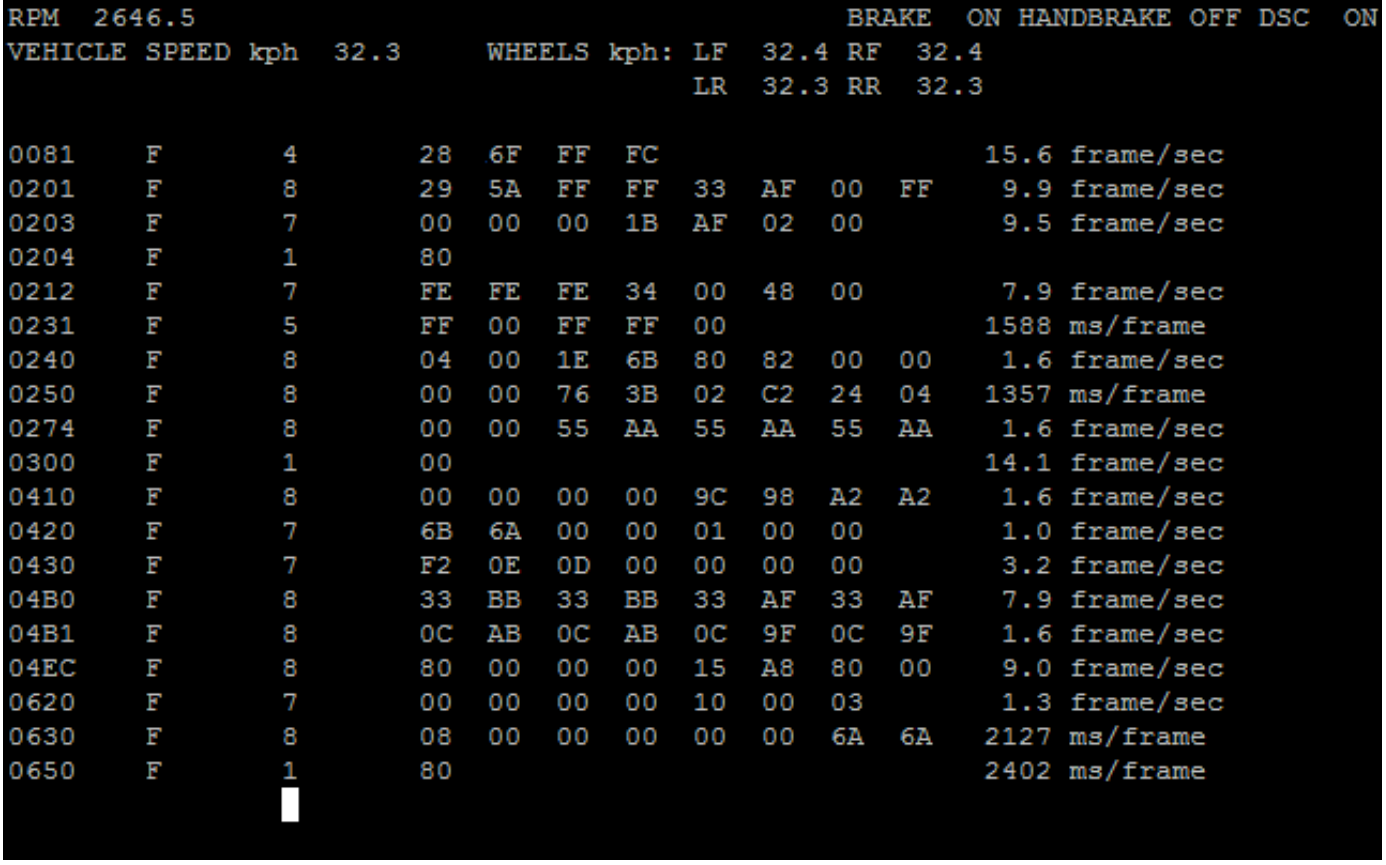
arduino-ecu-logger

Arduino + CAN-BUS shield to monitor fuel consumption and other vehicle parameters. I first wrote this to get a fuel economy meter on my RX-8, and later worked to reverse engineer the messages streaming across the CAN-BUS to see what sensor data is available on the car.

Features:

1. Live streaming of CAN-BUS content over serial link to connected PC (with logging and viewing software on PC side)
2. Computes fuel consumption/mpg and displays on attached serial LCD
3. Dumps available OBD-II PIDs to microSD card



```
RPM  2646.5                                        BRAKE  ON HANDBRAKE OFF DSC  ON
VEHICLE SPEED kph  32.3     WHEELS kph: LF  32.4 RF  32.4
                                        LR  32.3 RR  32.3

0081    F       4       28  6F  FF  FC                          15.6 frame/sec
0201    F       8       29  5A  FF  FF  33  AF  00  FF           9.9 frame/sec
0203    F       7       00  00  00  1B  AF  02  00               9.5 frame/sec
0204    F       1       80
0212    F       7       FE  FE  FE  34  00  48  00               7.9 frame/sec
0231    F       5       FF  00  FF  FF  00                      1588 ms/frame
0240    F       8       04  00  1E  6B  80  82  00  00           1.6 frame/sec
0250    F       8       00  00  76  3B  02  C2  24  04          1357 ms/frame
0274    F       8       00  00  55  AA  55  AA  55  AA           1.6 frame/sec
0300    F       1       00                                      14.1 frame/sec
0410    F       8       00  00  00  00  9C  98  A2  A2           1.6 frame/sec
0420    F       7       6B  6A  00  00  01  00  00               1.0 frame/sec
0430    F       7       F2  0E  0D  00  00  00  00               3.2 frame/sec
04B0    F       8       33  BB  33  BB  33  AF  33  AF           7.9 frame/sec
04B1    F       8       0C  AB  0C  AB  0C  9F  0C  9F           1.6 frame/sec
04EC    F       8       80  00  00  00  15  A8  80  00           9.0 frame/sec
0620    F       7       00  00  00  00  10  00  03               1.3 frame/sec
0630    F       8       08  00  00  00  00  00  6A  6A          2127 ms/frame
0650    F       1       80                                      2402 ms/frame
```

Table of Contents

Materials

1. Arduino Uno
2. [Serial LCD] (https://www.sparkfun.com/products/9394)
3. CAN-BUS shield (includes a joystick and microSD slot)
4. OBD-II to DB9 cable to connect between your car and

the CAN shield
5. microSD card

Arduino side

The Arduino can operate in one of four modes, selected on bootup using the joystick:

1. (down): live vehicle stats. Show MAF-based fuel efficiency (mpg) and consumption (oz/hr) on line 1 of LCD; coolant temperature and throttle position on line 2.
2. (up): CAN spy. Stream CAN-BUS frames over serial connection to attached PC for logging, reverse engineering, and analysis.
3. (left): query ECU for supported OBD-2 PIDs and write to microSD card.
4. (right): serial simulator. Send fake CAN-BUS frames over serial connection to test PC interface code.

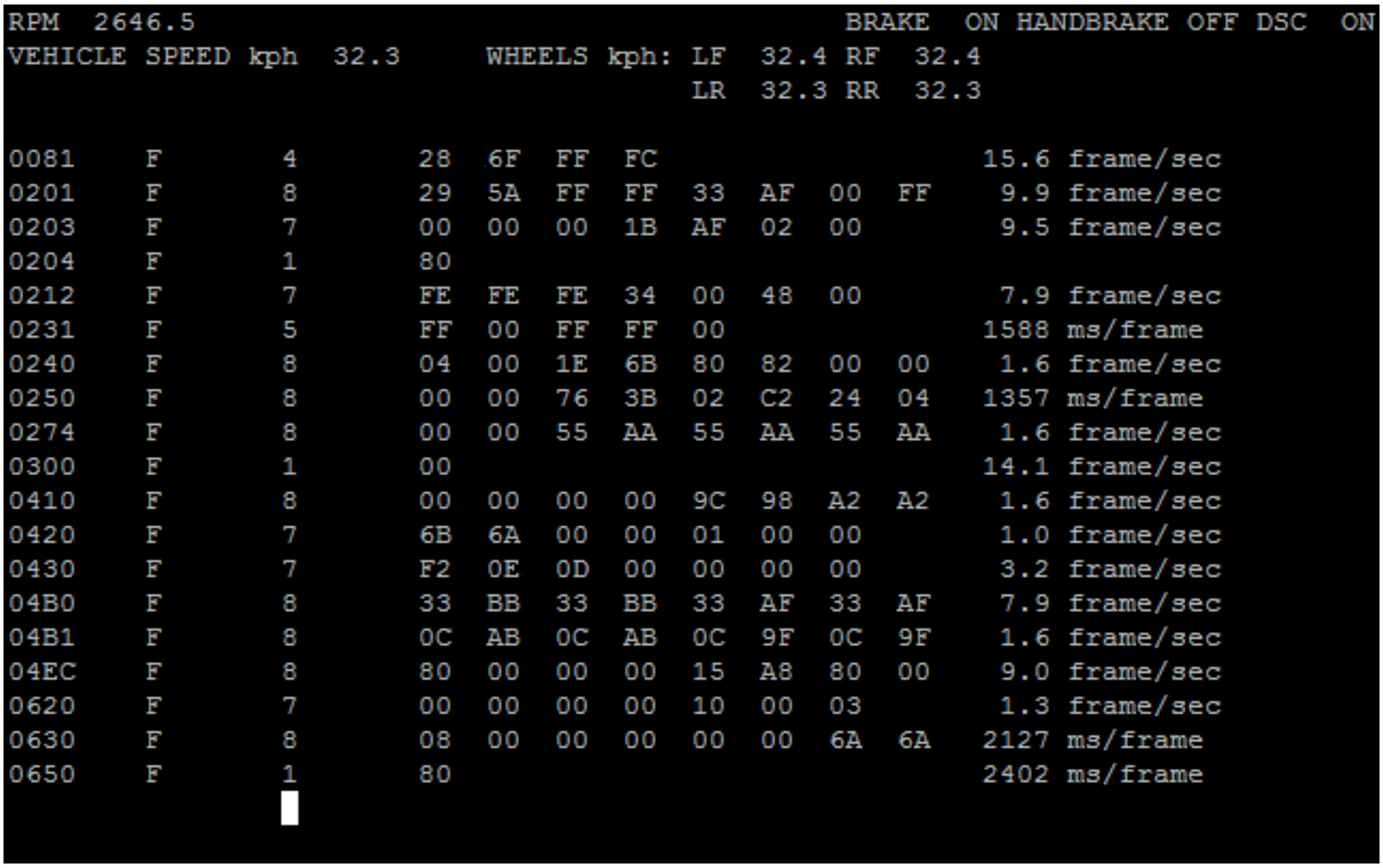
Hardware pin connections are described in logger/README.

The PC interface uses a custom framing protocol for high-speed reliable transmission of CAN frames to the PC. Once every 127 frames, a synchronization frame is sent over the wire; each frame starts with a sentinel byte, and each frame is protected by a CRC8.

PC side

python/can-dumper.py supports reading CAN frames either from a serial-connected Arduino (python/arduino.py:ArduinoSource) or from an on-disk log (python/hdf5_log.py:HDF5Source), and can stream frames simultaneously to a number of outputs, including an on-disk log or a curses-based live display of different CAN-BUS addresses. A demo logfile is available to play with the viewer; run `python can-dumper.py example_log.h5`.

The curses interface is shown below:



```
RPM  2646.5                                                BRAKE  ON HANDBRAKE OFF DSC  ON
VEHICLE SPEED kph  32.3     WHEELS kph: LF  32.4 RF  32.4
                                        LR  32.3 RR  32.3

0081     F       4       28  6F  FF  FC                          15.6 frame/sec
0201     F       8       29  5A  FF  FF  33  AF  00  FF           9.9 frame/sec
0203     F       7       00  00  00  1B  AF  02  00               9.5 frame/sec
0204     F       1       80
0212     F       7       FE  FE  FE  34  00  48  00               7.9 frame/sec
0231     F       5       FF  00  FF  FF  00                      1588 ms/frame
0240     F       8       04  00  1E  6B  80  82  00  00           1.6 frame/sec
0250     F       8       00  00  76  3B  02  C2  24  04          1357 ms/frame
0274     F       8       00  00  55  AA  55  AA  55  AA           1.6 frame/sec
0300     F       1       00                                      14.1 frame/sec
0410     F       8       00  00  00  00  9C  98  A2  A2           1.6 frame/sec
0420     F       7       6B  6A  00  00  01  00  00               1.0 frame/sec
0430     F       7       F2  0E  0D  00  00  00  00               3.2 frame/sec
04B0     F       8       33  BB  33  BB  33  AF  33  AF           7.9 frame/sec
04B1     F       8       0C  AB  0C  AB  0C  9F  0C  9F           1.6 frame/sec
04EC     F       8       80  00  00  00  15  A8  80  00           9.0 frame/sec
0620     F       7       00  00  00  00  10  00  03               1.3 frame/sec
0630     F       8       08  00  00  00  00  00  6A  6A          2127 ms/frame
0650     F       1       80                                      2402 ms/frame
```

The top two rows are a summary of the vehicle's current state, as inferred from decoding data on the CAN-BUS (see section below on the RX-8). Below that is a live-updating view of the last frame received for each CAN-BUS destination ID, including the `rtr` and `data` fields, as well as an estimate of the rate at which traffic is flowing to each ID. Following these fields as inputs are changed on a car (eg, throttle position, rpm, brake engagement, speed, steering angle) can help decode their meaning.

The RX-8 CAN

This blog post describes some reverse engineering of CAN messages from a Mazda 3; much of the data is the same on my Mazda RX-8, but not all. The spreadsheet in data/ (as well as the decoding logic in python/rx8.py) describe the CAN IDs that I have successfully mapped on the RX-8. HDF5 logs can also be plotted using python/plot_logs.py.