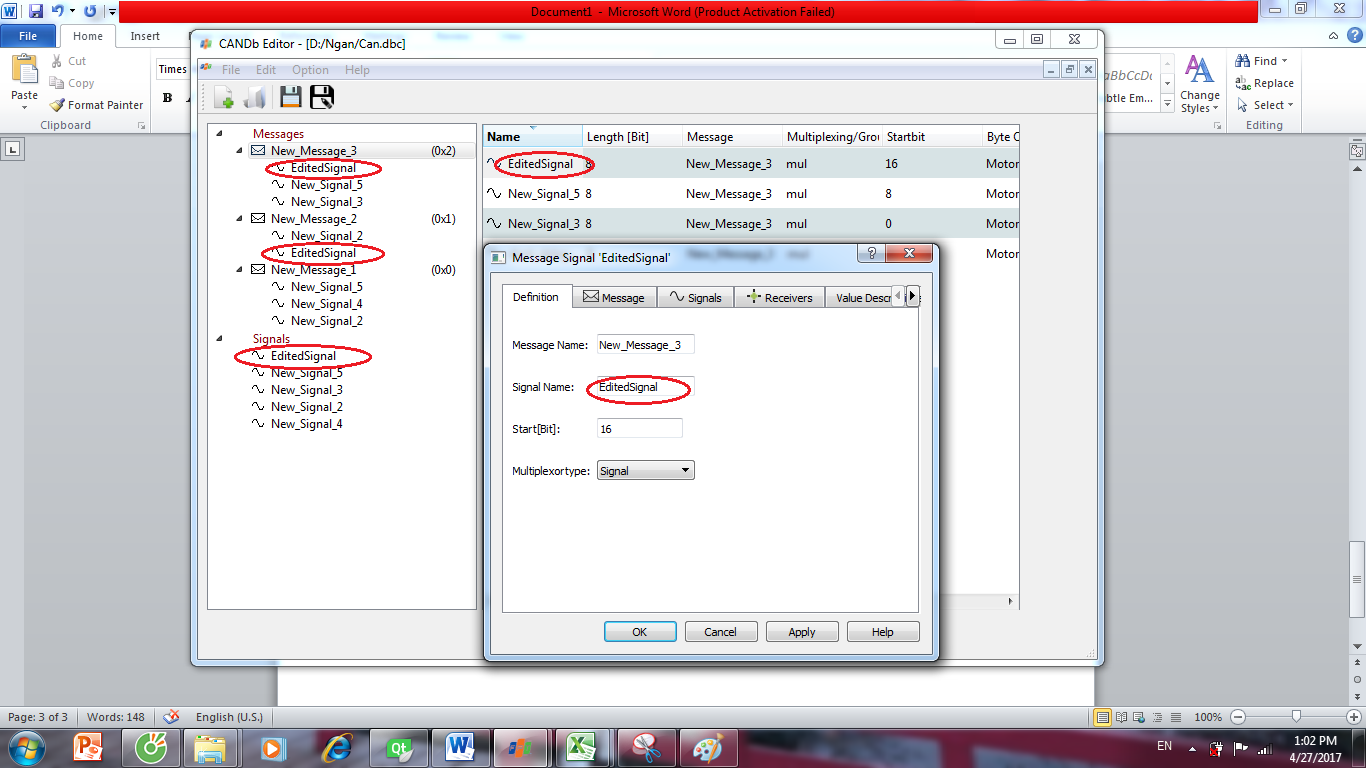
**3.2.1. Kết quả**

* Khi ta click đôi chuột pải vào table view thông tin cụ thể của các signal sẽ được hiển thị tại 1 dialog khác để tiện theo dõi vào chỉnh sửa



Hình 3.3. Thông tin Signal trước khi edit

* Sau khi chỉnh sửa phần Signal Name tại Dialog và nhấn OK,
* Thông tin của Signal đó lập tức thay đổi ở các vị trí:
* Tại Table View
* Tại Tree View
* Tại Source



Hình 3.4. Thông tin Signal sau khi edit

* Tương tự đối với các thông tin khác của Message hoặc signal ta đều có thể thay eidt được tại phần mềm

**3.2.2. Đánh giá**

* Dữ liệu của các Message và Signal đã được edit
* Tuy nhiên có 1 trường hợp eidt ngay tại Signal chứa trong Signal thì dữ liệu vẫn chưa được thay đổi và vẫn đang trong quá trình thực hiện
* Nút Apply cho nhiều tab cùng 1 lúc vẫn đang trong quá trình thực hiện nhưng chưa thành công.

# TÀI LIỆU THAM KHẢO

1. *Vector.com*
2. *Qt.io*
3. *Json.com*
4. *Ti.com*