Two Wires, and Two Wheels, Bikes Can Do CAN Too

1. WHOAMI
2. The Bike and Brand
3. Why? (Project Inspiration and Purpose)
4. The Hurdles
5. The Project begins(What I used and what I built)
    a. Hardware
    b. Engine Simulation
    c. CSV Parsing and Data Analyzing
6. What’s Next(Project Future)

WHOAMI

Derrick

Director of IT

Twitter: @canbusdutch

Email: CanBusDutch@gmail.com

Github: https://github.com/CircuitWorks1

My Bike

1190RX

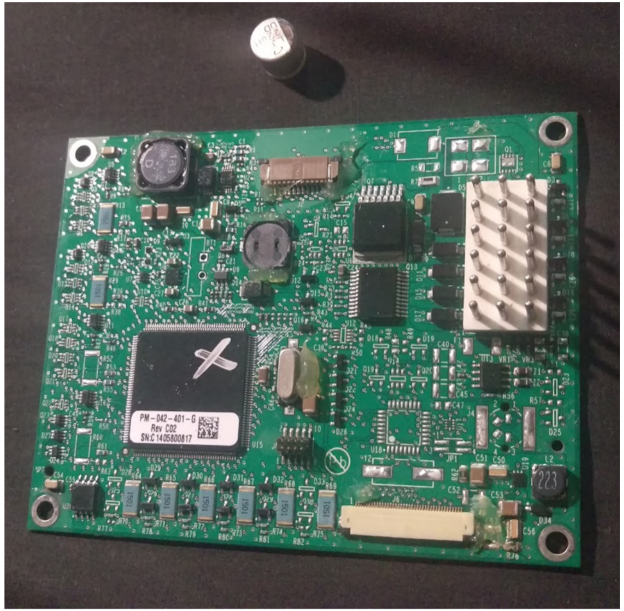
What is Buell and EBR(Erik Buell Racing)

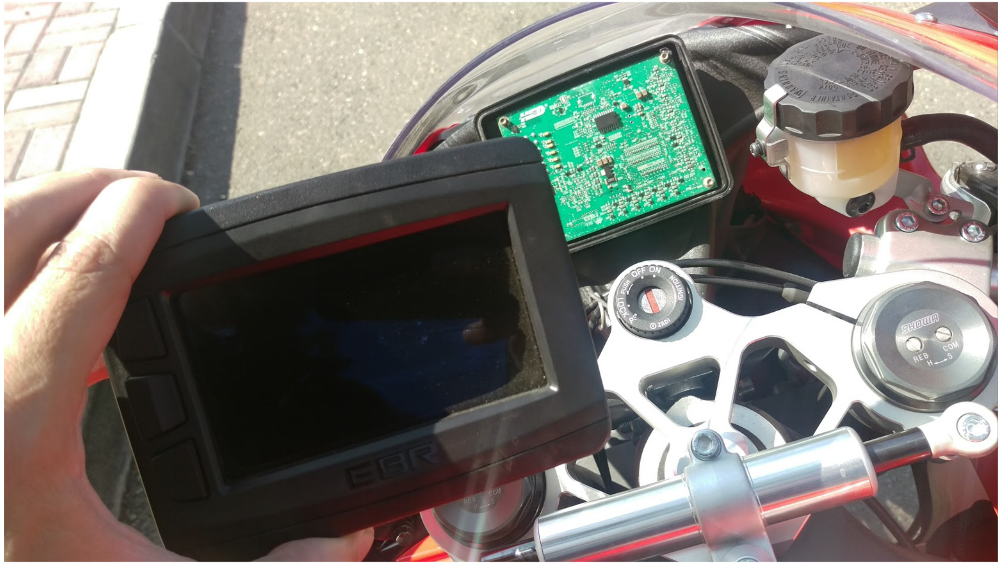
- The only production sportbike made and designed in the USA
- The only American made motorcycle, to ever score points in World Superbike
- Won the 2009 Daytona Superbike Championship without a single DNF(did not finish)
- They manufactured nearly 140,000 motorcycles in 15 years under Harley Davidson
- They sold 65 bikes in 18 months, garnering 3 million in revenue as a startup
- It’s a company now, teetering on the edge of existence


MOTORCYCLES

2010 Buell Blast

Why? Project Inspiration and Purpose





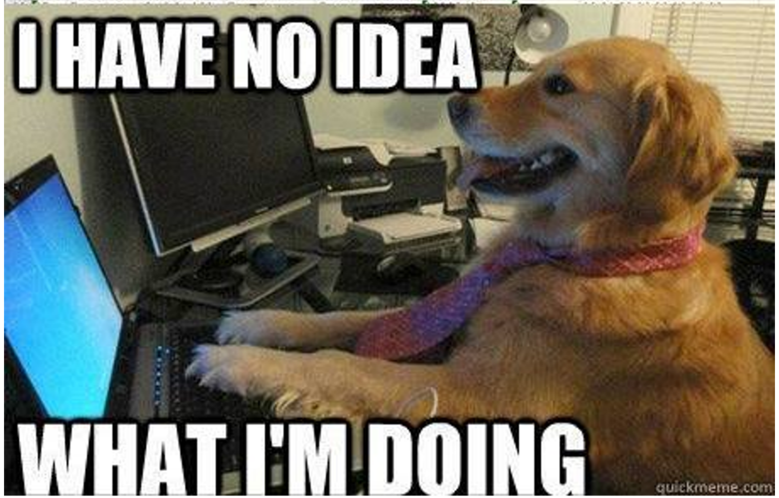

HISTORIC FAULTS
SEL
1. U0001
ODO: 0.0
COMMUNICATIONS ERROR
CLEAR FAULTS
EXIT
EBR


0
KPH
TRIP A
60.3
COMMUNICATION
ERROR


0
MPH
COMMUNICATION
ERROR
ODO


1
2
3
4
5
6
7
8
9
10
11
12
13
0
TC
3
0
MPH
02:47
COMMUNICATION
ERROR
ODO
2314.9

	A	B	C	D	E	F	G	H
1	No	Direction	Time_Scale	Frame_Type	Frame_Format	Frame_ID	Data_Len	Data
2	0	Receive	16:42:21:586	Data frame	Standard frame	00000201	8	00 00 00 00 00 00 00 00
3	1	Receive	16:42:21:595	Data frame	Standard frame	00000251	8	3e 00 00 00 00 00 00 00
4	2	Receive	16:42:21:598	Data frame	Standard frame	00000430	8	00 00 00 00 06 06 ff 0f
5	3	Receive	16:42:21:600	Data frame	Standard frame	00000440	8	12 2a 12 2a 00 00 00 00
6	4	Receive	16:42:21:603	Data frame	Standard frame	00000450	8	00 00 00 00 00 00 ff 0f
7	5	Receive	16:42:21:606	Data frame	Standard frame	00000201	8	00 00 00 00 00 00 00 00
8	6	Receive	16:42:21:608	Data frame	Standard frame	00000422	8	00 00 00 00 00 00 00 00
9	7	Receive	16:42:21:611	Data frame	Standard frame	00000423	8	00 00 00 00 00 00 00 00
10	8	Receive	16:42:21:615	Data frame	Standard frame	00000400	8	04 14 b2 c8 f0 64 64 04
11	9	Receive	16:42:21:625	Data frame	Standard frame	00000201	8	00 00 00 00 00 00 00 00
12	10	Receive	16:42:21:635	Data frame	Standard frame	00000410	8	1e 9c 19 19 ca da ff 39
13	11	Receive	16:42:21:637	Data frame	Standard frame	00000430	8	00 00 00 00 06 06 ff 0f
14	12	Receive	16:42:21:639	Data frame	Standard frame	00000440	8	6d 5b 6d 5b 00 00 00 00
15	13	Receive	16:42:21:641	Data frame	Standard frame	00000450	8	9a 06 9a 04 ff 0f ff 0f
16	14	Receive	16:42:21:643	Data frame	Standard frame	00000460	8	00 00 14 00 a6 01 26 01
17	15	Receive	16:42:21:646	Data frame	Standard frame	00000461	8	00 06 ff 3f 06 06 ed 00
18	16	Receive	16:42:21:649	Data frame	Standard frame	00000462	8	00 00 00 00 3a 00 00 00
19	17	Receive	16:42:21:656	Data frame	Standard frame	00000463	8	e8 03 e8 03 00 00 00 00
20	18	Receive	16:42:21:659	Data frame	Standard frame	00000464	8	49 ff 39 ff 9c 69 19 19
21	19	Receive	16:42:21:663	Data frame	Standard frame	00000465	8	da ca 1e ff ca 4e 4b ff
22	20	Receive	16:42:21:666	Data frame	Standard frame	00000466	8	00 00 00 00 00 00 00 00
23	21	Receive	16:42:21:670	Data frame	Standard frame	00000467	8	00 00 00 00 00 00 00 00
24	22	Receive	16:42:21:673	Data frame	Standard frame	00000468	8	00 00 00 00 00 00 00 00
25	23	Receive	16:42:21:677	Data frame	Standard frame	00000469	8	00 00 00 00 00 00 00 00
26	24	Receive	16:42:21:680	Data frame	Standard frame	00000201	8	00 00 00 00 00 00 00 00
27	25	Receive	16:42:21:682	Data frame	Standard frame	00000303	8	00 03 00 00 00 00 00 00
28	26	Receive	16:42:21:685	Data frame	Standard frame	00000420	8	08 00 20 00 00 00 00 00



NO-BD2

There is no diagnostics standard in the motorcycle industry.

There are “universal” diagnostics tools, although the byte offset and scaling factor for every manufacturer is different.

This is a struggle when working with motorcycles designed by smaller manufacturers.

MS458 Kawasaki 4-pin Cable
(Most Popular)

MS459 Kawasaki 8-pin Cable
(ABS Only)

MS460 Honda 4-pin Cable
(Most Popular)

MS461 Honda - Mondial 3-pin Cable
(Limited Models - Check Vehicle List)

MS463 Suzuki - Arctic Cat 6-pin
(Most Popular)

MS464 Suzuki - Kawasaki 4-pin Cable
(Limited Models - Check Vehicle List))

MS475 Yamaha - Malaguti - MBK
3-pin Cable (Limited Models - Check
Vehicle List)

MS476 Cagiva 10-pin Cable

MS477 Suzuki - Cagiva
(Injection Regulation Cable)

MS480 Harley-Davidson 4-Pin Cable

MS481 MZ - Piaggio - Triumph - Victory
OBDII Cable

MS483 Aprilia/Ditech Cable
(Limited Models - Check Vehicle List)

MS489 KTM - Husaberg - Husqvarna Cable

MS490 Aprilia/Sagem Cable
(Limited Models - Check Vehicle List)

MS491 Peugeot Cable

MS493 Kymco Cable

MS499 Packard Cable
(Adiva, Aprilia, Bimota, Cagiva, Derbi, Ducati, Garelli, Gas Gas, Gilera, KVN, Laverda, Malaguti, Moto Guzzi, Moto Morini, MV Agusta, Piaggio, Sherco, Vespa, Voxan,)

MS500 Kawasaki 4-pin Cable
(Model Year 2007)

MS501 BRP/CAN-AM Cable

MS502 Kawasaki
(Injection Regulation Cable)

MS505 Benelli 6-pin Cable

MS506 Buell/Harley Davidson Cable (Some H-D Ign - Check Vehicle List)

MS508 Ducati CAN 4-pin Cable
(Limited Models - Check Vehicle List)

MS509 Kawasaki 6-pin Cable
(Model Year - 2009)

MS510 Kawasaki 6-pin Cable
(Model Year - 2010)

MS512 SYM - Advia - Dafra 3-Pin Cable
(Check Vehicle List)

MS516 Polaris 8-pin Cable

MS518 Aprilia - Bimota Cable

MS525 BMW Cable

MS526 MV Agusta CAN 4-Pin Cable
WARNING: do not use on HONDA systems.

MS528 Honda Cable (Immobilizer only)

MS538 Kymco CAN 4-Pin Cable

MS539 Daelim CAN 4-Pin Cable

MS541 Harley Davidson CAN 6-Pin Cable

MS551 Husqvarna 6-Pin Cable

MS557 KTM injection regulation cable

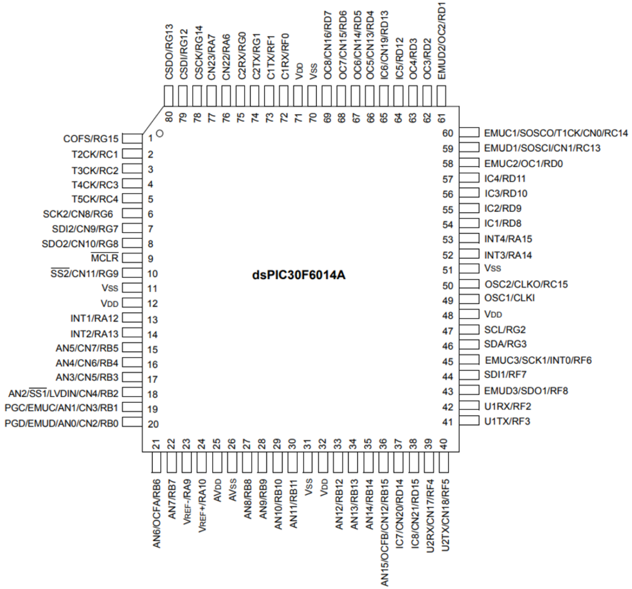
MS562 Benelli Kee Way 6-Pin Cable

History of CAN Systems and Motorcycles

Most European and Japanese manufacturers began implementing a CAN system on their bikes around 2003.(Ducati, BMW, Honda, Kawasaki)

American manufacturers were a little late to the CAN party. Buell first implemented a CAN system on the 2008 release of their 1125. Harley’s first CAN based bike was in 2011, and it wasn’t until 2014 that all Harleys were equipped with a CAN system.

My ECM and CAN Protocol



Microcontroller: Microchip Technologies dspic30f6014A-3oi/pf

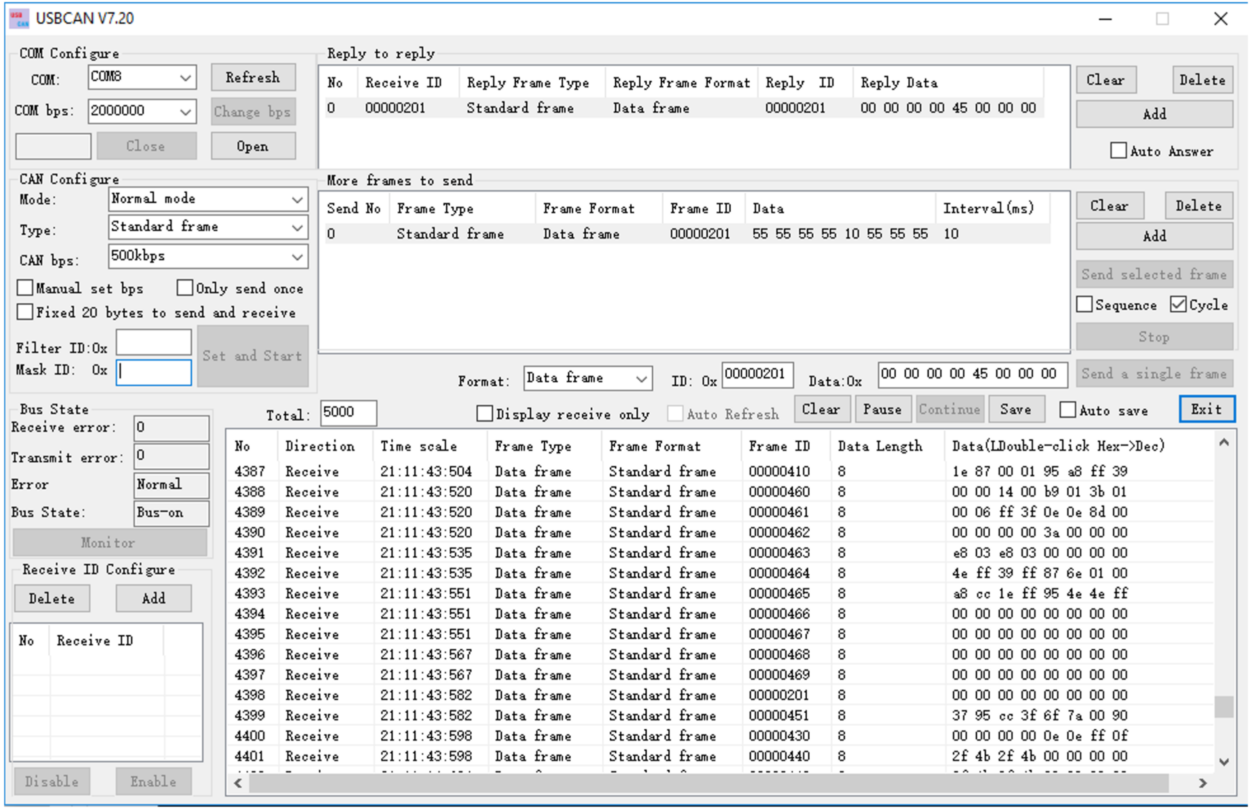
Diagnostics Data Protocol: J1850 VPW

CAN Protocol: 500Kb/s, 11 bit IDs, 120ohm termination, LSB first bit order, 0V nominal

The Hardware

Software

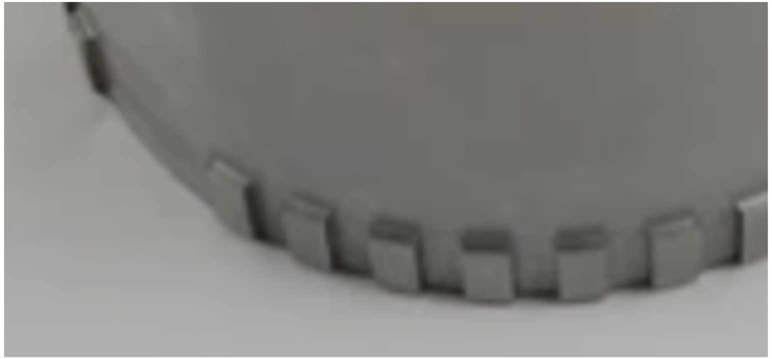
https://github.com/SeeedDocument/USB-CAN-Analyzer



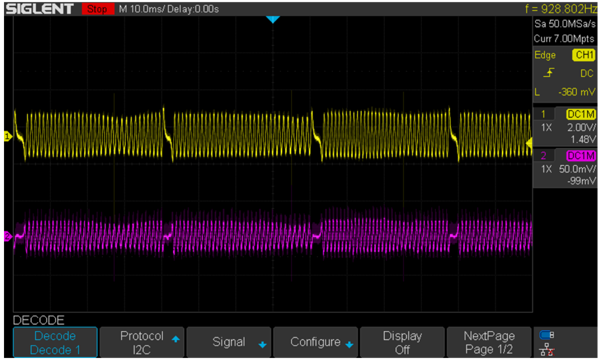
Engine Simulation

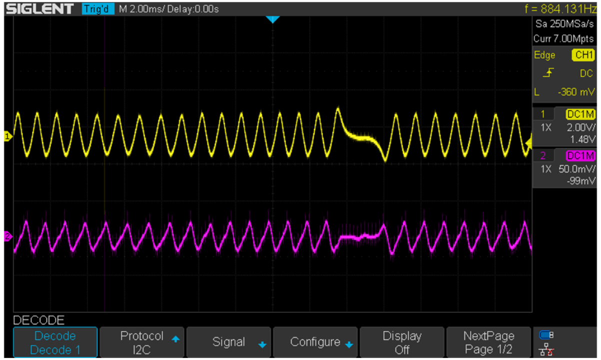
The most important piece when simulating the engine sensor data of the bike is the crankshaft sensor output. The crankshaft sensor produces 2 waves 180 degrees out of phase with each other. The wave is generated using a hall sensor and a stepped rotor/tone ring, surrounding the stator magneto. There are 34 steps and a timing gap the equivalent of 2 wavelengths/steps.

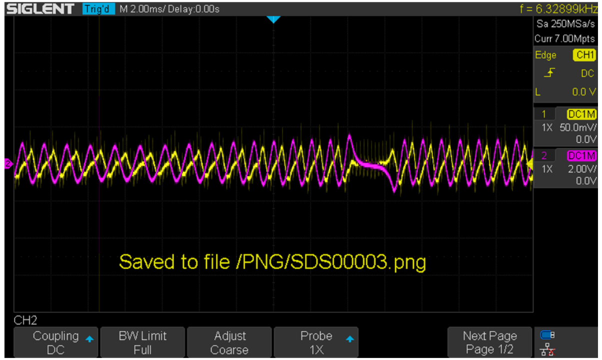
Stepped Rotor Design

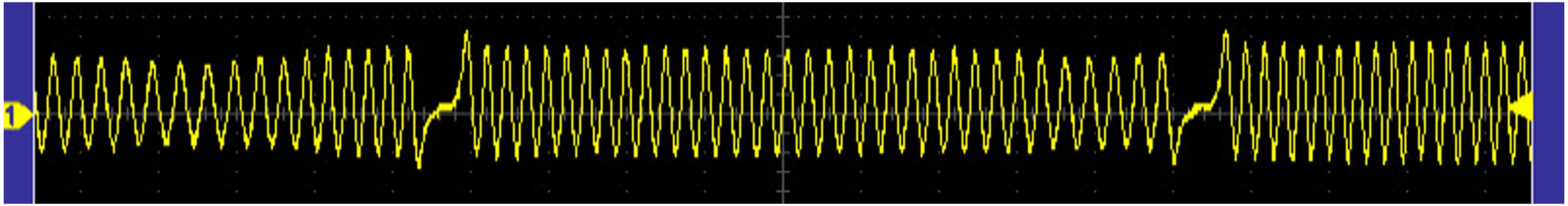


Let’s Take a Look At It









How Do We Simulate It?

There is 36 steps(34 steps then the timing gap which is 2 steps). 1 full step every 10 degrees(360 degrees in a circle). Let’s pick an RPM to simulate. 6000RPM, since RPM is a measurement by minutes, and Hz(wave frequency) is a measurement by seconds, we would have a math equation that looks like this.

$$(RPM/60seconds)36steps=frequency$$

$$(6000/60)36=3.6khz$$

The Code - RPM

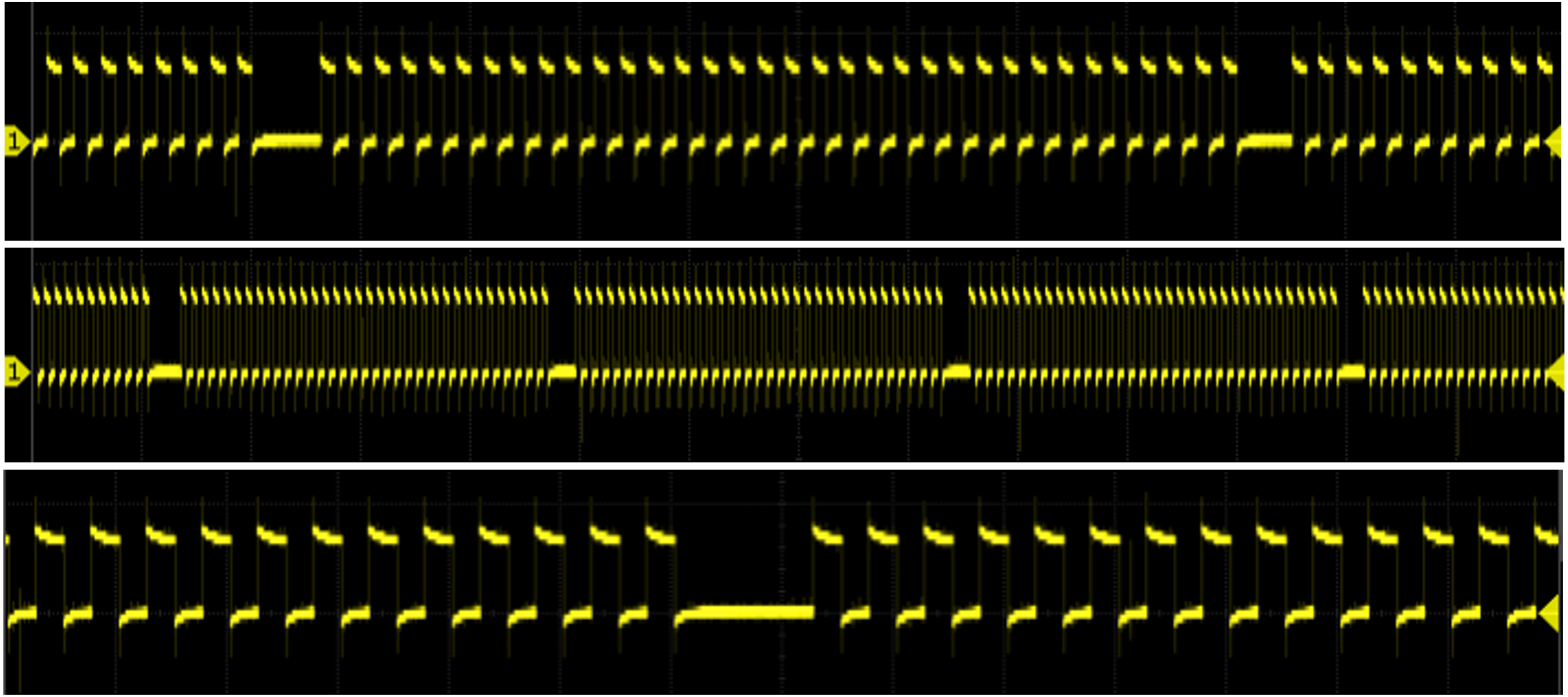
We take the frequency, input by the user, begin a PWM tone at the given frequency, and calculated the time to execute both 34 wavecycles, and 36 wave cycles in microseconds.

example:
rpmTime = 1 / rpmFreq * 1,000,000 * 34
and
rpmRestart = 1 / rpmFreq * 1,000,000 * 36

After we calculate and set our variables, our code logic would look something like the following

```
If waveStartTime > rpmTime
        wave(stop)
If waveStartTime > rpmRestart
        wave(start)
        waveStartTime = 0
```

The Waveform - RPM



The Code - MPH

The speed sensor simulation is a bit more basic because there is no timing gap. To simulate the speed sensor waveform we will take the given frequency, that was input by the user and rapidly turn on and off one of the pins. This will generate a square wave at the given frequency.

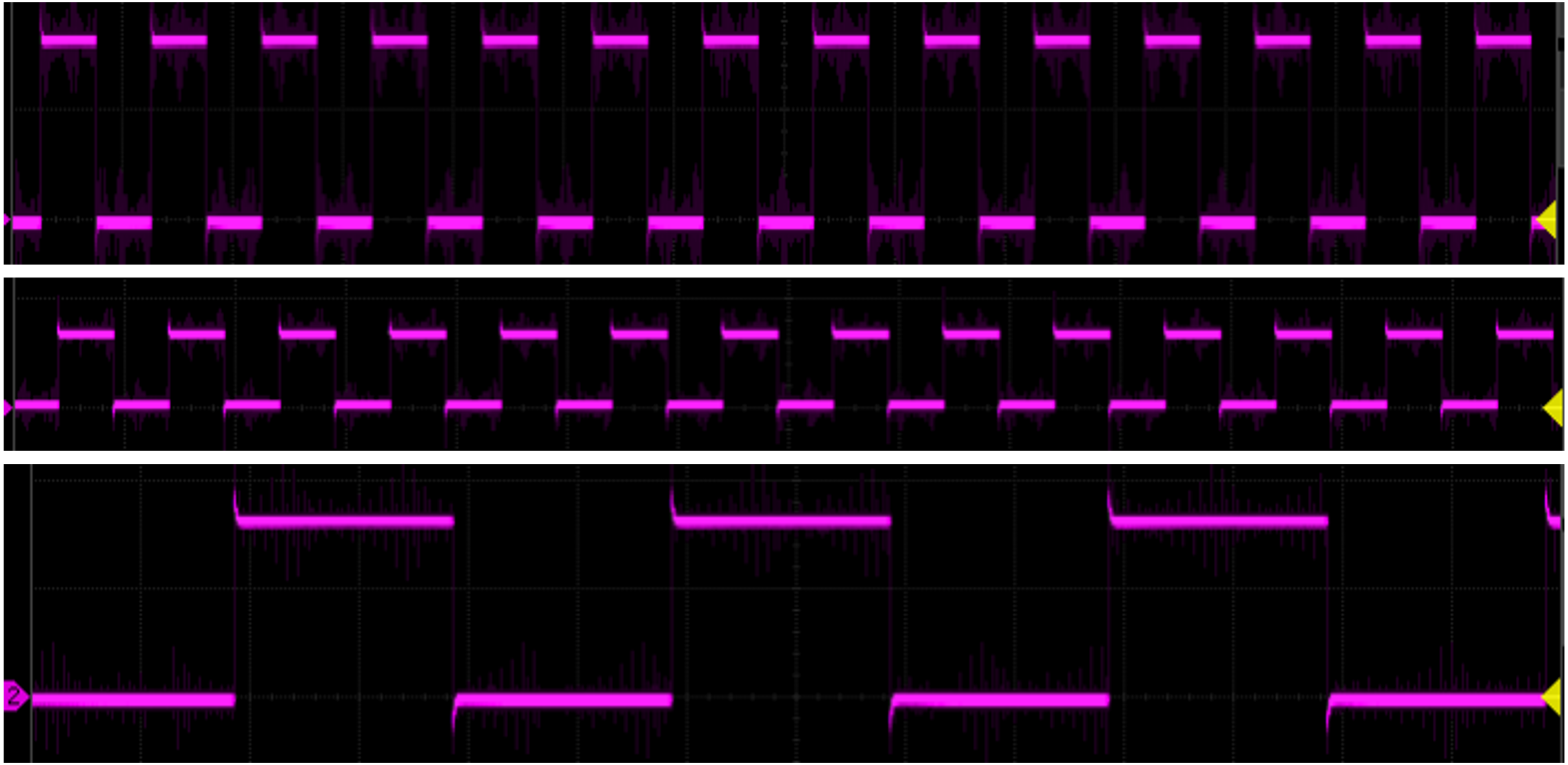
First we find out the period to complete 1 wave cycle in microseconds

speedTime = 1 / speedFreq * 1,000,000

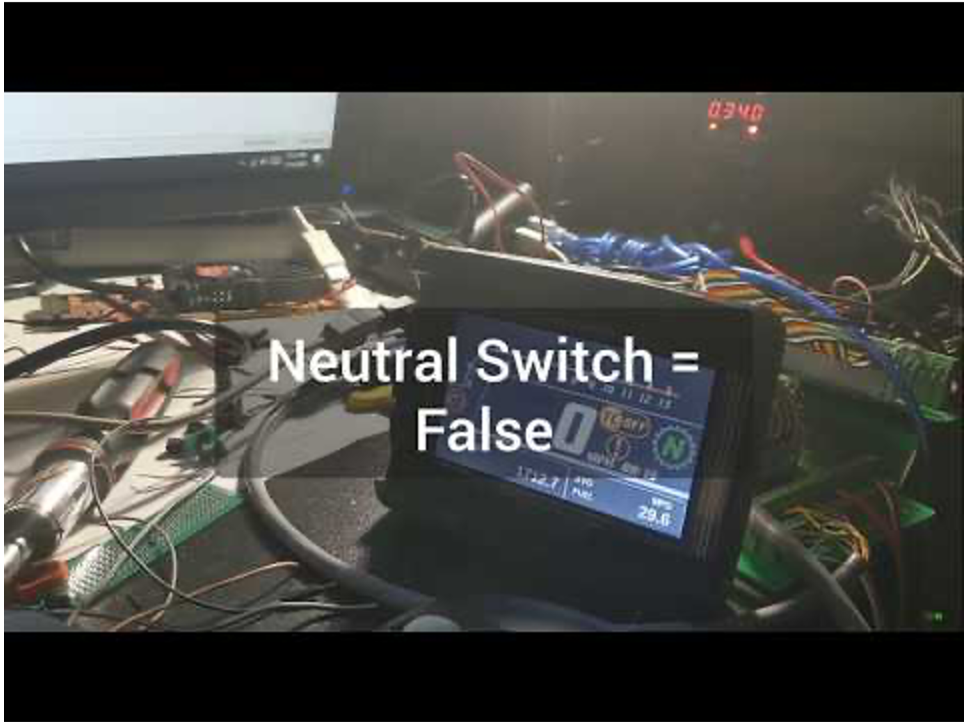
We can then divide that by 2, to find out how long the pin should remain in each state. We control this action with another non-blocking delay.

If delay timer < speedTime/2
pin off
If delay timer > speedTime/2
pin on
If delay timer > speedTime
restart delay timer

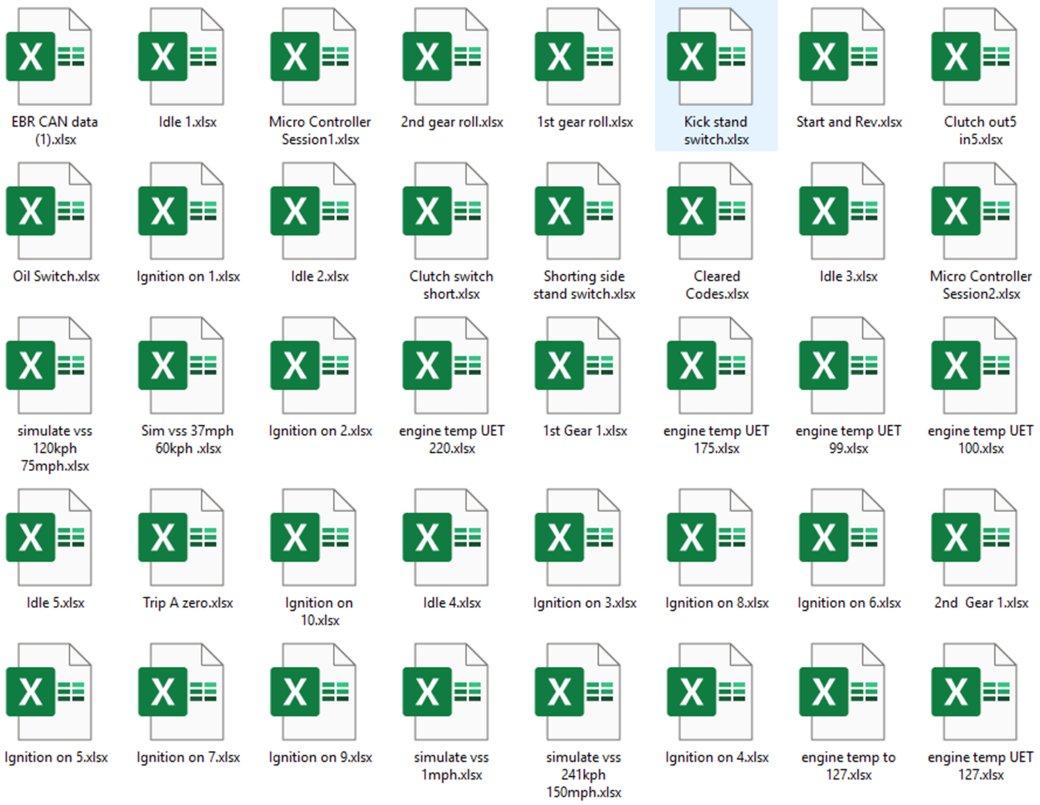
The Waveform - MPH



Let’s Give it a try



All This Data and Only One set of Eyes



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
24	Receive	14:27:28:270	Data fram	Standard	201	8	00	00	00	00	00	00	00	00
25	Receive	14:27:28:282	Data fram	Standard	303	8	00	03	00	00	00	00	00	00
26	Receive	14:27:28:293	Data fram	Standard	420	8	08	00	20	00	00	00	00	00
27	Receive	14:27:28:308	Data fram	Standard	421	8	80	01	00	00	00	00	00	00
28	Receive	14:27:28:319	Data fram	Standard	201	8	00	00	00	00	00	00	00	00
29	Receive	14:27:28:330	Data fram	Standard	430	8	00	00	00	00	06	06	ff	0f
30	Receive	14:27:28:342	Data fram	Standard	440	8	4b	56	4b	56	00	00	00	00
31	Receive	14:27:28:353	Data fram	Standard	450	8	e5	03	ed	02	ff	0f	ff	0f
32	Receive	14:27:28:365	Data fram	Standard	201	8	00	00	00	00	00	00	00	00
33	Receive	14:27:28:377	Data fram	Standard	251	8	3c	00	00	00	00	00	00	00
34	Receive	14:27:28:388	Data fram	Standard	460	8	00	00	14	00	fa	00	bb	00
35	Receive	14:27:28:399	Data fram	Standard	461	8	00	06	ff	3f	06	06	e5	00
36	Receive	14:27:28:411	Data fram	Standard	462	8	00	00	00	00	3a	00	00	00
37	Receive	14:27:28:422	Data fram	Standard	463	8	e8	03	e8	03	00	00	00	00
38	Receive	14:27:28:436	Data fram	Standard	464	8	2e	ff	39	ff	a5	3e	19	19
39	Receive	14:27:28:448	Data fram	Standard	465	8	d4	ce	1e	ff	b7	4e	4c	00
40	Receive	14:27:28:459	Data fram	Standard	466	8	00	00	00	00	00	00	00	00
41	Receive	14:27:28:470	Data fram	Standard	467	8	00	00	00	00	00	00	00	00
42	Receive	14:27:28:481	Data fram	Standard	468	8	00	00	00	00	00	00	00	00
43	Receive	14:27:28:492	Data fram	Standard	469	8	00	00	00	00	00	00	00	00
44	Receive	14:27:28:504	Data fram	Standard	201	8	00	00	00	00	00	00	00	00
45	Receive	14:27:28:516	Data fram	Standard	202	8	18	00	00	00	00	00	00	00
46	Receive	14:27:28:527	Data fram	Standard	400	8	04	14	b5	d7	e3	64	64	0c
47	Receive	14:27:28:538	Data fram	Standard	430	8	00	00	00	00	06	06	ff	0f
48	Receive	14:27:28:549	Data fram	Standard	440	8	4b	56	4b	56	00	00	00	00
49	Receive	14:27:28:562	Data fram	Standard	450	8	e8	03	ef	02	ff	0f	fe	0f
50	Receive	14:27:28:575	Data fram	Standard	201	8	00	00	00	00	00	00	00	00
51	Receive	14:27:28:586	Data fram	Standard	410	8	1e	a5	19	19	b6	d4	ff	39
52	Receive	14:27:28:597	Data fram	Standard	201	8	00	00	00	00	00	00	00	00
53	Receive	14:27:28:608	Data fram	Standard	451	8	28	b6	ce	3f	43	7a	00	30
54	Receive	14:27:28:620	Data fram	Standard	430	8	00	00	00	00	06	06	ff	0f
55	Receive	14:27:28:632	Data fram	Standard	440	8	4b	56	4b	56	00	00	00	00
56	Receive	14:27:28:643	Data fram	Standard	450	8	ea	03	f2	02	ff	0f	ff	0f
57	Receive	14:27:28:657	Data fram	Standard	201	8	00	00	00	00	00	00	00	00
58	Receive	14:27:28:668	Data fram	Standard	460	8	00	00	14	00	fa	00	bc	00
59	Receive	14:27:28:680	Data fram	Standard	461	8	00	06	ff	3f	06	06	dd	00

CSV Structure

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	No	Direction	Time_Scale	Frame_Type	Frame_Format	Frame_ID	Data_Length	ByteA	ByteB	ByteC	ByteD	ByteE	ByteF	ByteG	ByteH
2	0	Receive	14:29:18:886	Data frame	Standard frame	201	8	00	00	00	00	00	00	00	00
3	1	Receive	14:29:18:898	Data frame	Standard frame	251	8	3e	00	00	00	00	00	00	00
4	2	Receive	14:29:18:908	Data frame	Standard frame	430	8	00	00	00	00	06	06	ff	0f
5	3	Receive	14:29:18:919	Data frame	Standard frame	440	8	41	2a	41	2a	00	00	00	00
6	4	Receive	14:29:18:931	Data frame	Standard frame	450	8	00	00	00	00	00	00	ff	0f
7	5	Receive	14:29:18:943	Data frame	Standard frame	201	8	00	00	00	00	00	00	00	00
8	6	Receive	14:29:18:954	Data frame	Standard frame	422	8	00	00	00	00	00	00	00	00
9	7	Receive	14:29:18:965	Data frame	Standard frame	423	8	00	00	00	00	00	00	00	00
10	8	Receive	14:29:18:975	Data frame	Standard frame	400	8	04	14	b5	dd	f0	64	64	04
11	9	Receive	14:29:18:986	Data frame	Standard frame	201	8	00	00	00	00	00	00	00	00
12	10	Receive	14:29:18:996	Data frame	Standard frame	410	8	1e	a9	19	19	a8	d3	ff	39
13	11	Receive	14:29:19:007	Data frame	Standard frame	430	8	00	00	00	00	06	06	ff	0f
14	12	Receive	14:29:19:017	Data frame	Standard frame	440	8	c2	51	c2	51	00	00	00	00
15	13	Receive	14:29:19:029	Data frame	Standard frame	450	8	12	04	4c	03	ff	0f	ff	0f

Using Python to Analyze the CSVs

Input File1

- Number of times each ID appears
- Number of times a hex value appears in each given ID

Input File2

- Number of times each ID appears
- Number of times a hex value appears in each given ID

Output File

- Difference in hex value appearances
- Increase/decrease percentage of hex values
- Appearance of new hex value in a given ID
- Hex to binary

Output Example

This output shows the number of times a hex value appears in a given ID if its apperance count was zero in either of the can capture sessions. The line will be highlighted in red if the number of hex value appearences, is equal to the number of appearences of its given ID. Meaning, the value went from appearing zero times, to appearing 100% of the time.

---

hex value 19(00011001) in ID202, has had an increase from zero appearances to 61 appearances
hex value 99(10011001) in ID202, has had an increase from zero appearances to 61 appearances
hex value 26(00100110) in ID410, has had an increase from zero appearances to 3 appearances
hex value 27(00100111) in ID410, has had an increase from zero appearances to 1 appearances
hex value 19(00011001) in ID430, has had an increase from zero appearances to 2 appearances
hex value 24(00100100) in ID440, has had an increase from zero appearances to 2 appearances
hex value 26(00100110) in ID440, has had an increase from zero appearances to 2 appearances
hex value 32(00110010) in ID440, has had an increase from zero appearances to 2 appearances
hex value 38(00111000) in ID440, has had an increase from zero appearances to 2 appearances
hex value 47(01000111) in ID440, has had an increase from zero appearances to 2 appearances
hex value 69(01101001) in ID440, has had an increase from zero appearances to 2 appearances
hex value 30(00110000) in ID450, has had an increase from zero appearances to 1 appearances
hex value 38(00111000) in ID450, has had an increase from zero appearances to 2 appearances
hex value 20(00100000) in ID451, has had an increase from zero appearances to 61 appearances
hex value 60(01100000) in ID451, has had an increase from zero appearances to 62 appearances
hex value 38(00111000) in ID460, has had an increase from zero appearances to 2 appearances
hex value 26(00100110) in ID465, has had an increase from zero appearances to 1 appearances
hex value 27(00100111) in ID465, has had an increase from zero appearances to 1 appearances

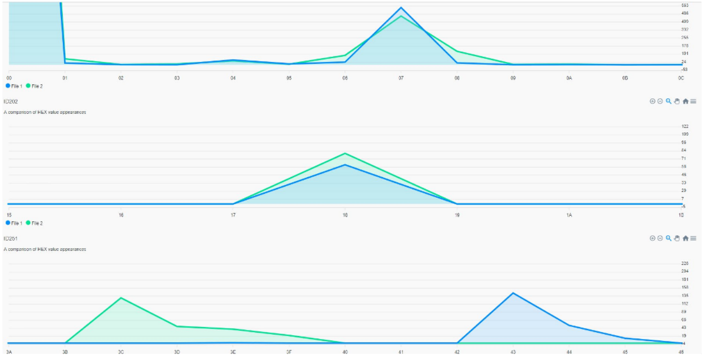
Prototype 1.0

Move to AIM MXS 1.2 Strada

What's Next?

Extensibility: Ideally an individual would be able to analyze a CAN-bus capture session from any manufacturer, with minimal script file editing

Data Visualization:



Byte Annotation: Right now my data is parsed only based on ID and hex values. There is no determination on where a value is in the 8 bytes of the packet which can lead to a misleading output.

Bit Flip Search: 01(00000001) is found in ID100 in file1.CSV , We would then search for 03,05,09,11,21,41,81 and 00 in ID100 in file2.CSV.

Thank You