Vehicle Predictive Maintenance - Technical Takeaways

Charlie Sands - April, 2024 - Northville, Michigan

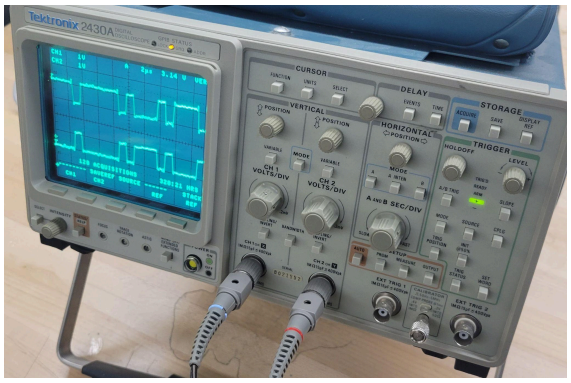
Overview

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. For testing purposes I generated artificial failures by deliberately modifying the vehicle's powertrain, then logged how the data was affected by these changes. Eventually, these labeled conditions are used to train an artificial intelligence system that runs on the head unit, constantly analyzing current conditions and predicting future component failures. Overall, I am happy with the current performance, it has already detected timing chain failure and overheating concerns.

CAN Reverse Engineering

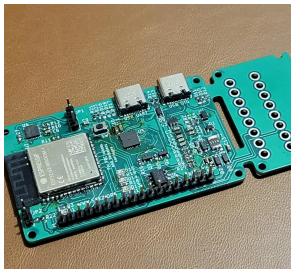
I started by using a commercial diagnostic CAN adapter to decode the data, however, General Motors uses multiple buses, with non-standard implementations. I was forced to replace the commercial adapter with a self made one that can interact effectively with both buses to log and decode data. I determined 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

Product Development

In an attempt to explore the possibilities of bringing a product like this to market I developed a smaller, much less expensive, CAN monitoring device based on the ESP-32 microcontroller. Written in C, the software on this custom board allows for rudimentary predictions to be made, then saves the data it gathers from the sensors to an SD card for deeper analysis on a more powerful computer system. A single test board currently costs around $200 to produce, however this could be reduced to a sub-$100 production cost at scale, allowing a retail cost of ~$200. In addition to exploring more cost effective implementations of my maintenance system, I applied for and was awarded a provisional patent. I am working toward filing a full utility patent application before my provisional patent expires to protect certain novel aspects of my design going forward.



The PCB I created to cost-optimize my design

Approximate Cost	$2500
Approximate time spent	~11 months
Skills developed	- Digital Printed Circuit Board Design - Embedded Software Development - Industrial Communication Protocols - Safe and Reliable System Design