Vehicle Predictive Maintenance - Technical Takeaways

Charlie Sands - October, 2023 (in progress) - Northville, Michigan

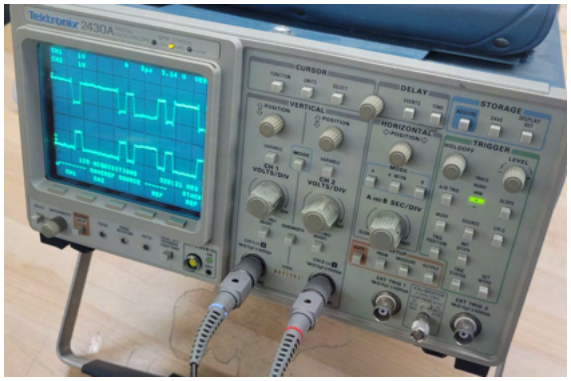
Overview

My vehicle's front seat, custom controls and display terminal

I am currently working on reverse engineering the CAN sensor communication in my vehicle, decoding the data transmitted by the sensors and using their information to preemptively determine what part of the vehicle is likely to fail next. The platform I chose for this project is a 2.4L 2012 Chevrolet Malibu which I purchased, damaged, in an insurance auction. I have designed a custom data acquisition system, head unit display terminal, a CAN bus adapter, and the vehicle's onboard computers. The vehicle's computers gather data from various onboard sensors. The data is sent over the CAN bus either by the sensor itself, or the computer it is connected to. The CAN bus adapter, which is based on a Raspberry Pi 4 running Linux, gathers, filters and decodes this data with the help of a built-in CAN driver before relaying it over HTTP to the head unit. The head unit aggregates the data, saves it to disk and displays it to the vehicle's operator. My goal is to generate artificial failures, log how the data is affected by these changes. Eventually, these labeled conditions will be used to train an artificial intelligence system that will run on the head unit, constantly analyzing current conditions and predicting future component failures. In conjunction with this, I am working on a small, inexpensive board that is able to collect and transmit data to a remote server for analysis, making this concept more marketable and affordable.

CAN Reverse Engineering

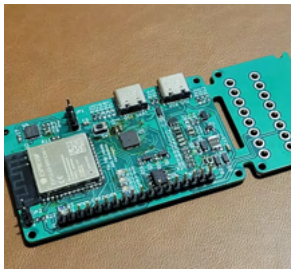
I started by using a commercial diagnostic CAN adapter to decode the data, however, General Motors uses multiple buses, with non-standard implementations. I was forced to replace the commercial adapter with a self made one that can interact effectively with both buses to log and decode data. I determined that the majority of the data is encoded with industry standard floating point and integer encoding schemes. My reverse engineering efforts have centered around determining which CAN message IDs correspond to what vehicle subsystems. This has proven difficult, presently I have deduced ~70% of the vehicle's sensors. Once I have reverse engineered the rest of the sensors, I will be able to focus on refining the analysis I am currently doing on the data and training the AI model.



Analyzing CAN signal from my car's ECU on my oscilloscope

Timeline and Side Projects

Currently my predictive maintenance system is on schedule to have a "minimum demonstrable product" complete by mid March 2024, in time for exposition at my local science and engineering faire. I wish to continue development of the concept after that, reaching a higher level of completion in the late summer of 2024. I am currently working on a provisional patent application for the unique aspects of the system. While working on this project I have discovered various interesting aspects of my vehicle that are not directly related to diagnostics, such as the ability to programmatically control engine speed. This has inspired other side projects including using computer vision to detect the distance of other cars in a line of traffic and programmatically optimize the timing and magnitude of vehicle acceleration to increase fuel efficiency and smoothness. I am concurrently working on this as well.



The PCB I created to cost-optimize my design

Approximate Cost	$2000
Approximate time spent	4 months so far, 6-10 month future timeline
Skills developed	- Digital Printed Circuit Board Design - Embedded Software Development - Industrial Communication Protocols - Safe and Reliable System Design