Project: UVic Formula Hybrid Team

1.1 About the Project

The UVic Hybrid Team is a group of engineering students that design and build hybrid vehicles. A model of the vehicle is shown in below Figure 1. The focus of the team is to research and implement powertrain technologies in a hybrid setting. The current powertrain architecture on the vehicle is a parallel hybrid design implementing a 250cc motorcycle engine from KTM and a brushed DC electric motor. A diagram of the implemented architecture is shown below in Figure 2.

Figure 1: Early CAD render of the vehicle in SolidWorks (2015)

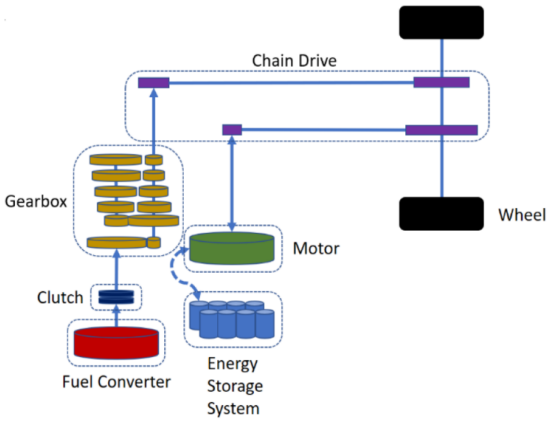


Figure 2: UVic Formula Hybrid parrallel power train architecture

1.2 My Contribution

In the end of my 2nd year as an undergraduate student, I joined UVic Hybrid and played an early role in the vehicle development as shown in Figure 1 and Figure 3. Most of the work I completed was focused on mechanical integration and lending a hand where needed. I helped lay out the mechanical and electrical components on the vehicle. Which included designing mounting brackets and a chain guard for the powertrain.

Figure 3: Helping the team mount the frame to the Jig for welding

1.2.1 Control System Development

After graduating, I joined the team once again with the focus being the control system. The supervisory controller the team uses for the vehicle is a MicroAutoBox II (Shorthand - MABx) from dSPACE. This controller is intended for rapid prototyping and not for long term deployment. Due to its size and weight, the team wanted to explore other alternatives. I used the vehicle as a platform to learn the dSPACE software tools and develop a new control scheme (using Simulink) that can be deployed to a smaller and lighter controller that is based on the Texas Instruments TMS320F28379D chip. Figure 4 provides a summary of the work I have done in a visual form. Figure 5 below shows a prototype I built of the new controller next to the MABx (previous controller).

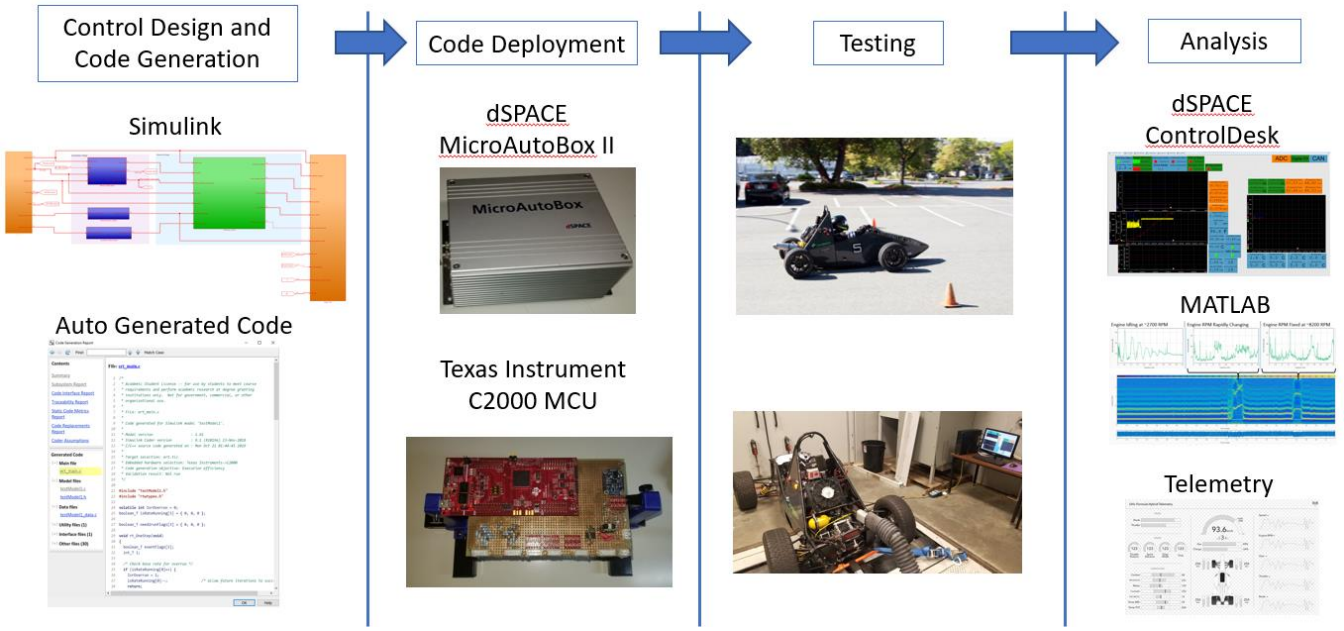


Figure 4: Visual layout showing the work flow and design cycle that I gone through to develop the control system

Figure 5: A prototype of the new vehicle supervisory controller next to the MicroAutoBox II (MABx), which it replaces. Compared to the MABx, the new controller is lighter and smaller making packaging easier. It also includes other features like IMU, GPS, and Wifi communication.

Similar to the MABx controller, the new controller algorithm is developed using Simulink. A sample of the control algorithm is presented in Figure 6. The code is autogenerated from the model using MATLAB Embedded Coder and Texas Instruments hardware support, which is then deployed into the target hardware.

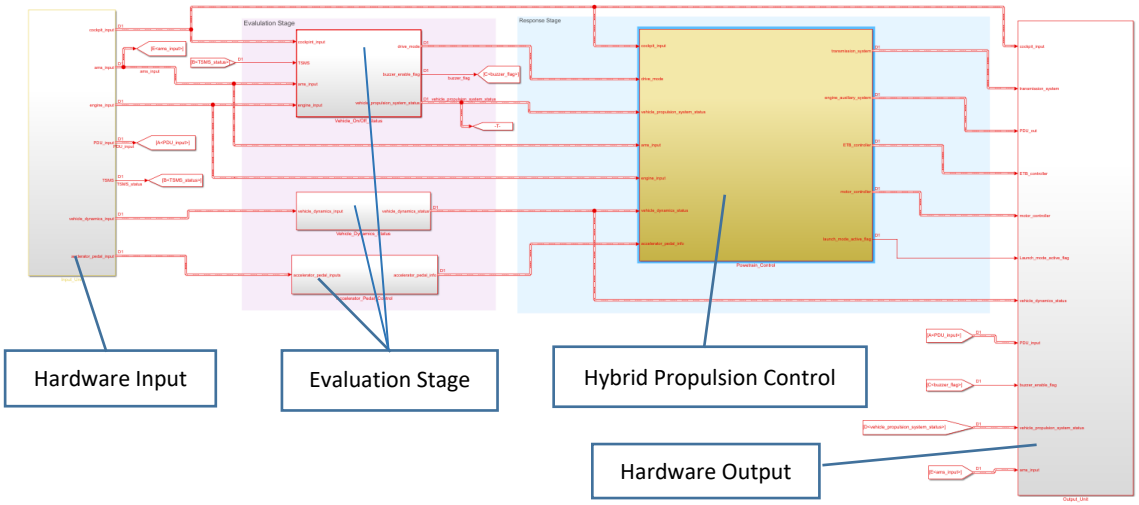


Figure 6: The new control model designed for the TI controller

While developing the logic for the new controller, I reverse engineered the old control scheme developed for the MABx. I used ControlDesk as a instrumentation tool to record sensor and logic data while the car is being operated. Figure 7 to Figure 9 provide images of the vehicle being tested and the graphical user interface made in ControlDesk, which is used as an instrumentation tool to monitor and record the data.

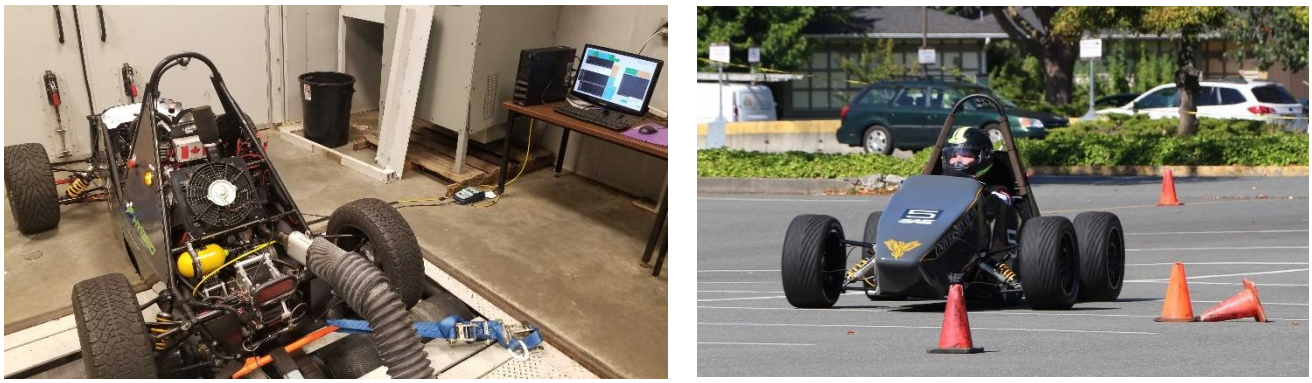


Figure 7: Dyno and track testing the vehicle. It is worth noting the dynomometer hasn't been fixed at the time of testing and we were only using the rollers on a moving surface (no load). Therefore, data regarding torque and horsepower from the dyno can't be measured.

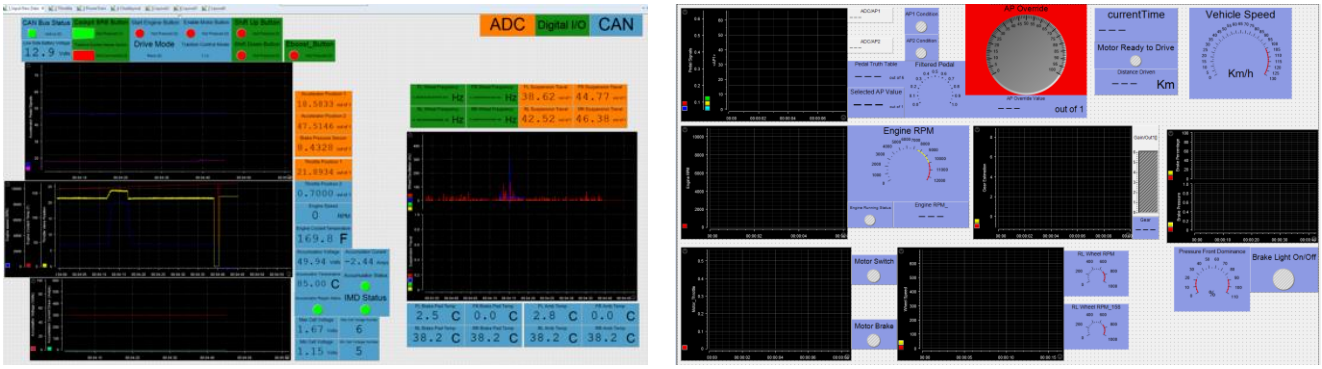


Figure 8: Examples the layouts I created in ControlDesk to monitor and record data (temperature sensors are not calibrated so they are sending wrong values)

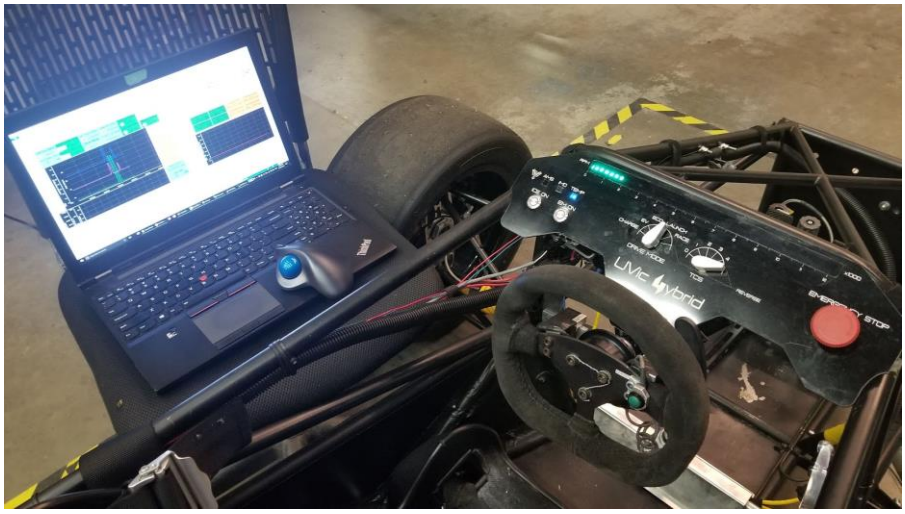


Figure 9: Calibrating the sensor values for the accelerator and brake pedal

The team was also lacking a model of the vehicle for MIL, PIL and HIL simulations. Therefore, I used data to create an approximate plant model to the vehicle using Simscape. This model will allow the team to use the model-based design approach for future control development as shown in Figure 10.

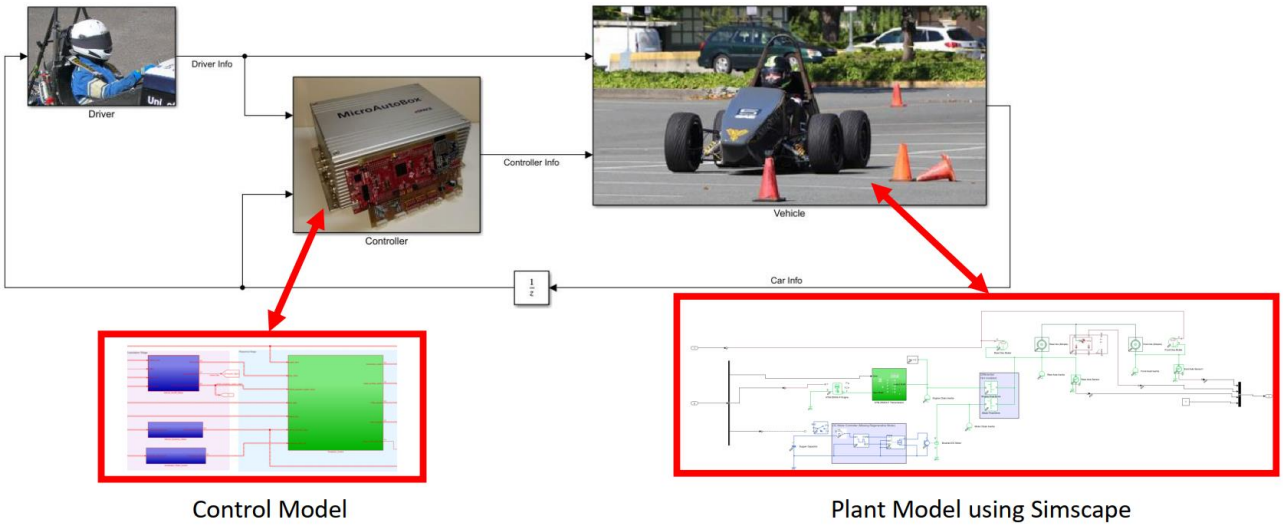
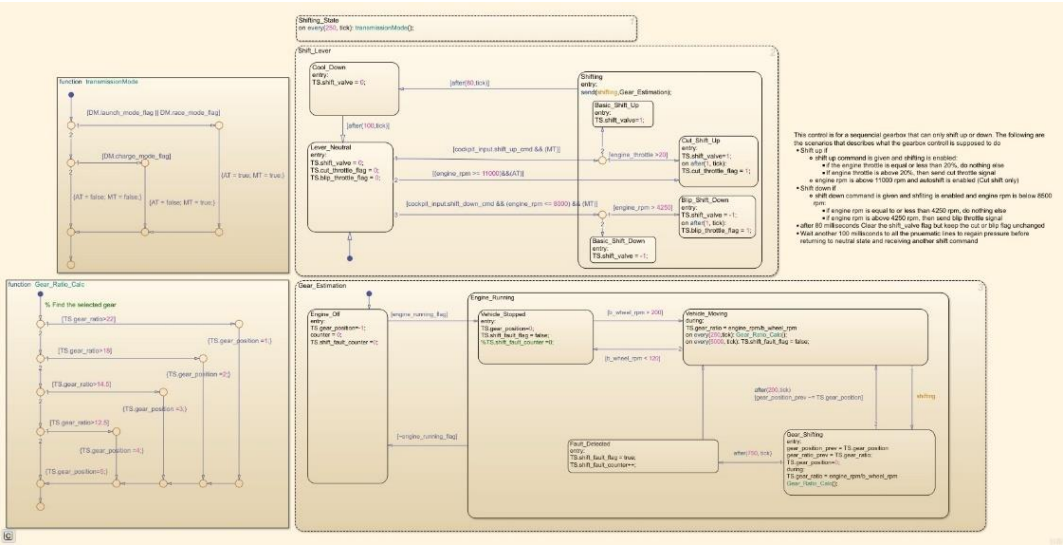


Figure 10: A Simulink model that includes plant model of the vehicle, which is made using Simscape. This model is underdevelopment and will be used for for future MIL, PIL and HIL simulations.

Figure 11 illustrates an example of reverse engineering and control design done on the shift mechanism. The old algorithm implements a hybrid of Stateflow charts, Switch blocks, and logical operators. This layout made debugging very challenging, therefore I redesigned it completely by putting the whole algorithm is a single Stateflow chart. I was still able to preserve previous algorithm's outcome. In addition, I implemented extra features like detecting mis-shifts, and enabling automated shifting based on the set drive mode.



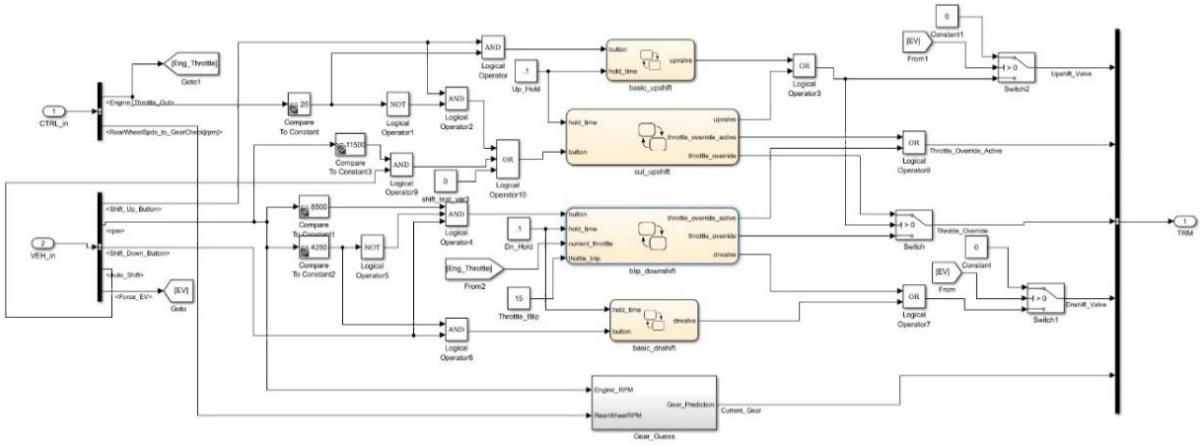


Figure 11 Comparison of the new (top) control scheme for the shift mechanism with the old one (button). The new algorithm is completely implemented in a single Stateflow chart, making it easier to debug and track the various states the shift mechanism can be in.

1.2.2 Signal Processing

For signal post-processing, I used MATLAB to analyse and visualize the data. I have conducted analysis on various components of the car like the engine unit, the electric storage system, the suspension system, and the CAN bus. I reported this analysis to the team to help make decisions on what needs improvement and what we should avoid when designing the next vehicle.

Figure 14 provide an example of this analysis, which is done on the engine section of the powertrain. While doing my analysis to calculate the gearing ratio, I noticed that the engine speed, when compared to the driven wheels speed, did not follow a defined straight line while accelerating. This is usually an indication of slip occurring in the system. As a primary suspect, the auto-clutch was inspected, and the Teflon pads on the auto-clutch were found to be worn out (see Figure 13). After servicing the clutch, the driver felt the engine responded better to throttle demands. I couldn't get data to confirm that the issue is resolved. A mechanical failure occurred on the front wheels when it was tested for the first time, and the data recorder wasn't running then. The car has been inoperable since.

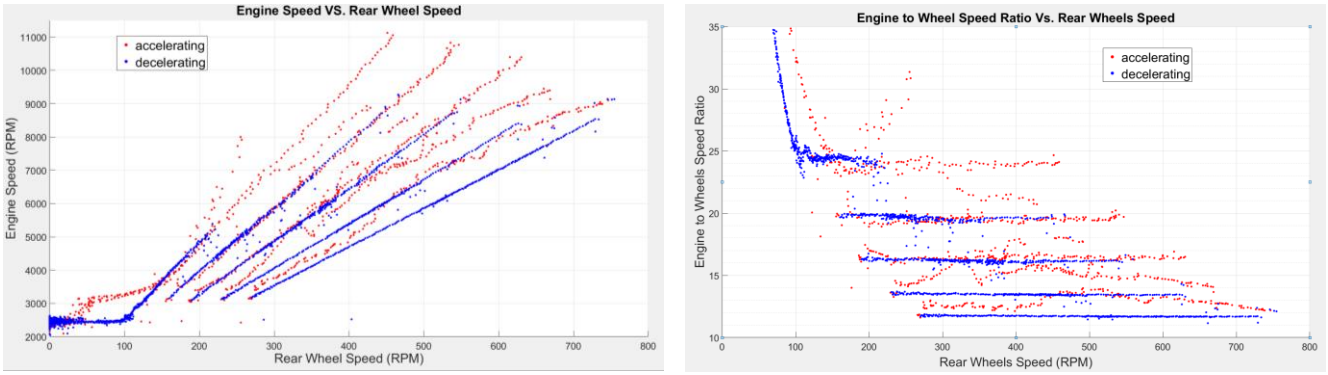


Figure 12: Two graphs representing the same data; one in form of the engine speed in RPM with respect to the driven rear wheels speed (right), and the other in form calculated speed ratio plotted agaist the rear wheels speed. The engine transmission has five gear ratios, which are individually represented by a line in the plot. The acceleration points deviate from those lines, which is an indication of slip occuring in the system. Upon further investigation into the auto-clutch's condition, it was found that it had worn out.

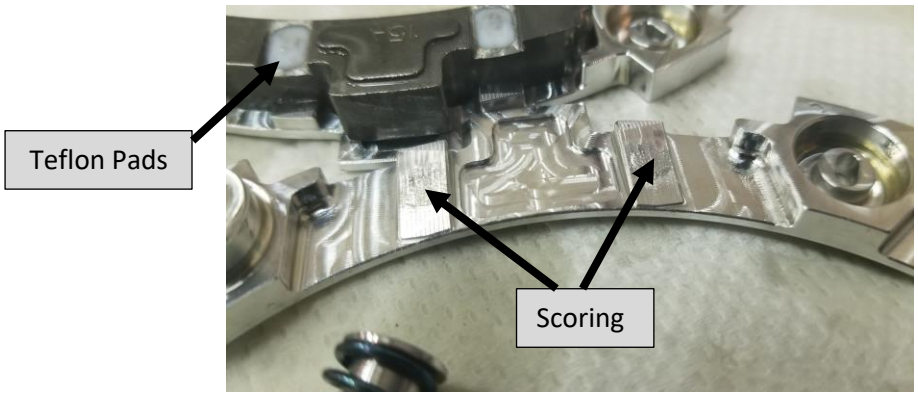


Figure 13: The teflon pads on the auto-clutch were worn out, causing scoring on aluminum mating surface. This affected the clutch engagment parameters allowing for slip to occur at rpm levels where it was supposed to be fully engaged.

Another highlight of my analysis is finding odd steady-state changes in the reading of the all analog sensors that were measured by the MABx when the vehicle is in operation. These changes were not driven by the physical parameters they measure. This investigation started when we noticed that the engine stalls in idle if the radiator fan is turned on. Our engine control unit (MicroSquirt) uses a feedback system to determine the amount of fuel based on the reading of the throttle position sensor (TPS) instead of the air to fuel ratio (AFR). When the radiator fan turns on, it draws about 5 amps. The return path for this current is the frame, which is shared with the analog data lines. This amount of current causes a voltage-drop across the frame that gets detected on the TPS line. This reduces TPS measured value, causing the MicroSquirt to deliver less fuel for the same amount of air. Hence, the engine leans out and stalls. This was eventually fixed by creating a dedicated return path for the radiator fan. However, similar effects are still caused by other loads like the fuel pump. With the root cause determined, the team decided to implement dedicated ground paths for all analog and communication lines in the future car design.

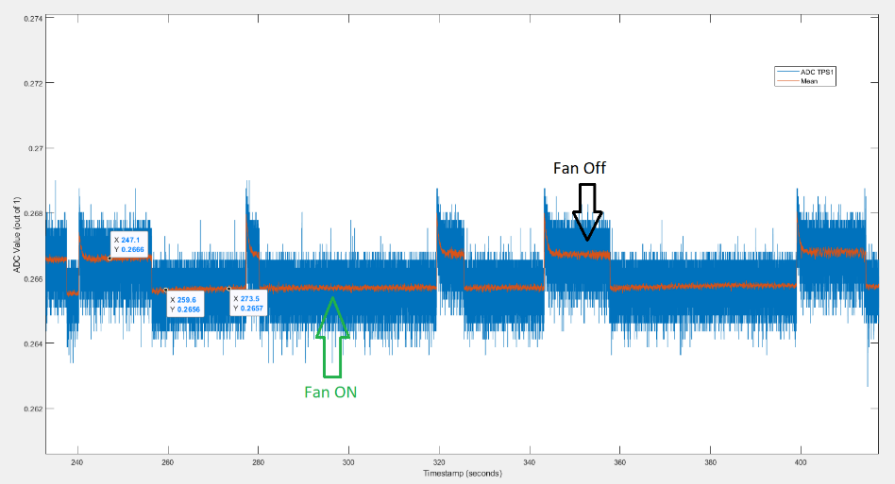


Figure 14: Analysis on the analog signals recorded by the MABx shows a change in the steady state value of a signal depending if certain power units like the fan motor is enabled. The vehicle frame is being used a common return path for all signals, and certain load units consume enough current to cause a votlage-drop across the frame that is measurable by the ADC units.

Figure 14 provides an indication on the amount of noise that we are dealing with on the car. I worked on creating better digital filters for the signals. I used a combination of the Signal Analyzer and Filter Designer apps on MATLAB to design these filters. The old control scheme used 200-point moving average filters. Giving the sample rate is 1 millisecond, this made the car relatively slow at responding to throttle commands. I switched to using IIR filter because it's computationally faster as it requires fewer samples compared to a FIR filter with similar frequency response characteristics. I started with a Chebyshev II filter, and I manipulated the poles and zeros in the Z domain until I reduced the overshoot to an acceptable level.

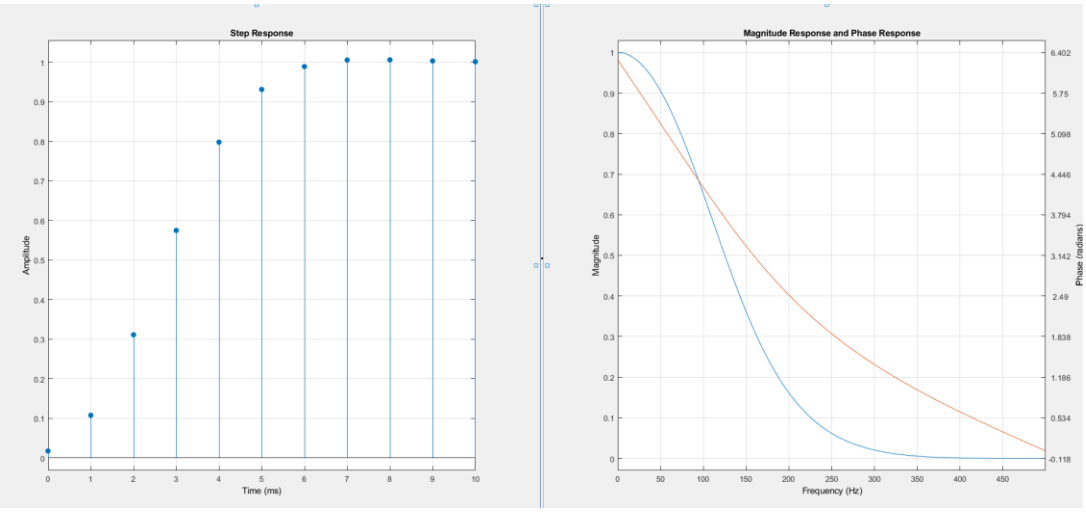


Figure 15: IIR Biquad filter properties implemented on the throttle position sensor

However, I noticed a peak at the Nyquist frequency (500 Hz) in some of the signals, so I decided to do more analysis by increasing the sample rate to 10 kHz on the MABx. It's worth noting that the MABx ADC hardware comes with a low pass filter with a corner frequency of 10 kHz, and a conversion time of 6.6 microseconds. Therefore, I kept this information in mind while performing the analysis. One of the results can be seen in Figure 16, which plots the frequency domain analysis conducted on a sensor data that informs the position of the rear left wheel suspension. The data was recorded while the car is stationary, the electric motor is disabled, and the engine is idling at ~2700 RPM. There are two regions in the data where the engine is rpm is increased, which are indicated

in the graph. The spectrogram in the figure shows frequency peaks starting at ~550 Hz. This 550Hz fundamental frequency and its harmonics do get slightly attenuated when increasing the engine rpm, but other noise are introduced in the process, which are getting partly aliased. I am currently using this information to work out both the sources of this noise. In addition to some hardware filters to minimize the noise. This work will improve the effective number of ADC bits (ENOB) for the analog signals on the car, which will ultimately increase the signal to noise and distortion ratio (SINAD).

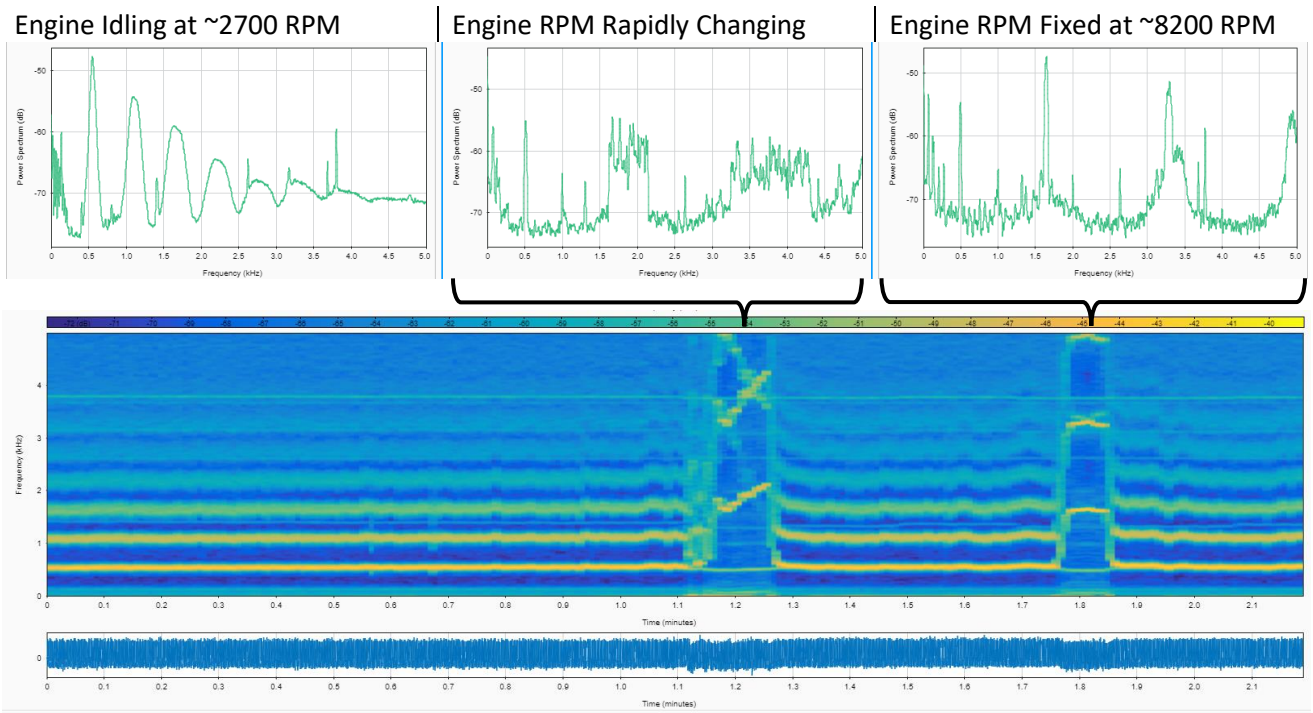


Figure 16: Spectrum analysis on the the rear left wheel suspension travel data. The car is staionary during recording, and the signal is sampled at 10 kHz, with the DC compoent of the signal removed. The graphs shown are: frequency spectrum (top), the spectrogram (middle), and the data in the time domain (buttom). There are three frequency spectrum graphs corresponding to three events in the data depending on the engine speed which are: engine at ~2700 RPM (idle), engine RPM changing, and engine at ~8200 RPM.

I have established a CAN database for the team using the CANdb++ Editor from Vector. I condensed the data as much as possible by minimizing the number of bytes reserved for each signal in a message. The number of reserved bytes depends on many factors like the max and min value of the signal, weather or not the value is signed, and if the value is scaled by a factor.

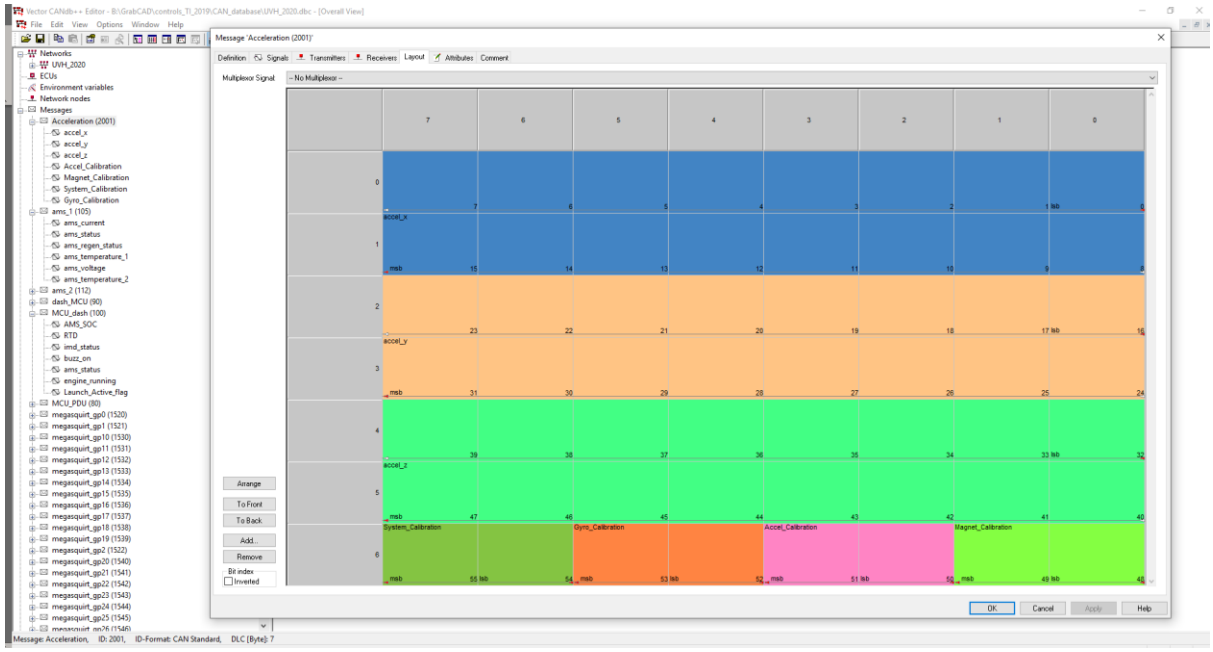


Figure 17: View of the Acceleration message content in the CAN database mde for the car

1.3 Project Conclusion

I was able to develop a new control scheme for the vehicle with new and improved features like better digital signal processing capabilities, launch control, optimized CAN database, battery protection system, and drive modes. This control scheme was ported to and tested with the new TI hardware. The MABx will remain in use for control development before getting ported to the TI board. For this reason, I created a workflow for recording data from the MABx and made MATLAB scripts for importing, post processing, and visualizing the data.