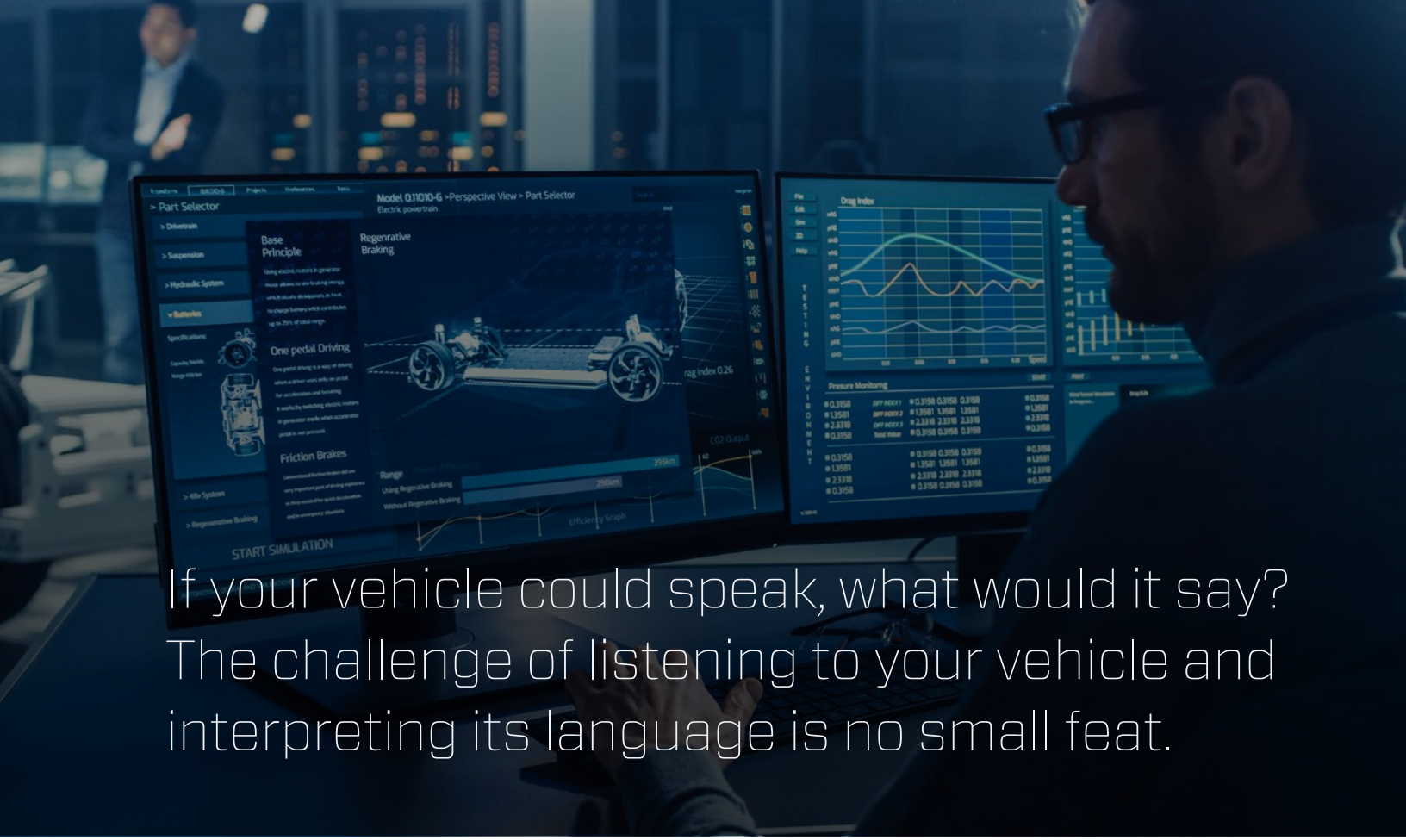
BlackBerry | CYLANCE

WHITE PAPER

We CAN Log To Big Data Analytics Platforms

Using COTS Products To Visualize Vehicle Data and More



If your vehicle could speak, what would it say? The challenge of listening to your vehicle and interpreting its language is no small feat.

Introduction

Have you ever wanted to visualize the data from your vehicle, whether a car, bus, boat, tractor, drone, submarine, or tank? We did too! To put it another way, if your vehicle could speak, what would it say? The challenge of listening to your vehicle and interpreting its language is no small feat. To get to this point, there were plenty of questions that needed answering:

- What are the hardware options available?
- Could this be accomplished using off-the-shelf hardware?
- Could this be accomplished with minimal software?
- What protocols are available to send the data?
- Can we interpret and visualize our performance metrics in real time?

In this paper, we will cover some background information on vehicle data and how to interface with the vehicle to obtain data. We describe the various methods used to send data from the vehicle to the analytics platform and how we processed the received data.

Why It Matters

Controller Area Network (CAN) bus data is at the heart of many industries such as:

- Passenger vehicles, trucks, buses (gasoline vehicles and electric vehicles), tanks, and ships
- Heavy machinery such as bulldozers, excavators, and other industrial vehicles
- Industrial automation and mechanical control

- Building automation, elevators, and escalators
- Medical instruments and equipment such as prosthetics

By developing a flexible solution for logging and interpreting CAN data, we accomplish the following:

- Performance monitoring and tuning, making the best possible products
- Predictive failure analysis, improving overall consumer safety and satisfaction
- Security alerting and prevention, thwarting sophisticated attacks
- Fleet management services (FMS), providing logistics insights and efficiencies

There are endless opportunities when creating a scalable process that could be applied to solve problems in such a vast amount of industries.

Key Component Summary

After a significant search effort, we narrowed our key components down to the CANLogger CL3000 from CSS Electronics (www.csselectronics.com) for interfacing with the car and Splunk Enterprise for receiving the data. We ended up needing the following hardware and software components:

Hardware

- CAN Logger CL3000
- Microsoft® Surface Go
- Mobile MiFi
- Cables
  - USB-C to micro
  - DB9 to OBD2
  - DB9 to J1939

Software

- Splunk Enterprise
- Splunk Universal Forwarder (for now...)
- CSS Electronics CANvas Software (used at the start, but then bypassed)
- Conversion/decoding databases

Data Flow Scenarios

We will cover three data flow scenarios. The first two were accomplished with the following hardware and steps:

- CL3000 logger + OBD2 to DB9 cable
- MiFi mobile hotspot
- Tablet/laptop

The third scenario of sending directly from the logger to the analytics platform is in development.

Scenario #1:

Local Data Processing in the Vehicle

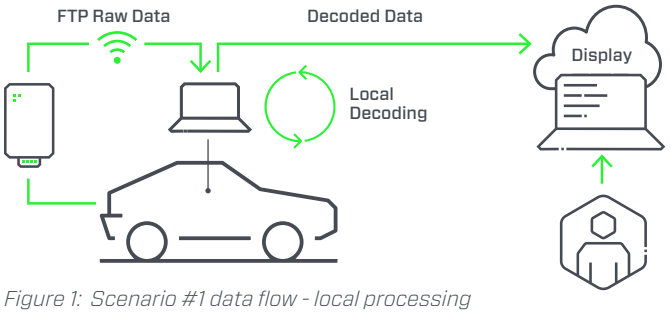


Figure 1: Scenario #1 data flow - local processing

The diagram above performs some local data processing of the raw data. In this case, the in-car laptop needs enough resources to perform the local processing. The step by step data flow is outlined below:

1. The in-car hotspot provides the bridge between the laptop and the CL3000
2. The CL3000 queries the car's ECUs and sends the data to the laptop via FTP
3. FTP software on the laptop receives the data from the CL3000 and drops it into an input folder
4. CANvas monitors the input folder and decodes the files as they arrive
5. CANvas writes the decoded results to the output folder
6. A Splunk Universal Forwarder monitors the output folder and sends the data to a Splunk indexer
7. The indexer parses the data to be displayed on the Splunk search head

Scenario #2:

No Data Processing in the Vehicle

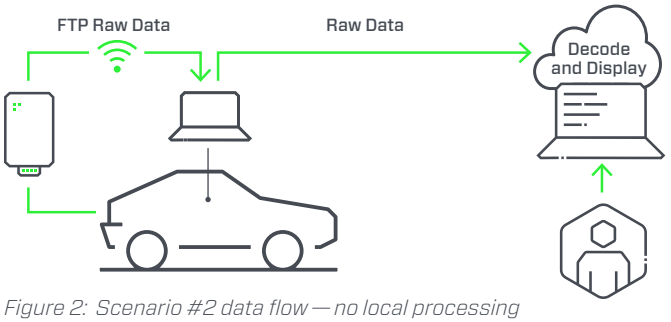


Figure 2: Scenario #2 data flow — no local processing

The diagram above performs no local data processing of the raw data. Thus, in this scenario, the in-car laptop only receives and sends the data to the analytics platform. The step by step data flow is outlined below:

1. The in-car hotspot provides the bridge between the laptop and the CL3000

2. The CL3000 queries the car's ECUs and sends the data to the laptop via FTP
3. FTP software on the laptop receives the data from the CL3000 and drops it into an input folder
4. A Splunk Universal Forwarder monitors the input folder and sends the data to a Splunk indexer
5. The indexer parses and decodes the data to be displayed on the Splunk search head

Scenario #3:

No in-car laptop — Data sent directly to analytics platform

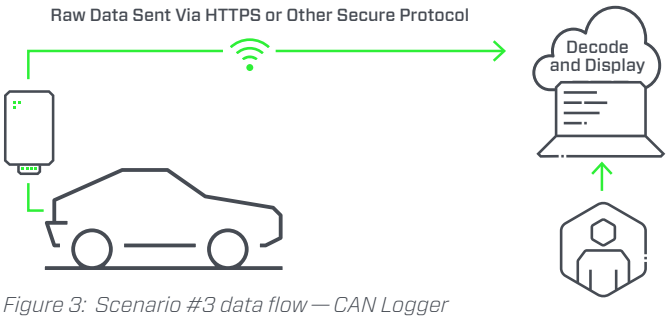


Figure 3: Scenario #3 data flow — CAN Logger sends data directly to the analytics platform

Unfortunately, for scenario #3, the existing hardware did not support directly sending the data securely to a cloud-based analytics platform. However, if it were possible, the data flow would follow the outline below:

1. The in-car hotspot provides the bridge between the CAN Logger and the data analytics platform
2. The CAN Logger sends the data directly to the Splunk indexer
3. The indexer parses and decodes the data to be displayed on the Splunk search head

This possibility is currently being explored using CSS Electronics' CANedge2 and CANcloud platform to reduce the complexity and components involved in such a project.

High-level Process and End Results

After selecting the hardware used to interface with the car, the data is sent, parsed, normalized, enriched, and displayed in Splunk to create real-time visualizations. Researchers created a custom Splunk application called CANLogger to organize various lookup tables applied to the CAN data. Once the lookup tables are applied to the raw data, the app provides panels and dashboards to graphically visualize vehicle performance. The sections below step through the analytics process of achieving the end-user experience.

Field Extraction

When information enters an analytics platform such as Splunk, it is raw data. Fields need to be parsed out of the data using structured format or regex, as shown in Figure 4, using OBD2 data.

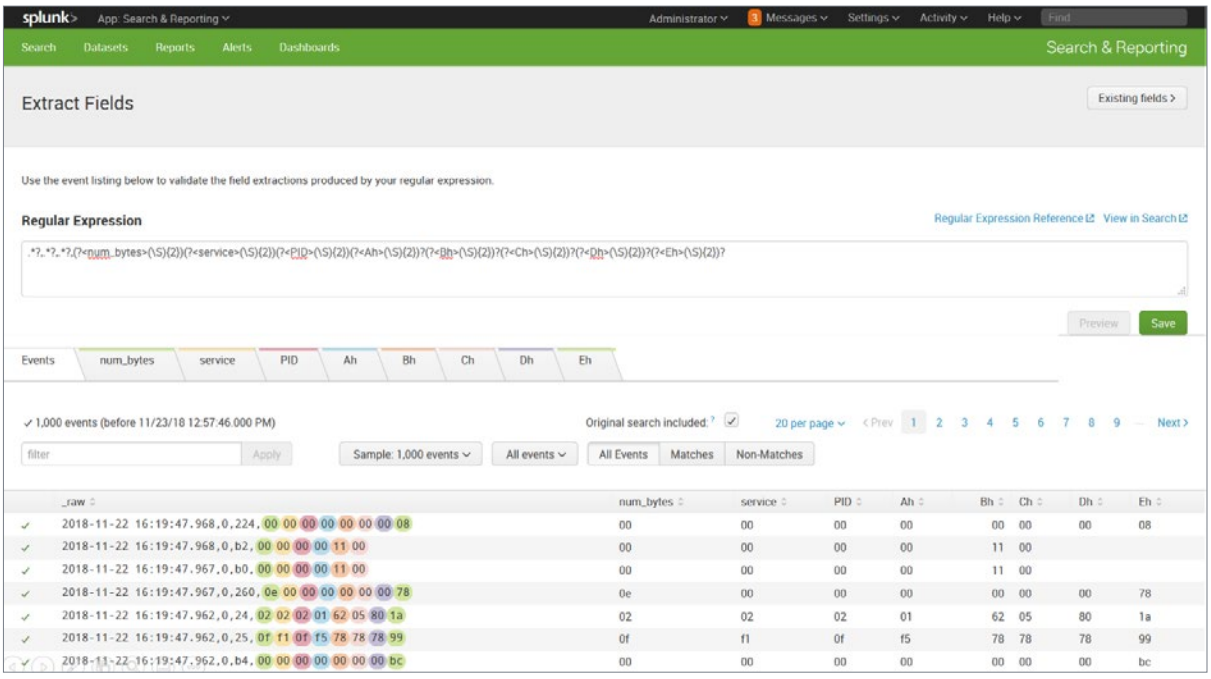


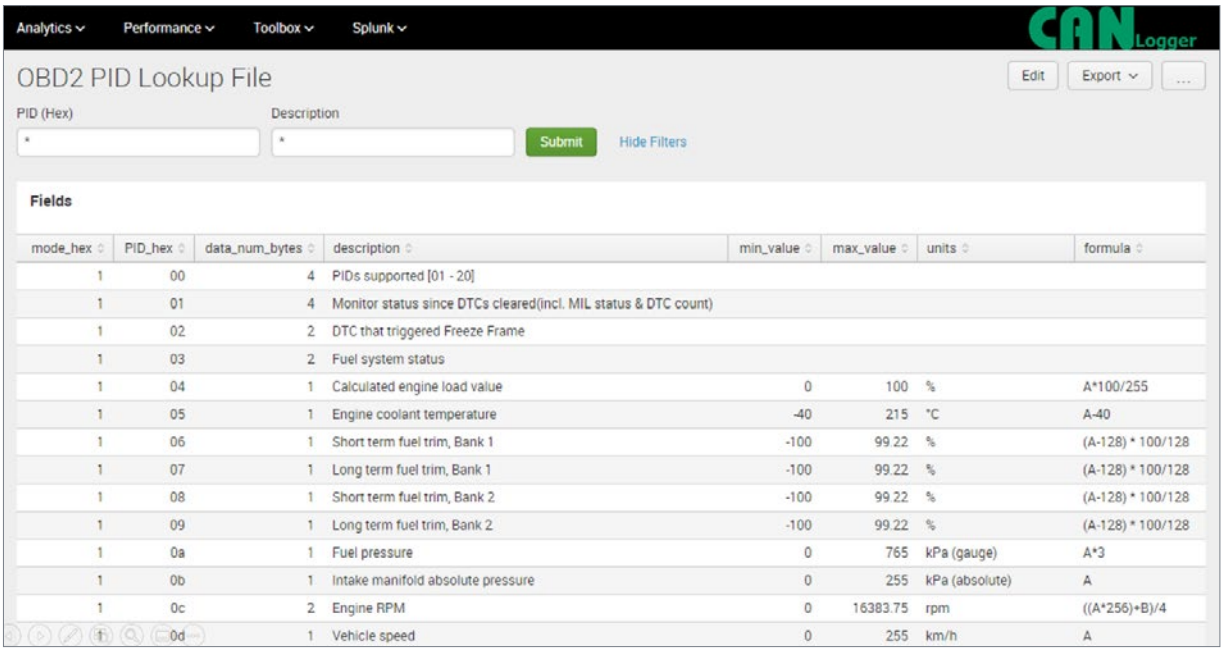
Figure 4: Data extraction within Splunk

Creating Lookup Tables

After parsing the data into fields or registers, the information needs to be interpreted. Unfortunately, the interpretation of raw CAN bus data is proprietary to the manufacturer and can even differ between makes and models. While this project can be used to reverse engineer the raw CAN bus data, it is beyond the scope of this paper. Fortunately, some formats, such as OBD2 and J1939, are relatively well-documented and somewhat standardized, which allows us to create lookup tables using knowledge found in the public domain. The lookup table, in Figure 5 below, is the result of taking that public knowledge and importing it into Splunk to then be applied to the parsed OBD2 data.

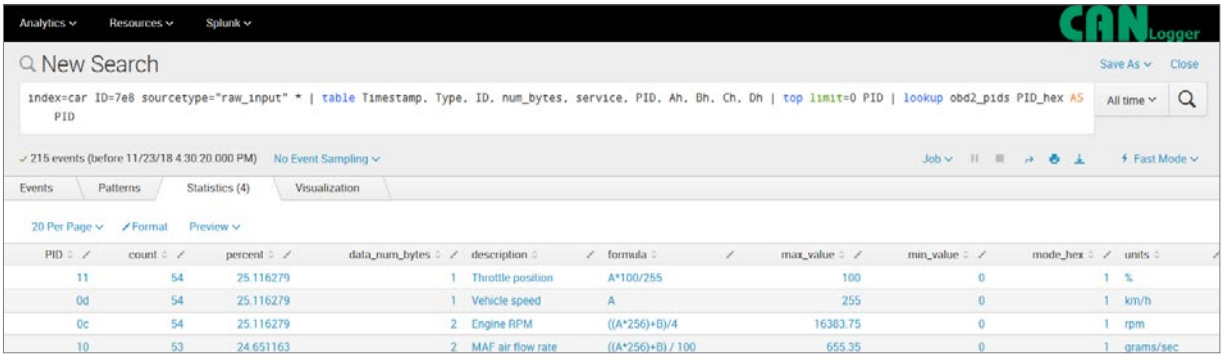
Enriching Data

Now that we have the fields parsed and the lookup tables created, we can enrich the data from the car by applying the lookup table information to the data from the car. The example, shown in Figure 6 below, overlays OBD2 PIDs to then retrieve other information, such as number of data bytes, human-readable description, formula to calculate the data, maximum and minimum value expected, and the units for the data.



mode_hex	PID_hex	data_num_bytes	description	min_value	max_value	units	formula
1	00	4	PIDs supported [01 - 20]				
1	01	4	Monitor status since DTCs cleared(incl. MIL status & DTC count)				
1	02	2	DTC that triggered Freeze Frame				
1	03	2	Fuel system status				
1	04	1	Calculated engine load value	0	100	%	A*100/255
1	05	1	Engine coolant temperature	-40	215	°C	A-40
1	06	1	Short term fuel trim, Bank 1	-100	99.22	%	(A-128) * 100/128
1	07	1	Long term fuel trim, Bank 1	-100	99.22	%	(A-128) * 100/128
1	08	1	Short term fuel trim, Bank 2	-100	99.22	%	(A-128) * 100/128
1	09	1	Long term fuel trim, Bank 2	-100	99.22	%	(A-128) * 100/128
1	0a	1	Fuel pressure	0	765	kPa (gauge)	A*3
1	0b	1	Intake manifold absolute pressure	0	255	kPa (absolute)	A
1	0c	2	Engine RPM	0	16383.75	rpm	((A*256)+B)/4
1	0d	1	Vehicle speed	0	255	km/h	A

Figure 5: Creating a Lookup File in the CANLogger Splunk app



PID	count	percent	data_num_bytes	description	formula	max_value	min_value	mode_hex	units
11	54	25.116279	1	Throttle position	A*100/255	100	0	1	%
0d	54	25.116279	1	Vehicle speed	A	255	0	1	km/h
0c	54	25.116279	2	Engine RPM	((A*256)+B)/4	16383.75	0	1	rpm
10	53	24.651163	2	MAF air flow rate	((A*256)+B) / 100	655.35	0	1	grams/sec

Figure 6: Enriching the data in the CANLogger Splunk app

Visualize Data Via Charts

Now that we understand how to interpret the data via the formulas, we can apply that to the raw fields. When performing that calculation over time, it provides a visual representation of what the vehicle is doing at various times. The example, in Figure 7 below, shows the vehicle speed over time.

Creating Heads Up Display

It is advantageous to be able to monitor multiple charts in a single heads up display, especially after creating individual charts for various vehicle metrics. For this, we create a dashboard, such as the one shown in Figure 8 below. This example dashboard shows speed, RPMs, throttle position, and mass air flow rate plotted over the same time range.

Future Roadmap

A project like this has unlimited potential in the areas of security, optimization, fleet management, and predictive failure analysis. There are factors that will help enable future success and applicability of this work, such as:

- Increased flexibility of data transmission
- Increased frequency and rate of data transmission
- Added features to the hardware such as native GPS for geographical plotting
- Embedded cellular modems for constant uplink to a data analytics platform
- Additional lookup tables for other public standards
- Addition of proprietary CAN bus lookup tables for interpreting specific make/model data
- Eventual utilization of OBD3 or CAN FD when available

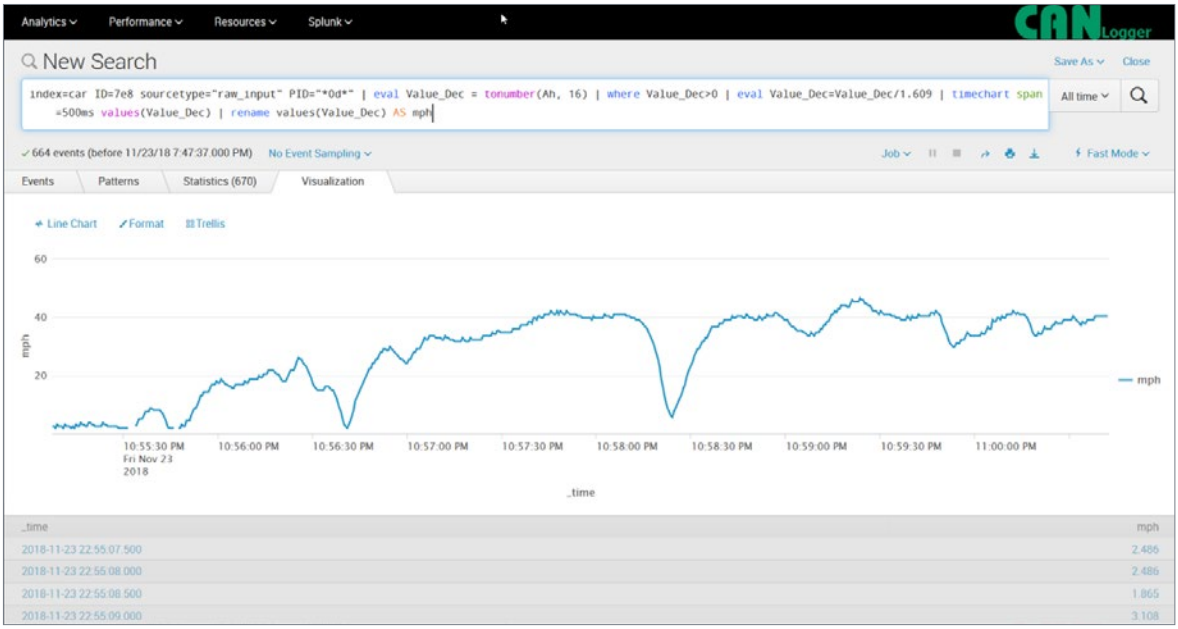


Figure 7: Plotting the data in the CANLogger Splunk app

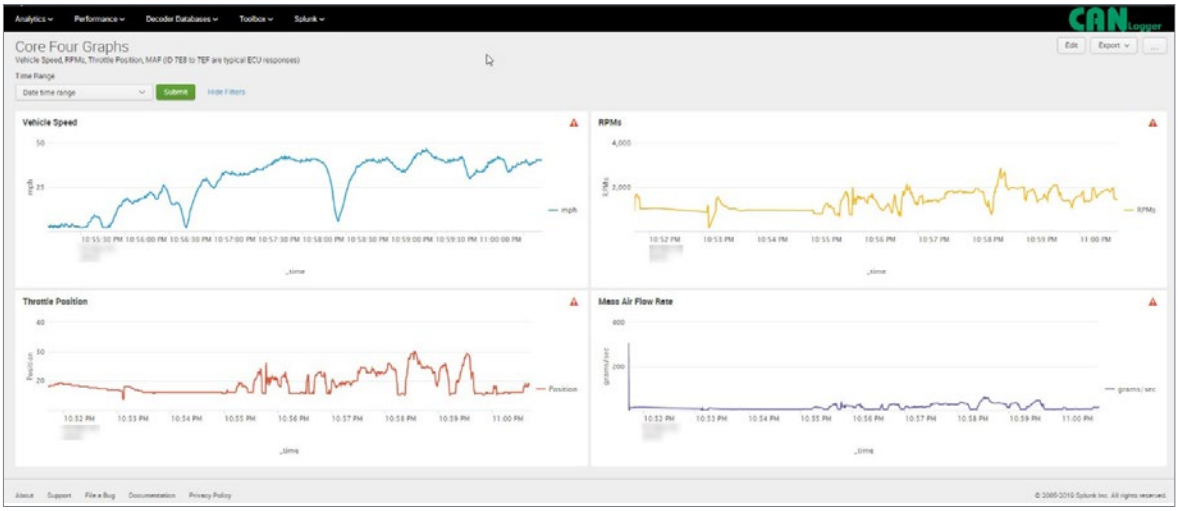


Figure 8: CANLogger app showing the Core Four Graphs

Technology Primer

Since everyone may not have a background in the various protocols and concepts used in this project, we will cover the major topics below at a high level. For more details, please visit the source links or references in the appendix.

Key Concepts

Electronic/Engine Control Unit (ECU)

Low-cost components within vehicles controlling anything from air bags to the radio. ECUs communicate with one another using the CAN bus. Modern vehicles can have 70 or more ECUs.

Logger vs. Interface

Logger is useful for collecting data over weeks or months of operation (depending on the size of the SD card).Interface allows real-time event-based analysis or diagnostics. The CL3000 used in this project is a hybrid and can act as both a logger and interface.

Protocols

Controller Area Network (CAN) Bus

Originally developed by Bosch in 1986 to replace expensive and complex point to point wiring as the common bus of communication between ECUs. Think of CAN bus as the infrastructure to pass data, but not necessarily defining the rules to interpret it. In this way, it can be analogous to the physical and data link layer of the OSI model — similarly often carrying higher-layer protocols (HLP) such as the ones we will cover below. Most of the data is broadcast and only requires a CAN logger to receive the data.

CAN bus is used in automotive, heavy machinery, shipping, and even prosthetic limbs. It is popular because it is low-cost, centralized, robust, efficient, and flexible. The most important fields, as shown in Figure 9, in CAN bus are the CAN ID, Control, and Data fields. Raw CAN bus data often requires access to proprietary conversion rules or car hacking/reverse engineering to make the machine data human readable.

Note:CAN FD is beyond the scope of this paper, but this newer protocol should add increased logging functionality and improve the world of CAN logging and telematics.

CANOpen

Higher-layer protocol (HLP) based on CAN bus typically used for non-automotive applications such as industrial robotics and machinery — however, it can extend into rail, construction, and marine equipment. It naturally lends itself to motor control applications communicating data such as command and speed. CANOpen could be logged using the same equipment we used for this project, however, since it was not logged, we will not examine this further.

SAE J1939

Higher-layer protocol (HLP) for commercial vehicles established by the Society for Automotive Engineers (SAE) leverages the CAN bus protocol for physical and data link transport:

- Standardized languages across different manufacturers of heavy vehicles such as buses, trucks, military (tanks), shipping, and construction equipment (cranes, bulldozers, excavators, etc.)

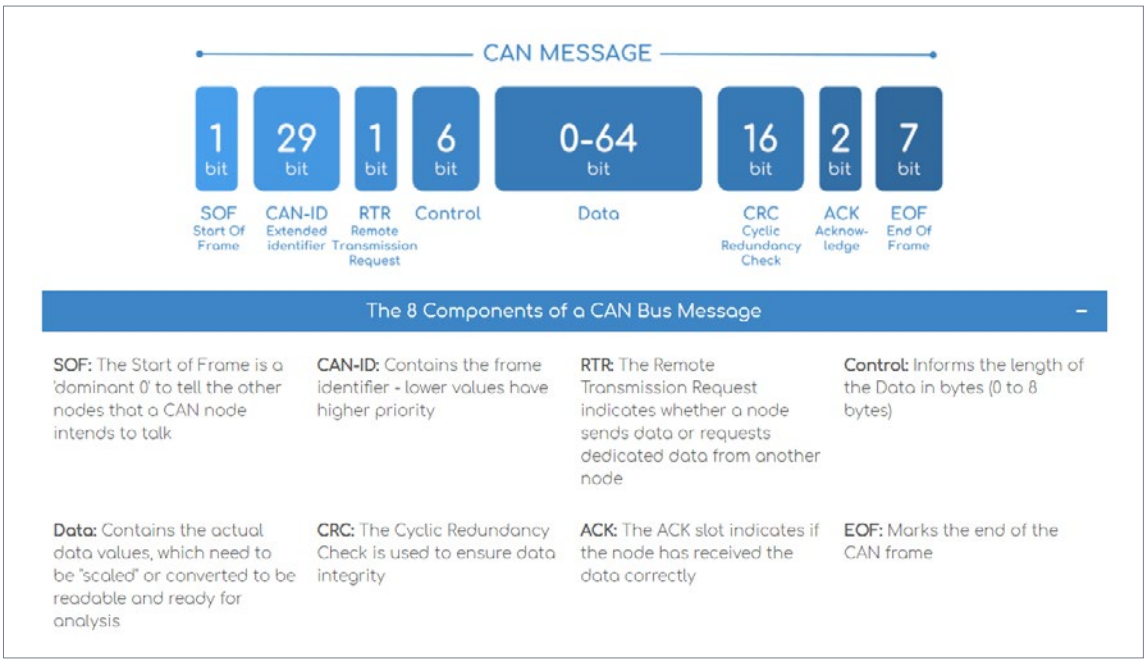


Figure 9: CAN bus message components — source: https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en

- Specifies how to convert machine data to human-readable data
- Allows for greater than 8-byte limit of CAN protocol by using multi-packet messages

Most messages are broadcast to the CAN bus, however some need to be requested from a device such as a CAN logger. This format can get very complicated when introducing multi-packet messages, however most messages will be constructed as follows:

- **Identifier —** First 29 bits in a J1939 message.
- **Parameter Group Numbers (PGN) —** Starting at bit 9, consists of 18 of the 29 bits in a CAN 2.0B message. Identifies messages and contains a number of Suspect Parameter Numbers (SPNs).
- **Suspect Parameter Numbers (SPN) —** 8 bytes reflecting parameters within a PGN — such as Engine Torque Mode, Engine Speed, Engine Demand, and more.

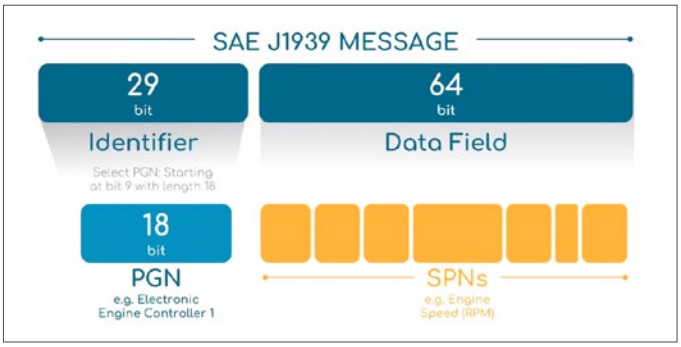


Figure 10: SAE J1939 message components — source:https://www.csselectronics.com/screen/page/simple-intro-j1939-explained/language/en

Even though the SAE J1939 standard is well-documented, it can be cumbersome to translate raw messages containing PGNs and SPNs into a human-readable message. For this purpose, DBC files can be purchased to assist in this task.

On-board Diagnostics (OBD2)

On-board Diagnostics (OBD2) originated as part of California emission control requirements in 1991. The OBD2 port can be found on all U.S. vehicles from 1996 to present day and E.U. gas cars and light trucks from 2001 to present day. Accessed through a 16-pin port, OBD2 is a higher-level protocol that uses CAN bus to provide self-diagnostic and reporting capabilities commonly used to identify what is wrong with a car. The driver is alerted to an issue via a malfunction indicator light (MIL), also known as the check-engine light. Mechanics then use a diagnostic tool to connect to the OBD2 port to pull diagnostic trouble codes (DTCs).

The OBD2 port can be active while the vehicle is at rest or being operated. This makes it convenient to pull telematics data from the vehicle and enables use cases such as fleet management, predictive vehicle failure analysis, and more. During operation, the CAN bus is actively transmitting data, but remember, this data is proprietary and needs to be interpreted or reverse engineered. Fortunately, OBD2 simultaneously supports real-time data query and response of metrics such as speed, RPMs, fuel consumption, and more. However, this does require more than a passive logger and that is one of the reasons we chose the CL3000 in our selection process below.

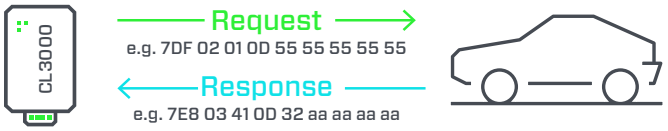


Figure 11:CAN Logger can query and receive OBD2 data

OBD2 contains a wide range of parameter IDs (PIDs) that can be extracted and converted to human-readable data. Much of the data is documented in various resources such as Wikipedia, however some can still be proprietary and needs to be decoded via a database of parameters requiring reverse engineering. Documentation and a converter can be found below:

- https://en.wikipedia.org/wiki/OBD-II_PIDs
- https://www.csselectronics.com/screen/page/obd-ii-online-message-converter/language/en

OBD2 data loggers can send queries to the vehicle to obtain responses from various ECUs. The message is split into an identifier and data as shown in Figure 12:

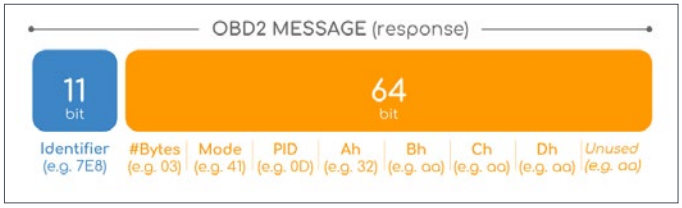


Figure 12:OBD2 message components - source:https://www.csselectronics.com/screen/page/simple-intro-obd2-explained/language/en

Message Breakdown

A quick explanation of the OBD2 message components follows:

- The start of an ODB2 message is an 11-bit identifier which contains information such as whether this was a “request message” (ID: 7DF) or a “response message” (IDs: 7E8 to 7EF).
- Next is the length of the data in bytes. This is dependent upon the PID which indicates the type of data being sent.
- There are 10 services (also known as modes). For example, service 01 shows current (real-time) data.
- Within each service, there are PIDs. For example, within service 01 vehicle speed is indicated by PID 0D and engine RPM is indicated by 0C.
- Data is shown in hex using registers: Ah Bh Ch and Dh. These values are converted to decimal before calculations are performed according to OBD2 conversion formulas.

The table below, shown in Figure 13, is a snippet from Wikipedia that shows PIDs contained within the real-time service (01). This represents a human-readable version of the OBD2 message which can be imported into a big data analytics platform and then applied to the messages for decoding.

Note: This process was used in the OBD2 real-time logging within this project since the OBD2 port is alive and communicates (query/response) while driving. ODB3 is beyond the scope of this paper, however, it should yield interesting new use cases for the field of telematics.

Selection Process

Now that we covered the goals of this project, the end results, and background on the protocols and key concepts, we will review the hardware and software selection process — including the key features for success.

Hardware Details

CAN Logger 3000 (CL3000)

The professional series CL3000 is manufactured by CSS Electronics, which is headquartered in Denmark. Their focus is on data acquisition from vehicles such as cars, trucks, and heavy machinery. With 16GB onboard memory, this device cyclically logs to a non-removable SD card. However, what makes this device special is on-board Wi-Fi and multiple transport modes useful for remote log collection.

Figure 14:CL3000 CAN Logger — Source:https://www.csselectronics.com/screen/product/can-bus-logger-canlogger3000/language/en

Service 01 [edit]

PID (hex)	PID (Dec)	Data bytes returned	Description	Min value	Max value	Units	Formula[a]
00	0	4	PIDs supported [01 - 20]				Bit encoded [A7..D0] == [PID $01..PID $20] See below
01	1	4	Monitor status since DTCs cleared. (Includes malfunction indicator lamp (MIL) status and number of DTCs.)				Bit encoded. See below
02	2	2	Freeze DTC				
03	3	2	Fuel system status				Bit encoded. See below
04	4	1	Calculated engine load	0	100	%	$\frac{100}{255}A$ (or $\frac{A}{2.55}$)
05	5	1	Engine coolant temperature	-40	215	°C	$A - 40$
06	6	1	Short term fuel trim—Bank 1	-100 (Reduce Fuel: Too Rich)	99.2 (Add Fuel: Too Lean)	%	$\frac{100}{128}A - 100$ (or $\frac{A}{1.28} - 100$)
07	7	1	Long term fuel trim—Bank 1				
08	8	1	Short term fuel trim—Bank 2				
09	9	1	Long term fuel trim—Bank 2				
0A	10	1	Fuel pressure (gauge pressure)	0	765	kPa	$3A$
0B	11	1	Intake manifold absolute pressure	0	255	kPa	A
0C	12	2	Engine RPM	0	16,383.75	rpm	$\frac{256A + B}{4}$

Figure 13: Service 01 table from Wikipedia - Source: https://en.wikipedia.org/wiki/OBD-II_PIDs

Key features for success

- Flexibility in logging all CAN bus applications (vehicles, boats, manufacturing, etc.) via OBD2 and J1939 cables
- Flexibility in log output to make big data analytics ingest easier
- Ability to transmit messages to query the ECUs
- Availability of multiple transport modes
- Conversion capabilities (Well-known OBD2 and J1939 standards)

CL3000 physical Interfaces

- Micro USB used for connecting to a computer (only used for programming CL3000)
- DB9 Female for connecting to vehicles
- Note:Product does not include cables (micro USB, DB9 to OBD2, DB9 to J1939, etc.)

CL3000 transport modes

All data is logged to the on-board 16 GB of memory, however, the flexibility of getting the data off the vehicle is unique compared to other CAN logger offerings. The following transport modes are supported:

- **Access Point Mode —** Shows up as a hotspot to a PC, phone, or tablet. Log in and access the data using a pre-specified link in your browser.
  - Use case — In the field and within range of the vehicle.
- **Station Mode —** Can be configured to auto-connect to an existing Wi-Fi hotspot (such as a mobile hotspot). Possible to turn this into a simple website for easy downloading of files.
  - Use case — Remotely diagnose vehicles that are Internet-equipped or when vehicles return to garages.
- **FTP Push Mode —** Leverages either AP or Station mode mentioned above to automatically push new log files to an FTP server. Once uploaded, local copies of the files are deleted to free up space and avoid duplicate transfers. Can also schedule auto conversion using CANvas.
  - Use case — Scale to process files from larger fleet of vehicles.

For this project, we used a combination of station and push mode to send the data from the CL3000 to a Microsoft® Surface Pro running an FTP server. The FTP server wrote the files to disk which were then sent to Splunk via a Splunk Universal Forwarder. Once the information arrived at Splunk, we used the custom written CANLogger Splunk app to parse, normalize, enrich, and display the CAN data.

Excellent reference:https://www.csselectronics.com/screen/page/canvas-configure-wireless-can-logger

Required Cables

The CL3000 does not plug directly into the vehicle via a DB9 port. An additional cable is needed to interface the CL3000 with the vehicle, but the cable depends on the type of port that is available. For example, cars will require a DB9 to OBD2 cable with the proper pin out. Trucks and heavy equipment vehicles will require a DB9 to J1939 cable (or a CANCrocodile, which uses induction to record data without the need of a J1939 port or splicing CAN wiring).

CSS Electronics sells premade cables, or you can make your own since CSS Electronics also provides the pinout specifications.

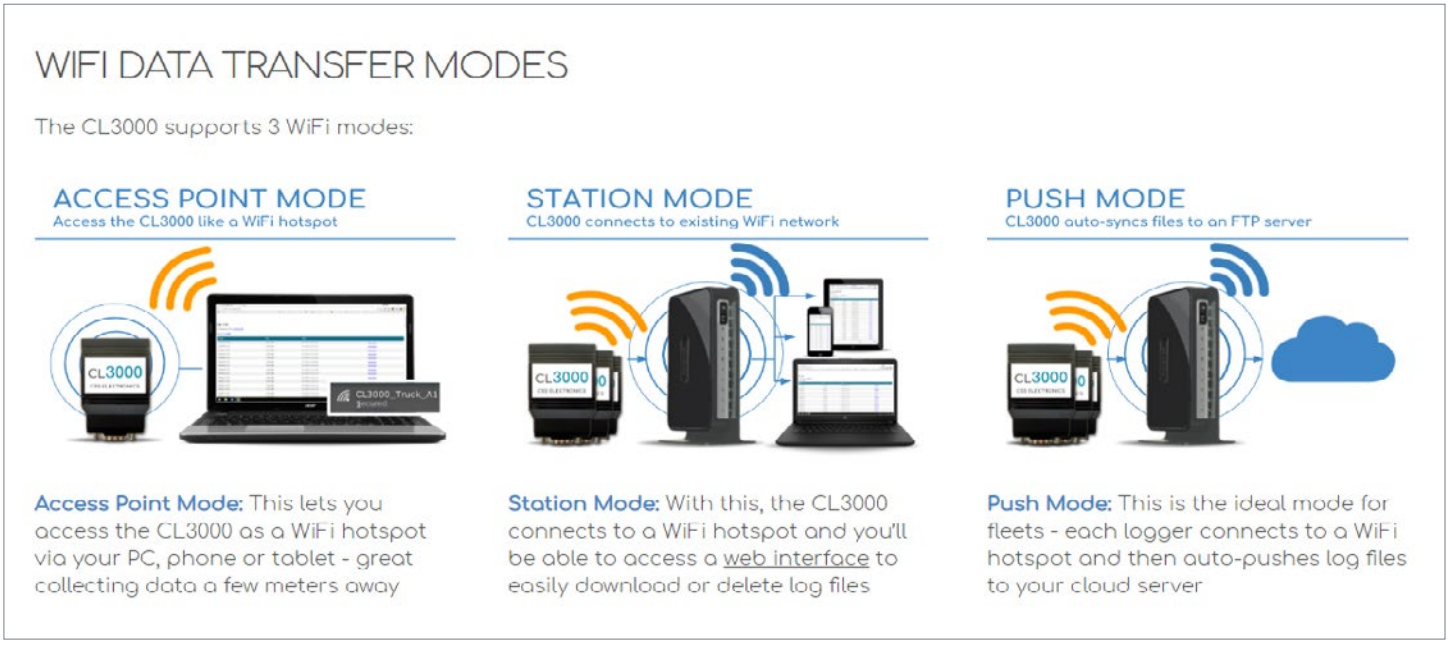


Figure 15: Data Transfer Modes of CL3000 - Source: https://www.csselectronics.com/screen/product/can-bus-logger-canlogger3000/language/en

- CSS Electronics DB9 to J1939 cable — https://www.csselectronics.com/screen/product/can-bus-logger-j1939-adaptor
- CSS Electronics DB9 to OBD2 cable — https://www.csselectronics.com/screen/product/can-bus-logger-obd-adaptor#void

Making your own cable is possible using a DB9 terminal breakout and an OBD2 open plug.

Software Details

Easily available off-the-shelf software is required for this project.Recommended software includes:

- Microsoft® Surface/Laptop Operating System — Windows® 7 or above
  - .NET 4.5.2 or higher
- CANvas software — Freely provided by CSS Electronics
- J1939 DBC (if processing and decoding J1939 data)
- Splunk Enterprise

Conclusion

As we near a future full of connected cars, it will be more important than ever that we harness the data within that ecosystem to be able to perform predictive failure analysis, security monitoring, and more. Much of the necessary data is directly accessible via the CAN bus. Furthermore, it is possible to use standard off-the-shelf hardware and software to log this CAN bus data to a big data analytics platform to gain a deeper understanding of the data. Our efforts to ingest, parse, enrich, and graph this data are easily extensible for others to log their vehicle or device of choice. Lastly, by applying proprietary decoding to the raw CAN bus data, manufacturers can glean much more insight than those without the proprietary decoders. These analytics could be used in development, test, quality assurance, and production environments to make massive improvements in the transportation industry and beyond. By sharing this data, we hope to spark curiosity and forge partnerships with others who share a similar vision. Let us know if that might be you.

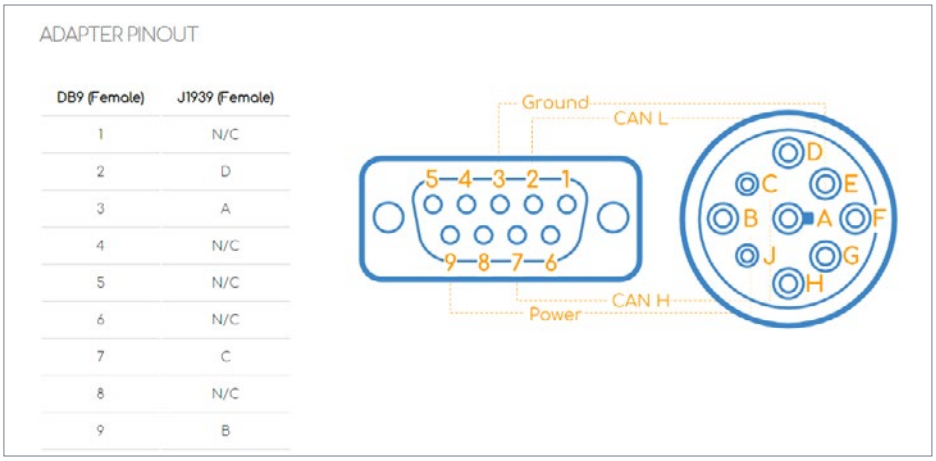


Figure 16: Pinout for DB9 to J1939 cable - https://www.csselectronics.com/screen/product/can-bus-logger-j1939-adaptor#void

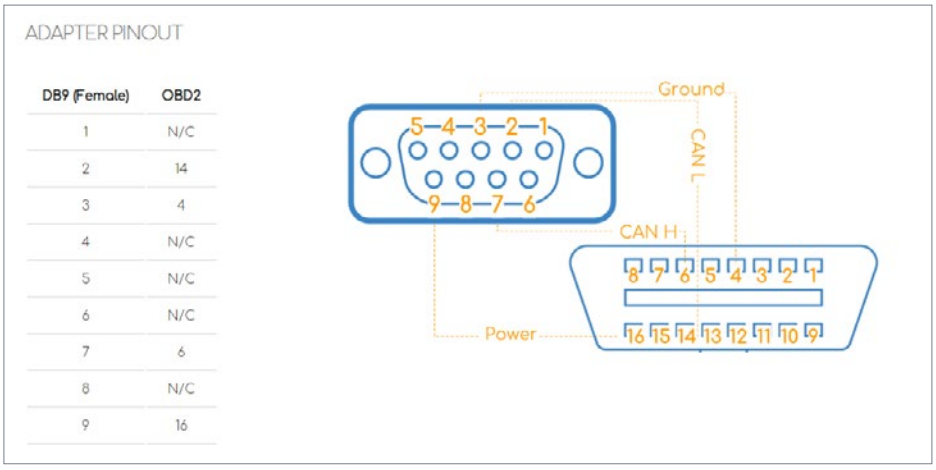


Figure 17: Pin out for DB9 to OBD2 cable - https://www.csselectronics.com/screen/product/can-bus-logger-obd-adaptor#void

About the Author

Tony Lee, Vice President, Global Services Technical Operations at BlackBerry Cylance, has more than fifteen years of professional research and consulting experience pursuing his passion in all areas of information security. As an avid educator, Tony has instructed thousands of students at many venues worldwide, including government, universities, corporations, and conferences such as Black Hat. He takes every opportunity to share knowledge as a contributing author to *Hacking Exposed 7*, a frequent blogger, researcher, and author of white papers on topics ranging from Citrix Security, the China Chopper Web shell, and Cisco's SYNFul Knock router implant. Over the years, he has enabled many incident analysts and responders by contributing to tools such as UnBup, Splunk Forensic Investigator Splunk app, and CyBot, the open-source Threat Intelligence Bot.

Discover the latest fun projects at:https://www.linkedin.com/in/tonyleevt/

Acknowledgements

- Martin Falch from CSS Electronics for the excellent technical support and willingness to implement new features
- Chris Wilson for being supportive of research projects such as this one

Appendix A:References

Below is a list of excellent references for the topics discussed in this paper.

The University of Tulsa Crash Reconstruction Research Consortium TU-CRRC: http://tucrrc.utulsa.edu/index.html

CAN bus primer: https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en

CAN bus wiki: https://en.wikipedia.org/wiki/CAN_bus

J1939 primer: https://www.csselectronics.com/screen/page/simple-intro-j1939-explained/language/en

J1939 wiki: https://en.wikipedia.org/wiki/SAE_J1939

Online J1939 converter: https://www.csselectronics.com/screen/page/j1939-pgn-conversion-tool/language/en

OBD2 primer: https://www.csselectronics.com/screen/page/simple-intro-obd2-explained/language/en

OBD2 wiki: https://en.wikipedia.org/wiki/On-board_diagnostics#OBD-II

OBD2 PID lookup table: https://en.wikipedia.org/wiki/OBD-II_PIDs

CANopen primer: https://www.csselectronics.com/screen/page/canopen-tutorial-simple-intro

CANopen wiki: https://en.wikipedia.org/wiki/CANopen

About BlackBerry Cylance

BlackBerry Cylance develops artificial intelligence to deliver prevention-first, predictive security products and smart, simple, secure solutions that change how organizations approach endpoint security. BlackBerry Cylance provides full-spectrum predictive threat prevention and visibility across the enterprise to combat the most notorious and advanced cybersecurity attacks, fortifying endpoints to promote security hygiene in the security operations center, throughout global networks, and even on employees' home networks. With AI-based malware prevention, threat hunting, automated detection and response, and expert security services, BlackBerry Cylance protects the endpoint without increasing staff workload or costs.

BlackBerry CYLANCE

+1-844-CYLANCE
sales@cylance.com
www.cylance.com

©2019 BlackBerry Limited. Trademarks, including but not limited to BLACKBERRY, EMBLEM Design and CYLANCE are trademarks or registered trademarks of BlackBerry Limited, its subsidiaries and/or affiliates, used under license, and the exclusive rights to such trademarks are expressly reserved.

MKTG 19-0781-20191212