Secure CAN Logging and Data Analysis

By Duy Van

Committee Members: Jeremy Daily, Steve Simske, Christos Papadopoulos, Stephen Hayne

Department of Systems Engineering

Fall 2020

Colorado State University

Introduction

About Me

- Bachelor of Science in Mechanical Engineering and a Minor in Cybersecurity at The University of Tulsa, 2018
- Pursuing Master of Science in Systems Engineering at Colorado State University, 2020
- Research in heavy vehicle network and cybersecurity since 2017
  - CyberTruck and CyberAuto Challenge participant
  - Chip level forensics research
  - Secure CAN logging project

Duy Van with the Super Truck at 2017 NMFTA Conference in North Carolina

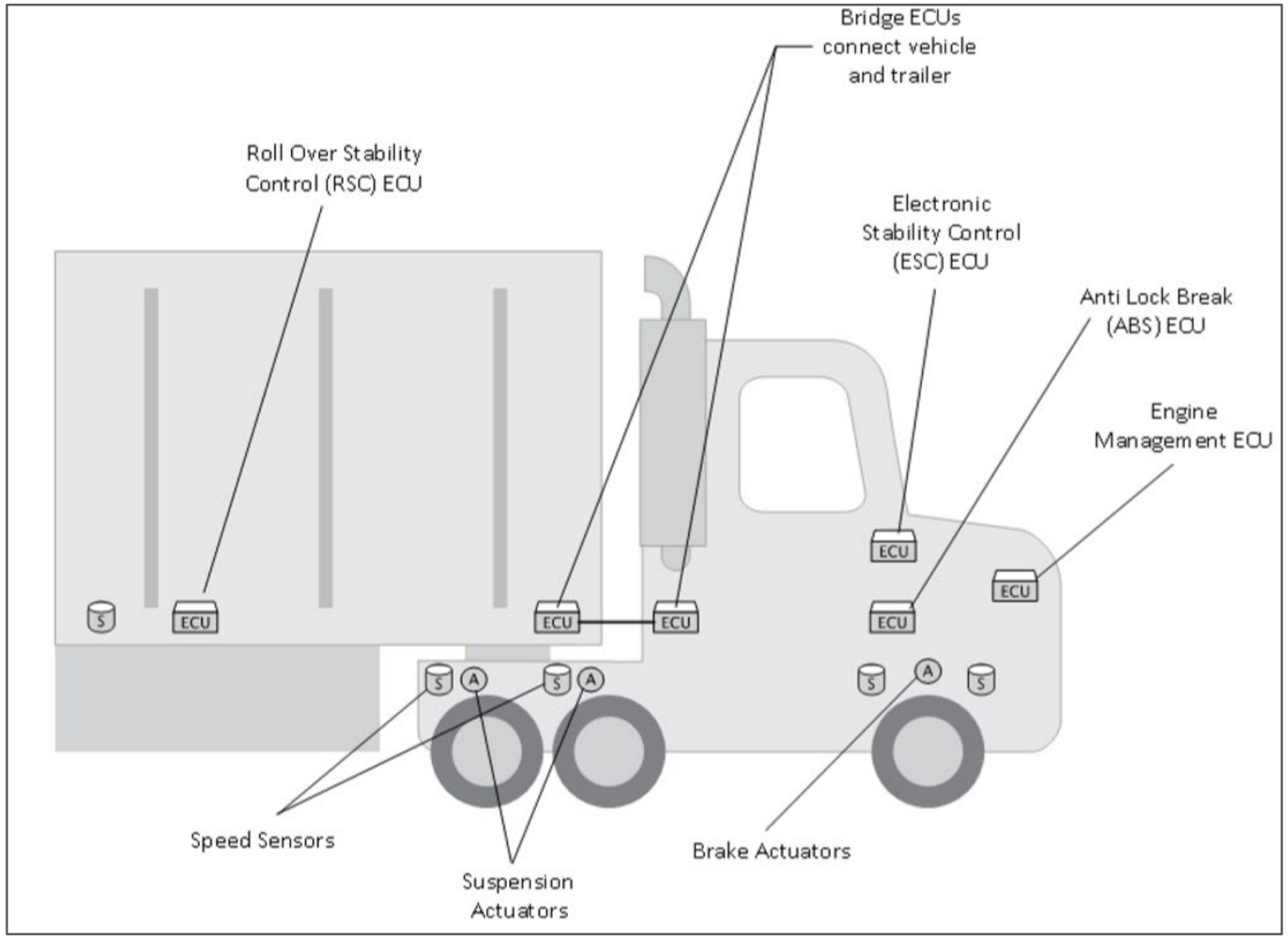
Background

Why heavy truck?

- Approximately 12.2 million registered heavy vehicles in the U.S. alone in 2017
- Approximately 730,000 new trucks on the road each year
- Often carry high-risk or high-value cargo
- Play an important role in the national as well as global economy
- A large-scale cyber-attack could:
  - Economic recession
  - Shortage of supplies and lack of essential services
  - Public Endangerment

Background

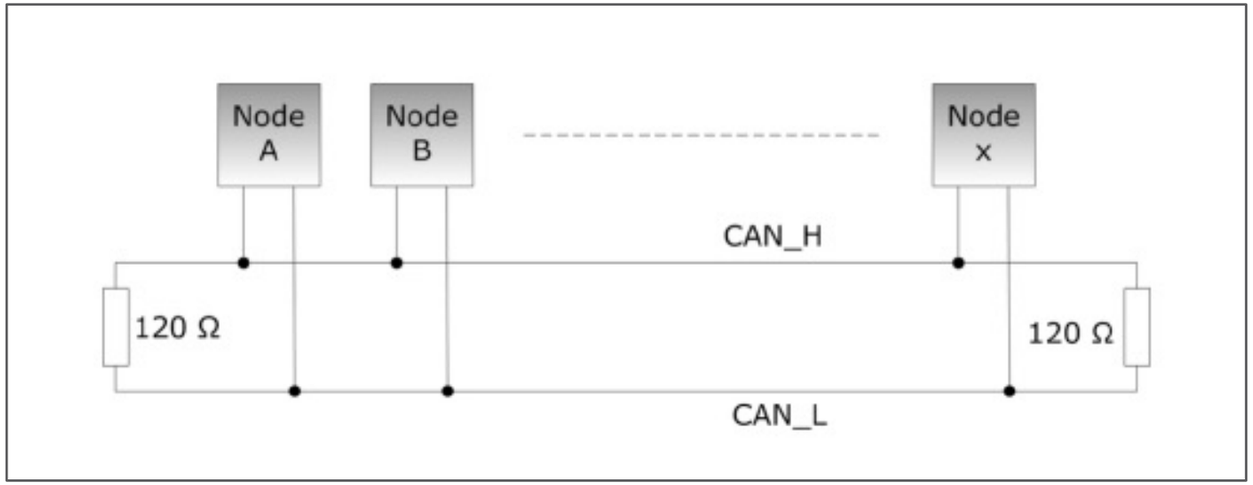
- Consisted of many Electronic Control Units (ECU) that help improve:
  - Efficiency
  - Safety
  - Durability
- Horizontal integration build
  - Customize different components from different brands



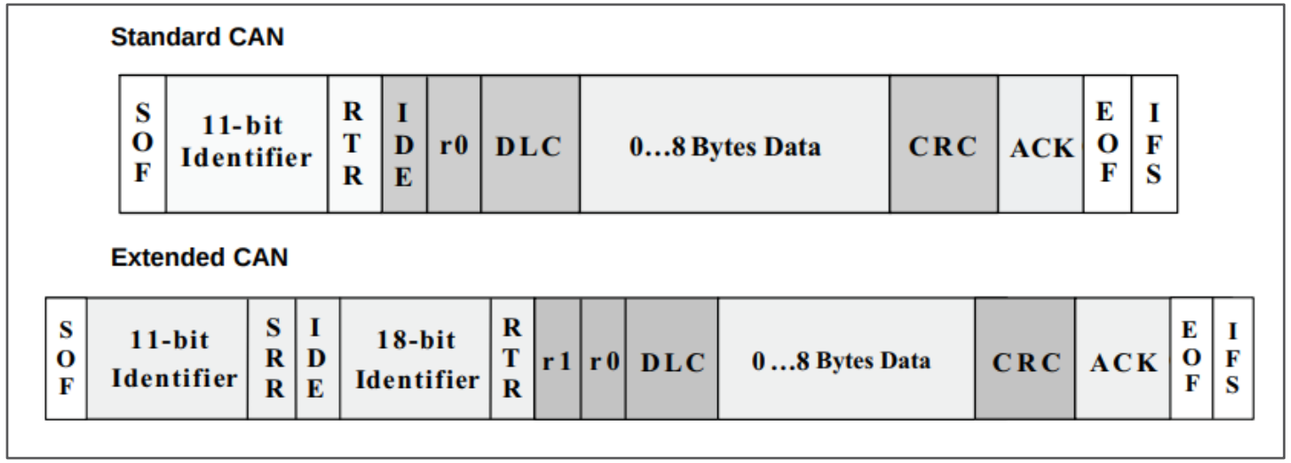
A typical heavy truck system

Background

- The ECUs communicate over an internal network called the Controller Area Network (CAN), which is known for its:
  - Robustness
  - Low cost
  - Speed
- CAN architecture:
  - Any node can talk, any node can listen
  - Any node can assert priority
  - Maximum of 8 bytes of data
  - No encryption or validation
- Heavy truck CAN network follows the J1939 standard



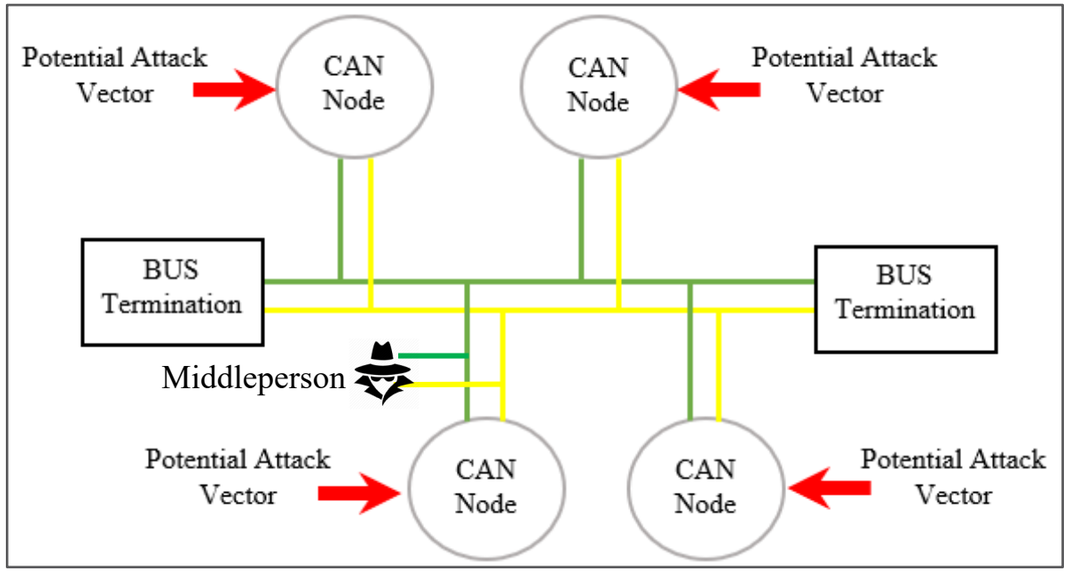
A typical CAN network



Standard and extended CAN frame structure

Background

- Heavy vehicle communication network is vulnerable due to its lack of data confidentiality and integrity protection
- If a node is compromised, possible attacks can be:
  - Denial of Service (DoS)
  - Middleperson
  - Diagnostic packets exploitation
  - ECUs firmware rewrite
  - Fuzzing
- Publicly available information from the J1939 open standard can be used to exploit truck systems



Abstracted CAN bus with vulnerabilities

Objective and Motivation

- Increasing overall cybersecurity posture, and mitigating risk and potential threats in heavy vehicles
- Understand the network utilization of fielded vehicles
  - Many third-party devices are added to the vehicles: telematics
  - Many researchers may not have access to realistic heavy vehicle logs
  - Trucks are highly customizable
- A large database of real CAN logs is necessary for referencing, verifying, and validating to develop an intrusion and anomaly detection mechanism
- NSF funded the project, with NMFTA as industry partner “SaTC: CORE: Small: Collaborative: GOALI: Detecting and Reconstructing Network Anomalies and Intrusions in Heavy Duty Vehicles - 1715409”
- Creating a pool of vehicle network data that is beneficial for industry and research

Approach

- Design and build an affordable, secure, open-source, standalone device for logging all heavy vehicle network traffic
- Implement a centralized management system for potential large data set
- An application is necessary for user to interact with the data

Hardware Design

Requirements

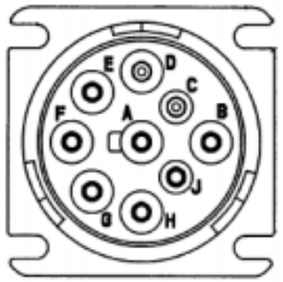
1. Support multiple CAN channels following the J1939 Deutsch 9-pin connector standards

Pin	Value
A	Ground
B	+12V
C	CAN/J1939 Hi
D	CAN/J1939 Lo
E	CAN/J1939 Shield
F	J1708/J1587 Hi
G	J1708/J1587 Lo
H	OEM Specific
J	OEM Specific

CAN0

J1708/CAN2

CAN1



SAE Standard 9-pin Deutsch connector

SAE J1939 type-1 (black) to type-2 (green) adapter cable

2. Inexpensive, with desired cost not exceeding $200
3. Capture all CAN messages, even at 100% bus load
4. Capture CAN error frames

Requirements

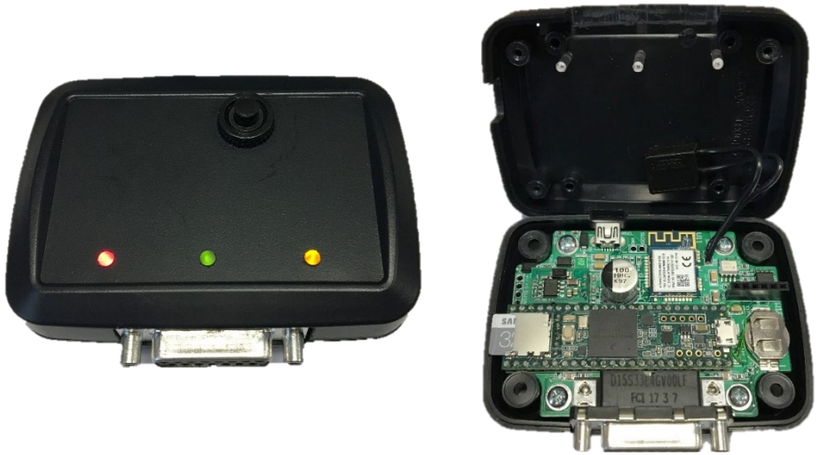
5. Utilize vehicle battery line from the diagnostic connector as a source of power
6. Withstand power failure without losing current logging data
7. Support a vehicle system up to 24V
8. Auto-detect CAN bus bitrate
9. Have removable external storage
10. Employ standard cryptographic implementations
11. Backend storage system needs to enable secure and a scalable access to the data
12. User-friendly and easy-to-navigate interface to upload and download files between the device and the server

A diagnostic port in a heavy truck

Design Alternatives

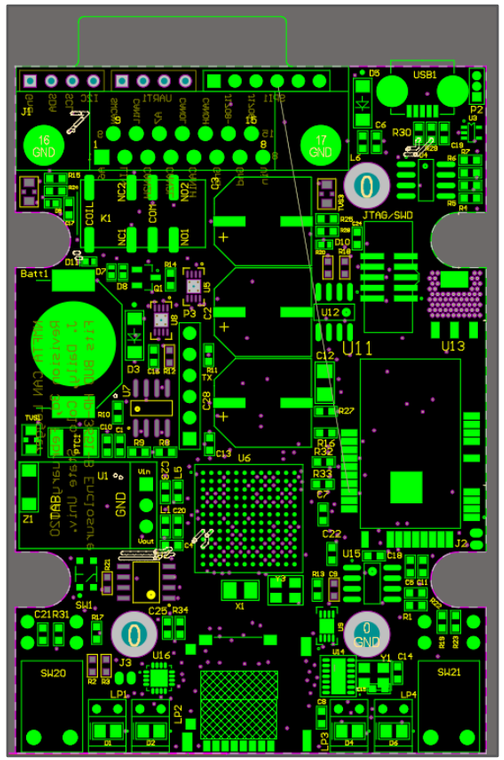
- Third-party devices are limited:
  - High cost
  - Proprietary information and limited modification
- Early CAN Logger versions:
  - NMFTA CAN Logger
    - 1 CAN channel
    - Low cost
  - CAN Logger 2
    - 3 CAN channels
    - Push button for programmable functionality
    - Hardware security module
    - WiFi module

NMFTA CAN Logger



CAN Logger 2

Final Design

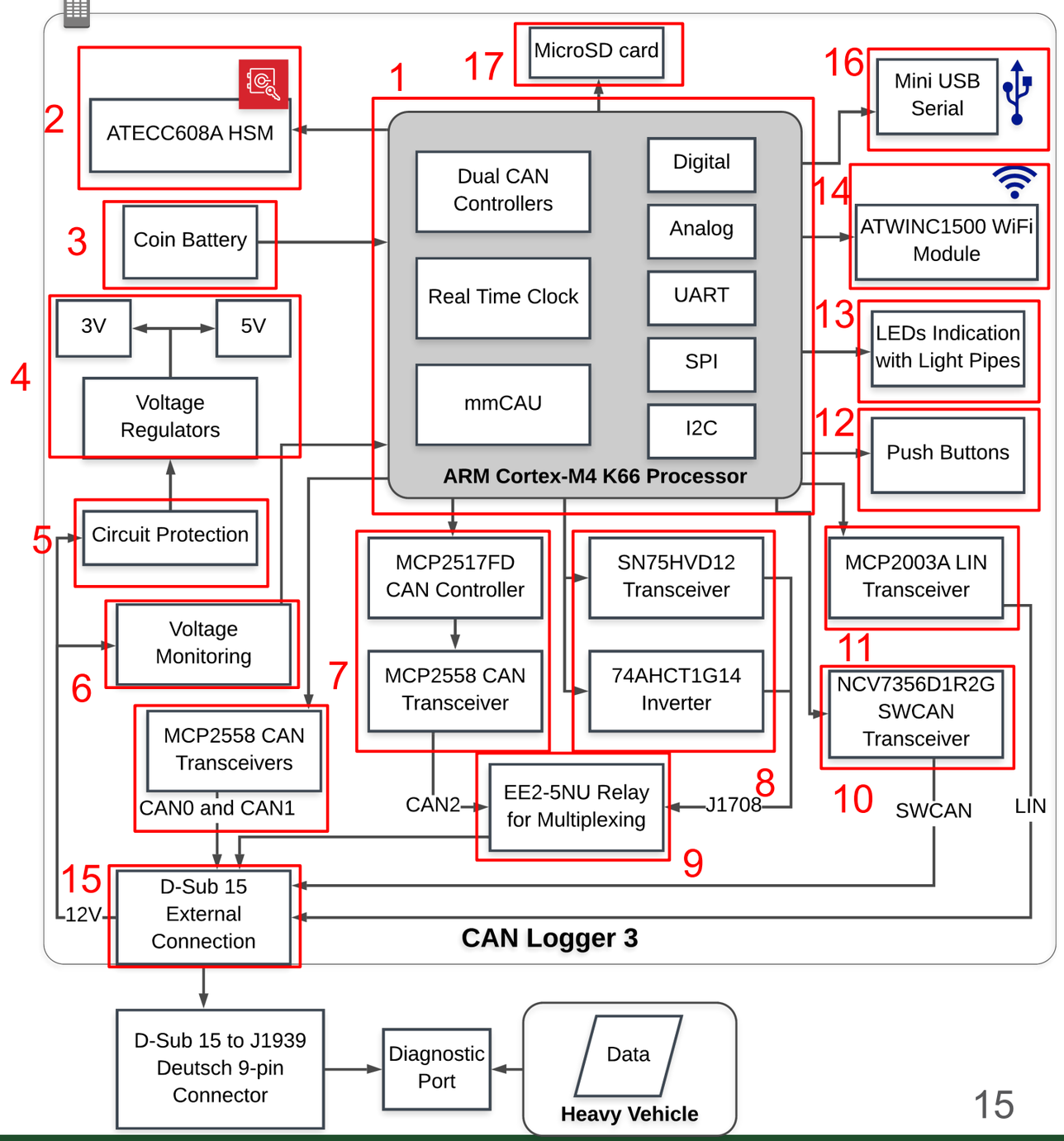


CAN Logger 3 Printed Circuit Board designed in Altium

CAN Logger 3, rev 3e

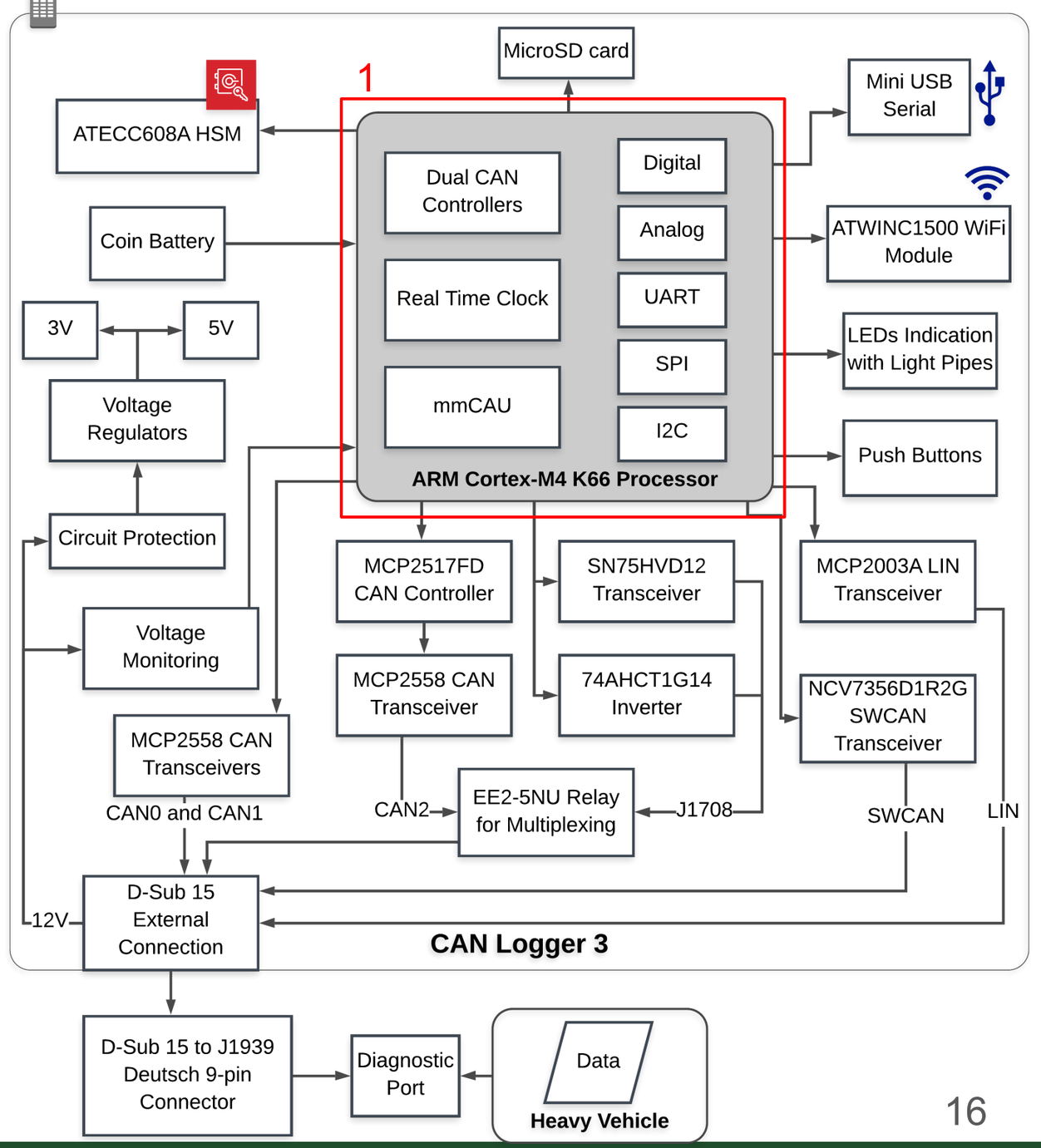
Block Diagram

1. Main processor
2. Coin battery
3. Hardware security module
4. Voltage regulators
5. Circuit protection
6. Voltage monitoring
7. CAN controllers and transceivers
8. J1708 circuit
9. Multiplexing switch
10. SWCAN circuit
11. LIN circuit
12. Push Buttons
13. LED indicators
14. WiFi module
15. D-Sub 15 connector
16. Mini USB serial
17. Micro SD card



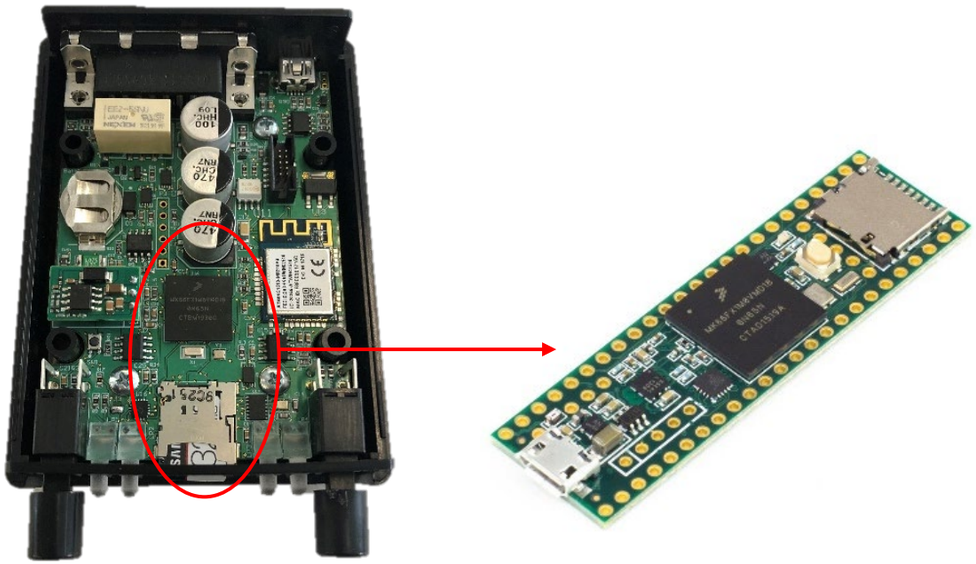
Key Components

1. Teensy 3.6 with K66 ARM Cortex M4F microprocessor
   - 180 MHz
   - CAN compatible with two onboard CAN controllers
   - Onboard SD card slot
   - Real-time clock
   - Memory-mapped Crypto Acceleration Unit (mmCAU)



Key Components

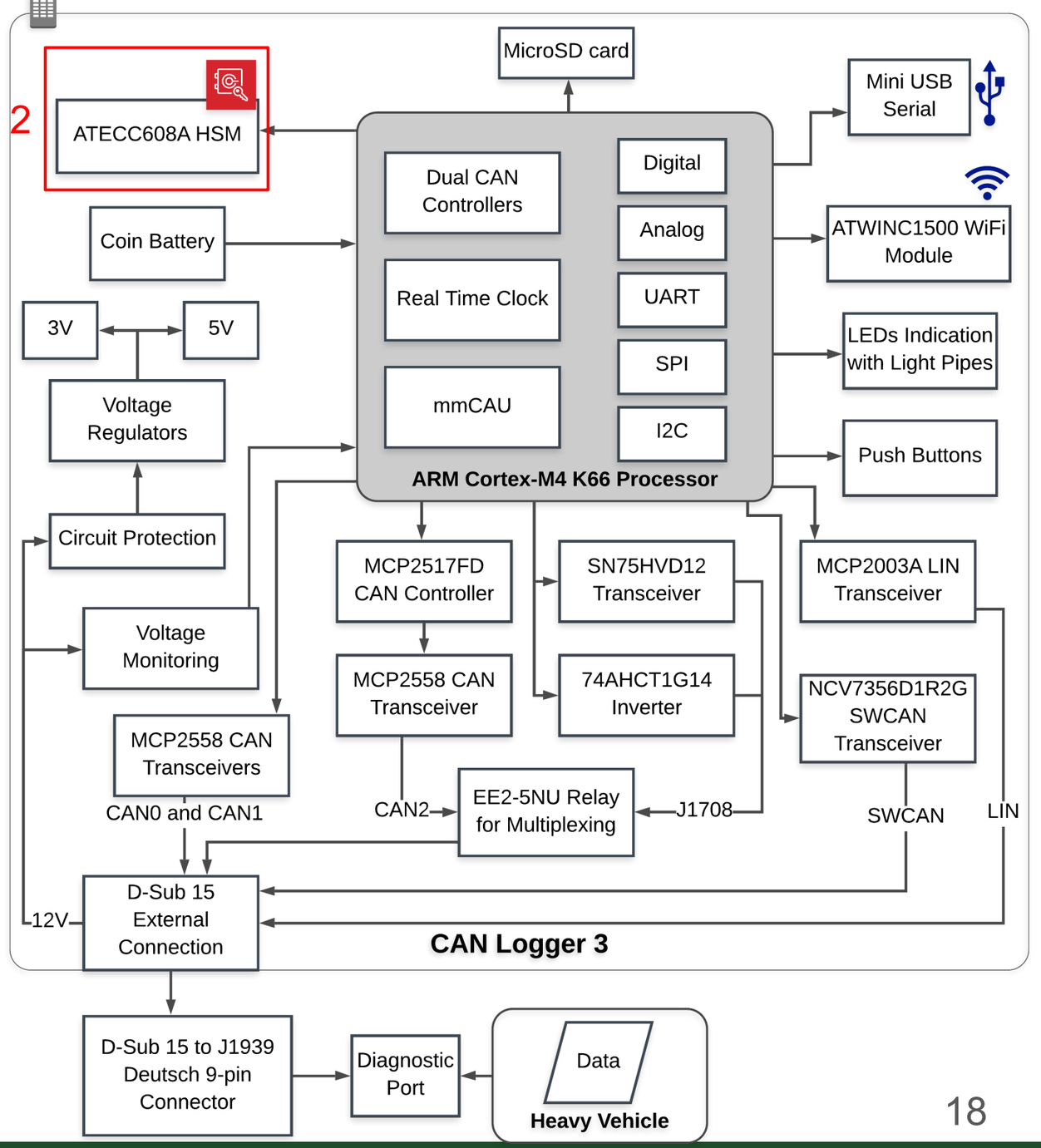
1. Teensy 3.6 with K66 microprocessor



Teensy 3.6 integrated into the CAN Logger 3

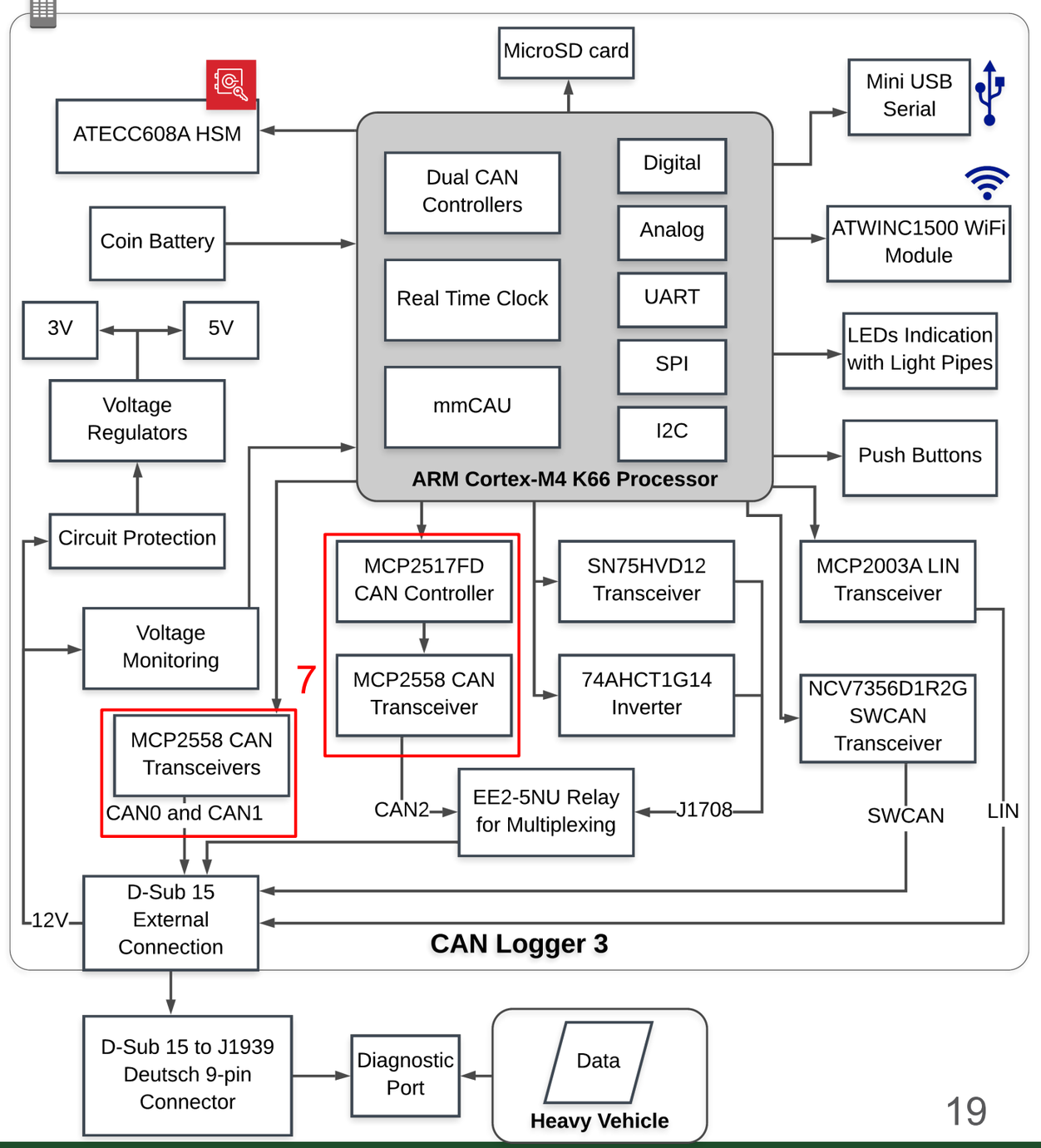
Key Components

2. ATECC608A hardware security module
   - Low cost
   - Communicate via I2C
   - Support AES-128
   - Support Elliptic-curve cryptography (ECC) P-256 following NIST standards
   - Secure key storage



Key Components

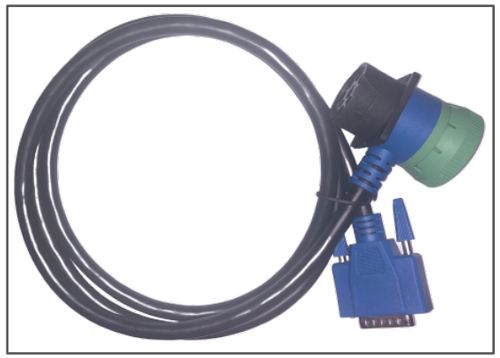
7. 3 CAN channels
   - CAN0 and CAN1 use the K66 processor's onboard CAN controller and separate CAN transceivers
   - CAN2 uses an external CAN controller (SPI) and a CAN transceiver



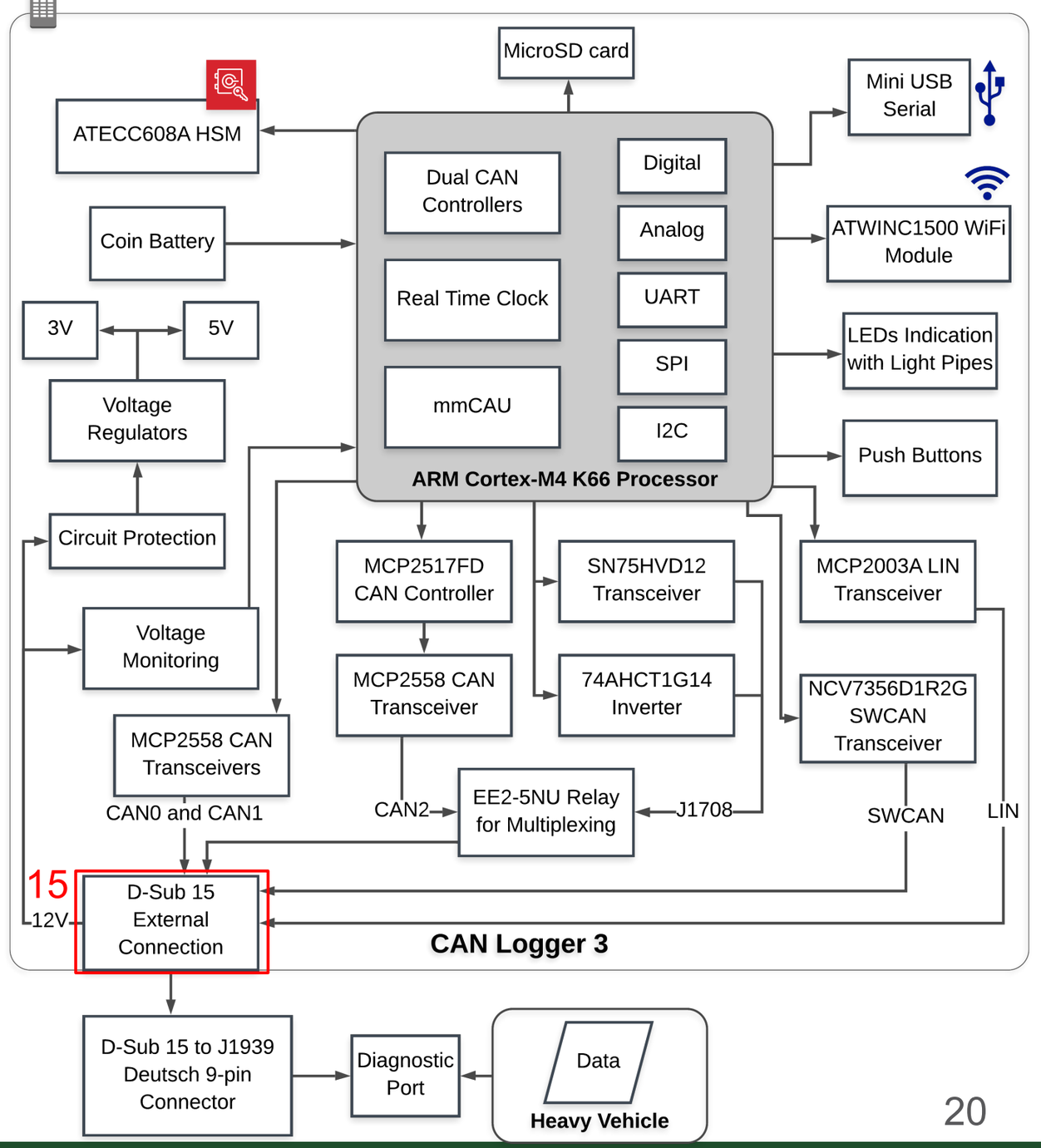
Key Components

15. D-Sub 15 to Deutsch 9-pin connector
    - Known for its commonality, robustness, and reliability
    - Output pins: 3 CAN channels, LIN, SWCAN, raw 12V, ground, and analog A6 and A7
    - A D-sub 15 to Deutsch 9-pin cable is used to connect the device to the vehicle

D-Sub 15 Female Connector

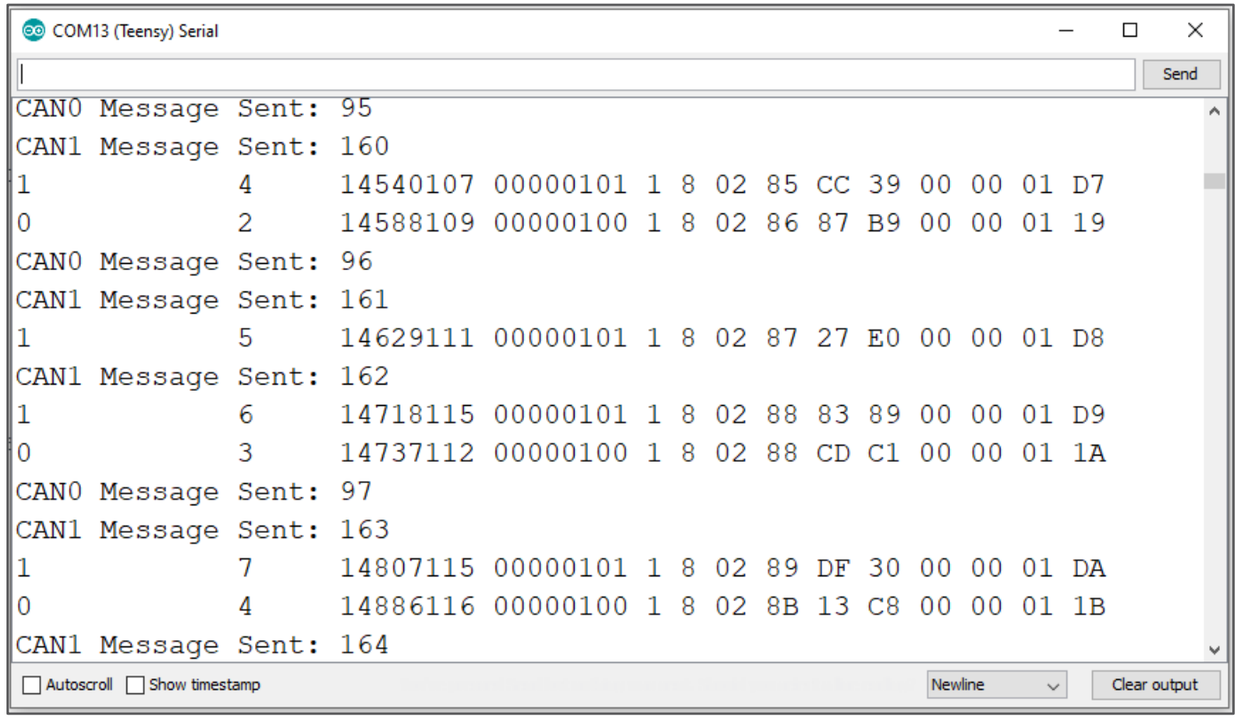


D-Sub 15 to Deutsch 9-pin cable



Functional Tests

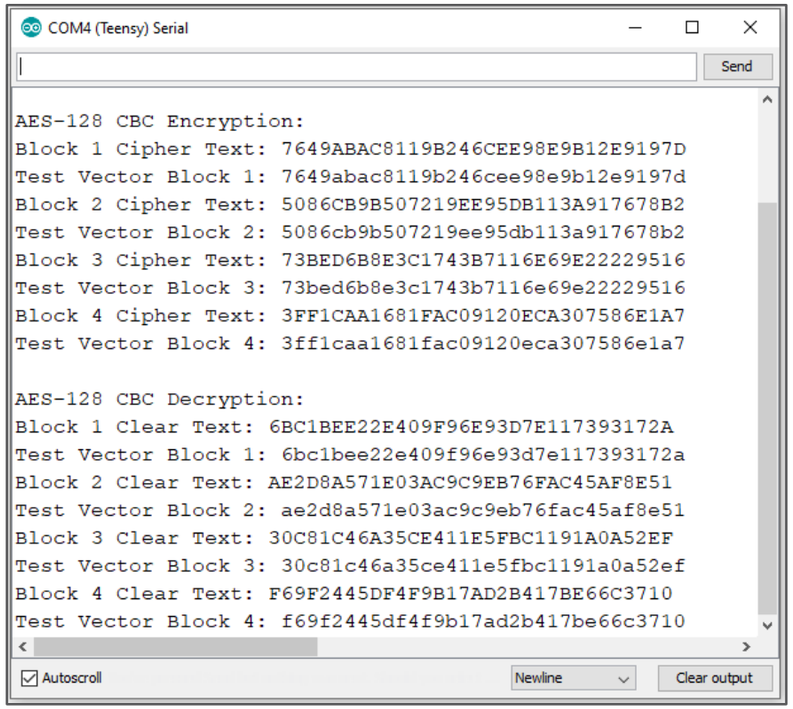
1. CAN0 and CAN1
   - Two CAN Loggers were connected to a CAN bus
   - One device sent, the other read on both channels
2. Autobaud
   - List of bitrates: 250,000, 500,000, 125,000, 666,666, 1,000,000
   - Poll for CAN messages, if any then at the correct bitrate
   - If none, check Receive Error Count (REC), if REC increases, iterate to the next bitrate in the list
   - If REC is 0, no CAN messages and keep polling



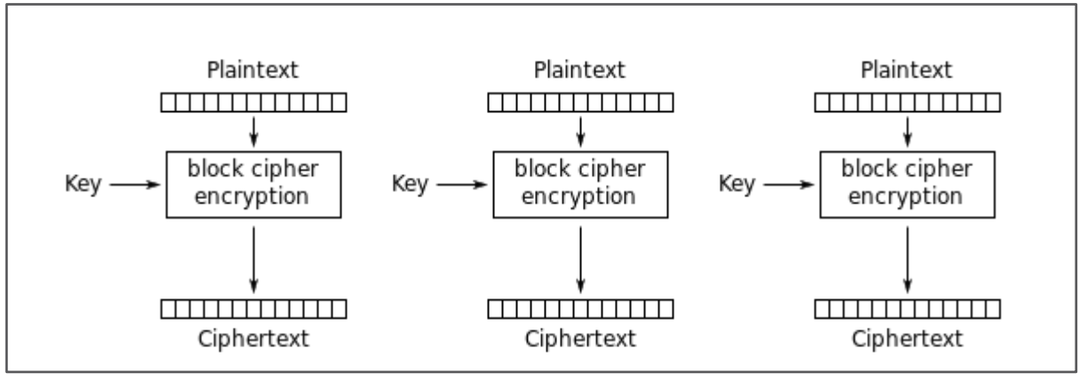
Results from one of the CAN loggers 3 serial monitor during the CAN0 and CAN1 testing

Functional Tests

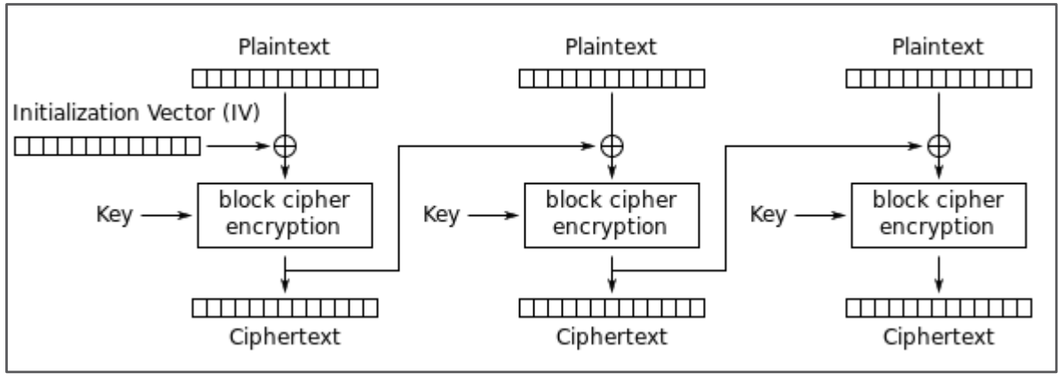
3. mmCAU AES-128 encryption and decryption
   - Cipher Block Chaining (CBC) referred over Electronic Codebook mode
   - AES-128 CBC encryption ad decryption were tested against NIST test vectors



Electronic Codebook mode encryption



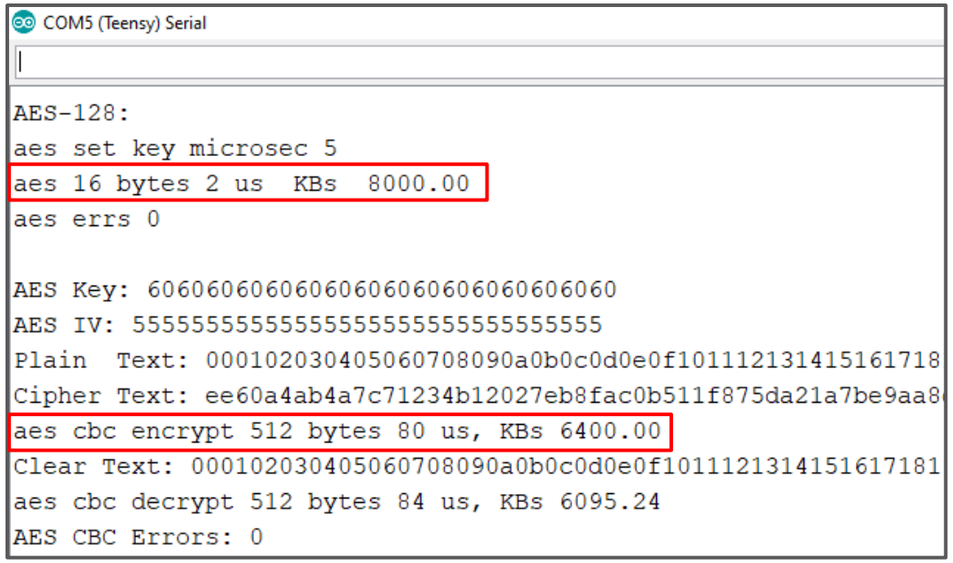
Electronic Codebook mode encryption



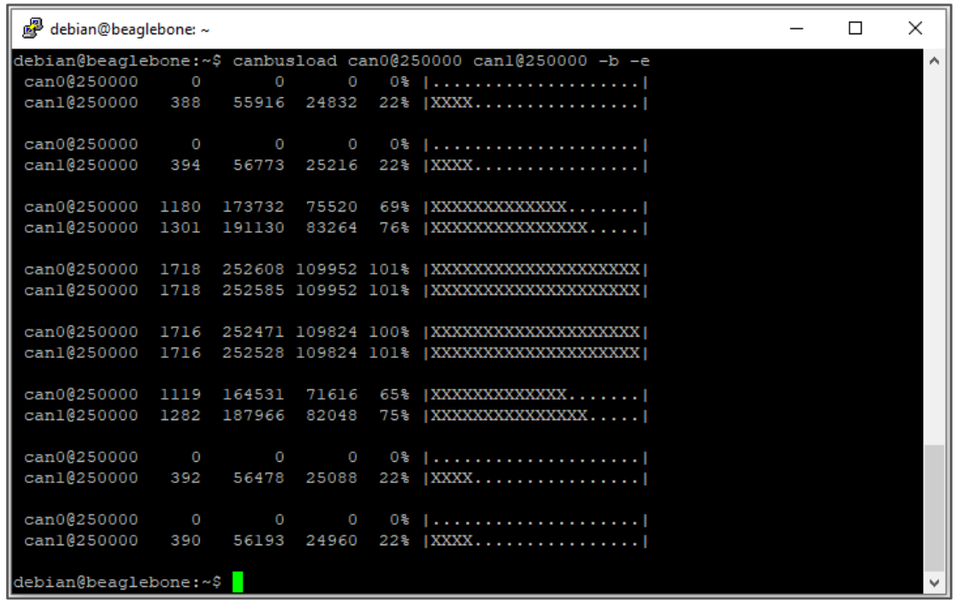
Cipher Block Chaining mode encryption

Functional Tests

4. Logging speed test
   - Encryption speed:
     - Encrypt a 512-byte buffer using AES-128 CBC: 6.4 Mbyte/sec
   - Full bus validation
     - Two CAN loggers connected to a CAN bus at 250 kbps
     - One for logging with encryption
     - One for flooding the bus by sending 4,000 CAN messages on both channels with ID of 0x15555555 and data of all 0xAA at interval of 520.5 microseconds
     - CAN bus load was measured using a CAN compatible TruckCape device with BeagleBoneBlack
     - All messages were captured
     - The test was redone for 1Mbps



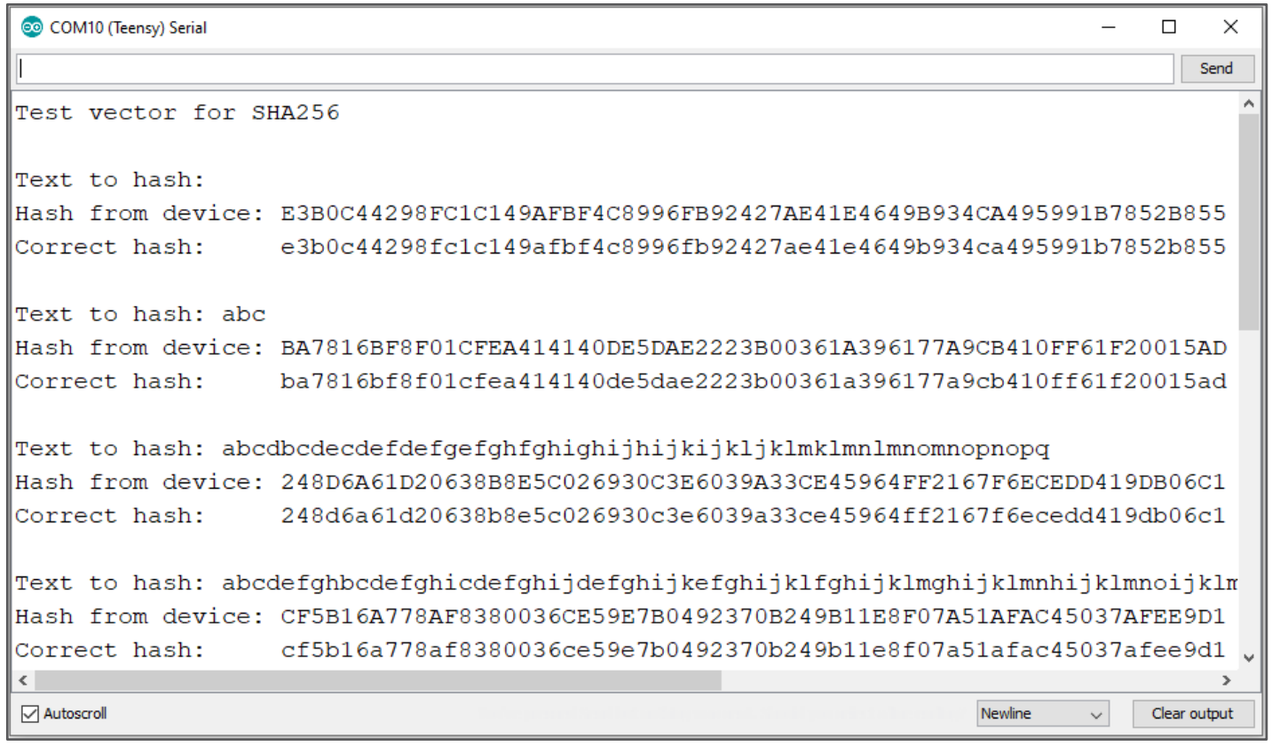
Encryption speed



CAN bus load measurement at 250kbps

Functional Tests

5. SHA-256 hash
   - SHA-256 hash is used for mapping data of arbitrary size to a unique fixed-size digest of 32 bytes
   - A good method to check if the data has been altered
   - SHA-256 functionality was tested against NIST test vectors



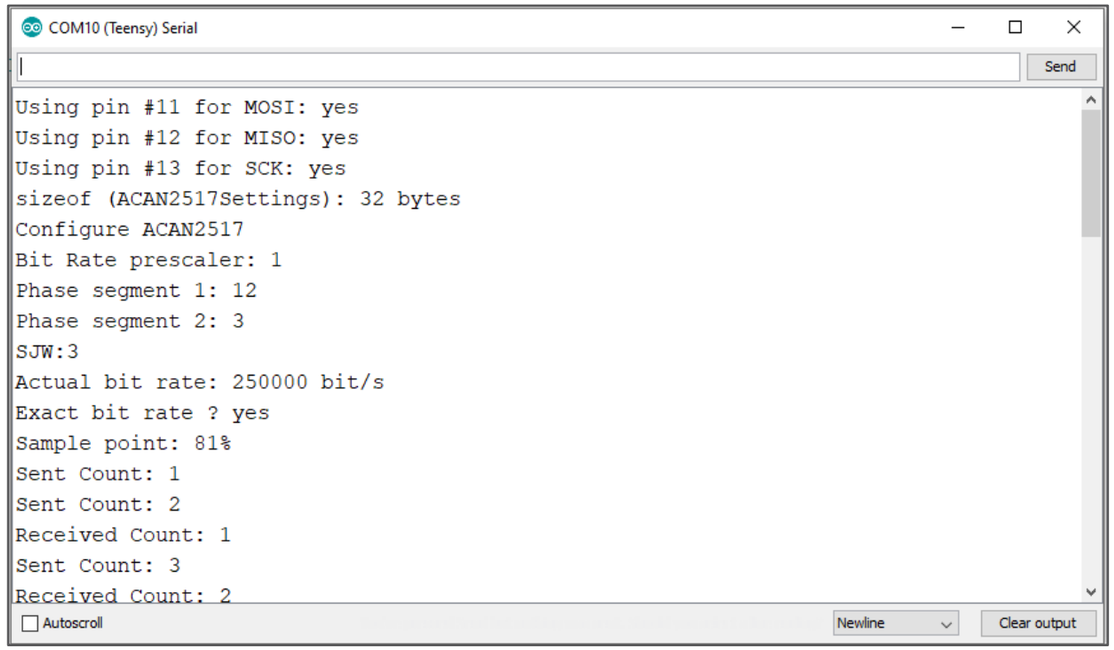
SHA-256 test

Functional Tests

6. ATECC608A
   - Key configuration
     - Specify how ECC keys are used and what slots they are generated/stored
     - Locked in order to use the ATECC608A functions
   - ECDH pre-master calculation
     - Calculate the same shared secret using the server private key and the client public key, or the other way around
     - Asymmetric cryptography to produce symmetric key
   - AES-128
     - 16-byte ECB mode encryption and decryption using the shared secret as the key
   - ECDSA sign and verify
     - Generate digital signature using the ECC private key and can be verified using its public key
     - Validate the integrity of the data and the sender

Functional Tests

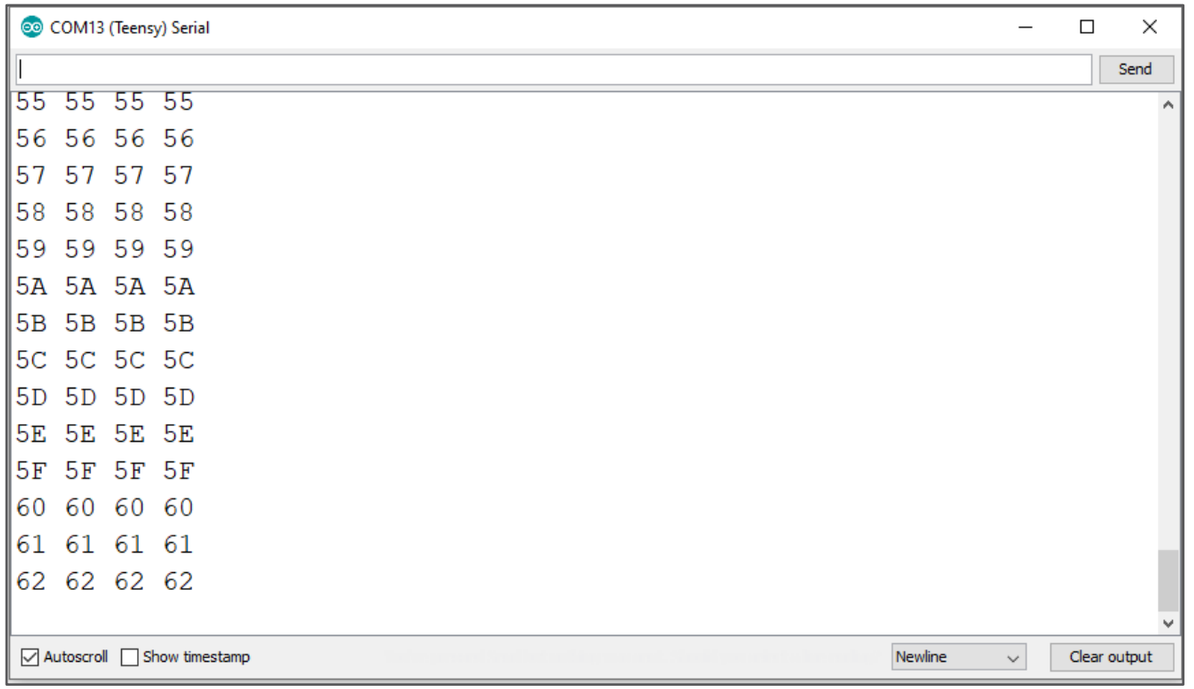
7. CAN2 and multiplexing
   - Multiplexing has default on J1708, CAN_switch pin needs to be pulled high to enable CAN2
   - Using ACAN2517 library instead of FlexCAN for CAN2 CAN controller
   - Perform internal loop back test to validate circuit
   - Full CAN2 functionality needs to be explored in future work



CAN2 internal loop back test

Functional Tests

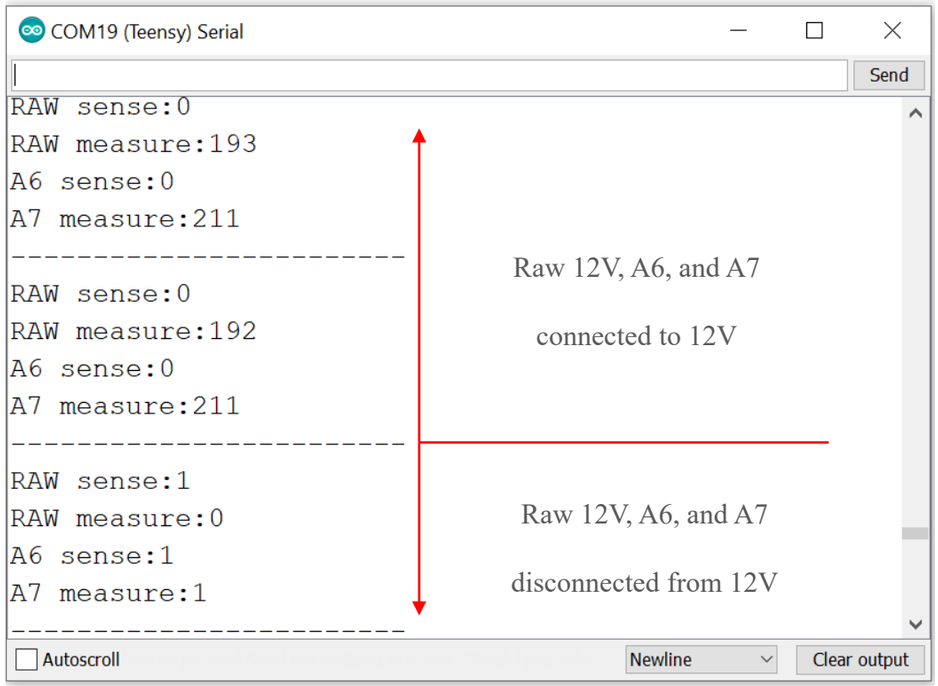
8. J1708
   - Two CAN Loggers were connected to each other for testing
   - One sending a message of 4-byte buffer, increasing by one for every message sent
   - One reading received messages



J1708 send and receive test

Functional Tests

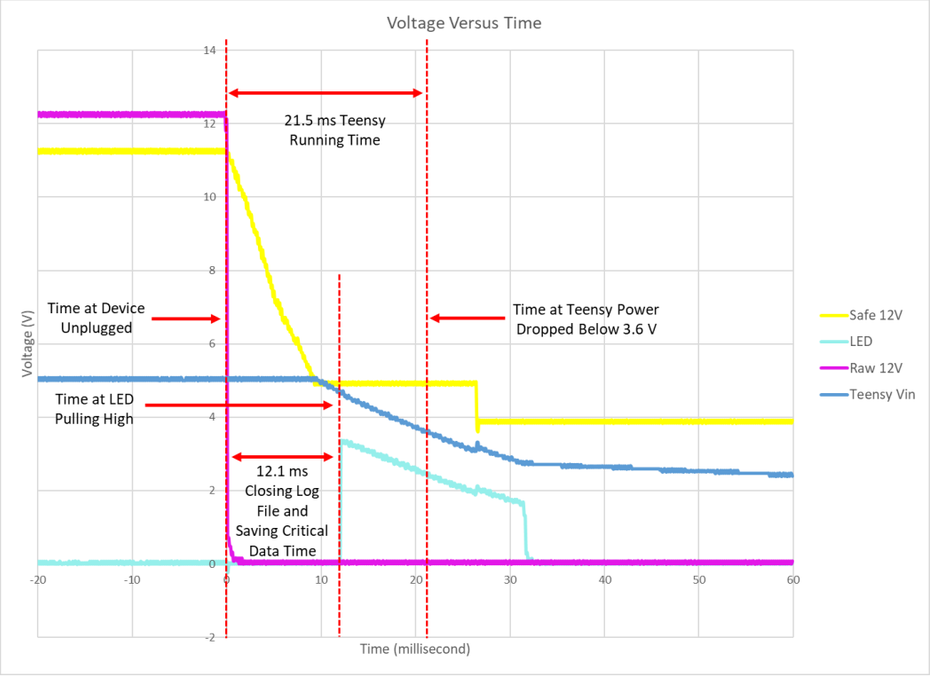
9. Voltage monitoring
   - External analog voltage measurement:
     - Raw 12V: 0-192 (0-12V)
     - Analog pin A7: 0-211 (0-12V)
   - Opto-isolator transistor:
     - Raw 12V and analog pin A6
     - 1 (true) for 0V and 0 (false) for 12V

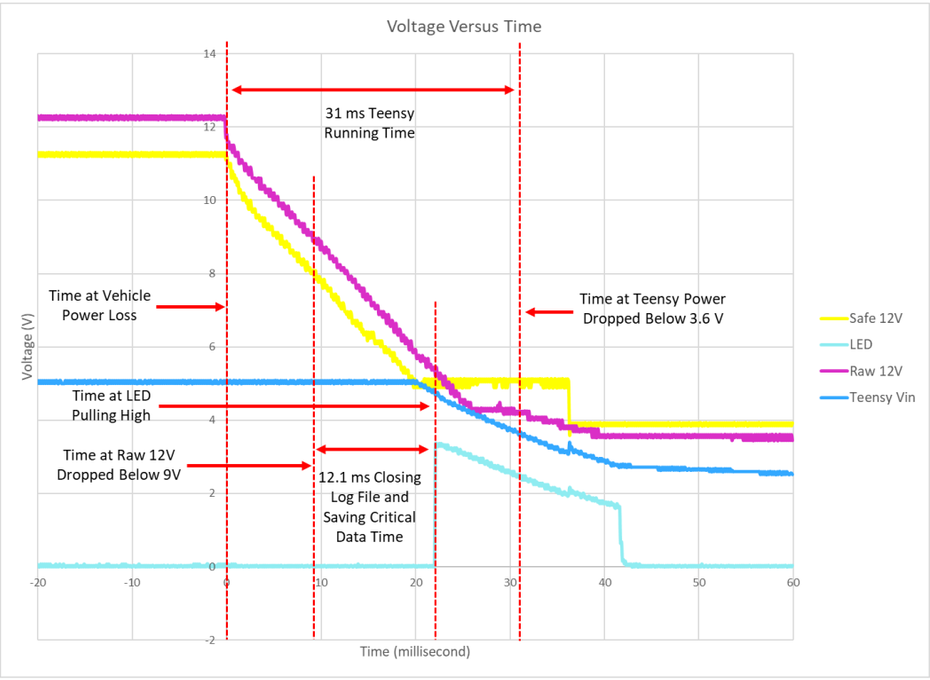


Voltage monitoring test

Functional Tests

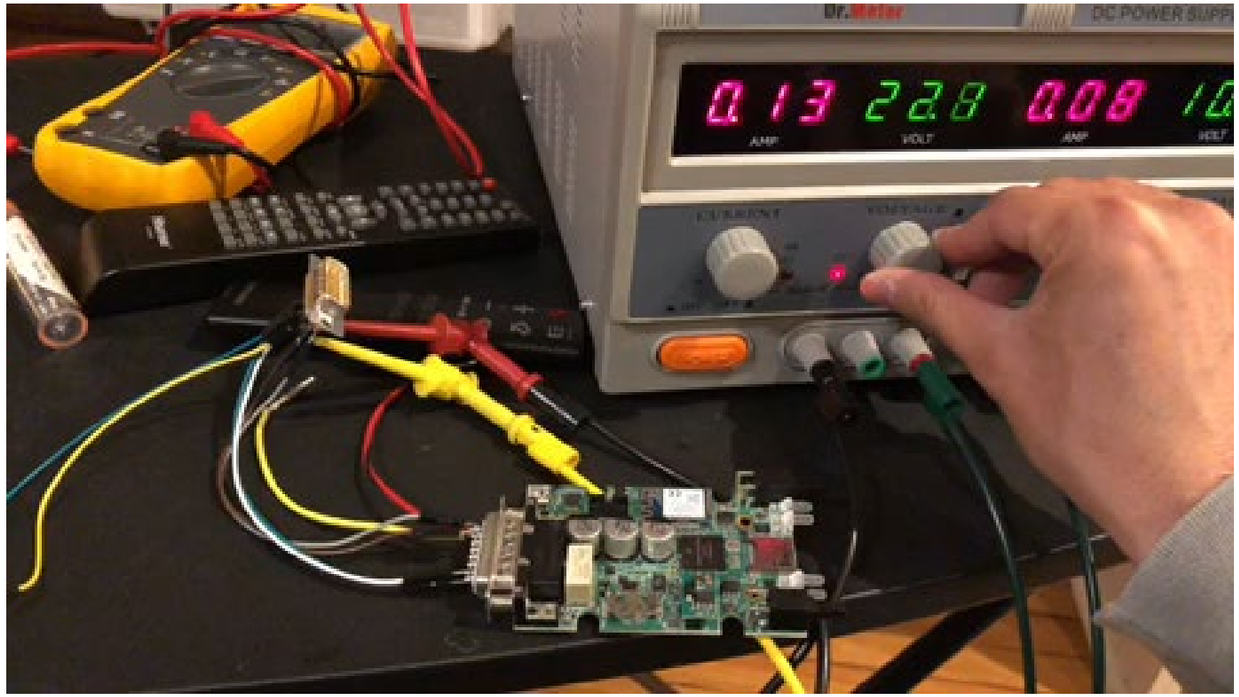
10. Power interruption
    - Cause by unplugging device
        - 12V power quickly drops (top picture)
    - Cause by heavy truck power loss
        - 12V power slowly decays (bottom picture)
    - Goal: Prevent data loss from power interruption
    - Solution:
        - Close file when device power drops below a threshold
        - Close bin file and store its hash digest in the memory
        - Generate missing metadata in the next power up





Functional Tests

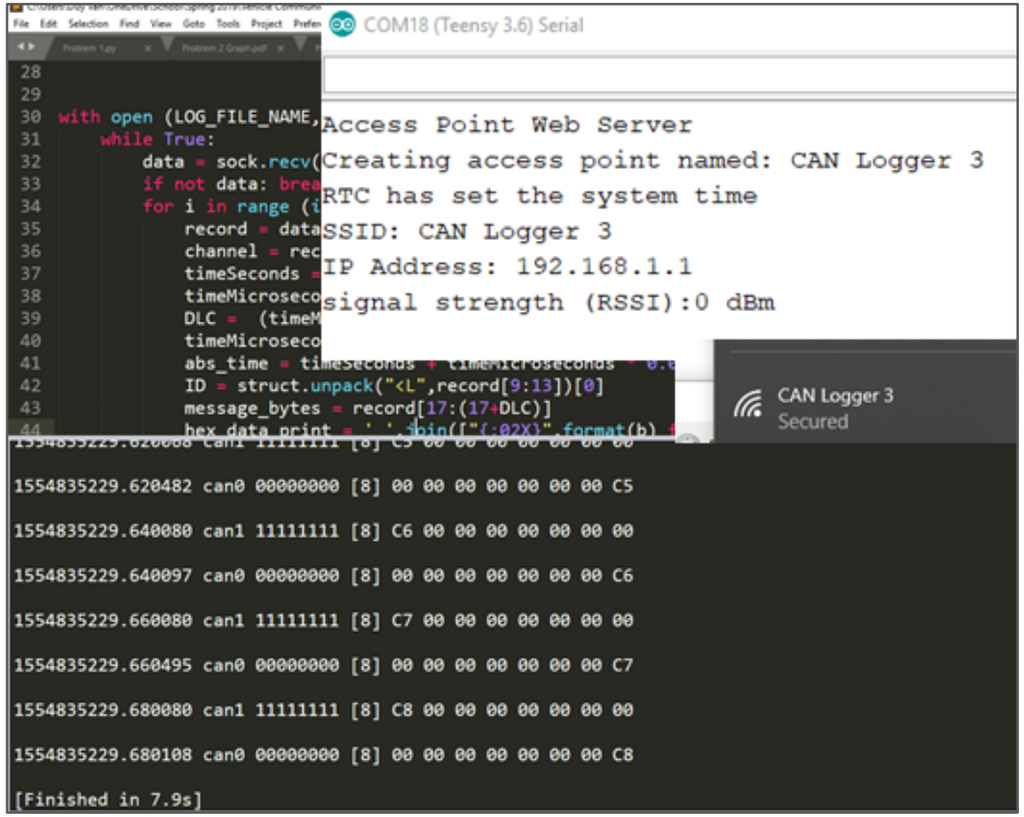
11. Destructive test
    - Max voltage of 36V test
    - Reverse polarity test



Max voltage test

Functional Tests

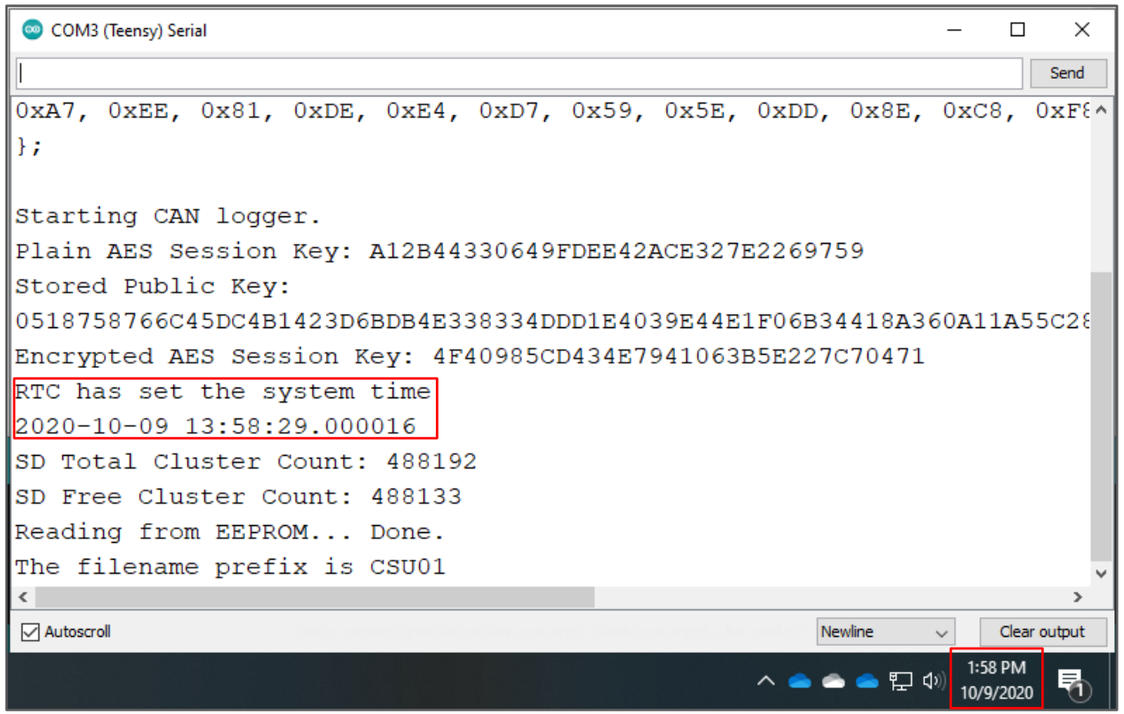
12. LEDs and buttons
    - Lights turned on when buttons were pushed
13. WiFi
    - Capture live CAN message
        - CAN Logger 3 broadcasts SSID as a host
        - A local PC connects as a client
        - CAN Logger 3 transfers live messages to the PC
        - Messages were dropped due to the limit speed
    - Can be used to transfer data from the device storage to a local PC for future work



Wireless CAN logging attempt

Functional Tests

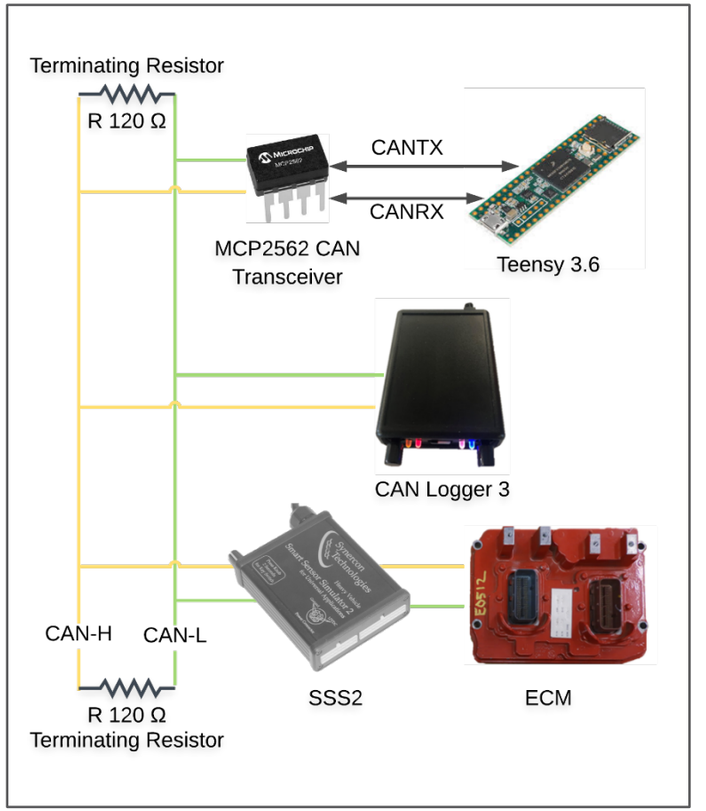
14. Real-time clock
    - https://www.pjrc.com/teensy/td_libs_Time.html
    - Synchronize the real-time clock with the PC
    - Run on the 3V coin battery
    - Test by logging data 7 days apart



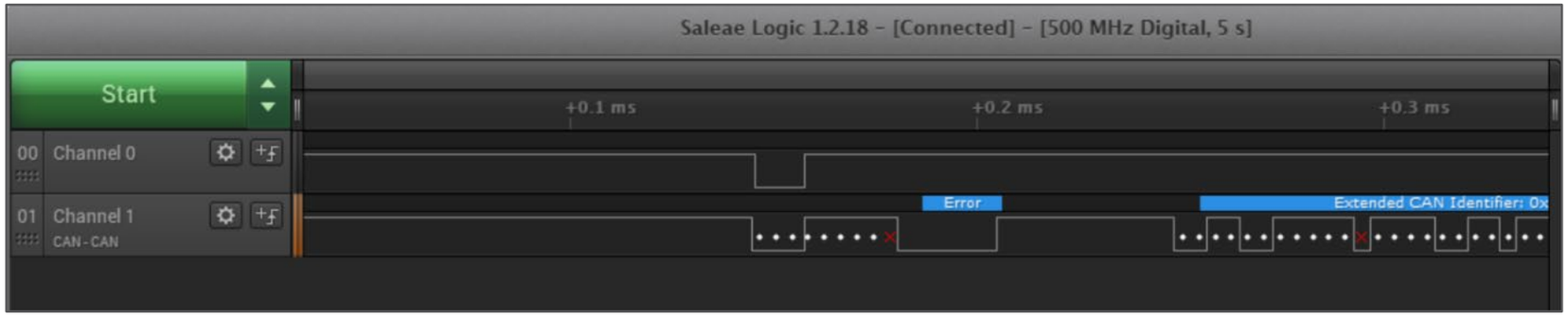
Real-time sync

Functional Tests

15. CAN Error frame
    - Test by injecting bit stuffing CAN error frame
    - Validated error occurred with Saleae Logic
    - Error frame captured by the CAN Logger:
        - ID of 0x20000008 and data of 0x0000040000000000
        - Error types are defined in error.h



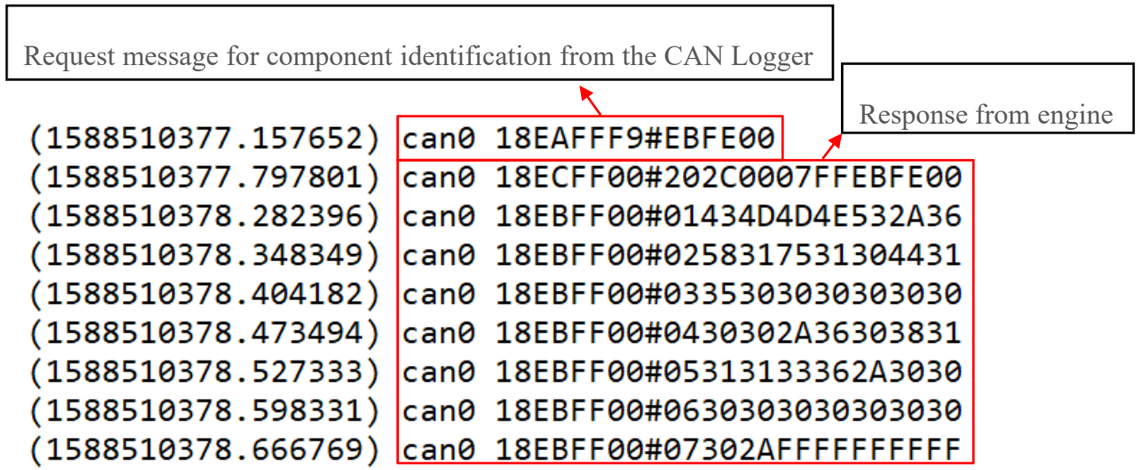
Bit stuffing error injection setup

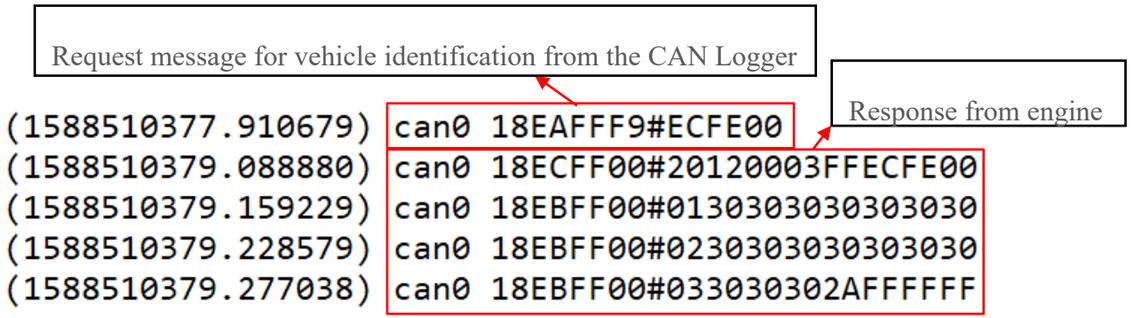


Saelae Logic signal capture

Functional Tests

16. Request messages
    - Based on PGN, as defined in the SAE J1939-71 Vehicle Application Layer
    - A series of request PGNs are sent when triggered
        - 65259 (0x00FEEB) – Component identification
        - 65260 (0x00FEEC) – Vehicle identification





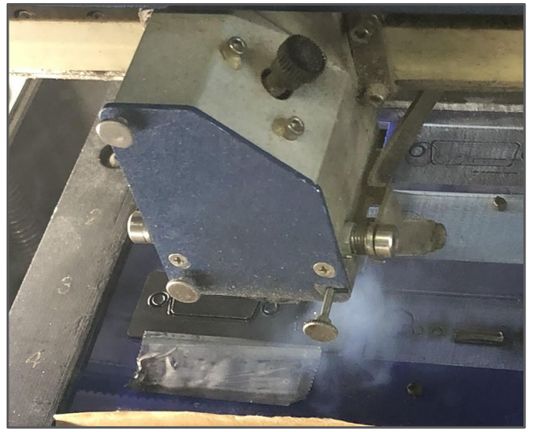

```
CMMNS*6X1u10D1500000000*60811136*0000000000* - component identification

00000000000000000* - vehicle identification (VIN)
```

Assembly and Manufacturing

- PCBs manufactured from third-party manufacture
- Final assembly:
  - Laser-cutting enclosure end panels
  - Attaching PCBs into their enclosures
  - Labelling
  - Configuring firmware

A batch of PCB boards



Laser-cutting end panels

Final products before shipment

Checklist

Complete this checklist before shipping a CAN Logger 3 to a Customer

☐ Customer Name: _____ ☐ Date: _____

☐ Remove SD card, connect logger to USB Serial and examine startup messages.

☐ With SD card removed, the red LED flashes.

☐ Logger time from USB Serial is within 1 minute of actual PC time.

☐ Logger and PC time zone: MDT (GMT-0700) or MST (GMT-0600) or Other: _____

☐ Red LED stops flashing after inserting an SD card.

☐ Enter the serial command for the ID (ID CSUXX) where XX is the logger number located on the enclosure.

☐ Device responds with Device ID: _____

☐ Enter the serial command to reset the count: COUNT 0.

☐ Device responds with Set current file to 000.

☐ Unplug and re-plug the USB Serial and observe solid green LED.

☐ Logger Number Printed on the enclosure: _____

☐ The filename prefix matches the number printed on the enclosure.

☐ Connect Logger to live CAN bus. Observe Green and Yellow LED flickering.

☐ Record the ATECC608 SN:_____

☐ Record first digits of the IV:_____ If zeros, then no encryption.

☐ Press the left button (near green). Observe red LED slow flash.

☐ Double click left button (near green). Observe a new file was created.

☐ Previous file showed SIZE, BIN-SHA, TXT-SHA, and SIG.

☐ Note filename from Serial console.

☐ Disconnect USB Power first, then disconnect 12V Power.

☐ Remove SD Card from Logger, connect to computer.

☐ Open last file in hex editor (HxD) and calculate SHA-256:_____

☐ Eject SD Card, reinsert to Logger, connect USB Serial.

☐ Previous file meta data shows BIN-SHA matching calculated SHA.

☐ Format SD card (FORMAT). Confirm with LS A being empty.

☐ Reset Counter to zero (COUNT 0).

☐ Logger Device ID and Serial Number match on https://systemscyber.github.io/CAN-Logger-3/loggers.html

Bill of Materials

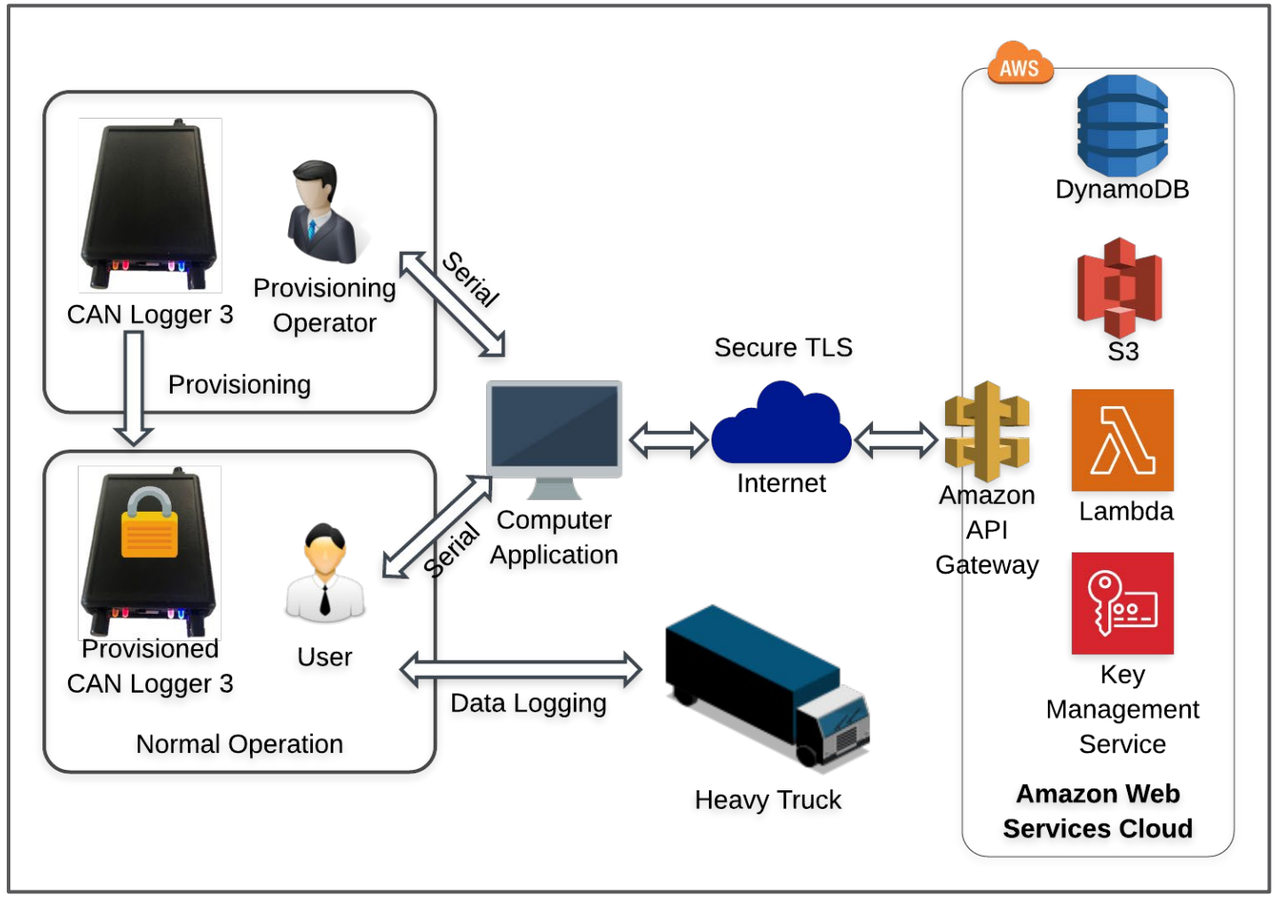
- PCB components: $86.58
- Colorado PCB Assembly: $3,787.50 for 25 devices
  - $151.5 per device
- With enclosure and final assembly:
  - Assume 0.5-hour work per device with $60/hour, final cost is ~$180 per device

Comment	Designator	Quantity	Supplier Part Number 1	Supplier Unit Price 1	Supplier 1
Yellow	D4	1	475-2560-1-ND	0.29	Digi-key
CR1225	Batt1	1	BAT-HLD-012-SMT-ND	0.29	Digi-key
0.1uF	C1, C4, C5, C6, C7, C8, C9, C11, C13, C17, C18, C19, C20, C21, C22, C23, C24, C25	18	399-6856-1-ND	0.101	Digi-key
2.2uF	C10, C16	2	587-3386-1-ND	0.12	Digi-key
22uF	C12	1	490-12451-1-ND	1.02	Digi-key
16pF	C14, C15	2	311-3964-1-ND	0.1	Digi-key
470µF	C2, C28	2	PCE3751CT-ND	0.81	Digi-key
100uF	C3	1	P19732CT-ND	0.81	Digi-key
Green	D1	1	475-1410-1-ND	0.27	Digi-key
Red	D2	1	475-1278-1-ND	0.29	Digi-key
ACURA107-HF	D3, D5	2	641-1884-1-ND	0.4	Digi-key
Blue	D6	1	516-1437-1-ND	0.95	Digi-key
Diode	D7, D8, D9, D10, D11	5	CCS15S30L3FCT-ND	0.38	Digi-key
Vehicle Interface Cable Connector	J1	1	609-1498-ND	2.3	Digi-key
JTAG/SWD	JTAG/SWD	1	1175-1735-ND	0.73	Digi-key
EE2-5SNU	K1	1	399-11056-5-ND	1.7	Digi-key
1k	L1, L5, L6	3	490-17350-1-ND	0.1	Digi-key
SLP3	LP1, LP2, LP3, LP4	4	492-2517-ND	0.64	Digi-key
PTC RESTTBLE 0.75A	PTC1	1	507-1765-1-ND	0.21	Digi-key
NUD3124	Q1	1	NUD3124LT1GOSCT-ND	0.41	Digi-key
4.7k	R1, R4, R7, R10, R21, R22	6	RHM4.7KAYCT-ND	0.14	Digi-key
10k	R18, R20, R24, R28, R34	5	RHM10.0KAYCT-ND	0.14	Digi-key
330	R3, R19, R23, R31	4	RHM330AYCT-ND	0.14	Digi-key
100	R2	1	RHM100AYCT-ND	0.14	Digi-key
47	R5, R6, R29, R30	4	RHM47AYCT-ND	0.14	Digi-key
100k	R8, R9, R11, R12, R13, R14, R15, R16, R17, R25, R27, R32, R33	13	RHM100KAYCT-ND	0.129	Digi-key
503182-1852	SD1	1	WM12834CT-ND	2.45	Digi-key
PTS830	SW1	1	CKN10587CT-ND	0.51	Digi-key
PB400	SW20, SW21	2	EG5548-ND	2.05	Digi-key
Varistor 42V	TVS1	1	478-2485-1-ND	0.62	Digi-key
Varistor 24V	TVS2, TVS3	2	478-2484-1-ND	0.79	Digi-key
OKI-78SR	U1	1	811-2692-ND	4.3	Digi-key
ATML-ATWINC1500-MR210PA-28	U11	1	ATWINC1510-MR210PB1140-ND	8.32	Digi-key
MOCD207R2M	U12	1	MOCD207R2MCT-ND	1.03	Digi-key
NCP1117LPST33	U13	1	NCP1117LPST33T3GOSCT-ND	0.46	Digi-key
MCP2517FD	U14	1	MCP2517FDT-H/JHACT-ND	2.31	Digi-key
ATECC608A	U15	1	ATECC608A-SSHDA-TCT-ND	0.75	Digi-key
MKL02Z32VFG4	U16	1	IC_MKL02Z32_QFN16	6.8	PJRC
NCV7356D1R2G	U2	1	NCV7356D1R2GOSCT-ND	1.8	Digi-key
74AHCT1G14	U3	1	74AHCT1G14SE-7DICT-ND	0.25	Digi-key
SN75HVD08DR	U4	1	296-37893-1-ND	3.97	Digi-key
MCP2558	U5, U8, U9	3	MCP2558FDT-H/MNYCT-ND	0.81	Digi-key
MK66FX1M0VMD18 (preprogrammed)	U6	1	568-13335-ND	17.65	Digi-key
MCP2003A-E/SN	U7	1	MCP2003A-E/SN-ND	0.82	Digi-key
UX60SC-MB-5S8	USB1	1	H11589CT-ND	1.05	Digi-key
32.768 KHz	X1	1	XC2292CT-ND	0.59	Digi-key
40MHz	Y1	1	XC3069CT-ND	0.5	Digi-key
16MHz	Y3	1	XC2866CT-ND	0.69	Digi-key
SMA6J	Z1	1	SMA6J24CA-TPMSCT-ND	0.41	Digi-key

Software Design

Process Overview

- Provisioning process
  - Exchange public key between the device and server
  - Prepare the CAN Logger 3 for normal operation
- Normal operation
  - Log, encrypt, and sign data from vehicles
  - Securely upload and download log files to and from server through a client computer application
- Cybersecurity factors are assumed to be uncompromised
  - Local computer with client application
  - Provisioning operator and users
  - AWS third-party
  - The Internet communication with TLS

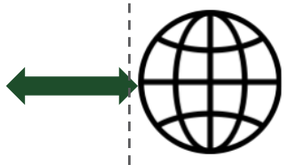


Provisioning

HSM

Login

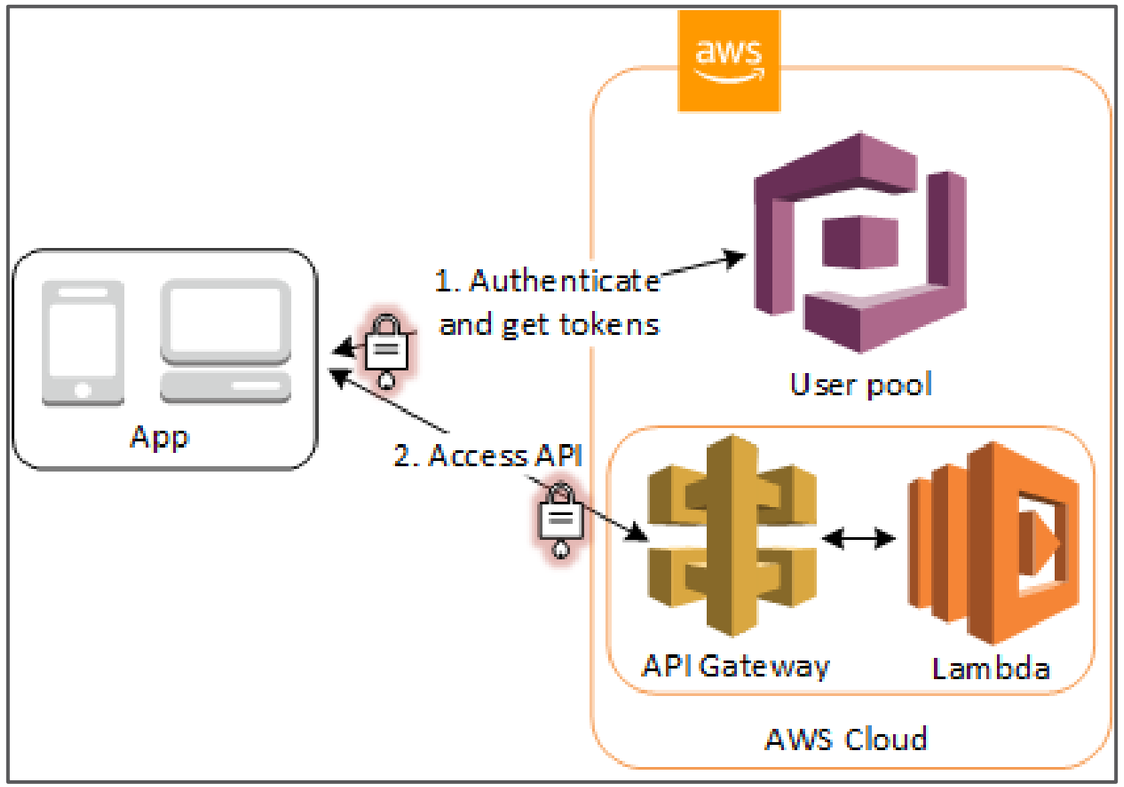
Provisioning Operator



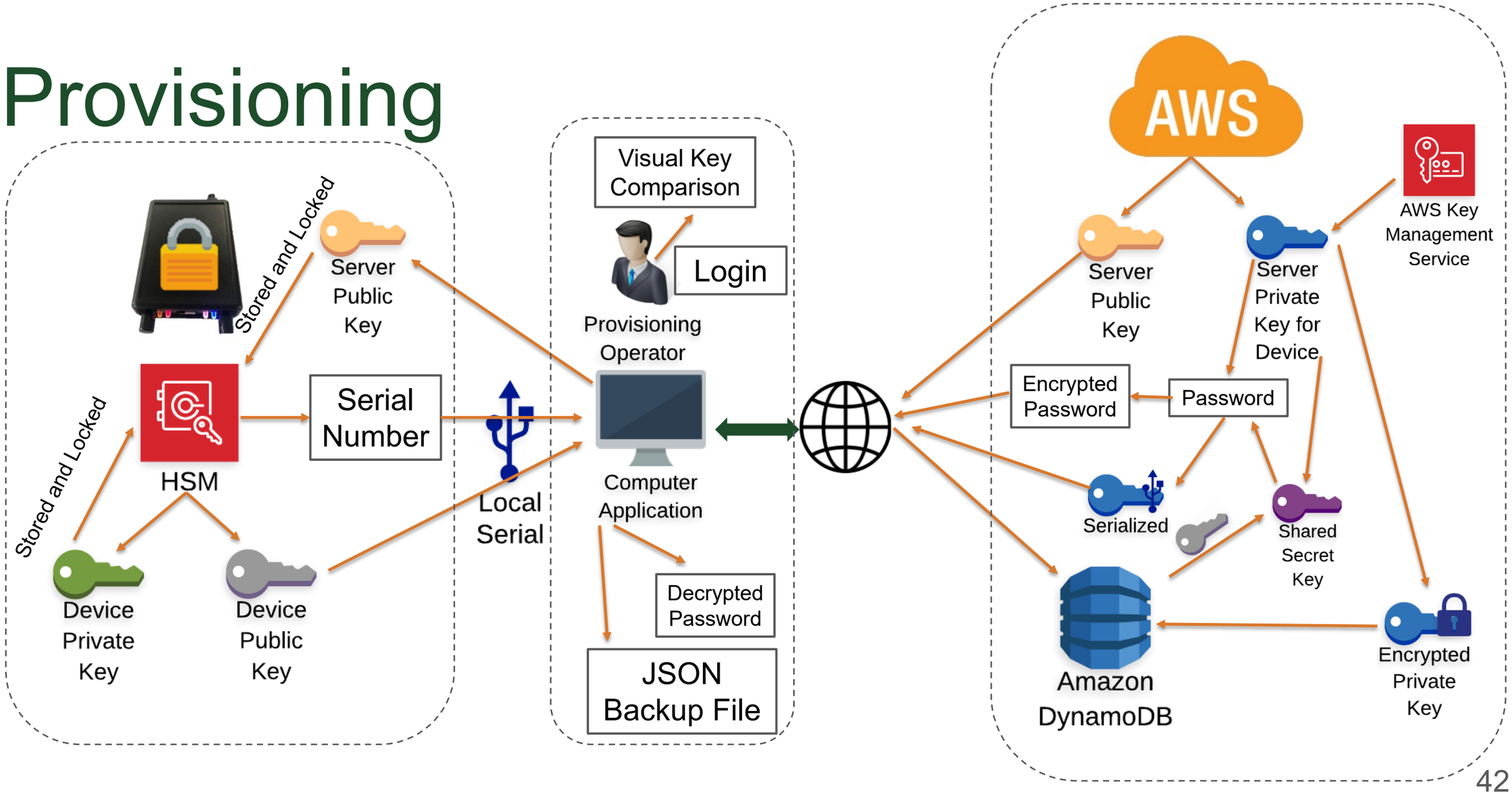
Computer Application

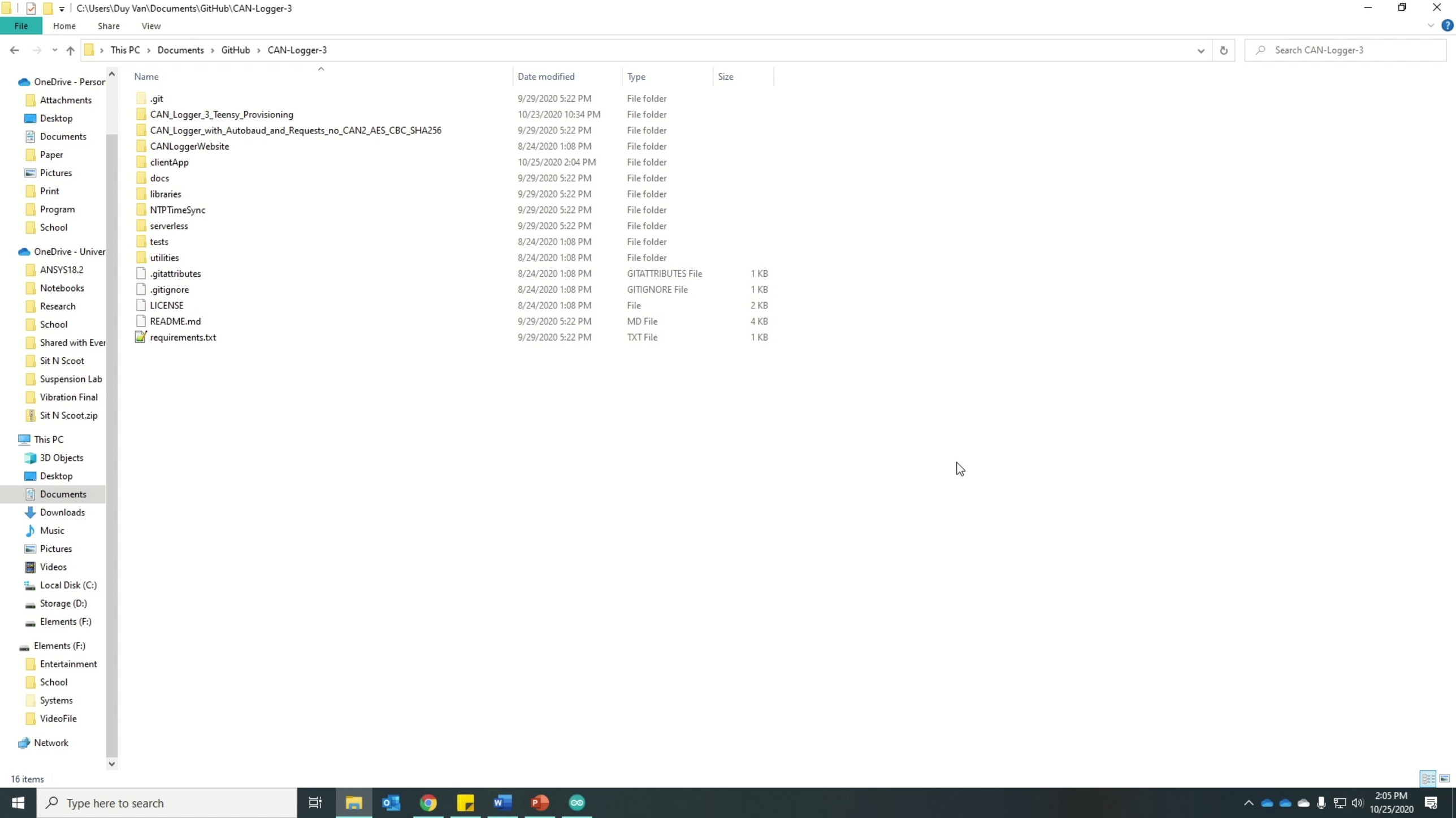
AWS

Login

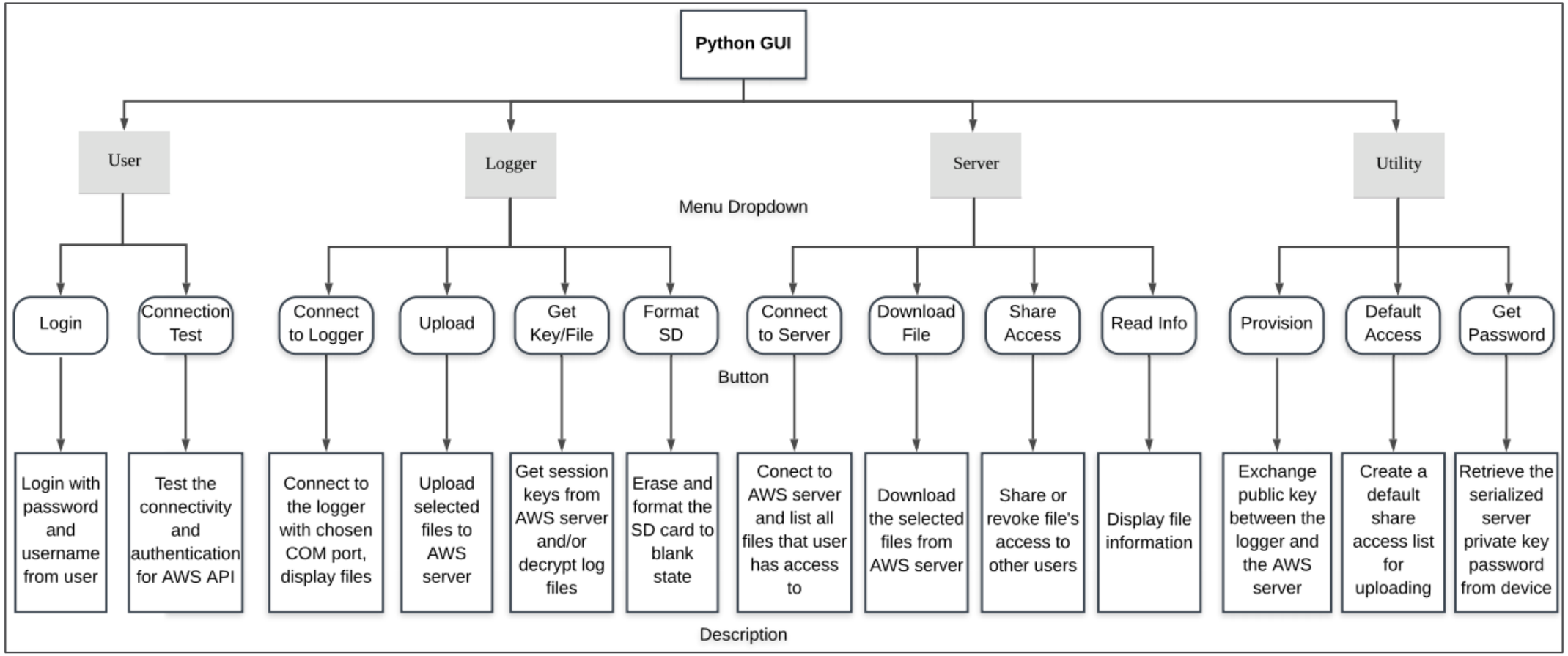

aws
1. Authenticate
and get tokens
User pool
App
2. Access API
API Gateway
Lambda
AWS Cloud

Provisioning




C:\Users\Duy Van\Documents\GitHub\CAN-Logger-3
File
Home
Share
View
This PC › Documents › GitHub › CAN-Logger-3
Search CAN-Logger-3
OneDrive - Persor
Attachments
Desktop
Documents
Paper
Pictures
Print
Program
School
OneDrive - Univer
ANSYS18.2
Notebooks
Research
School
Shared with Ever
Sit N Scoot
Suspension Lab
Vibration Final
Sit N Scoot.zip
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Storage (D:)
Elements (F:)
Elements (F:)
Entertainment
School
Systems
VideoFile
Network
Name
Date modified
Type
Size
.git 9/29/2020 5:22 PM File folder
CAN_Logger_3_Teensy_Provisioning 10/23/2020 10:34 PM File folder
CAN_Logger_with_Autobaud_and_Requests_no_CAN2_AES_CBC_SHA256 9/29/2020 5:22 PM File folder
CANLoggerWebsite 8/24/2020 1:08 PM File folder
clientApp 10/25/2020 2:04 PM File folder
docs 9/29/2020 5:22 PM File folder
libraries 9/29/2020 5:22 PM File folder
NTPTimeSync 9/29/2020 5:22 PM File folder
serverless 9/29/2020 5:22 PM File folder
tests 8/24/2020 1:08 PM File folder
utilities 8/24/2020 1:08 PM File folder
.gitattributes 8/24/2020 1:08 PM GITATTRIBUTES File 1 KB
.gitignore 8/24/2020 1:08 PM GITIGNORE File 1 KB
LICENSE 8/24/2020 1:08 PM File 2 KB
README.md 9/29/2020 5:22 PM MD File 4 KB
requirements.txt 9/29/2020 5:22 PM TXT File 1 KB
16 items
Type here to search
2:05 PM
10/25/2020

Client Application Function Description



512-byte Data Structure

- Byte 0-3: data structure version
- Byte 4-478: 19 CAN frames (25 byte each)
- Byte 479-490: RX count
- Byte 491-496: REC, TEC
- Byte 497-507: Logger version, number, file number, write time
- Byte 508-511: CRC checksum

SD Card Memory Block. 512 Bytes are stored at a time in the following format

Bytes	0	1	2	3	4 through 478	479	480	481	482	483	484	485	486	487	488	489	490
Data	C	A	N	2	Nineteen (19) CAN Frames	RXCount0				RXCount1				RXCount2			
Hex	43	41	4E	32	SEE CAN FRAME STRUCTURE	MSB			LSB	MSB			LSB	MSB			LSB
Notes	Characters					uint32_t				uint32_t				uint32_t			

491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511
Can0	Can1	Can2	Can0	Can1	Can2	T	U	2	_	_	N1	N2	N3	Write Time			CRC32			
uint8_t	uint8_t	uint8_t	uint8_t	uint8_t	uint8_t	54	55	32			ASCII Encoded			MSB		LSB	MSB			LSB
Receive Error Counts			Transmit Error Counts			Version			Logger Number		File Number			Microseconds for SDCard			Calculated from bytes 0 through 507			

CAN Frame Structure

Bytes	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Data	Channel	Timestamp				System				CAN Identifier				DLC	Microseconds per			B0	B1	B2	B3	B4	B5	B6	B7
Hex	0 | 1 | 2	LSB			MSB	LSB			MSB	LSB			MSB	8	LSB		MSB	01	02	03	04	05	06	07	08
Notes	Corresponds to Can0, Can1, or Can2	Number of seconds from the epoch (1970)				The system microsecond counter when the CAN registers were read.				CAN ID with the Error Flags and Extended Flag, like Socket CAN				Data Length Code	Fractional seconds per tick of the Timestamp			Message Data Bytes padded with x0FF if not used.							

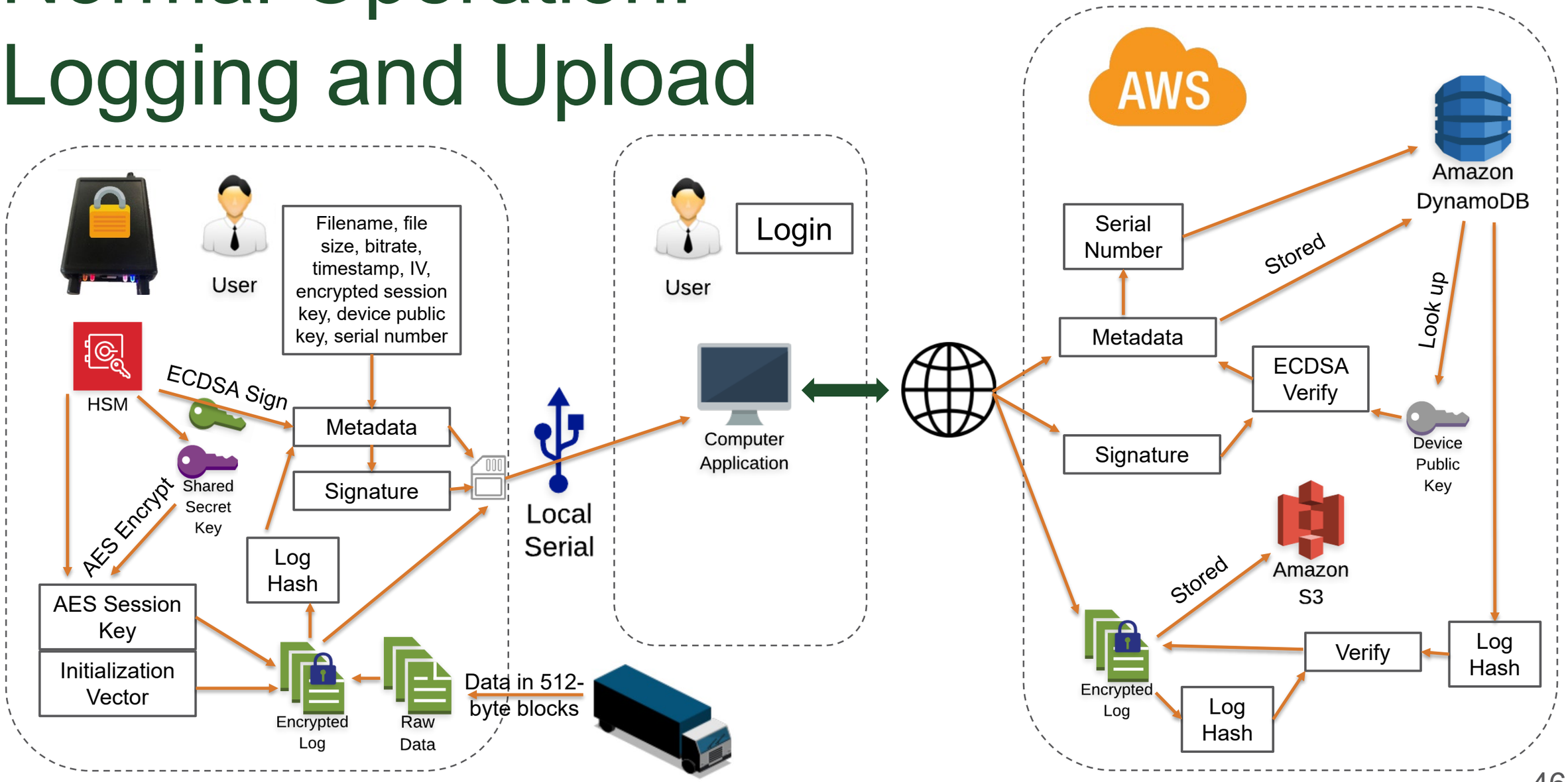
EEPROM Memory Map

0x00	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Data	Bitrate	Bitrate	Bitrate	RES	2	_	_	null	N1	N2	N3	null	T	U	null
Hex					32			0x00	ASCII Encoded						0x00
Notes	Can0 Bitrate	Can1 Bitrate	Can2 Bitrate		Logger Identifier of 2 uppercase letters				File ID. Each digit can be 0-9 or A-Z for a total of 36^3 = 46,656 files.				Brand Name of Logger (i.e. "TU") to start each filename.		

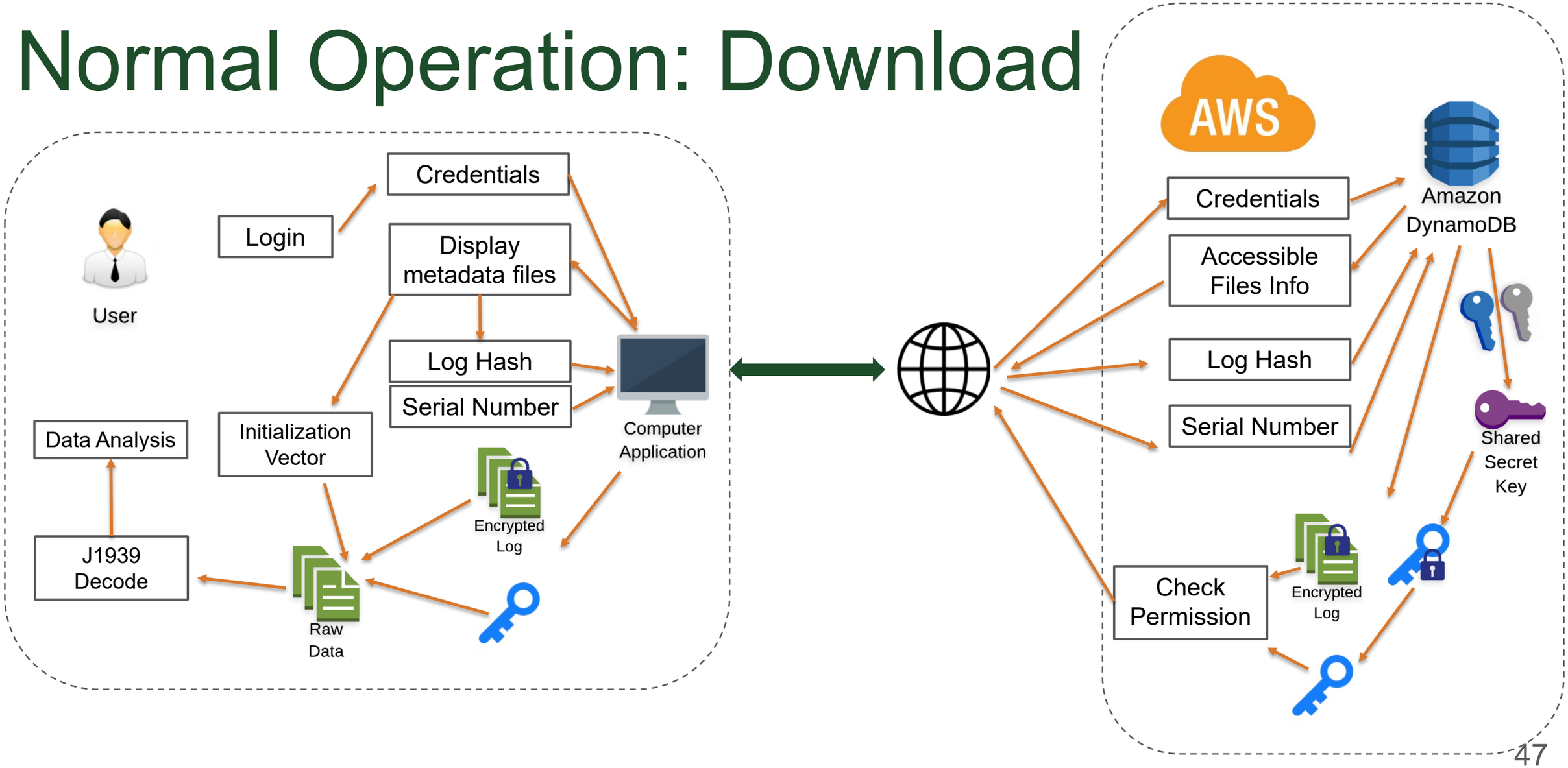
CAN_message_t

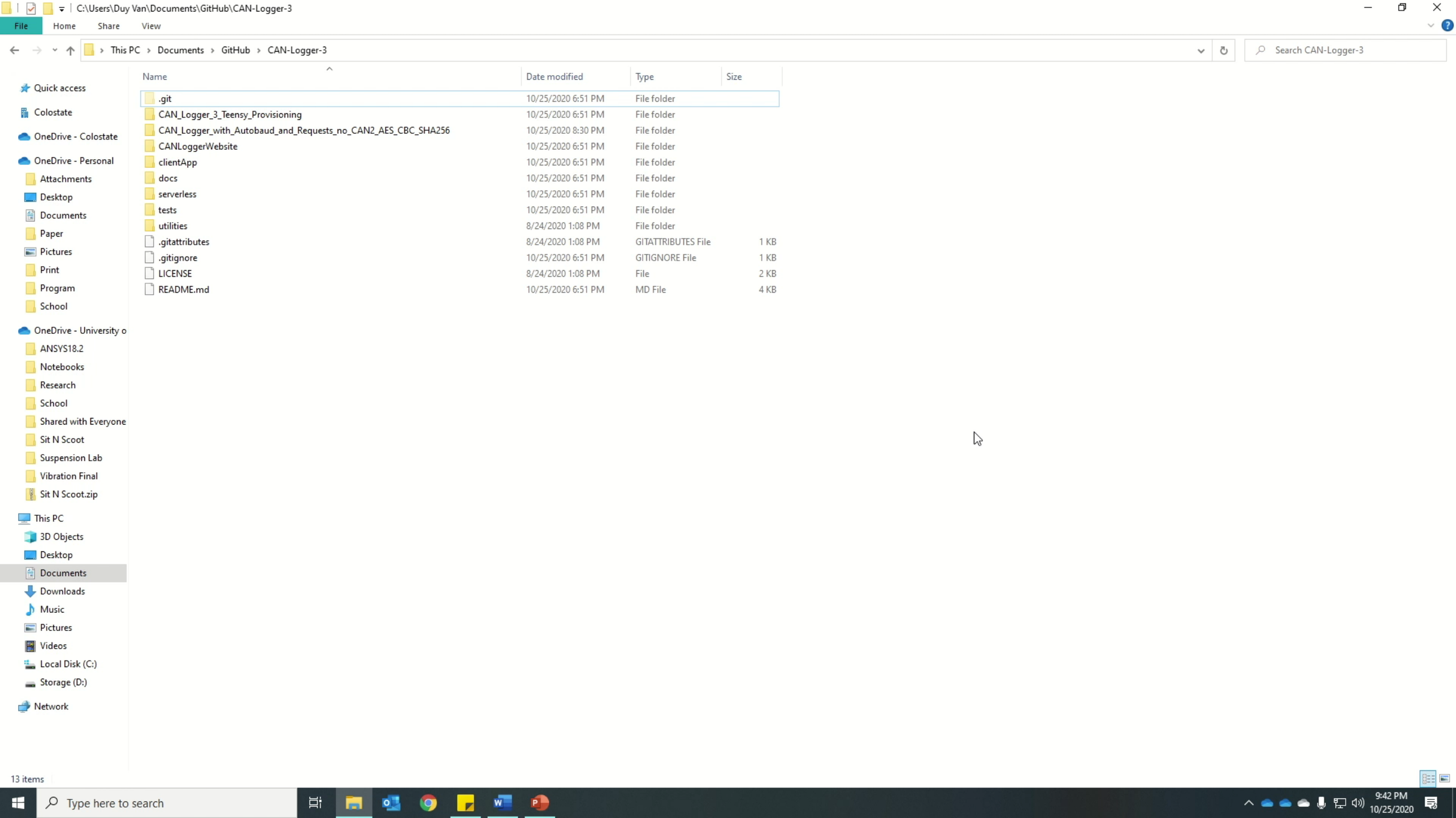
```
uint32_t id;          // can identifier
uint32_t micros;      // system microseconds
uint32_t rxcount;     // number of received messages
uint16_t timestamp;   // FlexCAN time when message arrived
struct {
  uint8_t extended:1; // identifier is extended (29-bit)
  uint8_t remote:  1; // remote transmission request packet type
  uint8_t overrun: 1; // message overrun
  uint8_t reserved:5;
} flags;
uint8_t len;          // length of data
uint8_t buf[8];       // data bytes
```

Normal Operation: Logging and Upload



Normal Operation: Download




C:\Users\Duy Van\Documents\GitHub\CAN-Logger-3
File
Home
Share
View
This PC › Documents › GitHub › CAN-Logger-3
Search CAN-Logger-3
Quick access
Colostate
OneDrive - Colostate
OneDrive - Personal
Attachments
Desktop
Documents
Paper
Pictures
Print
Program
School
OneDrive - University o
ANSYS18.2
Notebooks
Research
School
Shared with Everyone
Sit N Scoot
Suspension Lab
Vibration Final
Sit N Scoot.zip
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Storage (D:)
Network
Name
Date modified
Type
Size
.git 10/25/2020 6:51 PM File folder
CAN_Logger_3_Teensy_Provisioning 10/25/2020 6:51 PM File folder
CAN_Logger_with_Autobaud_and_Requests_no_CAN2_AES_CBC_SHA256 10/25/2020 8:30 PM File folder
CANLoggerWebsite 10/25/2020 6:51 PM File folder
clientApp 10/25/2020 6:51 PM File folder
docs 10/25/2020 6:51 PM File folder
serverless 10/25/2020 6:51 PM File folder
tests 10/25/2020 6:51 PM File folder
utilities 8/24/2020 1:08 PM File folder
.gitattributes 8/24/2020 1:08 PM GITATTRIBUTES File 1 KB
.gitignore 10/25/2020 6:51 PM GITIGNORE File 1 KB
LICENSE 8/24/2020 1:08 PM File 2 KB
README.md 10/25/2020 6:51 PM MD File 4 KB
13 items
Type here to search
9:42 PM
10/25/2020

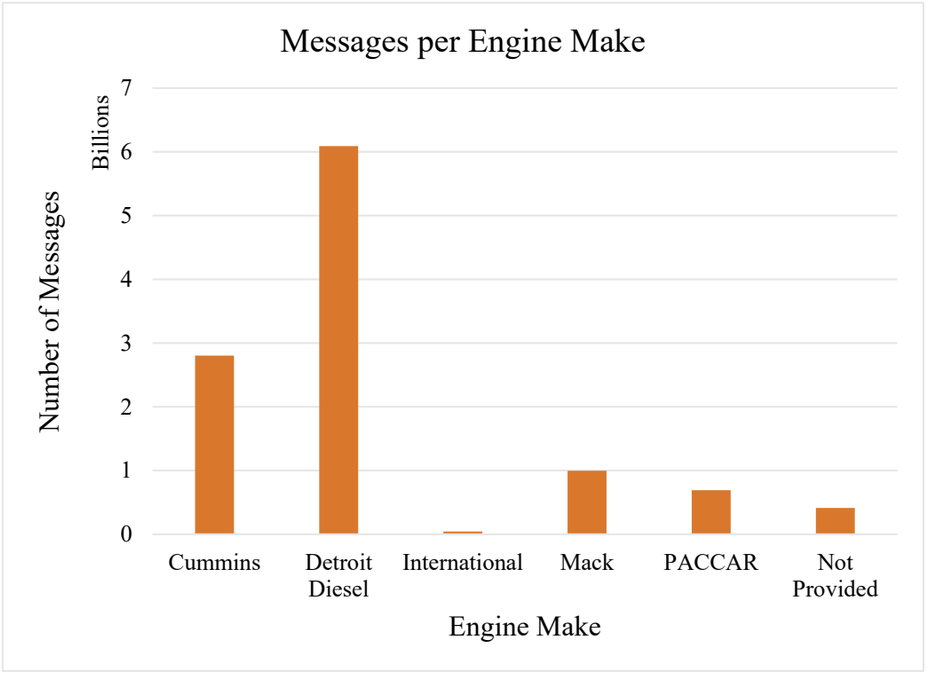
Software Summary

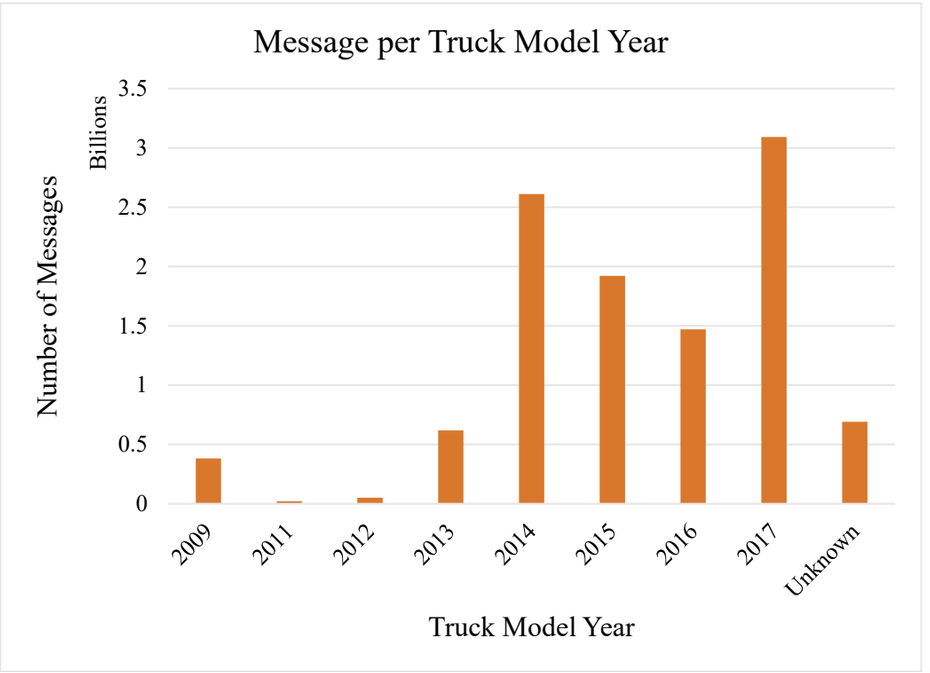
- Provide a secure end-to-end communication model with open source using off-the-shelf products and industry standards
  - ECC implementations
  - Public key exchange
  - Randomly-generated session key
  - Digital signature
  - Client application

Field Testing on Vehicle

Data Collection

- Data collection is one of the main objectives of this project
- NMFTA has been supporting by providing data resources from many volunteering companies
- A batch of 100 NMFTA CAN Loggers and 25 CAN Logger 2 devices have were built and sent to NMFTA for the data collection process since 2017
  - Total number of captured messages: 11,035,396,328
  - Total size of all log files: 667.83 GB
  - Number of different trucks: 21
  - Number of CAN Loggers used: 54
  - Number of Companies involved: 11





Data Collection

- In addition, the research team has been collecting data
- Access to actual platforms at the research facility
  - A 2007 Sterling was donated to the University of Tulsa in 2017
  - A 2014 Kenworth has been obtained by Colorado State University this year

Research team working on the CAN network of the 2007 Sterling truck

Kenworth truck of Systems Engineering department

Data Collection

- Access to actual platforms at the local dealerships

Data collection on a 2019 International truck at the Summit Truck Group dealership in Tulsa

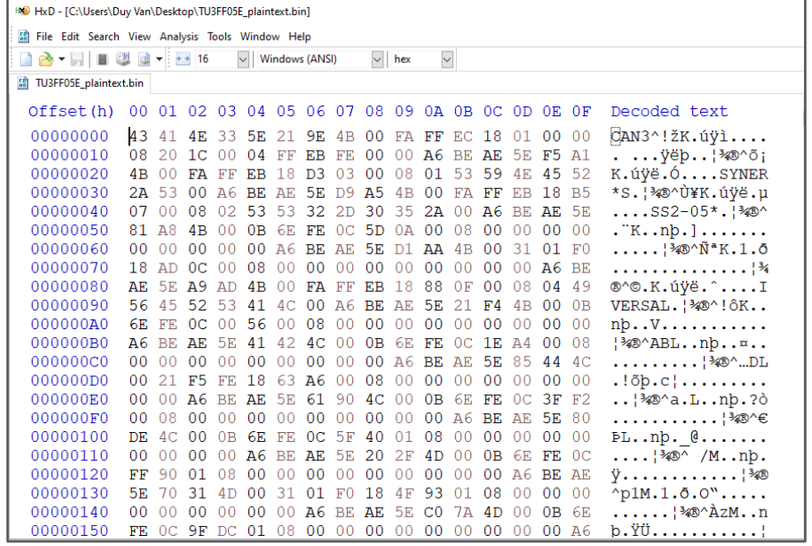
Data Collection

- Helping the engineering senior project team at The University of Tulsa with their pressure monitoring
  - Collect truck data for each experiment run
  - Parse data for vehicle speed and engine speed to support other data collected by the team

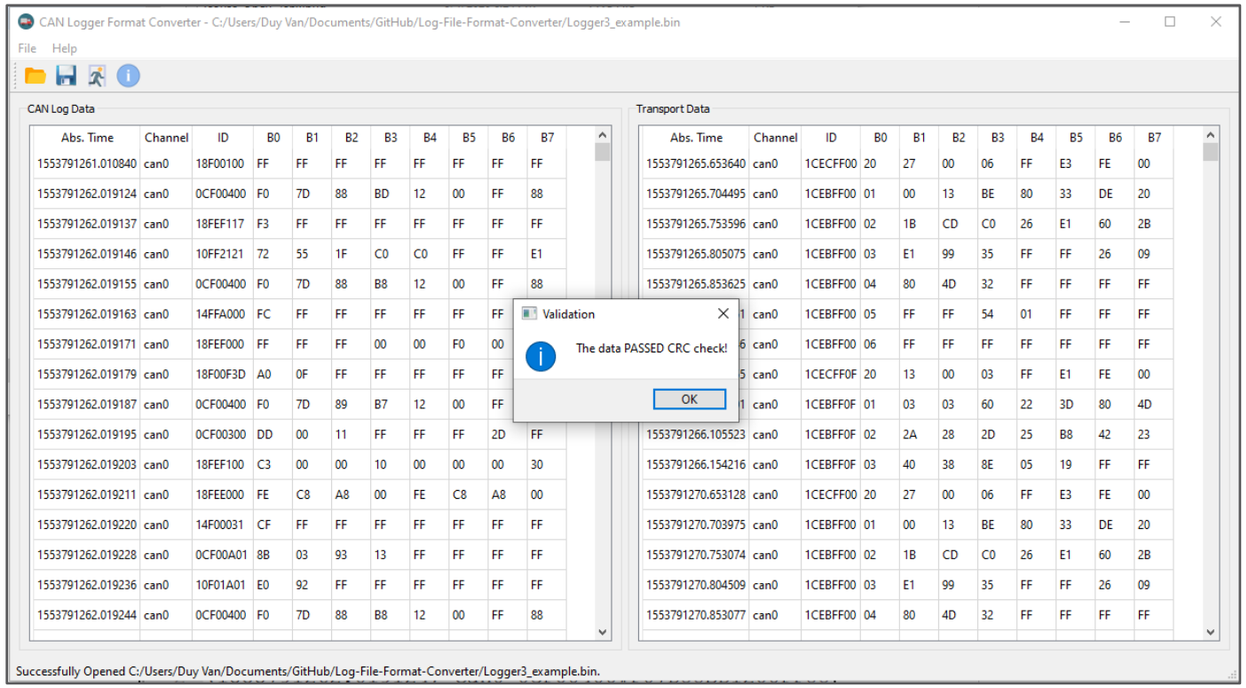
Data collection at ODFL facility

J1939 Decoding

- The raw data needs to be decoded into engineering units using SAE J1939
  - Log File Format Converter GUI was created
    - Convert raw data to J1939 CAN format based on the 512-byte data structure
    - Verify CRC for error
    - Save the log in SocketCAN candump format
    - Save the log in text format



Raw log data



Log file format converter GUI

J1939 Decoding

C:\Users\Duy Van\Desktop\logger3.txt - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

logger3.txt

```
1  (1553791261.010840) can0 18F00100#FFFFFFFFFFFFFFFF
2  (1553791262.019124) can0 0CF00400#F07D88BD1200FF88
3  (1553791262.019137) can0 18FEF117#F3FFFFFFFFFFFFFF
4  (1553791262.019146) can0 10FF2121#72551FC0C0FFFFE1
5  (1553791262.019155) can0 0CF00400#F07D88B81200FF88
6  (1553791262.019163) can0 14FFA000#FCFFFFFFFFFFFFFF
7  (1553791262.019171) can0 18FEF000#FFFFFF0000F000FF
8  (1553791262.019179) can0 18F00F3D#A00FFFFFFFFFFFFF
9  (1553791262.019187) can0 0CF00400#F07D89B71200FF89
10 (1553791262.019195) can0 0CF00300#DD0011FFFFFF2DFF
11 (1553791262.019203) can0 18FEF100#C300001000000030
12 (1553791262.019211) can0 18FEE000#FEC8A800FEC8A800
13 (1553791262.019220) can0 14F00031#CFFFFFFFFFFFFFFF
14 (1553791262.019228) can0 0CF00A01#8B039313FFFFFFFF
15 (1553791262.019236) can0 10F01A01#E092FFFFFFFFFFFF
```

SocketCAN candump format

C:\Users\Duy Van\Desktop\text_format.txt - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

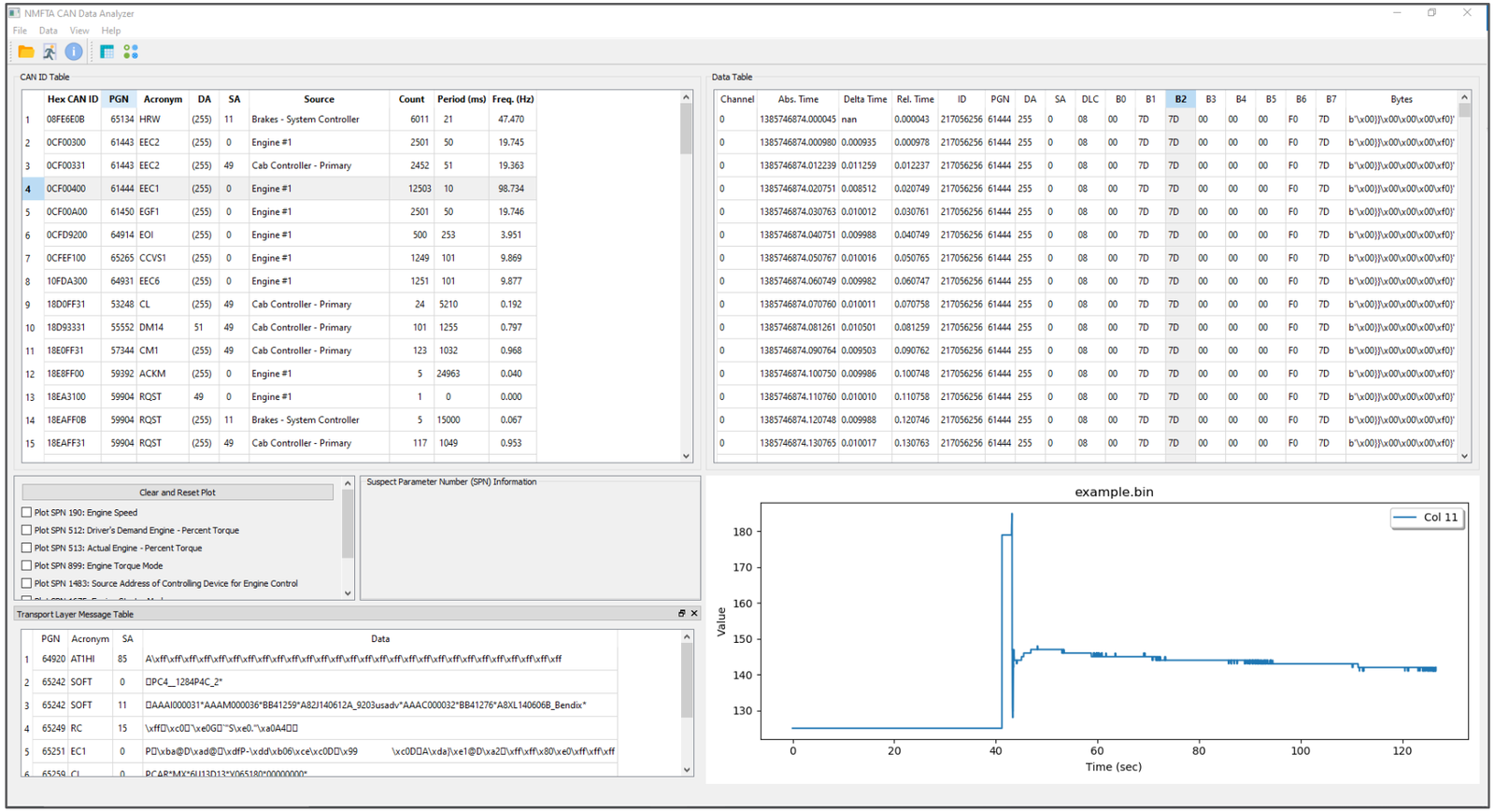
text_format.txt

		Abs. Time	Channel	ID	B0	B1	B2	B3	B4	B5	B6	B7
1	1	1553791261.010840	can0	18F00100	FF	FF	FF	FF	FF	FF	FF	FF
2	2	1553791262.019124	can0	0CF00400	F0	7D	88	BD	12	00	FF	88
3	3	1553791262.019137	can0	18FEF117	F3	FF	FF	FF	FF	FF	FF	FF
4	4	1553791262.019146	can0	10FF2121	72	55	1F	C0	C0	FF	FF	E1
5	5	1553791262.019155	can0	0CF00400	F0	7D	88	B8	12	00	FF	88
6	6	1553791262.019163	can0	14FFA000	FC	FF	FF	FF	FF	FF	FF	FF
7	7	1553791262.019171	can0	18FEF000	FF	FF	FF	00	00	F0	00	FF
8	8	1553791262.019179	can0	18F00F3D	A0	0F	FF	FF	FF	FF	FF	FF
9	9	1553791262.019187	can0	0CF00400	F0	7D	89	B7	12	00	FF	89
10	10	1553791262.019195	can0	0CF00300	DD	00	11	FF	FF	FF	2D	FF
11	11	1553791262.019203	can0	18FEF100	C3	00	00	10	00	00	00	30
12	12	1553791262.019211	can0	18FEE000	FE	C8	A8	00	FE	C8	A8	00
13	13	1553791262.019220	can0	14F00031	CF	FF	FF	FF	FF	FF	FF	FF
14	14	1553791262.019228	can0	0CF00A01	8B	03	93	13	FF	FF	FF	FF
15	15	1553791262.019236	can0	10F01A01	E0	92	FF	FF	FF	FF	FF	FF
16	16	1553791262.019244	can0	0CF00400	F0	7D	88	B8	12	00	FF	88
17	17	1553791262.019252	can0	18FEDF00	83	00	13	FF	7D	FF	FF	FF
18	18	1553791262.019260	can0	14FF3131	00	30	03	00	00	FF	FF	FF
19	19	1553791262.019269	can0	14F00131	FF	FF	FF	FF	00	FF	FF	FF
20	20	1553791262.020572	can0	0CF00400	F0	7D	89	B7	12	00	FF	89

Text format

Data Interpretation

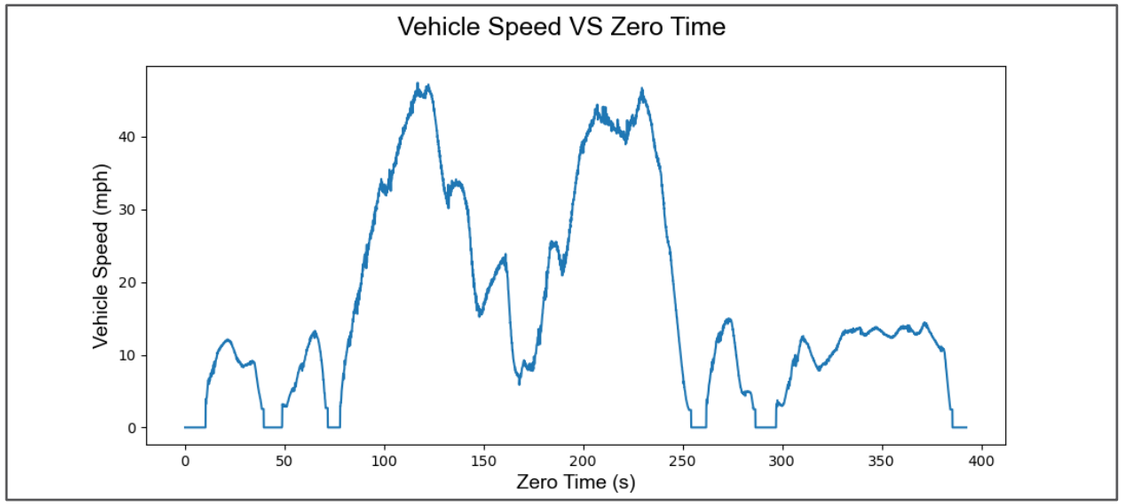
- CAN data analyzer GUI
  - Parameter Group Number (PGN) and its acronym
  - Suspect Parameters Number (SPN)
  - Destination address
  - Source Address
  - Statistical information
  - Transport Layer Protocol
  - Graphing

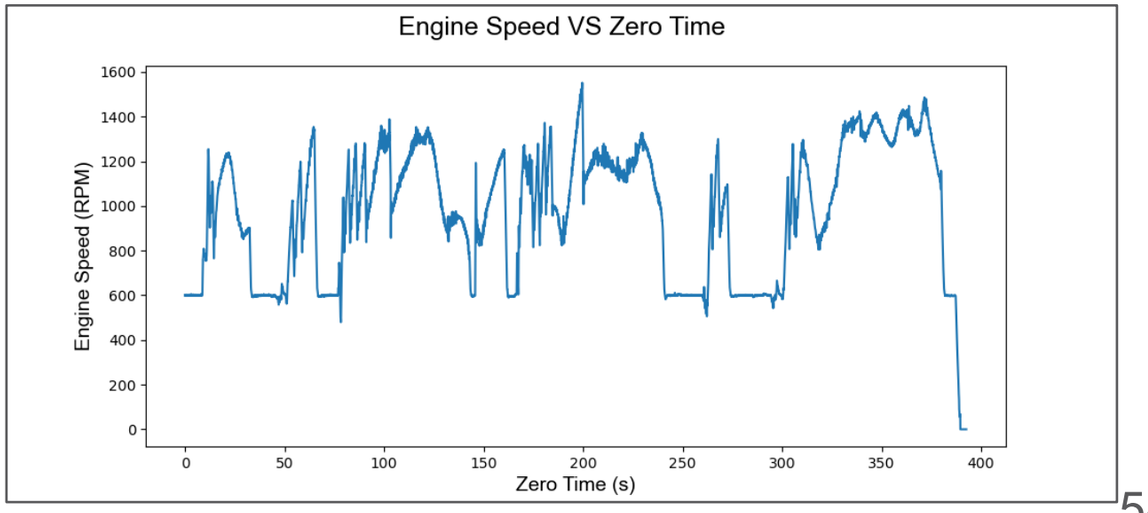


CAN data analyzer GUI

Data Interpretation

- Vehicle speed and engine speed of the log can be retrieved through their PGN and SPN
  - Vehicle wheel speed (PGN 65265, SPN 84)
  - Engine speed (PGN 61444, SPN 190)

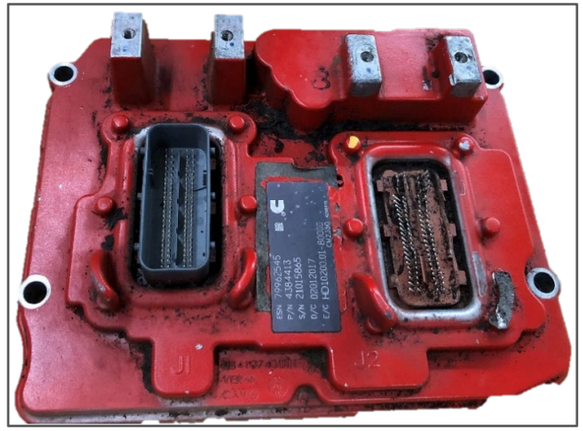




Chip Level Forensics Application

Chip Level Forensics Project Overview

- Heavy vehicles ECM records event data that can be helpful for law enforcement to reconstruct accidents
- Sometimes, ECMs are too damaged to be extracted using standard methods
- Common solution: removing processor and memory chips from the ECM for binary extraction or swapping module
  - Destructive
  - Costly
- New solution:
  - Extracting or cloning ECM firmware from its debugging port
  - Reverse-engineering binary to replicate forensics reports



A Cummins ECM with damage to the vehicle connector

Removing flash memory from an ECM to extract data

CAN Logger Application

- Locating sudden deceleration data in the firmware
  - Log CAN traffic when downloading the reports using OEM tools
  - Concatenate the all the messages
  - Find the similar pattern in the firmware

Cummins PowerSpec

1479051111.967139	can0	18EBFF00	08	04	AF	00	03	03	6C	00	
1479051111.976171	can0	18EBFA00	01	45	1A	A0	00	00	04	26	
1479051111.986092	can0	18EBFA00	02	1E	FA	15	9B	85	8A	12	
1479051111.996095	can0	18EBFA00	03	F6	01	D2	79	DC	00	00	
1479051112.006143	can0	18EBFA00	04	96	00	00	00	00	00	00	
1479051112.016150	can0	18EBFA00	05	00	00	00	00	01	00	00	
1479051112.026145	can0	18EBFA00	06	96	00	00	00	00	00	00	
1479051112.026713	can0	18EBFF00	09	04	01	66	00	04	01	94	
1479051112.036134	can0	18EBFA00	07	00	00	00	00	01	00	00	
1479051112.046148	can0	18EBFA00	08	96	00	00	00	00	00	00	
1479051112.056151	can0	18EBFA00	09	00	00	00	00	01	00	00	
1479051112.066164	can0	18EBFA00	0A	96	00	00	00	00	00	00	
1479051112.076132	can0	18EBFA00	0B	00	00	00	00	01	00	00	
1479051112.086140	can0	18EBFA00	0C	96	00	00	00	00	00	00	
1479051112.086713	can0	18EBFF00	0A	04	03	01	B9	04	04	03	
1479051112.096128	can0	18EBFA00	0D	00	00	00	00	01	00	00	
1479051112.106137	can0	18EBFA00	0E	96	00	00	00	00	00	00	
1479051112.116171	can0	18EBFA00	0F	00	00	00	00	01	00	00	

Sudden deceleration data from captured CAN traffic

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B
000F31BC	00	00	00	00	00	00	00	00	00	00	00	00
000F31C8	00	00	00	00	00	00	00	00	00	00	00	00
000F31D4	00	00	00	00	0C	31	00	00	00	00	00	00
000F31E0	00	00	00	73	00	02	1E	FA	15	9B	85	8A
000F31EC	12	F6	01	D2	79	DC	00	00	96	00	00	00
000F31F8	00	00	00	00	00	00	00	01	00	00	96	00
000F3204	00	00	00	00	00	00	00	00	00	01	00	00
000F3210	96	00	00	00	00	00	00	00	00	00	00	01
000F321C	00	00	96	00	00	00	00	00	00	00	00	00
000F3228	00	01	00	00	96	00	00	00	00	00	00	00
000F3234	00	00	00	01	00	00	96	00	00	00	00	00
000F3240	00	00	00	00	00	01	00	00	96	00	00	00
000F324C	00	00	00	00	00	00	00	01	00	00	96	00

Sudden deceleration data from firmware

CAN Logger Application

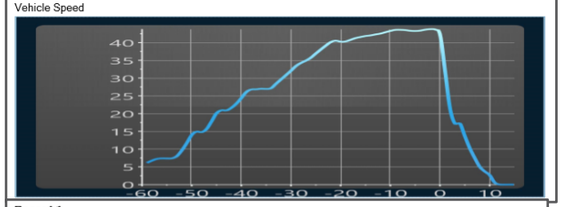
- Reverse-engineering sudden deceleration data
  - Trials and errors with J1939-71 standard
  - Middleperson attack to identify the unknown bytes

Sudden Vehicle Speed Deceleration Report Record 1

Engine Type	ISX 2010	Ecm Code	CL10132.39
Engine Serial Number	60811136	Software Phase	7.70.0.71
Unit Number	0000000000	Extraction Date	04-03-2019 04:37:51
Sudden Decel Threshold Rate:	N/A	ECM Run time	8227:56:32

Occurrence Date: N\A — ECM Run Time at Occurrence: 2899:18:36
Air Temperature (°F) at Occurrence: 72 — Occurrence Distance (mi): 109323.4

Vehicle Speed



Record 1

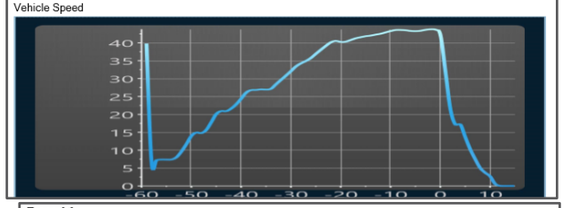
Time (Seconds)	Vehicle Speed (mph)	Engine Speed (rpm)	Engine Load (%)	Throttle (%)	Brake Status	Clutch Status	Cruise Status	Lamp Status
-59	6	683	34.8	30.1	-	-	-	-
-58	7	788	33.6	32.2	-	-	-	-
-57	7	813	21.9	27.5	-	-	-	-

Sudden Vehicle Speed Deceleration Report Record 1

Engine Type	ISX 2010	Ecm Code	CL10132.39
Engine Serial Number	60811136	Software Phase	7.70.0.71
Unit Number	0000000000	Extraction Date	04-03-2019 02:08:09
Sudden Decel Threshold Rate:	N/A	ECM Run time	8225:27:34

Occurrence Date: N\A — ECM Run Time at Occurrence: 2899:18:36
Air Temperature (°F) at Occurrence: 72 — Occurrence Distance (mi): 109323.4

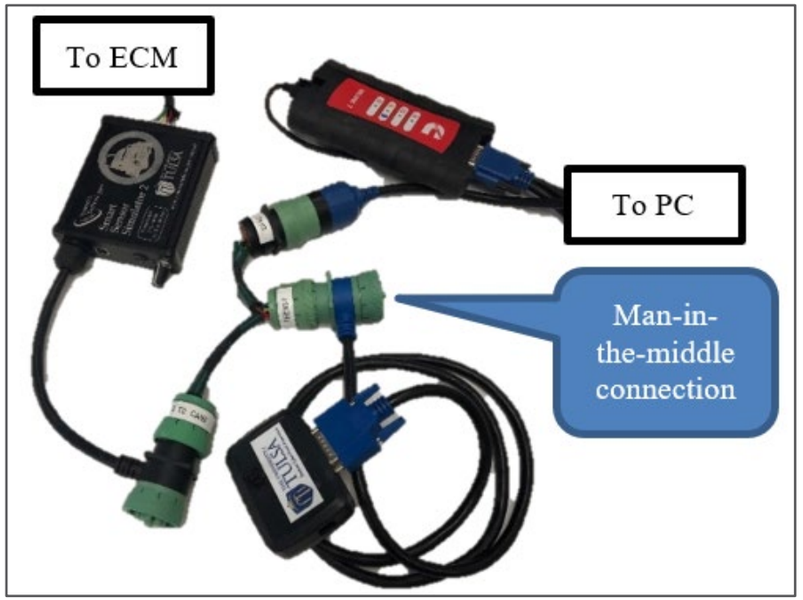
Vehicle Speed



Record 1

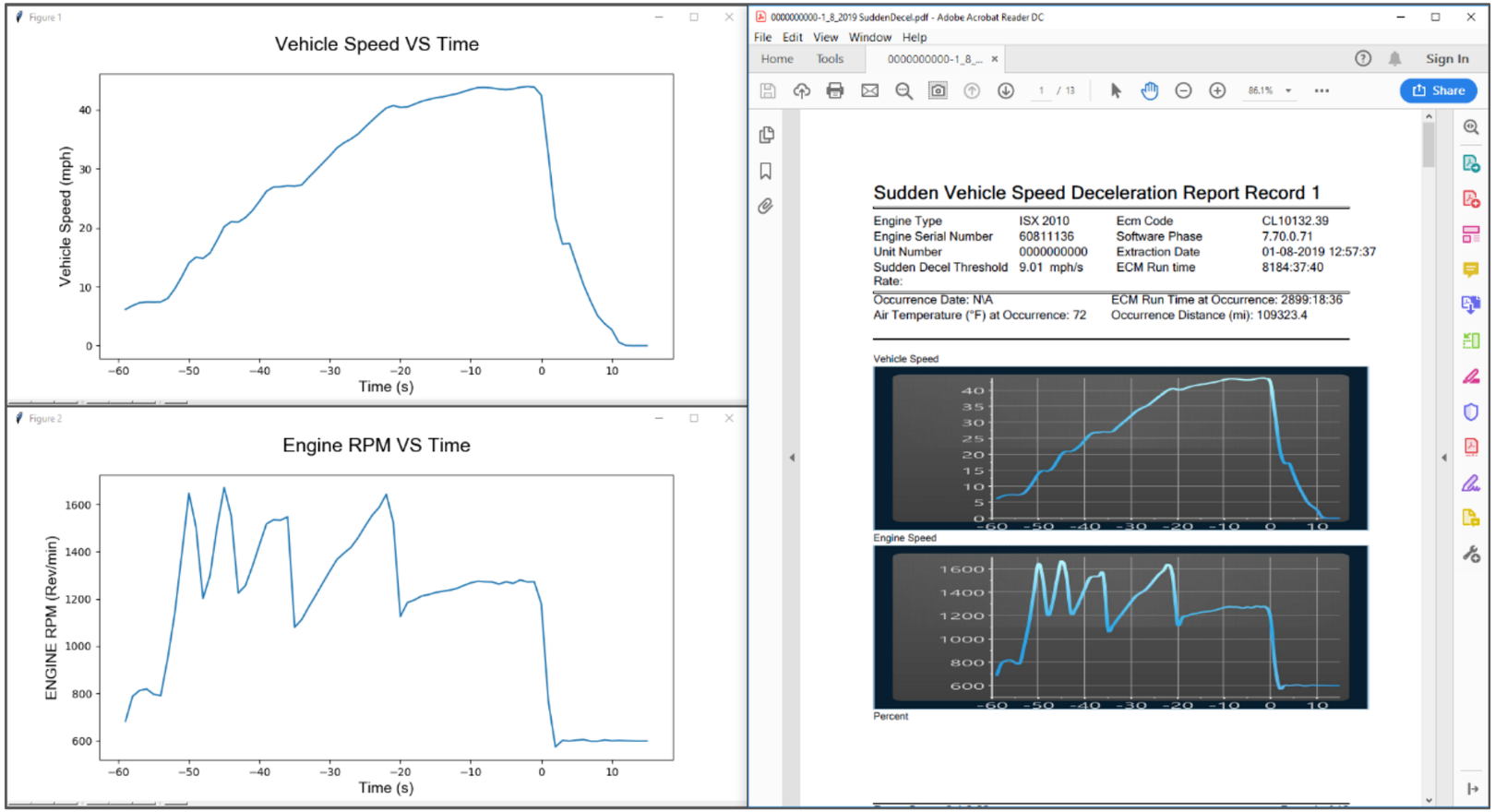
Time (Seconds)	Vehicle Speed (mph)	Engine Speed (rpm)	Engine Load (%)	Throttle (%)	Brake Status	Clutch Status	Cruise Status	Lamp Status
-59	40	683	34.8	30.1	-	-	-	-
-58	7	788	33.6	32.2	-	-	-	-
-57	7	813	21.9	27.5	-	-	-	-

Sudden deceleration report from genuine data (left) and attack data (right)



Man-in-the-middle attack setup

Results



Comparison of the custom decoding of the vehicle speed and engine speed records compared to the Cummins PowerSpec report

Conclusions

Contribution

- A secure open-source CAN logging system:
  - An affordable CAN Logger device with cryptography and secure key storage implementations
  - A cloud server for large data storage and management with access control
  - A user-friendly client application GUI for data transferring between the device and the cloud
- The project should be useful for inspiring future designs in automobile systems
- A current data pool of more than 11 billion CAN messages

Future Work

- Test and validate SWCAN and LIN
- Fully implement CAN2 and J1708 with auto-detection
- WiFi implementation to wirelessly transfer log to the local computer application
- A method to revoke server public key stored in the HSM
- RSA encryption alternative over ECDH shared secret

Acknowledgement

Special thanks to the sponsor

Disclaimer: "This material is based upon work supported by the National Science Foundation under Grant No. 1715409. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation."

and the supporters

Colorado State University

NMFTA
National Motor Freight Traffic Association, Inc.

THE UNIVERSITY of TULSA

Thank you

Colorado State University