Reverse Engineering of an OEM specific CANBUS and creating an application to control the instrument cluster.

Billy Irvin
C00251069
Richard Butler
South East Technology University
17 April 2023

Table of Contents

Abstract

A vehicle's internal CAN network is made up of a number of parts e.g. sensors,actuators and ECUs. With this project, I want to take control of the CAN network and alter its message structure so that the vehicle can perform activities it wouldn't normally be able to. I'm then going to develop a desktop application that will enable a user to have the ability to control the CAN bus by sending their own CAN messages onto the CAN bus and throughout the CAN network in order to accomplish this goal of reverse engineering and manipulating messages on a CAN network.

Introduction

Vehicles today can be very complex, with electronic and mechatronic systems in them that monitor and control lots of aspects of the car. Examples of these are the powertrain, ABS, and Comfort control systems which work together to provide better comfort and safety for the driver of the vehicle. The goal of this research report is to understand CAN and how it implements the protocols that CAN use and the message format, the prioritisation of messages also the reverse engineering of OEM-specific implementation. I will be looking at Mazda for my project and then create an application to control the sensors and actuators of a Mazda dashboard that I reverse-engineered. The systems communicate over CAN (Controller Area Network) buses which carry essential information about the workings of the car. We can learn by tapping into the CAN bus of the car and monitoring its behaviour. However, it's not as easy to tap into the CAN bus of the car and read the data. Although the CAN protocol is standardised, the implementation is different with each manufacturer and model of the car they make. The manufacturers can use any ID they want to represent the data that flows over the CAN bus. The reason why I chose this project is that I have a significant interest in physical hacking, and cars are things we depend on a day-to-day basis, so I thought, why not see what I can do?

Reverse Engineering

Reverse engineering is taking something like an object or software and analysing and gaining knowledge about the workings to try and enhance it, duplicate, or even manipulate it. For my project, I will be trying to manipulate the sensors and actuators of a car by reverse engineering the messages that are being sent across the CAN bus. (Lutkevich, B., 2021)

History of reverse engineering

Reverse engineering has been going on since 1600 the first object that was reversed is the telescope as Galileo heard about the telescope and gathered information about the telescope and figured out how to make one, but the original inventor was Hans Lipperhey who submitted a patent for the telescope in 1608. (History of reverse engineering, no date)

CAN Bus

Many different implementations utilise the underlying CAN message-based protocol in a vehicle. CAN allows separate systems, devices, and controllers in a network to communicate with one another. In layman's terms, CAN is a messaging service for you to communicate with components of a vehicle. A bus is a communication system that transports data between different ECMs in a car.

History of CAN bus

CAN was first developed by Bosch in 1983 and was called Control Area Network. It was then later codified into ISO 11898-1 standard. CAN entered the Automotive industry in 1986, only a year later in 1987 when intel introduced the CAN controller chips. Bosch in 1991 released CAN 2.0 which included a standard frame of 11 bit and an extended frame of 29 bit. CAN bus them became an international standard of ISO 11898 in 1993. Further on in 2003 ISO 11898 became a standard series. In 2012 came the release of CAN FD 1.0 which provided flexible data rate then CAN FD became standardised in ISO 11898-1. The latest update with CAN is the physical layer is up to 5Mbits/s and has been standardised in ISO 11898-2, in 2016. (CSS Electronics, 2021)

How CAN Bus functions

The CAN bus divides its traffic on to two lines CAN high (CANH) and CAN low (CANL). When a signal crosses one of these lines, the voltage on the high line is raised from 2.5 to 3.5 volts, and at the same time, the voltage on the CANL line is decreased by the same amount, making it 1.5 volts from 2.5 volts. As a result, each message's appropriate micro controller assigns an ID number to each message because a controller can send out many messages at once like how the speed of the car will increase and decrease constantly. A CAN message usually has an ID and then eight bytes of data after it which is in hex format. (CSS Electronics, 2021)

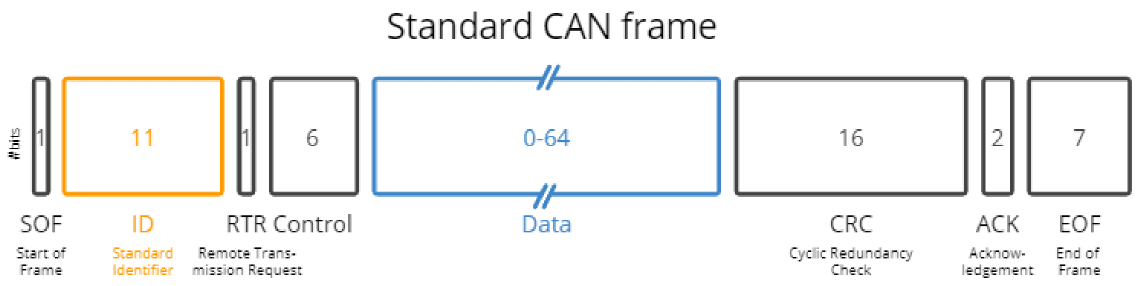

Figure 1: Standard CAN Frame

1. SOF – Start of Frame.
2. ID – Frame identifier.
3. RTR – Remote Transmission Request.
4. Control – control contains the IDE (Identifier Extension Bit) which is zero.
5. Data – This is the pay dirt where you can read this and decode it to find out information.
6. CRC – Cyclic Redundancy Check makes sure the data integrity is in tacked.
7. EOF – End of Frame.

Types of CAN Communication Protocol

Ther is four types of CAN communication Protocols.

1. High speed CAN – high speed CAN has a transmission rate up to 1M bits.
2. Low speed CAN – low speed CAN has a transmission rate up to 125 K bauds.
3. Single wire CAN – Single wire CAN has a transmission rate up to 83.3 K bauds.
4. Software selectable CAN – Software selectable CAN has two ports that could be used for High-speed CAN, Low speed CAN or Single wire CAN.

CAN vs Ethernet

CAN	Ethernet
CAN has a transfer speed of 1 Mbit/sec	Ethernet has a transfer speed of 10/100 Mbit/sec
Can only send 8 bytes per data frame	Can send unlimited amount of data
Secure communication	No secure communication
Detects and recovers from error within 23 micro-seconds at 1 Mbit/sec	Detects and recovers from error within 1 milli-seconds at 100 Mbit/sec
Collison Free and Predictable arbitration	unpredictable arbitration time

As described above in the table ethernet has a higher transfer rate compared to CAN and can send more data but CAN offers a faster time to recover from errors when they are detected. It also provides a secure communication compared to ethernet due to it being collison free for example a message, with CAN if you send the message and it fails to get to the actuator it will automatically resend the message. With ethernet there's no guarantee it will send the message again which is not very safe when dealing with a vehicle that goes at speed. This brings me to my next point, CAN is a collision free network which means if you have an important message being sent out and a not so important message CAN takes the important message first sends it through and once its sent it will then send the not so important message. An example would be if you are driving a vehicle you want to be able to monitor your speed which would be an important message but then you decide to change the radio channel which would not be so important CAN will take the message for the speed of the vehicle before the message to change the radio channel.

OBD

OBD in layman's terms is a device that is in your vehicle that diagnosis what's happening like if your trye pressure goes low it will indicate a light on the dashboard, so you know. It also holds the information about the status of the vehicle. Mechanics use this information when they want to find out precisely what faults are happening in the car to get a greater understanding of the problem. (CSS Electronics, 2021)

History of OBD

OBD first came about in 1968 by Volkswagen as the first OBD computer system that had scanning capability built in. Then in 1975 Datsun started to put OBD into consumer vehicles. In 1980 General Motors developed and configured an interface and protocol to test the Engine Control Unit. Eight years later in 1988 California air resources board made it mandatory that all vehicles sold from now on in California must have OBD. Then in 1994 California air resources board left OBD and made OBD 2 mandatory. 1996 all vehicles in the USA need to have OBD 2. Then in 2001 the EU made it compulsory for all petrol vehicles to have OBD2 and in 2004 all diesel vehicles had to have OBD2. Then 2006 Australia and New Zealand joined and made all their vehicles have OBD2. (CSS Electronics, 2021)

Difference between OBD I and OBD II

OBD I	Comparison	OBD II
Before 1995	Year of support	After 1996
Low	Popularity	High
Low	Accuracy	High
Basic	Features	Advanced
Over Console	Installation	Over Bluetooth or Wi-Fi

Structure of OBD II

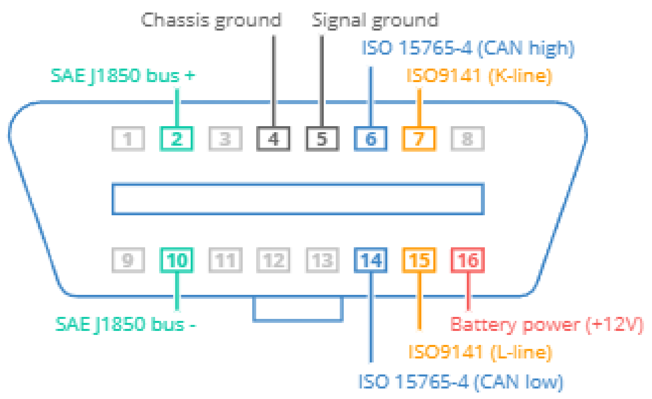

Figure 2: OBD II

As shown in the above figure is the OBD2 interface pin six is for High CAN. Pin fourteen is for Low CAN. We then have ground on pin five with these three pins you will be able to read the messages your vehicle is sending out. With this information I can decode these messages to find out the function they perform and send the message back in to determine the effect on different actuators. (CSS Electronics, 2021)

OBD II Message frame

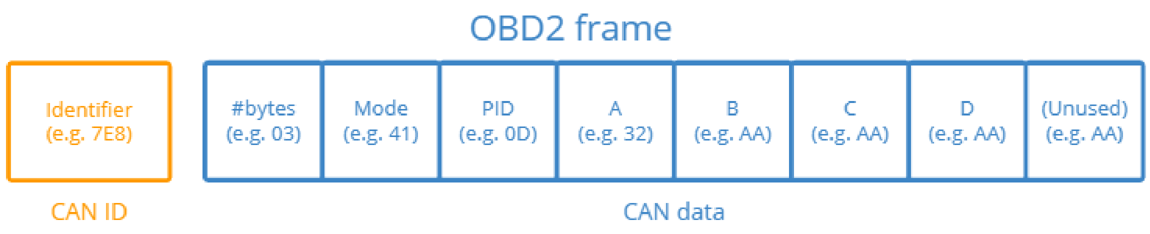

Figure 3: OBD II Frame

In the figure above is an OBD2 frame this is what I am trying to decode as you can see it starts off with the CAN id this id will be a specific component within the vehicle example would be like the switch for your headlights. Then it's followed by 8 bytes of data called the CAN data this is where it can tell you about the state of the light weather they are on or off. (CSS Electronics, 2021)

CAN Adapters

There is lots of data readers out there on the market the ones I will talk about is canable, PeakCAN and Ardunio uno with CAN-BUS shield.

CANable

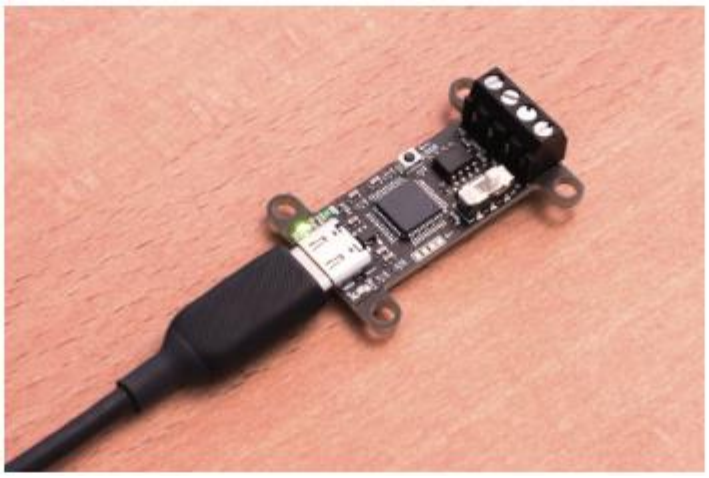

Figure 4: CANable adapter

CANable is small and compact its open source it has four pins. A pin to read all the High CAN messages. A pin to read all the Low CAN messages. A pin for ground and the last pin for power. The CANable is compatible with windows, Linux and has libraries to use python with it. It has a prebuilt GUI called CANgaroo which allows you to view message and send your own messages as well.(An open-source USB to can adapter, no date)

PeakCAN

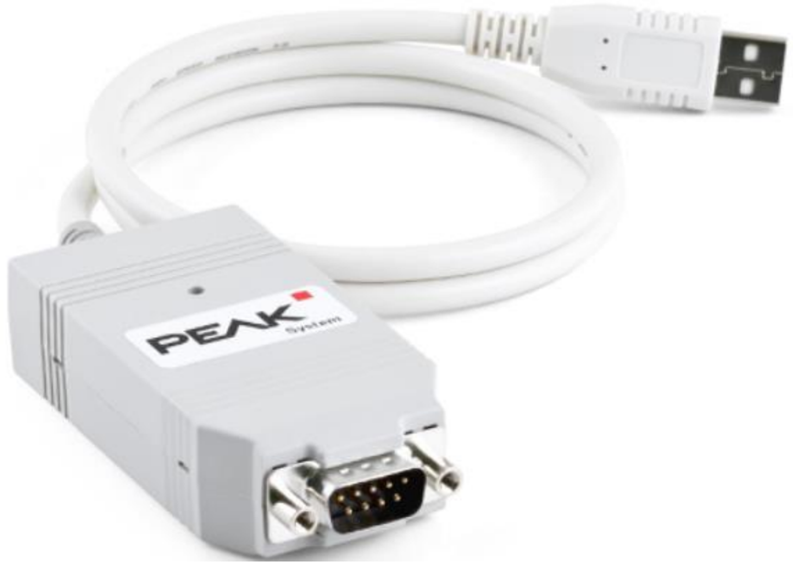

Figure 5: PEAKCAN adapter

PeakCAN has a serial connection interface giving it nine pins. Pin two on the adapter is CAN low and pin seven is CAN high pins three and six are ground pins. The adapter is compatible with windows, linux and python, C#, C++, Delphi, and java. The PEAKCAN also has their own software for reading the CAN data via the adapter.

CAN-BUS shield

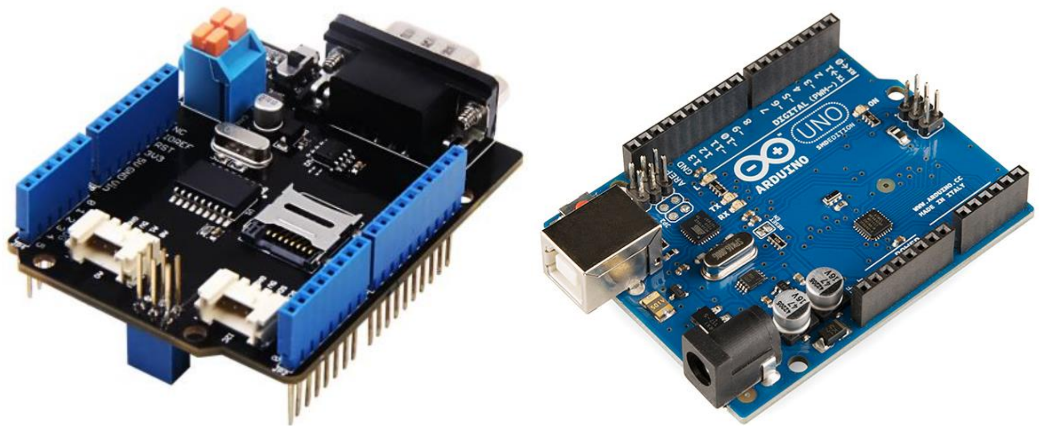

Figure 6: CANBUS shield 2.0 and Arduino uno

CAN-BUS shield has an DB9 interface where you can use a DBG-OBD cable to connect to the OBDII interface of the vehicle it also has a terminal for CAN high and Low. The CAN-BUS shield is compatible with windows and Linux and Arduino.

The adapter that I have chosen is the PEAKCAN adapter as it has a lot more support with different coding languages which in turn will help with when I am going to develop the application.

Software

Java

Java was released in 1995 by James Gosling and his team from sun microsystems. Java's greatest feature was that it was platform-independent which brought along a slogan WORA (write once, run anywhere). Java was first introduced as a programming language that would connect the likes of VCR, TV, etc. Oracle Corporation bought sun Microsystems in 2009 and became the owner of java. (Khoirom, S., Sonia, M., Laikhuram, B., Laishram, J. and Singh, T.D., 2020)

features of java

1. Java is platform independent as when compiling its complied into byte code which is then executed using a Java virtual machine (JVM)
2. Java is Robust as it has strong memory management and neglects objects that are not being used also has exception handling.
3. Java is Object-oriented this helps when you are looking to develop application as you have polymorphism, abstraction, and encapsulation.

Applications of java

1. Android applications.
2. Desktop GUI applications.
3. Cloud-based applications.
4. Web-based applications.

Advantages of java

1. As java is platform-independent it supports multi-threading which allows for garbage collection.
2. Java has APIs prebuilt in to help with connecting to databases, networking, input/output, etc.
3. Java has lots of well-established libraries and frameworks.

Disadvantages of java

1. Java does not have a lot of implementation techniques when it comes to desktop GUI development.
2. Java requires your system to have a lot of memory.
3. Java does not have the feature of multiple-inheritance and pointers.

Python

Python was released in 1989 by Guido van Rossum its first purpose was to be the successor of the ABC language as it adept at handling exceptions and interacting with the operating system Amoeba. Python became popular due to programmers not needing to define datatypes of variables and python had an interactive command-line.(Beazley, D.M. ,2018 | Khoirom, S., Sonia, M., Laikhuram, B., Laishram, J. and Singh, T.D., 2020)

Features of python

1. Python is expressive as it can execute a complicated function in just a few lines of code.
2. Python is a high-level language which means there is no need for memory management and remembering the architecture.
3. Python is also extensible as you can write code in C or C++ and compile it. Python is also embeddable as you can use python with other coding languages.

Applications of python

1. Game development.
2. Artificial intelligence and machine learning.
3. Desktop GUI Application.
4. Web development.

Advantages of python

1. Python needs less code compared to other languages.
2. Python has a large library to help you when executing complex functions.
3. Python is free and open source.
4. Python is also a very flexible programming language.
5. Python also has multiple inheritances.

Disadvantages of python

1. Python is not used as much in larger businesses due to the limitation with database connections.
2. Python is not as fast to execute compared to other languages.
3. Python is not really used in mobile development.

Reverse engineering of CAN bus

Reverse engineering the traffic that flows on the CAN bus is a very time-consuming process and there is not that many techniques out. The technique I will be using for my project will be a manual analysis with fuzzing as I will be reading the messages that are being sent across the CAN bus from the sensor as the sensors read the signals and generates a message which in turn is then sent over the CAN bus to the actuators where they read the message and perform their action. Just before the message reaches the actuators I will be stepping into the frame and changing the data randomly then sending it to see what happens and record if something occurred.

Example of Message

Take the accelerator of the car and it has a CAN ID of 100 and after the id we have eight bytes of data. These eight bytes of data will fluctuate all the time because the speed of the car is changing all the time. So, we first start off with byte zero and enter random hex data in it like 20 we then send the message and see if anything happens and if so, record it. We do this process for each byte until we discover what each byte does for example byte tree and four could be for the RPM of the car and Bytes five and six could be the speed of the car.

What could go wrong

1. The car might not turn back on this could be due to the battery being drain as if the car is not fully running the adapters take power from the battery.
2. The car might not turn off as you might have flooded the CAN bus when sending messages.
3. The car might be recklessly driving this could be that you are trying to inject messages while the car is driving.

References

Comparison of CAN bus and ethernet features for Automotive Networking Applications (no date) Copperhill. Available at: https://copperhilltech.com/blog/comparison-of-can-bus-and-ethernet-features-for-automotive-networking-applications/ (Accessed: November 15, 2022).

Figure 1: CSS Electronics (2021) *Can bus explained - a simple intro [2022], CSS Electronics*. Available at: https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial (Accessed: November 15, 2022).

CSS Electronics (2021) *Can bus explained - a simple intro [2022]*, *CSS Electronics*. Available at: https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial (Accessed: November 15, 2022).

Figure 2: CSS Electronics (2021) *Can bus explained - a simple intro [2022], CSS Electronics*. Available at: https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial (Accessed: November 15, 2022).

Figure 3: CSS Electronics (2021) *Can bus explained - a simple intro [2022], CSS Electronics*. Available at: https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial (Accessed: November 15, 2022).

Figure 4: An open-source USB to can adapter (no date) CANable. Available at: https://canable.io/ (Accessed: November 24, 2022).

An open-source USB to can adapter (no date) CANable. Available at: https://canable.io/ (Accessed: November 24, 2022).

Figure 5: GmbH, P.E.A.K.-S.T. (no date) PCAN-USB, PEAK. Available at: https://www.peak-system.com/PCAN-USB.199.0.html?&amp;L=1 (Accessed: November 24, 2022).

GmbH, P.E.A.K.-S.T. (no date) PCAN-USB, PEAK. Available at: https://www.peak-system.com/PCAN-USB.199.0.html?&amp;L=1 (Accessed: November 24, 2022).

Figure 6: Zuo, B. (no date) Can-bus shield v2.0, seeedstudio. Available at: https://wiki.seeedstudio.com/CAN-BUS_Shield_V2.0/#hardware-overview (Accessed: November 24, 2022).

Zuo, B. (no date) Can-bus shield v2.0, seeedstudio. Available at: https://wiki.seeedstudio.com/CAN-BUS_Shield_V2.0/#hardware-overview (Accessed: November 24, 2022).

AutoPi.io (no date) Can bus protocol: The ultimate guide (2022), AutoPi.io. Available at: https://www.autopi.io/blog/can-bus-explained/ (Accessed: November 16, 2022).

AutoPi.io (no date) What is OBD2? the easiest guide (2022), AutoPi.io. Available at: https://www.autopi.io/blog/what-is-obd-2/#8 (Accessed: November 16, 2022).

CSS Electronics (2021) OBD2 explained - a simple intro [2022], CSS Electronics. Available at: https://www.csselectronics.com/pages/obd2-explained-simple-intro (Accessed: November 16, 2022).

Smith, C., 2020. The car hacker's handbook. 2nd ed. chapter 5. Available at: https://docs.alexomar.com/biblioteca/thecarhackershandbook.pdf (Accessed: November 21 2022).

History of reverse engineering (no date) History of Reverse Engineering | Dan's Docs. Available at: https://danrlyon.github.io/history_of_reveng.html (Accessed: November 21, 2022).

Hemant and Madhu (2020) Can communication protocol, working, types and applications, Microcontrollers Lab. Available at: https://microcontrollerslab.com/can-communication-protocol/#:~:text=Types%20of%20CAN%20Communication%20Protocol%201%20High%20Speed,3%20Single%20Wire%20CAN%204%20Software%20Selectable%20CAN (Accessed: November 21, 2022).

Khoirom, S., Sonia, M., Laikhuram, B., Laishram, J. and Singh, T.D., 2020. Comparative analysis of Python and Java for beginners. *Int. Res. J. Eng. Technol*, *7*(8), pp.4384-4407. Available at: IRJET_V7I8755-with-cover-page-v2.pdf (d1wqtxts1xzle7.cloudfront.net) (Accessed: November 21, 2022)

Huybrechts, T. (2018) Automatic reverse engineering of CAN bus data using machine learning ..., Automatic Reverse Engineering of CAN Bus Data Using Machine Learning Techniques. Available at: https://www.researchgate.net/profile/Gregory-Van-Barel/publication/320837400_Automatic_Reverse_Engineering_of_CAN_Bus_Data_Using_Machine_Learning_Techniques/links/5bea2b9c299bf1124fce25ad/Automatic-Reverse-Engineering-of-CAN-Bus-Data-Using-Machine-Learning-Techniques.pdf?origin=publication_detail (Accessed: November 22, 2022).

Lutkevich, B. (2021) What is reverse-engineering? how does it work?, SearchSoftwareQuality. TechTarget. Available at: https://www.techtarget.com/searchsoftwarequality/definition/reverse-engineering (Accessed: November 22, 2022).

Beazley, D.M. (2018) Python essential reference. Lieu de publication non identifié: Pearson Education.