UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

CRET - CanBus Reverse Engineering Toolkit

Presentado por Adrián Marcos Batlle
en Universidad de Burgos — 20 de enero de 2019
Tutor: Álvar Arnaiz-González

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Álvar Arnaiz-González, profesor del departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Adrián Marcos Batlle, con DNI 71310384B, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado CRET - CanBus Reverse Engineering Toolkit.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 20 de enero de 2019

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

El bus CAN (CAN-Bus - Controlled Area Network) es un protocolo de comunicación utilizado en los desarrollos de multitud de sectores críticos como la industria, la automoción y la aviación entre otros, para la comunicación entre los componentes internos que forman la infraestructura desarrollada.

Los datos que fluyen por dicho bus son propiedad de cada uno de los fabricantes a pesar de ser un protocolo libre. El desarrollo de esta herramienta viene motivado a realizar un análisis de la información que fluye por estos buses, así como su clasificación y monitorización en tiempo real.

Para dicho objetivo, se ha desarrollado tanto una parte de software (para el análisis y clasificación de los datos), como una parte de hardware para conectarse a dicho bus y poder acceder a los datos que fluyen a través del mismo.

Descriptores

CAN-Bus, Automoción, Industria, Aviónica, Ingeniería inversas, Análisis de protocolos.

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

Can-Bus, Automotive, Industry, Avionics, Reversing, Protocol analysis.

Índice general

Índice de figuras

Índice de tablas

Introducción

Todos los vehículos que utilizamos en el día a día, maquinaria utilizada en las empresas, el sector náutico o de la aviación utilizan el bus CAN para la inter-conexión de los componentes electrónicos que hacen funcionar dichas máquinas.

Durante los años se han desarrollado nuevos protocolos, todos ellos basados en (habiéndose incrementado, por ejemplo, la velocidad en los buses actuales), pero utilizando las bases del protocolo que en su primer desarrollo.

El estándar CAN (CITA) es

El estándar del bus CAN únicamente hace referencia a las dos primeras capas del protocolo, la capa física, y la capa de enlace de datos (siguiendo el modelo OSI).

A través del uso de esta herramienta, sería posible identificar y clasificar los datos que los distintos elementos del vehículo analizado comparten entre ellos, para su funcionamiento. De esta manera, por ejemplo, si necesitásemos realizar una aplicación para la monitorización de un vehículo, no sería necesario introducir nuevos sensores (para la velocidad, las revoluciones del motor, el GPS), sino que estos datos serían extraídos del bus CAN, ahorrando costes y posibles problemas.

1.1. Estructura de la memoria

Introducción: Descripción breve sobre el proyecto, motivación por la que se ha realizado y soluciones propuestas. Estructura de la memoria y listado de materiales adjuntos proporcionados.

Objetivos del proyecto: Exposición de los objetivos, clasificados en objetivos generales, objetivos técnicos y objetivos personales.

Conceptos teóricos: Conceptos básicos y necesarios para entender el propósito del proyecto.

Técnicas y herramientas: Metodologías y herramientas utilizadas durante el desarrollo del proyecto.

Trabajos relacionados: Aplicaciones, proyectos y empresas que ofrecen soluciones en el mismo campo que el estudiado.

Conclusiones y líneas de trabajo futuras: Conclusiones a las que se ha llegado tras la realización del proyecto, así como mejoras y futuro desarrollo de la aplicación.

1.2. Materiales adjuntos

Los materiales adjuntos a la memoria son los siguientes:

- Aplicación desarrollada en Java: CRET.
- Fotos del hardware desarrollado.
- Esquemas del hardware desarrollado.
- JavaDoc.

Además, los siguientes recursos están accesibles a través de internet:

- Repositorio del proyecto TODO—-.

Objetivos del proyecto

A continuación se definen los objetivos del proyecto realizado, estructurados en tres secciones:

2.1. Objetivos generales

- Desarrollar una aplicación para el análisis y monitorización de los datos que fluyen por el bus CAN.
- Desarrollo de un hardware libre el cual permita conectarse a dicho bus de datos y monitorizar distintas velocidades de forma simultanea.

2.2. Objetivos técnicos

- Desarrollar un *hardware* propio desde 0 siguiendo la metodología para el desarrollo del mismo, así como su producción y montaje.
- Desarrollar una aplicación en Java y JavaFX.
- Aplicar la arquitectura MVP (*Model-View-Presenter*) en el desarrollo de la aplicación.
- Utilizar Zenhub (basado en el método Kanban) para realizar un seguimiento y gestión del proyecto.
- Utilizar Git (en la plataforma GitHub) para realizar un control de versiones de software.
- Utilizar librerías para la recolección de datos del bus CAN.

2.3. Objetivos personales

- Profundizar en el conocimiento de hardware y en el desarrollo del mismo.
- Adquirir conocimiento sobre el funcionamiento del bus CAN en distintos escenarios.
- Profundizar en el conocimiento del análisis de datos y monitorización en tiempo real.

Conceptos teóricos

Las partes del proyecto con mayor desconocimiento y complejidad están enfocadas principalmente en el funcionamiento del bus CAN y en el desarrollo del hardware, el cual requiere de unos conocimientos básicos y unas metodologías específicas las cuales serán detalladas a continuación:

Bus CAN

El bus CAN (*Controller Area Network*) es un protocolo desarrollado para la comunicación entre los distintos micro-controladores y dispositivos que son necesarios para el funcionamiento de un vehículo. Este protocolo no necesita de un host principal, sino que sigue una topología de tipo "bus".

Este protocolo está basado en el uso de mensajes para el intercambio de información entre los distintos dispositivos que lo componen.

Es utilizado en multitud de escenarios como la aviación, la navegación, la automatización industrial, instrumentos médicos, maquinaria pesada y ascensores entre otros.

Nodo: Cada uno de los dispositivos físicos que están conectados a la red CAN. Al menos es necesario que existan dos nodos conectados a la red para que se produzca una comunicación.

Cada uno de los nodos es capaz de enviar y recibir mensajes, pero no simultáneamente.

FOTO DE UN SOLO NODO.

Frame: Cada uno de los mensajes enviados a través de la red CAN. Estos contienen 3 campos en los que nos centraremos: ID, Length y Data.

ID: Se trata del identificador del Frame. A través de el, se establece la prioridad que tiene ese Frame durante el acceso al bus de datos.

Length: En este campo, se define la longitud del campo de datos del Frame. En las versiones estándar, esta longitud puede ir desde 0 hasta 8 bytes de datos.

Data:

Bitriate: Se trata de la velocidad a la que los datos son transmitidos a través del bus. Todos los nodos deben de transmitir a la misma velocidad. Esta, puede cambiar dependiendo de los sistemas. Los valores más comunes son 10000 bit/s, 20000 bit/s, 50000 bit/s, 100000 bit/s, 125000 bit/s, 250000 bit/s, 500000 bit/s, 800000 bit/s y 1000000 bit/s.

Además, el estándar dispone de varias medidas para la detección de errores y seguridad las cuales no son relevantes para la exposición de este proyecto.

A través del bus CAN, es posible el flujo de datos a distintas velocidades de forma simultanea, a través de los mismos cables. Lo que no es posible, es la lectura de datos a dos velocidades distintas, por lo que se desarrolló un hardware específico para dicha tarea.

Siguiendo el modelo OSI, el estándar bus CAN especifica únicamente las dos primeras capas:

Capa física: Esta capa define como son transmitidas las señales a nivel eléctrico.

Capa de enlace de datos:

CanBus Esquemas can Explicación ID Explicación Velocidad Explicación hardware Explicación señales low-level Explicación capas del canbus Explicación modo activo-listenonly

Explicación conceptos de hardware:

PCB:

Footprint: Hace referencia al espacio necesario para colocar un elemento del hardware.

Gerber: Se trata de un formato de fichero el cual contiene la información necesaria para realizar la fabricación de una PCB (Placa de Circuito Impreso).

3.1. Secciones

Las secciones se incluyen con el comando section.

Subsecciones

Además de secciones tenemos subsecciones.

Subsubsecciones

Y subsecciones.

3.2. Referencias

Las referencias se incluyen en el texto usando cite [**?**]. Para citar webs, artículos o libros [**?**].

3.3. Imágenes

Se pueden incluir imágenes con los comandos standard de LaTeX, pero esta plantilla dispone de comandos propios como por ejemplo el siguiente:

Figura 3.1: Autómata para una expresión vacía

3.4. Listas de items

Existen tres posibilidades:

- primer item.
- segundo item.

1. primer item.

Herramientas	App AngularJS	API REST	BD	Memoria
HTML5	X			
CSS3	X			
BOOTSTRAP	X			
JavaScript	X			
AngularJS	X			
Bower	X			
PHP		X		
Karma + Jasmine	X			
Slim framework		X		
Idiorm		X		
Composer		X		
JSON	X	X		
PhpStorm	X	X		
MySQL			X	
PhpMyAdmin			X	
Git + BitBucket	X	X	X	X
MikTEX				X
TEXMaker				X
Astah				X
Balsamiq Mockups	X			
VersionOne	X	X	X	X

Tabla 3.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

2. segundo item.

Primer item más información sobre el primer item.

Segundo item más información sobre el segundo item.

-

3.5. Tablas

Igualmente se pueden usar los comandos específicos de LATEXo bien usar alguno de los comandos de la plantilla.

Técnicas y herramientas

Metodologías

Scrum: A través del uso de esta metodología, se adopta una estrategia de desarrollo incremental, en lugar de realizar una planificación y ejecución completa del proyecto desde el principio.

Pomodoro: El uso de este método de gestión de tiempo ayuda a incrementar la productividad. Esta técnica consiste en realizar una división del tiempo en fragmentos. Estos fragmentos son periodos de tiempo de 25 minutos (llamados pomodoros). A su vez, estos fragmentos están separados entre ellos por pausas de 5 minutos. Cuando se han realizado 4 periodos de 25 minutos (es decir, 4 pomodoros), se realiza una pausa más larga de unos 30 minutos.

Patrones de diseño: MVC (Model, View, Controller): Aplicando este patrón de arquitectura, se utilizan 3 componentes, las vistas, los modelos y los controladores, de tal forma que se separa la lógica de la vista de la aplicación. De esta forma, si realizamos una modificación en una parte de nuestro código, la otra parte no se ve afectada. Por ejemplo, si modificamos la base de datos, solo modificaríamos el modelo encargado de esa acción, sin tocar el resto de la aplicación.

Control de versiones:

Git: Con el uso de este sistema de control de versiones, se realiza de manera eficiente y confiable el mantenimiento de versiones de aplicaciones o proyectos, cuando estas poseen una gran cantidad de código fuente. Además, guarda un registro de los cambios realizados en la aplicación o proyecto, de forma que podemos volver a una versión anterior en cualquier momento.

Hosting: GitHub: Se trata de una plataforma de desarrollo colaborativo en la cual es posible alojar proyectos de forma gratuita a través Git (el sistema de control de versiones utilizado por GitHub). Con el uso de una cuenta de estudiante, es posible crear un repositorio privado en esta plataforma.

Gestión del proyecto:

Zenhub: Es una herramienta integrada con GitHub desde la cual se puede realizar una gestión de proyectos. Para ello dispone de un tablero con tarjetas, las cuales son tareas del proyecto. Estas tareas pueden estar en diversos estados a lo largo del proceso. (RELLENAR MÁS)

Entorno de desarrollo:

Eclipse: Es un IDE para el desarrollo de aplicaciones Java. Dispone de un gran número de plugins para facilitar el desarrollo, refactorización y revisión del código fuente.

SceneBuilder: Es una interfaz de usuario con la cual a través del método darg and drop, es posible maquetar interfaces gráficas las cuales son utilizadas por JavaFX. Esa información es almacenada en ficheros FXML, un formato especial de XML.

SQLite:

Documentación:

TextMaker: Editor de LaTeX multiplataforma, el cual integra diversas herramientas necesarias para la generación de documentos LaTeX.

Impresión 3D:

Solidworks: Software de tipo CAD (Computer Aided Design) el cual es utilizado para modelar piezas en 3D. En este caso ha sido utilizado para el desarrollo de la caja que contiene el hardware utilizado para conectarse al bus CAN.

Ultimaker Cura: Se trata de un silicer gratuito para la impresión 3D. Es el encargado de transformar el diseño 3D en un formato que la impresora 3D sea capaz de procesar, en este caso un fichero gcode.

Librerías:

USBtinLib: Librería utilizada para mediar entre el hardware utilizado para conectarse al bus CAN y Java. Es la encargada de realizar al conexión, además de que nos permite tanto enviar como recibir datos del bus.

sqlite-jdbc: Librería para realizar conexiones con bases de datos SQLite.

JSSC (Java Simple Serial Connector): Librería multiplataforma a través de la cual es posible obtener los puertos serie de comunicación disponibles en el equipo en el que se ejecute.

JavaFX: Se trata de una librería para Java, enfocada a la creación de interfaces gráficas.

Medusa: Conjunto de componentes los cuales pueden ser añadidos a JavaFX para ampliar sus funcionalidades y características visuales.

Diseño de hardware:

KiCAD: Está compuesto por un conjunto de herramientas para el diseño de circuitos electrónicos. Esto incluye la creación de esquemas para circuitos electrónicos así como la conversión de esos esquemas a diseños de PCB (Printed Circuit Board). Además nos permite exportar los ficheros necesarios para la producción de dichos componentes.

PAGINA WEB JLCPCB?

CAN bus:

canAlyser3 mini: Herramienta privada desarrollada para la utilización de un hardware concreto (USB-to-CAN), la cual ha sido utilizada para la generación de tramas (Frames) CAN para la realización de las pruebas y los test correspondientes durante el desarrollo de la aplicación.

SocketCAN: Se trata de un conjunto de herramientas y drivers .ºpen Source"para el bus CAN, desarrollado para Linux. Es posible la creación de interfaces CAN virtuales, o añadir hardware CAN como si de una tarjeta de red se tratase.

Can-utils: Se trata de un set de herramientas con las cuales podemos interactuar con el bus CAN. A continuación se describen las dos herramientas utilizadas y más interesantes.

candump: Nos permite capturar todo el tráfico que es transmitido a través del bus, y almacenarlo en un fichero.

canplayer: Esta herramienta nos permite generar tramas (Frames) y enviarlas al bus CAN. También es posible enviar un fichero capturado anteriormente con la herramienta candump.

https://tomato-timer.com/

Esquema global del programa, con input de can, etc

Aspectos relevantes del desarrollo del proyecto

Inicio del proyecto

Motivacioens ,etc logo

Metodologías

Formación

Desarrollo del software

Desarrollo de la app

Desarrollo del hardware

Trabajos relacionados

Cada fabricante dispone de sus propias herramientas (al igual que muchos de ellos disponen de su propio hardware). Estas herramientas, como se ha mencionado en la introducción, tienen un coste muy elevado en la mayoría de los casos.

Las principales ventajas de este proyecto sobre los otros mencionados anteriormente son:

El hardware desarrollado puede ser producido por cualquier persona, ya que los esquemas y los ficheros necesarios para su producción son open source.

Aplicación multiplataforma, compatible tanto con sistemas Windows como Linux.

No es necesaria la instalación de drivers en el equipo en el que se va a utilizar la herramienta.

Es la única herramienta libre que permite "graficar"los datos que fluyen por el bus CAN, así como su identificación y posterior monitorización.

Las principales desventajas son:

Actualmente solo funciona con un hardware en concreto. Como se menciona en la sección –Lineas futuras de trabajo–, el desarrollo de este proyecto continuaría con la incorporación de un módulo para el soporte de los drivers SocketCAN en sistemas linux.

Conclusiones y Líneas de trabajo futuras

Conclusiones:

Lineas de trabajo futuras:

Es interesante proseguir con el desarrollo de la herramienta y adaptarla para su utilización con los módulos de Linux SocketCAN.

De esta manera no es un requisito necesario el uso de un hardware en concreto, ya que SocketCAN haría de intermediario entre cualquier software soportado y la aplicación.

Además sería posible la creación y utilización de interfaces virtuales dentro del equipo, para realizar diversas pruebas.