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
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
Why? Project Inspiration and Purpose
I HAVE NO IDEA

WHAT I'M DOING
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.
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)
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

USB-CAN Analyzer
SKU 114991193
4 Reviews

The USB-CAN Analyzer is a cost-effective high quality and easy to use USB to CAN adapter.

$24.90
10+ In Stock

Tags: CAN BUS, USB to CAN, CAN Analyzer
Software

https://github.com/SeeedDocument/USBCAN-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.

\[
\frac{\text{RPM}}{60\text{seconds}} \times 36\text{steps} = \text{frequency}
\]

\[(6000/60)\times36=3.6\text{khz}\]
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.

equation:
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

\[ \text{speedTime} = \frac{1}{\text{speedFreq}} \times 1,000,000 \]

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

Neutral Switch = False
All This Data and Only One set of Eyes
<p>| | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>M</td>
<td>N</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>-------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---</td>
<td>---</td>
<td>----</td>
<td>---</td>
<td>----</td>
<td>----</td>
<td>---</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>No</td>
<td>Direction</td>
<td>Time_Scale</td>
<td>Frame_Type</td>
<td>Frame_Format</td>
<td>Frame_ID</td>
<td>Data_Length</td>
<td>ByteA</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>Receive</td>
<td>14:29:18:886</td>
<td>Data frame</td>
<td>Standard frame</td>
<td>201</td>
<td>8</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Receive</td>
<td>14:29:18:898</td>
<td>Data frame</td>
<td>Standard frame</td>
<td>251</td>
<td>8</td>
<td>3e</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Receive</td>
<td>14:29:18:908</td>
<td>Data frame</td>
<td>Standard frame</td>
<td>430</td>
<td>8</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
<td>06</td>
<td>ff</td>
<td>0f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>Receive</td>
<td>14:29:18:919</td>
<td>Data frame</td>
<td>Standard frame</td>
<td>440</td>
<td>8</td>
<td>41</td>
<td>2a</td>
<td>41</td>
<td>2a</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>Receive</td>
<td>14:29:18:931</td>
<td>Data frame</td>
<td>Standard frame</td>
<td>450</td>
<td>8</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>ff</td>
<td>0f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>Receive</td>
<td>14:29:18:943</td>
<td>Data frame</td>
<td>Standard frame</td>
<td>201</td>
<td>8</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>Receive</td>
<td>14:29:18:954</td>
<td>Data frame</td>
<td>Standard frame</td>
<td>422</td>
<td>8</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>Receive</td>
<td>14:29:18:965</td>
<td>Data frame</td>
<td>Standard frame</td>
<td>423</td>
<td>8</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>Receive</td>
<td>14:29:18:975</td>
<td>Data frame</td>
<td>Standard frame</td>
<td>400</td>
<td>8</td>
<td>04</td>
<td>14</td>
<td>b5</td>
<td>dd</td>
<td>f0</td>
<td>64</td>
<td>64</td>
<td>04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>9</td>
<td>Receive</td>
<td>14:29:18:986</td>
<td>Data frame</td>
<td>Standard frame</td>
<td>201</td>
<td>8</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>Receive</td>
<td>14:29:18:996</td>
<td>Data frame</td>
<td>Standard frame</td>
<td>410</td>
<td>8</td>
<td>1e</td>
<td>a9</td>
<td>19</td>
<td>19</td>
<td>a8</td>
<td>d3</td>
<td>ff</td>
<td>39</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>11</td>
<td>Receive</td>
<td>14:29:19:007</td>
<td>Data frame</td>
<td>Standard frame</td>
<td>430</td>
<td>8</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>06</td>
<td>06</td>
<td>ff</td>
<td>0f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>12</td>
<td>Receive</td>
<td>14:29:19:017</td>
<td>Data frame</td>
<td>Standard frame</td>
<td>440</td>
<td>8</td>
<td>c2</td>
<td>51</td>
<td>c2</td>
<td>51</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>13</td>
<td>Receive</td>
<td>14:29:19:029</td>
<td>Data frame</td>
<td>Standard frame</td>
<td>450</td>
<td>8</td>
<td>12</td>
<td>04</td>
<td>4c</td>
<td>03</td>
<td>ff</td>
<td>0f</td>
<td>ff</td>
<td>0f</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
This output shows the number of times a hex value appears in a given ID if its appearance count was zero in either of the can capture sessions. The line will be highlighted in red if the number of hex value appearances, is equal to the number of appearances 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