Development of a stand-alone embedded system for controlling and distributing power in a FSAE Race Car

ENEL 500 Capstone

Team Name: Ownin with Onen

Team Number: 500-ES-4

Academic

Project Manager

Aidan Johnson

aidan.johnson1@ucalgary.ca

Project Members

Aidan Johnson 30043501

Dylan Rae 30020151

Adam Trepac 30019593

Tyler Sawatzky 30020880

Guillaume Raymond-Fauteux 30053731

Academic Advisor

Denis Onen | donen@ucalgary.ca

Industry

Industry Sponsor

University of Calgary FSAE Schulich Racing Team

Sponsor Representative

Kevin Chow | kevin.chow@ucalgary.ca

Table of Contents

List of Figures

List of Tables

Glossary

PDM: Power Distribution Module

CAN: Controller Area Network

PCB: Printed Circuit Board

LED: Light Emitting Diode

MCU: Microcontroller Unit

LDO: Low Dropout

ECU: Engine Control Unit

DAQ: Data Acquisition

GPIO: General Purpose Input/Output

IC: Integrated Circuit

PC: Personal Computer

SD: Secure Digital

TCP: Transmission Control Protocol

SMD: Surface Mounted Device

MVP: Minimum Viable Product

Cube IDE: Integrated Development Environment for STM32 Cube

STMicro: STMicroelectronics

RTOS: Real Time Operating System

eFuse: An active fuse-style integrated circuit that can switch power safely in fault scenarios

SWD: Serial Wire Debug

JTAG: Joint Test Action Group

HAL: Hardware Abstraction Layer

ADC: Analog to Digital Converter

HSD: High Side Drive

PWM: Pulse Width Modulation

PTC: Positive Temperature Coefficient

eLoad: Electronic Load

Short Team Bios

Aidan Johnson

Aidan Johnson is currently in his final year of Electrical and Computer engineering at the University of Calgary. He recently completed an internship with a start-up company called FUS Instruments in partnership with the Foothills Hospital leading the development of their Focused ultrasound therapy software platform. Aidan is passionate about hardware and software development and has previous experience with project management, embedded systems, PCB design and software application development. During his spare time, you can find Aidan in the outdoors, hiking, backpacking and taking scenic pictures.

Dylan Rae

Dylan is a student at the University of Calgary in his final year of the dual degree program studying electrical engineering and finance. He is interested in fields such as software development, machine learning, and FinTech. He is passionate about how we can bridge engineering and business to solve some of the world's most important problems.

Tyler Sawatzky

Tyler is in his final (6th) year of a dual degree in electrical engineering and finance. Through his experience in internships and extracurriculars Tyler has primarily gained skills in hardware development and PCB design, but he is also interested in building experience in embedded systems and software development.

Adam Trepac

Adam is in his final year of electrical engineering with a minor in computer engineering. Through his experience on extracurricular clubs and internship Adam has learned more about the

Automotive sector and has gained skills in embedded systems as well as hardware development and testing. Adam is looking forward to applying his experience to this project.

Guillaume Raymond-Fauteux

Guillaume is in his final year of electrical engineering with a minor in computer engineering. His interests lie in the embedded systems space, enjoying both programming on microcontrollers and designing hardware around them. His internships gave him experience developing software and systems level testing skills which he aims to bring to this project.

Project Motivation and Objectives

Schulich Racing is an interdisciplinary team that designs, builds, and tests a 2/3rd scale Formula 1 car. The team competes in an SAE-regulated competition against 80 other collegiate teams from around the world. To continually improve the performance of the vehicle, the team relies heavily on electrical systems that support data collection. This data is vital to the team's success and is used to make educated engineering decisions that further improve vehicle performance. To collect this data, the team deploys numerous sensors whose values are transmitted to a cloud database while simultaneously being logged to an onboard SD card.

Previously, all these sensors and other electrical systems were powered by a third-party PDM. This year, the Schulich Racing team is planning to deploy more sensors and electrical systems than ever before. Because of this growth, the third-party PDM is no longer capable of meeting the requirements that necessary to properly power all electrical systems. Specifically, the existing power management solution is lacking in three main areas. First, the third-party PDM only offers outputs at 12V. Since the race car has systems that don't run at 12V, the team previously solved this issue by using a breakout board that down-regulates the 12V outputs from the PDM to their required voltage level. This is not ideal since it adds additional mass and components to the race car and could be accomplished in a centralized solution. Second, the existing solution does not have enough output pins to support the number of devices the new car will require. Finally, the third-party PDM offers limited capabilities for logging power usage and faults. An improved logging system would improve the troubleshooting process and help with improving general vehicle reliability.

Creating a customized solution for a PDM also acts as a steppingstone in developing a fully centralized electronics system for the vehicle. Currently, the car's electronics system contains several off the shelf modules such as the engine control unit and data acquisition unit. A long-term goal for the team would be to deploy a fully customized solution that integrates many of these modules into a single system. Doing so would reduce the weight of the car, open up opportunities for further customization, and increase the control that team members have on the car's behaviour.

The objective of the project is to design and manufacture a new PDM that solves the problems with the existing third-party solution. The new solution should offer a regulated voltage output for every voltage level required by the onboard electrical systems. In addition, it should have enough pins to supply power to every electrical component on the vehicle without having to use one pin to power multiple outputs. Finally, the new PDM should implement detailed logging features that can be broadcast over CAN for making troubleshooting easier. In addition to solving these core problems, a custom-built PDM will offer the opportunity for the addition of other new features that can be used to further improve the performance of the race car.

Project Scope and Deliverables

The scope of this project is to develop a standalone system that will safely control and distribute power to all the endpoints on the race car. The embedded system will be implemented on a custom PCB and will use eFuse modules to detect faults and safely switch the power supply ON or OFF for each endpoint on the race car.

The specific endpoints that our system will be required to distribute power to are as follows:

DC motors

Our team took measurements of each of these devices with the exception of the starter motor to obtain transient and steady-state current values. See Appendix Appendix A: High Power Device Oscilloscope Current MeasurementsAppendix B: Shutdown Circuit FSAE Rules and Clarification Requestfor the oscilloscope traces of the current measurements our team took while powering on the fans and pump. The values for the starter motor are using measurements that the team has from earlier, and our team will be redoing the measurements to confirm.

Node/Endpoint Name	Voltage	Transient Current (A)	Steady State Current (A)
Starter Motor	12	120	65
Fan 1	12	18.7	4.9
Fan 2	12	24	4.7
Water Pump (x2)	12	4.48	1.5
Fuel Pump	12	21.3	8

Table 1: DC Motor Endpoint Voltage and Current Measurements

Sensors

The team does not currently have current/power measurements for these, but based on the types of sensors we expect that all of these sensors will have a steady state current << 1A. Our team will be conducting measurements to confirm and more precisely quantify the power/current requirements of these devices.

SR Subteam	Sensor	Quantity	Required voltage
Powertrain	TPS (Throttle Position)	1	5V
Powertrain	EGT (Exhaust Gas Temperature)	4	5V
Powertrain	Wideband O2	1	12V
Powertrain	Neutral Switch	1	5V
Powertrain	Gear shift potentiometer	1	5V
Suspension	Pressure Transducer (Active ARB)	2	5V
Suspension	Brake Pressure	2	5V

Suspension	Wheel Speed	4	12V
Suspension	Potentiometer	4	5V
Suspension	Rotary Pot (Steering angle)	1	5V
Suspension	Wheel Temp/Pressure	4	5V
Suspension	Accelerometer	1	5V
Suspension	Brake temperature	4	TBD

Table 2: Sensor Endpoints Quantity and Voltage

Other Devices

These current/voltage values were obtained either from previous measurements by Schulich Racing or datasheet values. Our team will be redoing these measurements to obtain more precise values for our PDM design.

Node Name	**Voltage**	**Steady State Current (A)**
Novatel Receiver	12	0.44
ECU	12	0.5
DAQ	12	0.1
Display	5	0.2
Radio	8	0.3 (1.7A when transmitting)
Injection	12	< 8
Ignition coils	12	< 8
Starter switch power	12	< 1
Shutoff circuit power	12	< 1

Table 3: Other Device Endpoints Voltage and Current Measurements

Hardware

The hardware for this project consists of a custom PCB that controls each of its components with a microcontroller. The microcontroller will be responsible for the control and monitoring of the eFuse modules as well as broadcasting and reading CAN messages. The PDM should be designed to be powered by a 12-volt battery and should regulate this supply as needed based on the requirements listed above for each endpoint. The below lists summarize the specific requirements the hardware design must meet:

Must-have:

- Output circuits must withstand and protect from:
    - Over voltage
    - Under voltage
    - Over current
        - Must trip within 100ms
    - Thermal overload due to large current draw
- Reverse battery protection
- Provide 3 different voltage rails on output pins:
    - 8 Volt Rail +/- 5%
    - 5 Volt Rail +/- 5%
    - 3.3 Volt Rail +/- 5%
- Provide output pins with the following voltage:
    - 19 12V pins
    - 1 8V pin
    - 25 5V pins
- Provide output circuits with the following current limits: (Output circuits connect to one or more output pins, these are preliminary estimates subject to change based on further testing)
    - 12V:
        - x1 65A
        - x4 20A
        - x2 10A

        - x12 1A
    - 8V:
        - x1 2.5A
    - 5V:
        - x25 <1A
- SWD interface
- Current monitoring for high current critical endpoints
- Communicate over CAN
- Battery voltage monitoring
- All components and circuitry integrated into a custom PCB

Should-Have:

- Provide extra output rails to supply the following:
    - 4 extra 12V devices:
        - x4 12V 10A outputs
    - 4 extra 5V devices:
        - x4 5V 1A outputs
- Adjustable current limit circuit for select output circuits
- Voltage monitoring of regulated 8V and 5V rails
- Board temperature monitoring
- Shutdown circuit
    - Turn off Ignition and Fuel pump output when shutdown input circuit opens
- Modular design to allow for future expansion

Nice-to-Have:

- 2.4GHz transceiver (for ANT, Wi-Fi, Bluetooth, etc.)
- Integrate PDM solution with DAQ onto single PCB
- LEDs to indicate if an output circuit is switched on
- Current monitoring for all outputs
- Back-up battery to run MCU and CAN modules in event of main power supply failure

Firmware

The firmware will be responsible for making all control decisions for the PDM while it is in operation. The firmware should be able to seamlessly interact with the eFuse modules and react to any faults that are detected. In addition, the firmware should support CAN communication and will need to broadcast and interpret messages related to fault occurrence, battery status, and power usage. The below lists summarize the specific requirements the firmware design must meet:

Must-have:

- Ability to read from the following sources ADC at 20 samples/second at 8-bit resolution:
    - Current draw from each output circuit
    - Battery voltage
- Control output circuit current flow
- Broadcast and receive data over CAN
    - Broadcasted data includes current draw from each output circuit and battery voltage levels
    - Broadcasted data must be sent at 20Hz
- Ability to assign customized fault retry procedure for each eFuse in case of failure
    - Includes detecting component fault, resetting to initial state, powering back on, and repeating the process up to 3 times before retiring the faulty component.
    - This requirement is only relevant if the eFuses used do not implement fault resetting itself. Supply chain constraints have meant that an ideal choice of eFuse with this functionality may not be possible.

Should-Have:

- Ability to use ADC at 12-bit resolution at 20 samples/second
- Ability to control current limit on select individual output circuits

Nice-to-Have:

- Interface with a 2.4GHz transceiver to receive and transmit data
- Functionality to selectively turn off low priority devices in the event of a low battery voltage or when engine starting to ensure continued operation

- Ability to run DAQ firmware from previous capstone project
    - Includes reading from ADC, writing data to SD card through SPI, reading data from PWM input, and transmitting over Novatel radio

System

When viewed from a system level, there are several design considerations relevant to the automotive racing application that should need to be incorporated into the design. First, the most important system-level design consideration is the total mass of the PDM. This is because reducing the total mass of the race car is the best way to improve the overall performance of the car. When designing the PDM, it must be ensured that the total mass of the new solution is not greater than the mass of the existing third-party design. Along with reducing mass, the new PDM must minimize the overall dimensions to ensure the PDM can be mounted on the race car with ease as space for mounting components is often limited.

Another important consideration is ensuring that the design is as modular and scalable as possible. Due to the student-run nature of Schulich Racing, every year the more experienced members are graduating and leaving the team bringing with them their specific knowledge. Because of this turnover, it is vital that designs can be easily replicated and scaled up to accommodate for future growth. To ensure the new PDM design meets these requirements, two crucial philosophies must be considered when designing the solution. First, accurate and thorough documentation detailing the design for each model must be kept so that future Schulich Racing team members can understand the design process. Second, the overall design should be as modular as possible with high levels of abstraction built into each module. This will assist future team members working on related projects to utilize our designs more easily when attempting to integrate them into other projects or scaling them up to accommodate changes in requirements.

Must have:

- All functionality implemented with one PCB
- Maximum mass of: 600g
- Maximum dimensions of 200mm x 200mm x 50mm
- Enclosure that encapsulates the PCB
- Automotive specification wiring connector to connect to harness

Should have:

- Maximum mass of: 500g
- Maximum volume of 108mm x 128mm x 39mm (size of current PDM)
- Water resistant enclosure

Nice to have:

- Maximum mass of: 350g
- Maximum volume of 96.8mm x 114.8mm x 34.8mm (10% reduction each dimension)
- Water resistant metal enclosure

Main Deliverables

Deliverable Name	**Deliverable Description**
System Architectures for the Hardware and Firmware	Complete Hardware and Firmware block diagram illustrating how the system and their modules will communicate data and power one another.
Components Selected and Schematics created for each module and circuit	Schematics for each module developed in Altium each including their appropriate circuit(s) with the specific components selected for the initial prototype design. These components will also be ordered after the schematics of the different modules are completed.
PCBs developed and tested for each module	Each module's PCB developed in Altium and verified with 3D models of components and Altium's Design Wizard. Once each PCB is verified, they will be ordered and once arrived assembled and tested.
Complete System Circuit Schematic Created	Complete Circuit Schematic in Altium connecting every module together with finalized components.
Final PCB developed and tested	Final PCB developed in Altium based on the complete circuit design and once arrived assembled and tested.
Fully Functioning Firmware tested on the final PCB	The Firmware for the entire PDM unit will be tested and refined on the final PDM PCB unit.

Table 4: Main Project Deliverables

Preliminary Design of the Proposed Solution

Hardware:

The preliminary design for our hardware solution in the development of a standalone power distribution module consists of a custom designed PCB without the enclosure. Our hardware solution has been split into multiple different modules consisting of each of the different components to support power distribution and management. The main modules are:

1. Power Regulation module
2. Diagnostics Module
3. CAN Module
4. MCU Module
5. eFuse Module

The diagram below shows a preliminary high-level block diagram detailing the mentioned modules as well as the basic flow of voltage and data signals:

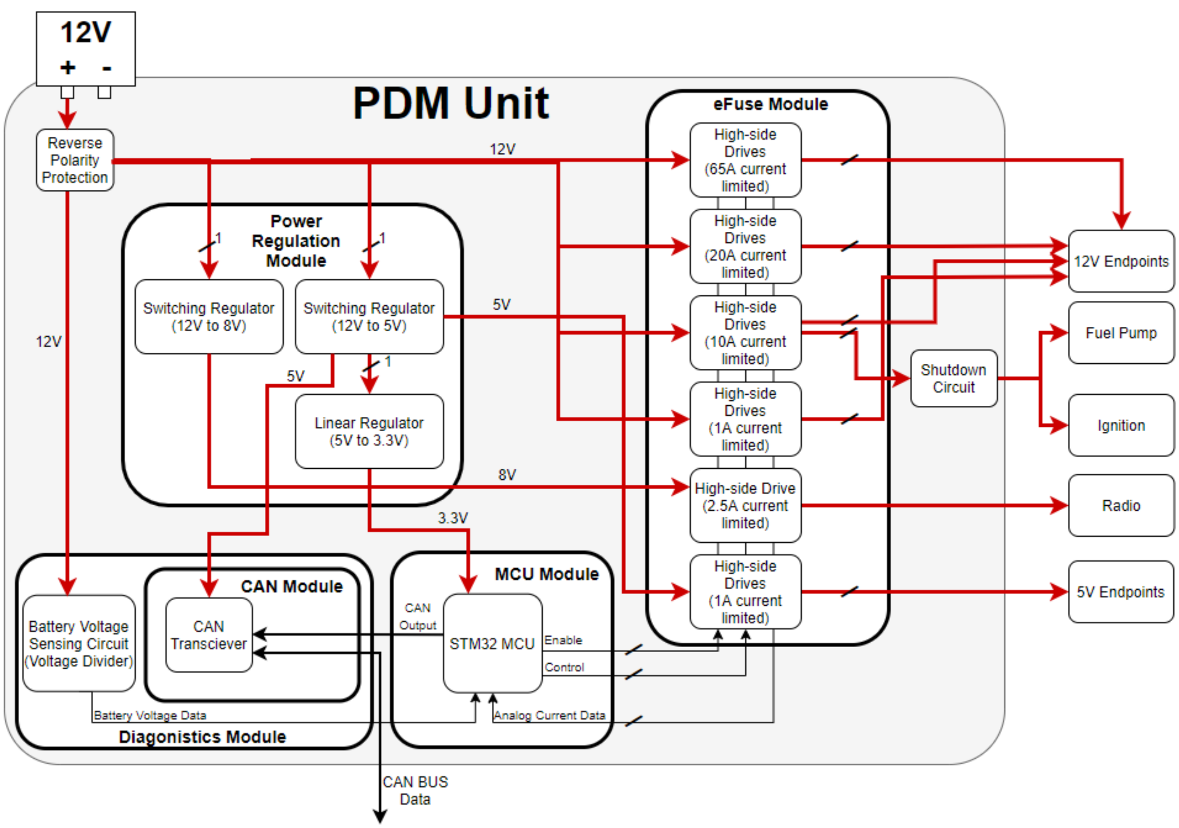


Figure 1: Preliminary High-level System Block Diagram illustrating different hardware modules & signals

This system includes a reverse polarity circuit that will be connected directly to the 12-volt line (battery) to protect the unit if the battery is connected in reverse. Our system also includes a shutdown circuit to properly shut down two specific endpoints: the Fuel pump and Ignition, as required by the FSAE rules.

For the PCB design of the power distribution module, we will be developing a multi-layered PCB utilizing different sized external traces to deal with the different current draws as well as tented vias. The PCB will be mounted to the enclosure with screws and include necessary heat sinks for specific components and the unit where necessary to improve heat dissipation. To protect the sensitive electrical components and the unit from external forces an enclosure will be developed that will include openings for the output and input connectors. The enclosure may also be made with a different material or include heat sinks to also help mitigate overheating of the unit.

Power Regulation Module

The power regulation module's main goal is to properly regulate the power/voltage from the battery on the board for each respective output and onboard component. Voltage regulation is a critical component of our power distribution system as the 12V battery will need to be downregulated to the appropriate voltage for each of the various endpoints such as the display, fuel pump, fans, and sensors shown in table 1, 2 and 3. Currently on the car, the third-party MOTEC PDM does not perform any voltage regulation for any of the output pins, so an external breakout board shown below, is required to regulate the 12 volts down to only 5 volts for some of the various sensors.

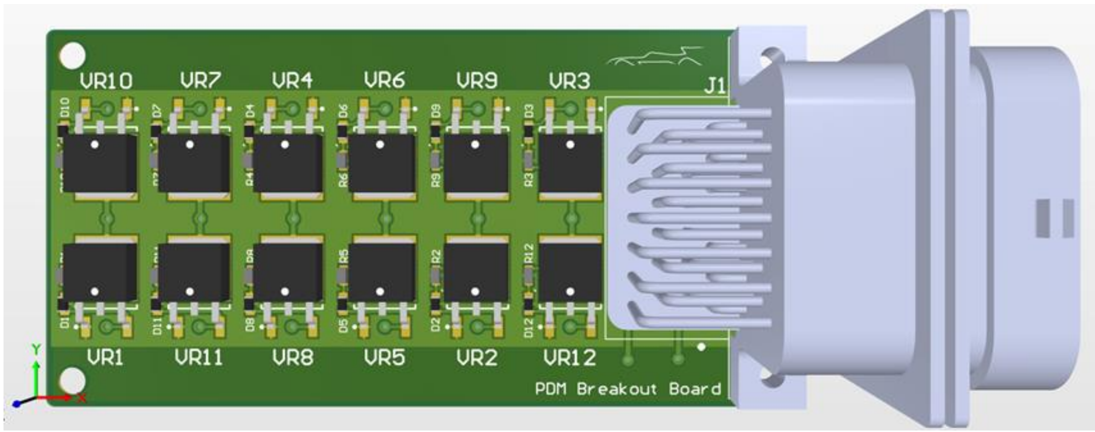


Figure 2: Current PDM Breakout Board

The voltage regulation provided by this PCB board along with the regulation of 8 volts and 3.3 volts will be provided in our design. This will enable the ability for our system to provide an accurate 8V, 5V and 3.3V rail to power all required endpoints as well as the on-board electrical components such as the microcontroller, CAN Transceiver, and any other onboard components we plan to add on in the future such as onboard temperature sensor.

Voltage regulation is the conversion of a DC voltage to a different DC voltage and the two main ways to regulate voltage is through either using a linear or switching regulator [1]. The main considerations for our design when choosing an appropriate voltage regulator for a specific scenario are:

- Efficiency (Power Dissipation)
- Dropout Voltage
- Current Rating
- Operating Temperature Range
- Voltage Output's Accuracy
- Input Voltage Range

To improve the efficiency of voltage regulation a switching regulator can be utilized over a linear regulator, however a linear regulator which uses a resistive load in a voltage divider configuration to transform the provided voltage into the desired voltage at the output [2] are normally cheaper and have a low amount of noise.

For our preliminary design it was decided to use a switching regulator specifically the TPS5431 manufactured by Texas Instrument, to regulate the 8V supply rail to properly power the radio endpoint on the car. This component was selected due to that it provides up to 3A controlled current output as well as provides an 8V output voltage with an accuracy of +/- 1.5% which is within our requirements [3]. A switching regulator was also chosen for this supply rail due to its higher efficiency than a linear regulator as it can have an efficiency of up to 95%. This is important as when the radio is transmitting it draws approximately 1.7 amps with a voltage difference of 4V (12V-8V), causing there to be 6.8W of power dissipating which would generate a substantial amount of heat. Below is a schematic that was created for the TPS5431 switching regulator to provide a fixed 8V output with a max 2A load using Texas Instruments Webench software:

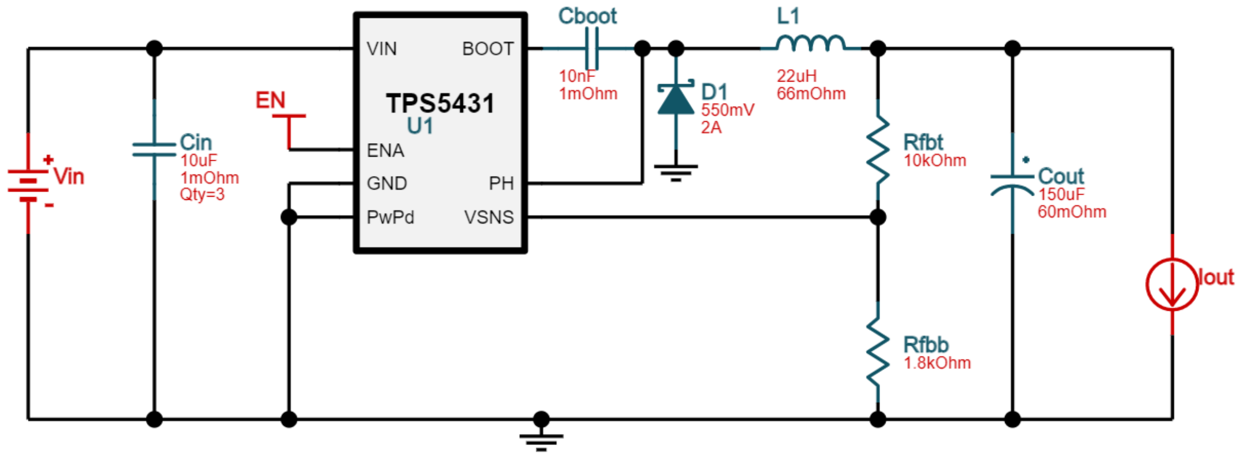


Figure 3: Schematic for the TPS5431DDA Switching Circuit

For the 5V supply rail a switching regulator specifically the LM2575HVT-5.0 manufactured by Texas Instrument [4] will be used to step down the battery's voltage for the various 5V sensor endpoints and the CAN Transceiver. This component was selected due to that it provides a 1A controlled output as well as provides a 5V output voltage with an accuracy of +/- 4% which is within our requirements. A switching regulator was also chosen for this supply rail due to its high efficiency. This would help mitigate power dissipation as well as there is quite a large voltage difference (7V) from the battery and the 5V rail and with a current load around 1 A, a linear regulator dissipates 7W of power. The schematic of the LM2575HVT-5.0 switching regulator for a fixed 5V output with a max 1A load is shown below:

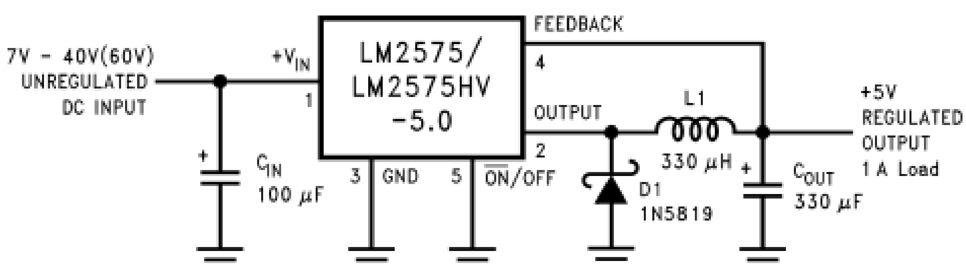


Figure 4: Schematic for the Switching LM2575HVT-5.0 Circuit [4]

From the 5V supply rail a linear regulator specifically the LM3940IS-3.3 [5] manufactured by Texas Instruments will be used to step down 5V to 3.3V to create the 3.3V rail. This rail will be used to power the onboard microcontroller as well as any other onboard components or sensors such as an onboard temperature sensor that may be implemented in the future. This linear regulator was selected as it is inexpensive, has an LDO voltage of only 0.5V and has an accurate output voltage of 3.3V +/- 2%, which is below the requirement of 3.3V +/- 5%. The schematic for the LM3940IS-3.3 linear regulator for a fixed 3.3V output with a max 1A load is shown below:

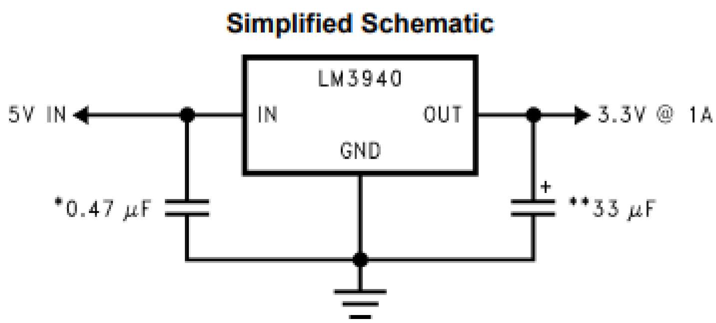


Figure 5: Schematic for the Linear LM3940IS-3.3 Circuit [5]

eFuse Module Design

The eFuse module will handle power distribution from the power regulation module to all the endpoints of the vehicle. The module will allow the MCU to switch power to specific endpoints or groups of endpoints while also providing diagnostic and fault information back to the MCU. Along with delivering power, the module will also be capable of safety features to help protect itself, the endpoints, the car battery, and the rest of the PDM.

An eFuse, or electronic fuse, is an integrated circuit that acts as a power switch while also providing fuse-like overcurrent protection and other electrical protections including over voltage protection, undervoltage lockout, and thermal shutdown in case the device’s internal temperature becomes too high. An eFuse has the advantage over a traditional fuse of having significantly faster short circuit shut off protection, typically in the hundreds of microseconds to hundreds of milliseconds range. [6] This makes the eFuse a more precise, and therefore, safer device in automotive applications. Furthermore, some eFuses have the added benefit of having a hardware programmable current limit which allows for the added flexibility of swapping out endpoints with different current requirements. [7]

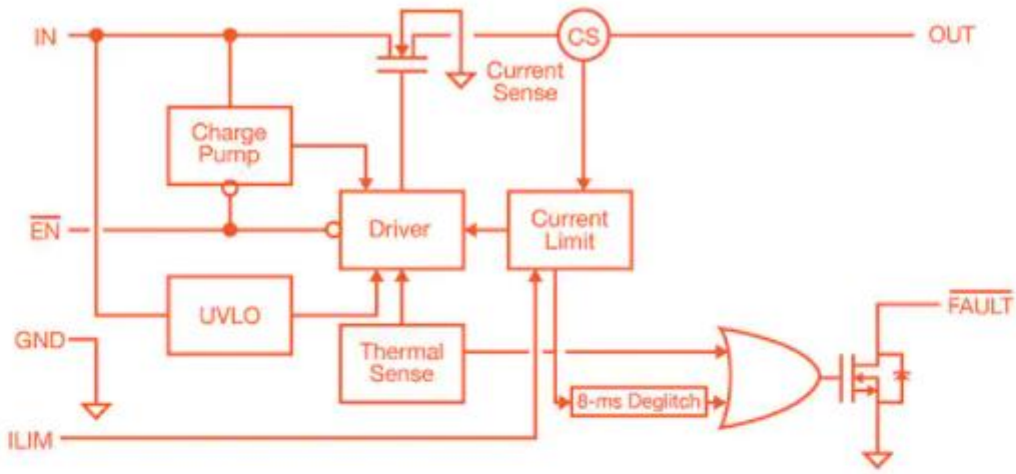


Figure 6: eFuse Internal Schematic [8]

Internally, an eFuse consists of high side switch, typically an N-MOSFET, that switches power on or off. The FET is controlled by an enable (EN) input that connects to an MCU. The enable line control logic activates a driver circuit to power the N-MOSFET with the use of a charge pump that generates a voltage higher than the supply voltage to properly bias the N-MOSFET's V_{gs}. While all eFuses have their own maximum internal current limit, the ILIM input can be used to program a specific decreased limit depending on the application. The remainder of the internals are made up of various protection circuits that are all OR'd to a diagnostic/fault line (FAULT). The Current Limit protection switches off the output if the output current surpasses a certain threshold. The Under-Voltage Lockout protection (UVLO) shuts off the output in the case where the output voltage dips below a safe threshold. Some eFuses also contain overvoltage protection which switches off the output and protects the eFuse in the case of large voltage spikes. The Thermal Sense protection monitors the temperature of the eFuse, and if it exceeds a certain value, shuts off the output to protect the eFuse IC. The diagnostic fault line in some eFuses also outputs a current monitor of the output to track current in real time [8] [6].

The following eFuses have been selected to meet our design requirements to power our endpoints:

BV2HC045EFU-CE2 – Automotive High Side Switch with Variable OCD and OCD Mask Function	
Overcurrent Limit	21A
Voltage	6V to 19V
Outputs	2
Protections	Overvoltage, Overcurrent, Undervoltage, Thermal Shutdown,

Table 5: 6-19V 21A eFuse

Details: Has active clamp to limit inductive load spikes at 48V. Has adjustable OC with an external resistor.

TPS1HA08BQPWPRQ1 – 40-V, 8-mΩ Single-Channel Smart High-Side Switch	
Overcurrent Limit	80A
Voltage	8 - 18V
Outputs	1
Protections	Overvoltage, Overcurrent, Undervoltage, Thermal Shutdown,

Table 6: 8-18V 80A eFuse

Details: Large variety of onboard diagnostics available.

BD2069FJ-MGE2 – 2CH 2.4A Current Limit High Side Switch ICs	
Overcurrent Limit	1.5A
Voltage	2.7V – 5.5V
Outputs	2
Protections	Overvoltage, Over Current, Undervoltage, Thermal Shutdown

Table 7: 2.7-5.5V 1.5A eFuse

Details: Includes a soft start circuit and clamps current output on overcurrent fault instead of shutting off output.

Below is an example circuit of the BV2HC045EFU-CE2 when connected to a load RL and to an MCU. Note that R-GND and D-GND are for reverse battery protection which will not be required in our application since reverse battery protection is implemented upstream in another module. R-OLD is for open load detection which is also unlikely to be needed in our application. C-DLY is used to adjust the overcurrent protection delay.

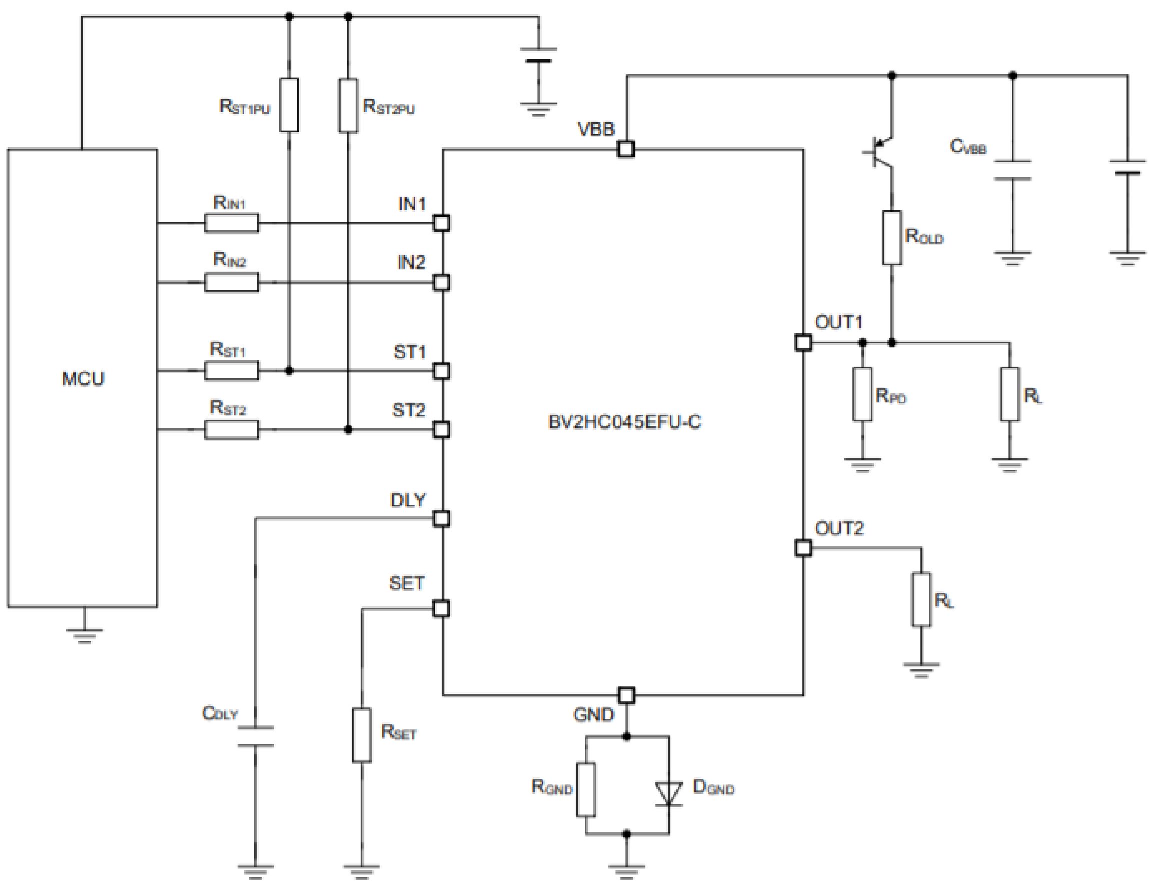


Figure 7: Application Example Schematic for the BV2HC045EFU-C eFuse [9]

MCU Module Design

The MCU module operates as the central hub of the proposed PDM system. It is responsible for ensuring appropriate connections from all necessary devices to the PDM's microcontroller. The microcontroller requires connections to each of the high side drives enable pins and diagnostic pins, battery voltage sensing circuitry and to the CAN transceiver module. The MCU module also includes a multiplexing component to accommodate the numerous analog inputs required, as the typical microcontroller we considered contains only 12-16 analog enabled pins [10], too few for our design. A SWD interface is also a key requirement to enable live debugging of the system, as well as allow for firmware flashing.

At the heart of the MCU module is the actual microcontroller unit. In order to meet the projects requirements with regards to the number of devices being connected, sampling rates, communication protocols, operating conditions, and prototyping concerns, we developed the following list of requirements for our choice of specific microcontroller:

- At least 1 ADC with 12 analog channels.
- At least 1 UART or USART port.
  - For communication with current drive ICs and debugging.
- At least 1 SPI port.
  - For writing data to SD card, keeping in mind that future expansions of this project will require this.
- At least 1 CAN port.
- At least 2 PWM controllers/timers.
- At least 70 I/O pins total.
- DMA controller.
  - Necessary functionality to achieve reasonable speeds when sampling ADC peripherals.
- Able to function in a 100°C environment.
- Come with at least 128KB of storage.
  - To accommodate RTOS and HAL use.
- Free deployment environment and basic peripheral libraries.
- Easily accessible implementation documentation and samples.
- Available in a SMD, non-ball mount package to allow for hand soldering.
- Cost of less than $20 per unit.
- In stock on Digikey, Mouser, or other online electronics store which can ship components in a reasonable timeframe.
- Available on a development board for experimentation and learning purposes.
  - A development board with a microcontroller in the same family is also acceptable.

With the help of the above, we determined that the most appropriate choice of microcontroller would be an STM32 device. We identified several device families within this product line which meet our requirements and wait only to place an order for components before solidifying our exact choice, as MCU availability changes rapidly in current times.

As previously stated, one of the main functions of the MCU module is to control current flow through our high side drive circuitry. This is done by connecting each high side drive's enable pin to its own digital I/O pin on the microcontroller. Since the HSD's are 3.3V CMOS compatible, the microcontroller can simply drive a pin high or low to control the flow of current through any of the output circuits.

Another equally important functionality is the ability to sample the diagnostic pin on each HSD. The diagnostic pin of the HSD produces a voltage proportional to the current flowing through the device, which needs to be measured by the microcontroller's ADC. In order to accommodate the over 20 signals, an external multiplexing component is required. By connecting an analog compatible multiplexer to one of the analog pins on the microcontroller, we can effectively increase our analog pin count. We control the connected multiplexers with at most 3 digital pins, which can be shared with all multiplexers as we can only sample 1 signal at a time. Through our firmware, we can cycle through which signal is being sampled at any moment.

In addition to sampling the current draw from the HSD's, we also monitor the 12V battery which powers all electric components on the car. However, the ADC of the microcontroller uses a 3.3V voltage reference when sampling signals. Thus, we plan to use a voltage dividing circuit to scale the battery's typical operating range of ~9-15V down to measurable levels.

CAN messaging functionality is made possible through the interaction between the microcontrollers onboard CAN peripheral, and the external CAN transceiver module. The microcontroller chosen contains an internal CAN peripheral which handles the upper layers of the CAN protocol stack and exposes 2 pins which we connect to the CAN transceiver module, responsible for connecting to the CAN bus [11].

Certain devices on the car, namely fans and pumps, require a pulse modulated current source for proper operation. This is achieved using the available PWM pins on the microcontroller, which would connect to the enable pins of the specific HSD circuits which power those devices. The PWM pins can be programmed to provide the required signal to operate at the desired conditions.

The SWD interface is accomplished by simply connecting the 5 pins on the microcontroller designated for use in the SWD protocol to 5 headers on the PCB [12]. Then, a user may debug or flash new firmware onto the microcontroller by connecting those 5 header pins to a SWD to USB converter device, which can be connected directly to a computer.

To power the microcontroller, the 12V source battery is regulated to the necessary 3.3V level in the power regulation module, then fed to several of the microcontroller's power supply pins. In order to ensure a smooth power supply, decoupling capacitors are connected to each of the microcontroller's supply pins [13].

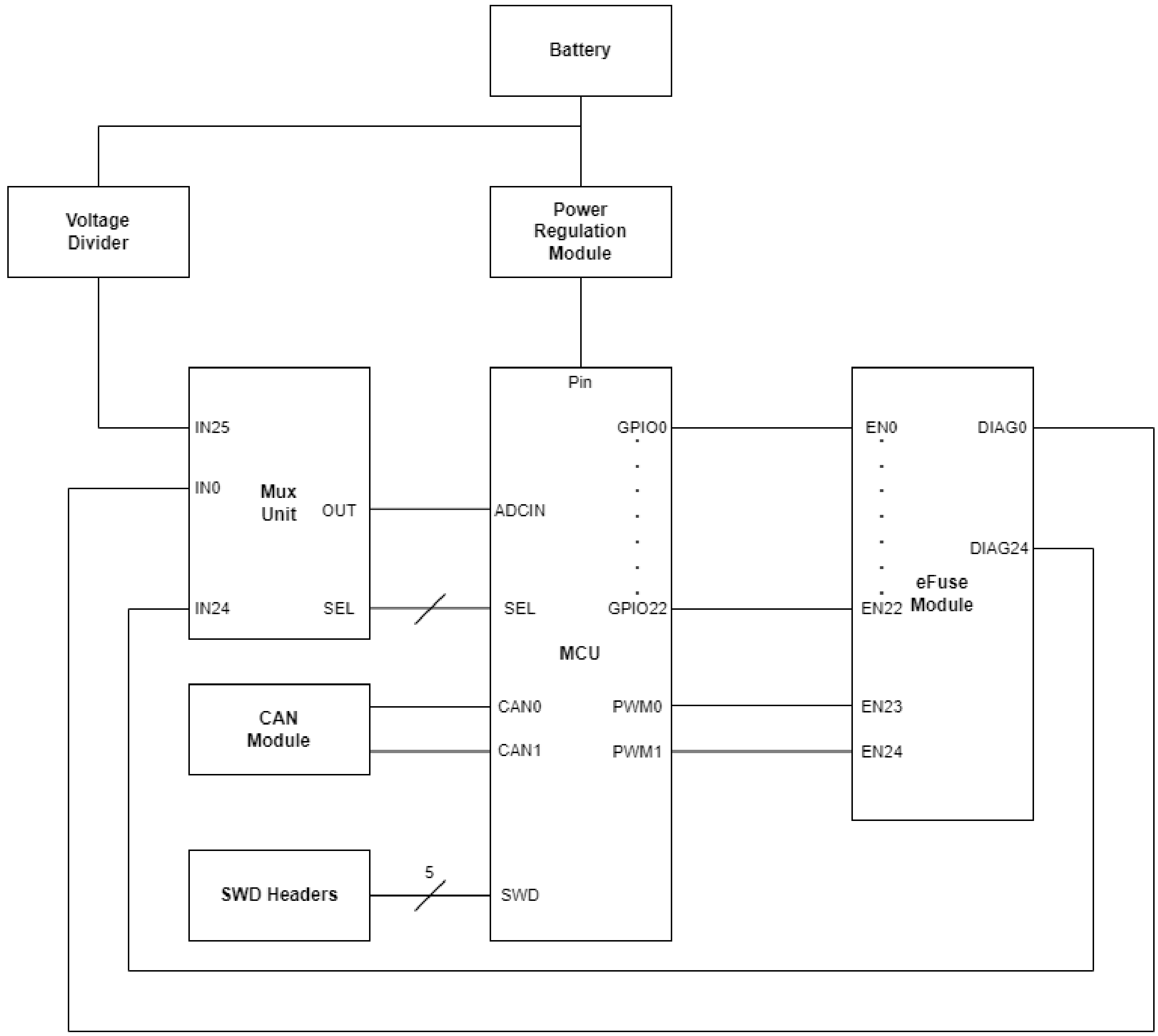


Figure 8: MCU module block diagram

CAN Transceiver Module Design

CAN is a message-based protocol designed to allow devices to communicate with each other without the need for a host computer. The protocol uses a *half-duplex* service scheme with *multicast* transmission. *Half-duplex* communication means that all nodes can communicate with each-other, but not simultaneously [14]. CAN supports *multicast* transmission with functionality to address messages to a group of nodes simultaneously. CAN is often considered a standard communication protocol for automotive applications partly due to historical reasons but also because of the robustness of the protocol [15]. CAN is often considered a standard communication protocol for automotive applications partly due to historical reasons but also because of the robustness of the protocol [15].

The Schulich Racing Team uses CAN as the standard protocol for communication between all the on-board equipment. As such, our design solution requires the ability to both read and send messages over a CAN network. The CAN module is designed to act as an interface between a microcontroller with CAN communication support and the CAN bus (physical layer). The microcontroller will utilize the hardware in the CAN module to reliably send and receive messages. For this solution, the CAN messages broadcast to the network by the MCU module will include fault events, supply voltage values, and current supply levels.

To successfully design a module for facilitating CAN communication, it is important to have a thorough understanding of how the protocol functions. The physical layer in CAN is commonly known as the CAN bus and is a twisted wire pair terminated by a 120 Ω resistor on either end [16]. One of the wires is known as CAN_H and the other is CAN_L. A dominant bit (1-bit) is defined by the CAN_H wire being more than CAN_L + 0.9V. A recessive bit is defined by the CAN_H wire being less than CAN_L + 0.5V [16]. The most common method for creating a dominant bit is to drive a 2V differential across the CAN bus, while a recessive bit is indicated by connecting the CAN_H and CAN_L wires together [16]. Any voltage common to both lines is referred to the common mode voltage. The use of a twisted-pair cable and a differential voltage to indicate bit state causes any external noise to be coupled onto both lines and rejected by the CAN receiver. Additionally, the differential data transmission technique radiates less electromagnetic noise energy than single-ended signals due to the cancellation between the complementary bus signals. The below figures illustrate the topology of a CAN bus and example CAN bus states with 0 common-mode voltage that are being driven by a 5V CAN node.

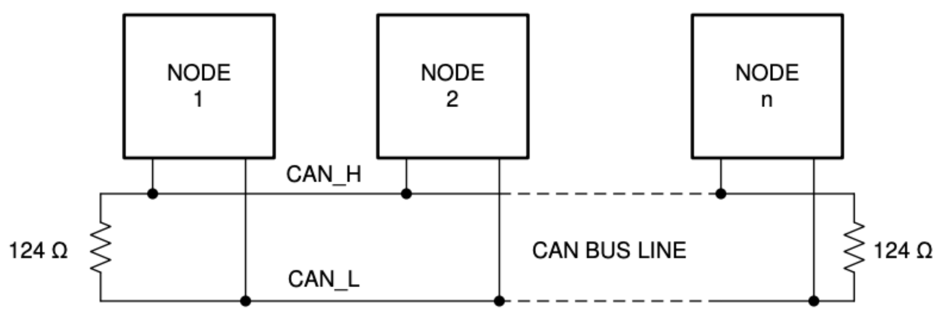


Figure 9: CAN Network Topology

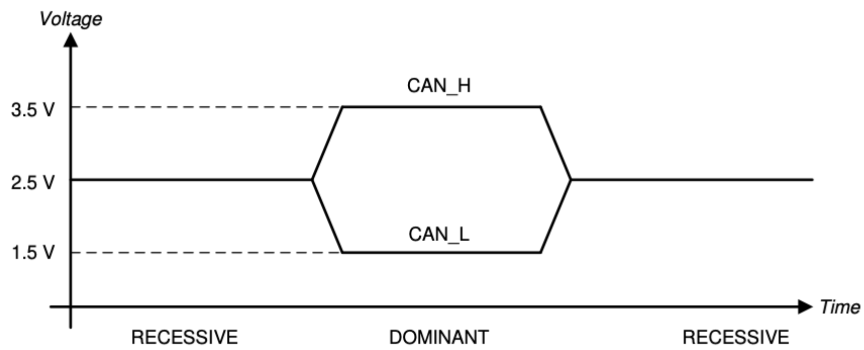


Figure 10: CAN Bus States

The microprocessor family specified in the MCU module has support for CAN communication and implements the data-link layers that handle's functionality such as message acknowledgement, error detection, and packet interpretation [17]. If no microprocessors with CAN support end up being compatible with the final design, ICs are available that implement the data-link layer and can communicate with a different microprocessor via SPI. Special hardware is required to interface the data-link layer with the physical layer as CAN has electrically demanding requirements for maximum voltage tolerance that are typically incompatible with those supported by microcontrollers. For this function, ICs known as CAN transceivers are available to interface the microprocessor with the CAN bus. The transceivers connect to the CAN bus via CAN_H and CAN_L wires and connect to the microprocessor via the CAN_TX and CAN_RX pins.

The transceiver selected for this module is the MCP2551 by Microchip Technology. It is an 8-pin device and is offered both in through-hole and SMD variants which makes it well suited to prototyping and manufacturing a final product. Additionally, it offers many features such as thermal shutdown protection, short-circuit protection, and protection against high-voltage transients. The below diagrams demonstrate the pin out of the MCP2551.

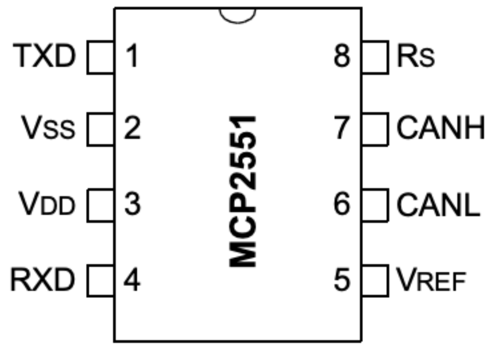


Figure 11: MCP2551 Package

Pin Number	Pin Name	Pin Function
1	TXD	Transmit Data Input
2	V_{SS}	Ground
3	V_{DD}	Supply Voltage
4	RXD	Receive Data Output
5	V_{REF}	Reference Output Voltage
6	CANL	CAN Low-Level Voltage I/O
7	CANH	CAN High-Level Voltage I/O
8	R_S	Slope-Control Input

Figure 12: MCP2551 Pinout

Pins 1 and 4 will connect directly to the CAN_TX and CAN_RX pins from the MCU module. The datasheet specifies a V_{DD} (pin 3) of between 4.5 and 5.5V. This pin will be connected to the output of the 5V regulator from the power regulation module as it will provide a steady current and voltage supply within the ranges specified in the datasheet for the MCP 2551. A decoupling capacitor of 0.1μF will be used between the V_{DD} and V_{SS} (pin 2) pins. Pin 5 will be unused for our application. Pins 6 and 7 will connect directly to the CAN_H and CAN_L wires and will also need to be connected with a 120 Ω resistor if the transceiver is a terminating node for the bus. Finally, R_S (pin 8) will be set to an initial value of 1k Ω, but may need to be adjusted depending on testing results. R_S is used for slope-control mode for reducing electromagnetic interference by limiting the rise and fall times of CAN_H and CAN_L. Implementing this pinout produces the below schematic.

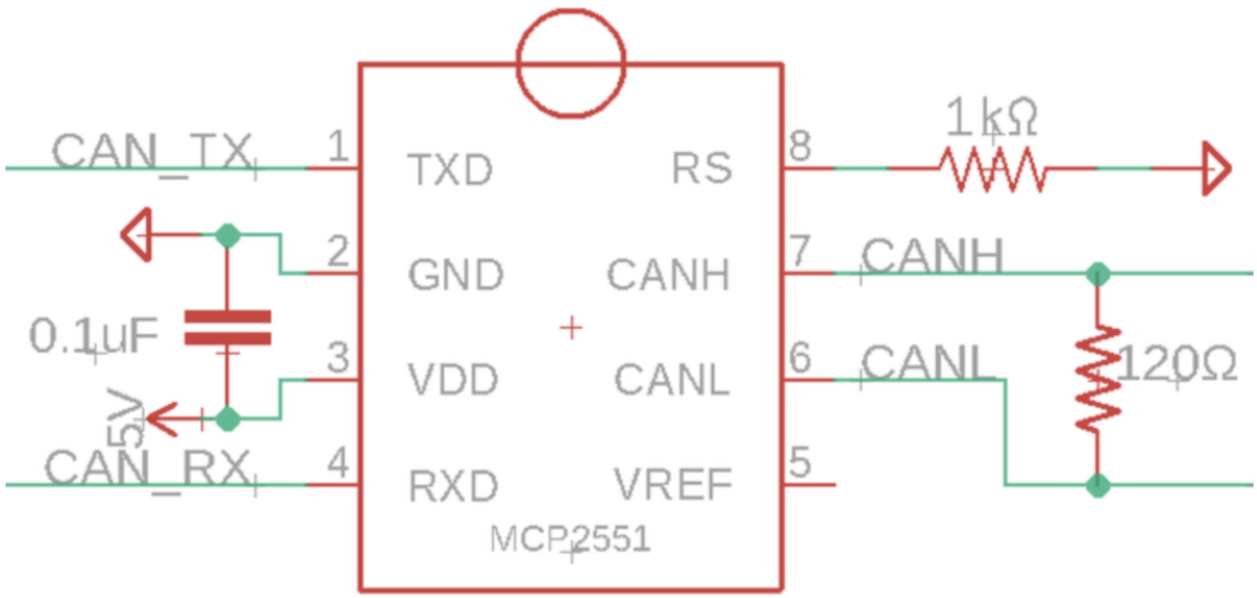


Figure 13: MCP2551 Schematic

CAN transceivers offered by different manufactures have very similar features and the same base functionality (driving and interpreting voltage differentials to/from a CAN bus). The MCP2551 was chosen for this module because of extensive online examples with documentation, our sponsor's prior experience with the device, and compatibility with a 5V supply which will be widely available on the PDM. Future design work will involve finalizing the parameters that will be used in the CAN network such as 11 vs 29-bit ID's, baud rate, and if any additional network layers such as CANopen will be useful. Additionally, a test PCB will be created, and a multitude of hardware and firmware tests will be conducted to verify reliable operation of the CAN module.

Diagnostics Module Design

The diagnostics module encompasses the CAN module (the CAN Transceiver circuit) as well as the battery voltage sensing circuit, which is a voltage divider that will be connected to one of the microcontrollers ADC pins to read the current battery's voltage. In the future other components and circuits may also get added such as a temperature sensor to record the onboard temperature of the PCB to make sure that it is maintaining a sustainable temperature and not over heating. A schematic of the voltage divider circuit that will be used to measure the battery's voltage is shown below:

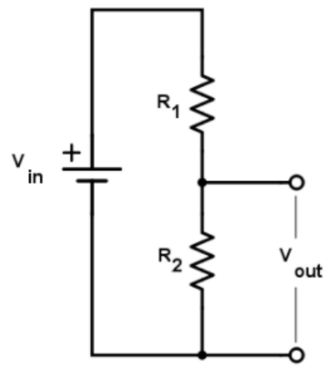


Figure 14: Voltage Divider Schematic

The maximum expected operating voltage of the LFX14L2-BS12 12V battery is approximately 15V so the voltage divider's resistors (R1 and R2) will be calculated based off of the V_{in} value being 15V and the V_{out} voltage being the max voltage the MCU's ADC pin can handle, which is 3.3V. The voltage divider equation used to calculate the R1 and R2 value is shown below:

$$V_{out} = V_{in} * \frac{R_2}{R_1 + R_2}$$

$$V_{out} = 15V * \frac{1.13 * 10^3 \Omega}{5.13 * \ 10^3 \Omega} \approx 3.3V$$

A R1 value of 4 kΩ and an R2 value of 1.13 kΩ were chosen as they produce an output voltage of approximately the max ADC voltage (3.3V) when the battery is at its max operating voltage value. The resistors were chosen to be in kilo ohms as this will greatly minimize the current flow, thus reducing the power dissipation provided from this circuit. The calculation used to determine the max current flow is shown below:

$$I = \frac{V_{in}}{(R_1 + R_2)}$$

$$I = \frac{15V}{(5.13 * 10^3 \Omega)} = 2.924 * 10^{-3} A$$

With the R1 and R2 values that we selected the maximum current draw will be 2.924 mA and the circuit will dissipate a maximum of 43.86 mW which is a negligible amount. The equation used to calculate the power dissipation is shown below:

$$P_{max} = {I_{max}}^2 * (R_1) + {I_{max}}^2 * (R_2)$$

$$P_{max} = (2.924mA)^2 * \left((4 * 10^3\Omega) + (1.13 * 10^3\Omega)\right) = 43.86mW$$

Shutdown Circuit Design

According to the Formula SAE rules, Schulich Racing must implement a shutdown circuit into their electrical system as a safety mechanism for emergency or malfunction situations [18]. Specifically, the shutdown circuit must disconnect power to the Fuel Pumps and Ignition when one of two switches opens the circuit:

- Brake Over Travel Switch (BOTS)
- Cockpit Main Switch

The rules further specify that the shutdown functionality cannot be implemented using programmable logic controllers or other engine control unit functionality – it must be implanted using analog components. Due to this rule, Schulich Racing was not able to use the previous PDM to implement the shutdown circuit, as the PDM outputs were all controlled by programmable logic.

In the past, Schulich Racing has implemented this functionality using two relays as shown below:

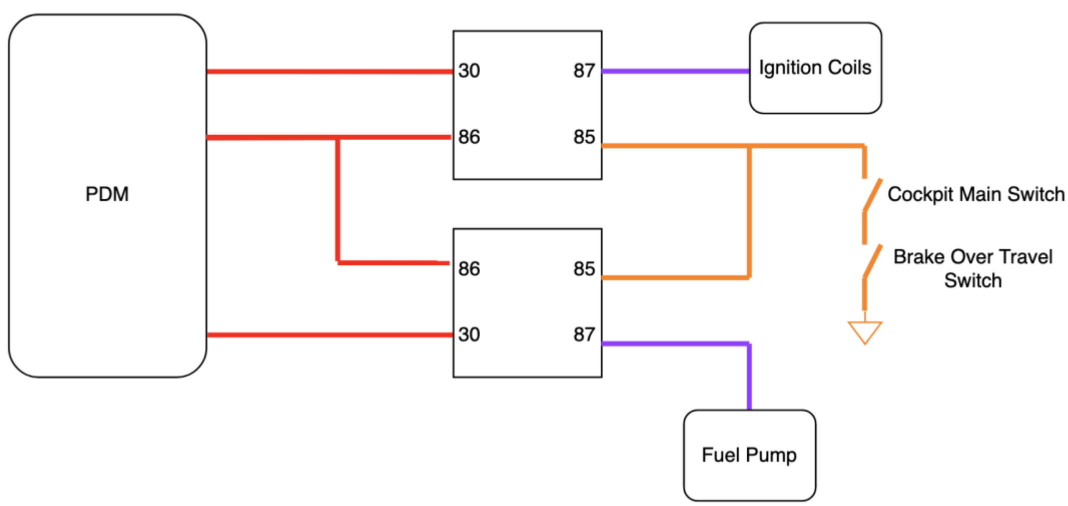


Figure 15: Block diagram of Schulich Racing's previous implementation of the shutdown circuit

However, now that a custom PDM is being built, Schulich Racing is interested in revisiting the idea of implementing the shutdown circuit functionality in the PDM, as the relays for the shutdown circuit are the only relays on the car – the idea of the PDM was to move away from relays and keep the power system compact and centralized.

Looking into a design for the shutdown circuit on this project, MOSFETs were chosen as the switching mechanism due to their compact size and low price, as well as availability. See the below diagram which shows the initial high-level design for the shutdown circuit:

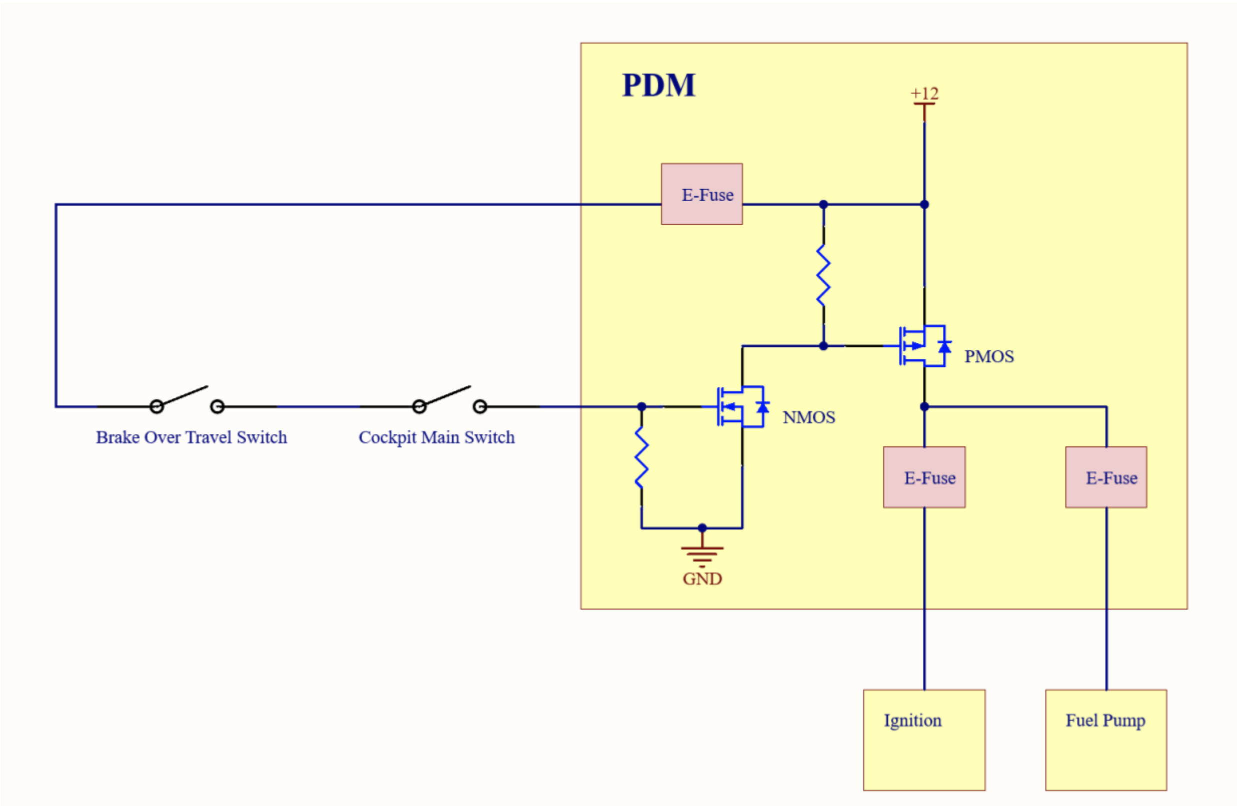


Figure 16: Shutdown Circuit preliminary high-level schematic

A p-channel MOSFET was chosen to switch power to the ignition and fuel pump despite its size and efficiency costs due to the difficulty in using n-channel transistors to power devices from the high-side – with the floating source node it would not be possible to maintain a sufficient V_{GS} when supplying power to both the drain and the gate of the NMOS transistor from the same 12V bus. With the source node of the PMOS transistor connected directly to the 12V bus we can reliably toggle power to the endpoints by pulling the gate low.

While connecting the input from the shutdown switches directly to the PMOS driver and connecting the other end to ground was considered, there is a potential failure mode where the harness could short to ground, permanently latching the PMOS driver on and bypassing the safety functionality of the shutdown switches. Therefore, an NMOS transistor was chosen as an input to the PMOS driver, with the switches opening/closing a loop that connects to the 12V bus.

The Formula SAE rules do not appear to disallow using MOSFETs for the shutdown circuit, but they also are not completely clear that it is legal, so a request was submitted to the Formula SAE Rules Representatives with the above proposed circuit, requesting that they confirm whether it is legal (See Shutdown Circuit FSAE Rules and Clarification Request for details). In the event that it is determined not to be allowed, solid state relays may be considered as an alternative, or the team may decide to revert back to the previous relay solution.

Reverse Polarity Circuit Design

The reverse polarity circuit's main goal is to provide protection to the unit from the battery being connected with its polarity reversed. It is not uncommon for a battery to be accidently connected in reverse when the battery is being changed on the car or during maintenance/testing work on the electronic system when the battery needs to be removed. This circuit will be implemented on the 12V rail right after the connection to the battery to preserve the integrity of the other modules and unit in the case that the battery is connected in reverse as well as the battery itself. There are multiple options to protect a circuit from reverse polarity and the two main ones are using a Schottky Diode or using a PMOS MOSFET. A Schottky diode can placed in series with the power rail to properly disconnect the circuit (blocks the flow of current) when the polarity is reversed, however this circuit is inefficient due to the constant flow of load current through the diode [19] . A more efficient way to protect for reverse polarity is using a PMOS MOSFET which controls the flow of current in an application based on the voltage applied at the terminal's gate which alters the conductivity of the source and drain terminal changing the flow of current. A Zener diode is also used and is connected to the PMOS drain and a 100 Ω resistor to protect the gate from over voltage. A schematic of the reverse polarity circuit with the PMOS MOSFET is shown below and will be the circuit that we will design due to its high efficiency and over-voltage protection on the gate:

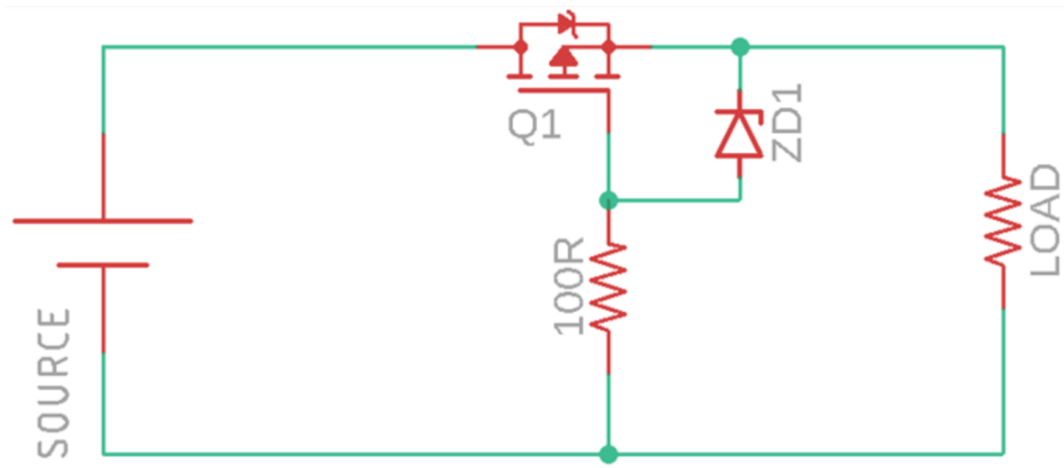


Figure 17: Schematic for a PMOS MOSFET Reverse Polarity Protection Circuit [19]

Firmware Design

The firmware of the PDM system refers to the program flashed onto the central microcontroller within the MCU module. Its responsibilities include configuring the peripherals and pins of the microcontroller, controlling the sampling of the several analog signals on board the PDM, setting pin levels to control the current flow through the HSD circuits, analyzing the diagnostic data from analog signals to determine any necessary action required, broadcasting CAN messages, and utilizing an RTOS to ensure that these functionalities occur in a determinate fashion.

Fundamental to the design of the PDM firmware is the use of CubeMX, the STM32 HAL, and FreeRTOS. CubeMX is a development tool created by STM which enables a user to configuring peripherals and pins on the microcontroller through a visual interface [20]. It generates code for the user which would other have to be done manually, which allows our team to implement more complex functionalities in less time. Complementary to CubeMX is the STM32 HAL which is a library containing members that allow our team to forgo bare metal programming of the microcontroller [21], again saving time and effort. Finally, FreeRTOS provides a basic scheduling framework and other primitive OS functionalities [22]. We plan to use FreeRTOS to implement our goal of achieving a determinate program, so we can guarantee execution of safety and mission critical code at designated times. And while we could implement our own scheduler, using FreeRTOS will again save us time and effort as we develop the PDM system.

The most important responsibility of the firmware is to control when each HSD circuit is permitted to have current flow through it. Enabling current flow is as simple as setting an associated GPIO pin high. However, if all devices on the vehicle were powered up simultaneously, the total inrush current poses risks to the integrity of the PDM system. As a result, a start-up procedure will be implemented such that high power devices, as noted in the appendix, will be switched on in a predetermined, staggered order. Additionally, some devices on the car, namely fans and motors, require a PWM signal in order to control their operating speed. In such cases, the firmware configures designated PDM pins on the microcontroller to pulse at the required duty cycle. The duty cycle of these PWM signals will also need to be adjusted dependent on certain operating conditions, which is trivial.

The firmware must also handle the sampling and analysis of all diagnostic analog signals on the PDM. As one of our requirements is to sample these signals at 20Hz intervals, a routine will be created which would sample all analog signals, and this routine would be executed by the RTOS to occur at the designated rate. In addition to sampling these signals, we must also monitor any fault conditions, which may be battery over/under voltage or a fault in the HSD circuit. The diagnostic lines from the eFuse module are pulled high when the HSD detects a fault, which may indicate several issues. In any case, once a fault is detected, the firmware must execute two tasks. First, a CAN message must be broadcasted to the CAN network for other onboard systems to interpret and log the fault. Although there is no requirement on how quickly we must a detect and report a fault, we plan to execute a routine which looks at previously collected diagnostic data and determines if a fault occurred at a 1Hz interval, again utilizing the RTOS to ensure timely execution. Second, the firmware must initiate a retry routine for the pin that experienced the fault. Depending on the characteristics and criticalness of the endpoint that experienced the fault, these retry routines could include reenabling the eFuse a prespecified number of times or simply not reenabling the eFuse until the PWM is restarted.

To implement communication with the CAN network, the firmware will interface the MCU module with the CAN module. The STM32 HAL library includes functions for configuring CAN network parameters and filters, receiving CAN messages, and preparing CAN packets for broadcasting. Currently, the PDM does not have any need to receive CAN messages, however this functionality will be included in the design of the firmware to support future development and projects. Since the PDM currently does not need to receive any data over CAN, all message filters will be enabled so that the CAN peripheral will filter out all messages before they are sent to CPU.

This will prevent the CPU from spending unnecessary resources processing the multitude of messages that will be present on the CAN network. When implementing the CAN functionality in code there are several reference designs that will be used. First, the sponsor has example code of CAN communication with an STM32 microprocessor which will be extremely relevant. Second, the STM32 Cube IDE has reference files which can be used for further reference. Finally, there are several online resources the team has saved in resources documents for other examples and to help with troubleshooting.

The overall firmware design is demonstrated by the below diagram. Note that certain events such as an eFuse fault or a failed diagnostics check will be implemented with interrupts such that the firmware responds immediately to the event.

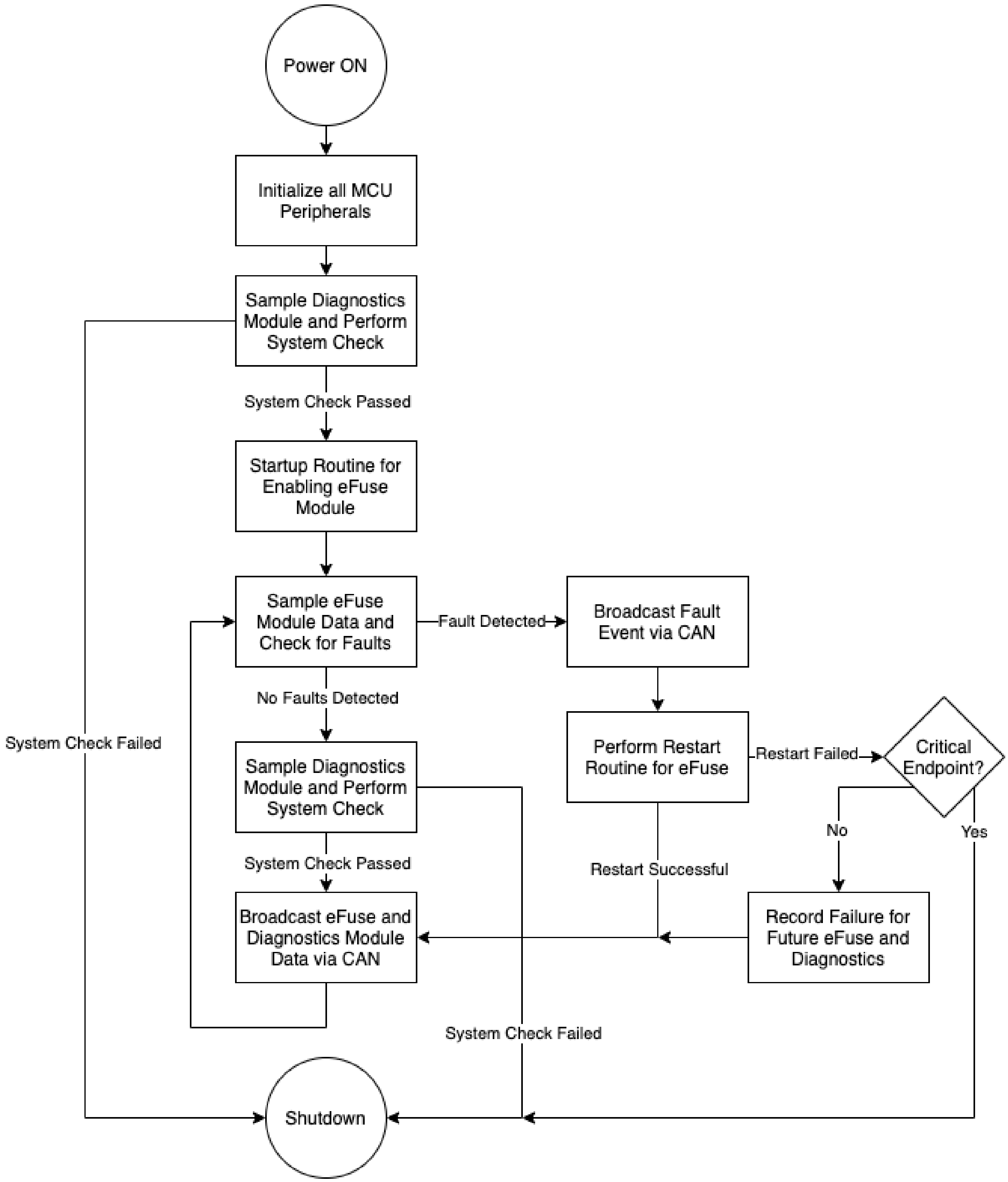


Figure 18: Firmware Design Block Diagram

Alternative Solutions/Methodologies

For our preliminary design there are many different alternatives that we considered for different modules as well as specifically for different components. On the hardware side we looked at different alternatives for the regulators to regulate the batteries voltage down to the appropriate line, different ways to protect the endpoint, different microcontrollers and alternate CAN Transceivers and different ways to implement the shutdown circuit. On the Firmware side we investigated different OS systems for the MCU.

Hardware

Power Regulation Module Alternatives

There were multiple different design decisions and components considered for the power regulation module to create three different supply voltage lines with an accuracy of +/- 5%, have minimal power dissipation (high efficiency) and be rated respectively for the end point’s current draw. The biggest decisions we made regarding the power regulation module was selecting the step-down process for each voltage supply as well as choosing to use a linear regulator over a switching regulator or vice versa for each voltage rail.

Linear regulation uses a linear component like a resistive load to regulate the output and they have their main advantages and disadvantages shown below:

Advantages	Disadvantages
Simple Circuit Design	Poor Efficiency
Require Minimal External Components	High Heat Generation
Low Noise (Accurate Output Voltage)	Only Perform step-down (buck) regulation

Table 8: Linear Regulation Advantages and Disadvantages

Switching regulation on the other hand uses a switching element such as MOSFET to transform the input voltage into a pulsed output voltage which is smoothed by a capacitor and other elements. The switching process is done at high speeds increasing the efficiency of the voltage regulation however in return providing more noise into the system. The main advantages and disadvantages of switching regulation is shown below:

Advantages	Disadvantages
Low Heat Generation	Increased Noise (Less Accurate Output Voltage)
High Efficiency	Complicated Circuit Design
Step-up (boost), stepdown (buck) and both operations are possible	More external components are needed

Table 9: Switching Regulation Advantages and Disadvantages

The decision to use a single buck regulator for the 5V rail was made due to that it is common to find 5V regulators that have a high efficiency as they are used in many different consumer products as well as hand held instruments versus using multiple different regulators to regulate 12V down to 5V such as 12V to 9V, 8V or even 7.5V then down to 5V, which are non-standard/less common regulated voltages [23]. Some advantages however to using an intermediate voltage regulator down to 5V would be that it would make using a linear regulator more feasible for that rail as the voltage difference would be only from 8 or 9V for example to 5V instead of 12V, dissipating less power as well as providing a slightly more accurate voltage output with a less complicated circuit. In our design we also are required to have an 8V power rail, so it already has an 8V line that could be down regulated. However, from using a single buck convertor (switching down regulator) specifically the LM2575HVT-5.0 it greatly minimizes the power dissipation and still provides a fairly accurate 5V +/- 4% [4] which is within our requirements. We are also able to have more flexibility with the current rating of the linear regulator which we choose to be 1 A based on our 5V end points which are mainly sensors that draw minimal current.

For the radio endpoint an alternative design that we considered was using an 8V linear regulator instead of a switching regulator to create the 8V rail. The main reason why a switching regulator was chosen for this rail was due to the high steady state current draw of approximately 1.7 A from the radio when it is transmitting. Even though for most of a race the radio will not be transmitting any messages from the driver, designing the module for an edge case where the driver could be broadcasting longer messages the linear regulator would be dissipating 6.8W of power which for power dissipation greater than about 5 watts an external heat sink should be used [23]. The alternative linear regulator that we looked at using was the NCV7808 [24] which can output a voltage with 1.5% accuracy and requires minimal external components. The switching regulator that we decided on using is the TPS5431 which has a wide input voltage range of 5.5V to 23V, allows for up to 3 A of steady state current, high efficiency (95%) and an adjustable output with an initial accuracy of 1.5% [3]. The regulator also has over current limiting, over voltage protection and thermal shutdown. A different buck switching regulator that we considered using was RICHTEK's RT7274 [25] however it only provided thermal shutdown protection, worse efficiency and had less resources provided in the development of its circuit.

eFuse Module Alternatives

Regarding safely switching power to endpoints with electrical protections, there are generally four approaches to achieving this:

1. Switch and fuse
2. Switch and PTC resistor
3. Discrete eFuse/driver
4. Integrated eFuse chip (our approach)

The classic approach of using a switch (typically a relay but can also be a FET) and fuse can achieve power distribution like our integrated eFuse approach but with fewer protections and decreased control. Fuses are inherently non precise devices, meaning that there is a large margin between its rated value and where it will practically trip, this value also being dependent by how much the current rating is exceeded. The downside to this is that more sensitive electronics may require tighter operating margins are likely to be damaged before the fuse is able to trip, essentially defeating its purpose. Furthermore, because a regular fuse must be reset or replaced manually after tripping, it is not possible to perform software reset attempts on the fly and the act of physically replacing the fuse adds increased maintenance to the vehicle. The other downside is that any other protections such as undervoltage lockout would need to be separately implemented. [6] [26]

PTC (Positive Temperature Coefficient) resistors behave like standard fuses except they can perform automatic overcurrent protection resets. This has the added benefit over standard fuses of not requiring physical intervention to reset them. The downsides being that the reset time is variable, can take several hours, and the PTC resistor becomes increasingly sensitive to trips on each reset. Both downsides make PTC resistors not suitable in our application. Furthermore, PTC resistors, like fuses, require additional protections to be separately implemented. [6]

Discrete eFuses operate very similarly to our integrated eFuses except they are built out from individual electrical components instead of all being packaged in one IC. The advantage of this method is that one can design each eFuse to match the exact operating requirements of the endpoint and system. For more standard devices this approach is typically overkill as existing eFuse ICs on the market can perform all the same functionality for a lower cost, smaller footprint, and decreased complexity for PCB designer. The discrete approach can be very effective for higher current and more unique power distribution requirements that is required by some endpoints. In these situations, there may not be a well sized or cheap IC eFuse equivalent. While the discrete eFuse approach could offer some of the benefits listed above for powering certain endpoints on our PDM, the discrete eFuse approach was ruled out due to the much higher complexity and larger physical footprint required to implement them. [6]

MCU Module Alternatives

The most impactful decision we made with regards to the MCU module was our choice of microcontroller. Considering our requirements, it is highly likely that an STM32 device will turn out to be overly powerful for our application. However, a big factor in our decision making was ease of use as we did not want to spend an unnecessary amount of time learning how to implement our design on a particular platform. The STM32 ecosystem has this in mind as it offers a free end to end suite of development tools with complimentary, thorough documentation. Not to mention how its ubiquity has resulted in a significant developer community which provides a great deal of tutorials, examples, and advice. Other popular microcontroller lineups such as the PIC devices from Microchip, Atmel AVR's, and ESP32 devices would likely have met our hardware requirements just as well as STM32 devices did but developing a prototype on these devices when the team's practical experience is somewhat limited in this field would have been an unnecessary challenge. Additionally, many of these devices are very similar hardware wise and are offered at similar price points.

Our method of expanding our analog sensing ports also presents an opportunity to examine alternative solutions. One potential alternative to the multiplexing method would be to use an external ADC IC. Doing so would take up about as much space as the multiplexers we plan to use, while also allowing the PDM to sample multiple analog signals at once as it works in conjunction with the MCU's ADC, and potentially increase the accuracy of our measurements [27]. The external ADC would communicate to the MCU through SPI or I2C protocols and sample just as fast or even faster than the MCU's own ADC. The only real downsides are the additional costs and complexity when compared to the multiplexing method. However, since we have not identified a need to simultaneously sample multiple signals, do not anticipate exceeding the sampling rate possible with the MCU's ADC, and do not require such high precision measurements, we opted for the simpler and cheaper solution.

Shut Down Circuit Alternatives

Three main alternatives were considered for the shutdown circuit: Using an NMOS transistor as the high-side drive, using a solid-state relay as the switching mechanism, and implementing the shut-down as a low-side drive.

An NMOS transistor was considered for the high side drive due to its inherent benefits over PMOS including power efficiency and small size. However, in order to implement this a charge pump circuit would be required to achieve the V_{GS} necessary to enable the transistor. In the interest of maximizing reliability and limiting complexity, our team decided against this option.

The other way which would allow the team to use an NMOS transistor and obtain the same benefits over PMOS is to implement the switch as a low-side drive. However, due to the architecture of the car we are designing for, it is not possible to intercept the power before it reaches ground as the fuel pump and ignition are immediately grounded to the chassis.

The third alternative is using a solid-state relay. The solid-state relay has benefits in that it is quite reliable and completely decouples the input signal from the output power. However, solid state relays are quite physically large compared to MOSFETS, and especially in this project where size and weight are significant concerns this makes solid-state relays less than ideal for this project. Additionally, solid-state relays are extremely expensive compared to MOSFETs. With the proposed circuit implementation with MOSFETs we will be able to obtain a reliable and sufficiently decoupled circuit, so there is no need for a solid-state relay.

CAN Transceiver Module Alternatives

The most basic implementation of a CAN transceiver is a high-speed buck converter that steps down the CAN bus signals to a voltage suitable for reading by a microcontroller. Since the CAN protocol has defined standards, manufacturers of CAN transceivers all offer very comparable products that implement a high-speed buck converter suitable for microcontroller to CAN voltage conversions. Beyond this basic functionality, manufacturers offer additional features in their transceivers for protection against events like short circuits and other features that support different bus termination schemes. When selecting a suitable transceiver for the CAN module, this meant that all the transceivers available would be suitable to implementing the basic functionality required from the module. Thus, the selection of an ideal transceiver was not constrained by having to select a device that best met the modules functional requirements as all the manufacture's transceivers implemented the basic functionality we required. Finding transceivers that supported niche functionality such as biased split termination for the CAN bus was not necessary for our application. Instead, the selection criteria for a suitable transceiver consisted of the following:

1. Estimated time required to implement
2. Availability of the device
3. Hardware protection features

The ideal CAN transceiver will have the minimum time required to implement successfully, be widely available, and offer a wide array of protection features. The below table summarizes the alternatives and considerations for each:

Transceiver	**Summary**	**Time to Implement**	**Availability**	**Protection**
MCP2551	Microchip transceiver used previously by sponsor. Extensive documentation and examples available online. 5V supply requirement.	Minimal	Mid-low	High
MCP2561	Second generation version of the MCP2551 transceiver with additional features not required for our solution.	Minimal	None	High
XR31235ED	MaxLinear transceiver with low power consumption and 3.3V supply requirement.	Medium	None	Medium

Custom-built	Create our own high-speed buck-converter specially for CAN to microprocessor interfacing.	Very High	High	Very Low

Table 10: CAN Transceiver Alternatives

From the above table, the MCP2551 transceiver is the best alternative. Previous sponsor experience should make this transceiver the easiest to implement and it has a wide variety of protection features that will prevent the transceiver from getting damaged. Due to the worldwide auto shortage, CAN related hardware currently has low availability on any electronic part website. As such, designing a custom transceiver from scratch was considered but this alternative would take a long amount of time and is not guaranteed to work as expected without extensive testing.

Firmware

RTOS vs OS

At the heart of our firmware is the operating system, FreeRTOS. FreeRTOS is one of the de facto standards of embedded RTOS', designed to be small and simple. While FreeRTOS fits the needs of the PDM system, there are several other embedded RTOS' which provide similar functionality and more. However, a key consideration is the cost, time and effort that we would have to put in in order to learn to use an RTOS. As a result, we considered only RTOS' which were free, officially supported by STM32, and had considerable official and community documentation. This lead us to examine FreeRTOS and MbedOS. MbedOS, much like FreeRTOS, provides a ready to use scheduler and OS primitives, but also contains additional libraries which build upon the STM32 HAL, as well as some connectivity and security libraries [28]. Another positive of MbedOS is its integration with Mbed Studio, an accompanying development environment which provides many of the same functions as STM32's development tools. Mbed Studio touts itself as an Arduino-esque tool to accelerate development by simplifying the end-to-end process of programming a microcontroller. However, one of our team members has previously used Mbed Studio in a prior project and found that the Mbed ecosystem is suffering from development pains as it is still in its infancy. This, in addition to the fact that the extra benefits MbedOS has on FreeRTOS would not be overly relevant to the PDM's design, led us to conclude that FreeRTOS was the best choice.

IP and Other Legal Agreements

This project does not have any formal IP contracts that have been signed, however the team has discussed IP ownership with the sponsor and both parties have verbally agreed to the following terms:

1. All hardware paid for by Schulich Racing is the property of Schulich Racing.
2. All IP including firmware, circuit schematics, and system-level designs are the property of Ownin' with Onen with all team members having an equal share of any IP produced.

Should Schulich Racing or Ownin' with Onen have concerns arise regarding IP, these concerns should be discussed as soon as possible between the relevant parties to avoid any conflicts or misunderstandings in the future.

Technical Specifications

Hardware

Specification	**Expected Value**
Output Endpoint Connections	45 outputs
8V Supply Rail Voltage	8V +/- 5%
5V Supply Rail Voltage	5V +/- 5%
Internal Voltage (3.3V Supply Rail)	3.3V +/- 5%
Battery Supply Voltage Measurement	+/- 5%
Reverse Polarity Battery Protection	Yes
Number of 12V Outputs	• 1x 50-80A outputs • 4x 14-20A outputs • 2x 5-12A outputs • 12x >2A outputs
Number of 8V Outputs	• 1x 2-3A outputs
Number of 5V Outputs	• 25x <1A outputs
Endpoint Fault Detection	Yes
Max Overcurrent Trip Time	500ms
Budget for Final Unit	$500

Environmental	Water-resistant
Enclosure Dimension	200mm x 200mm x 50mm
Weight	600g

Table 11: Hardware Specifications and their expected values

Firmware

Specification	**Expected Value**
CAN Communication	The unit should be able to broadcast and also receive messages using the CAN Protocol.
ADC sampling rate	20 samples/second
ADC resolution	8 bits
CAN Broadcast Frequency	20Hz
SWD Interface/Debugging	The unit needs to have the ability to connect over the SWD interface with an external debugger to flash the firmware and debug and troubleshoot the system in real time.
Operating System	RTOS

Table 12: Firmware Specifications and their expected values

Materials, Supplies, Tools and Cost Estimates

Hardware

Material/Tools/Supplies	**Estimated Cost**
PCB manufacturing and Stencils	$100
Electrical Components (Voltage Regulators, Resistors, Capacitors, Inductors, High-side Drives, Temperature Sensor, Connectors, etc)	$300
Microcontrollers & Development board	$80
Enclosure	$100
Soldering Supplies & Equipment	$60
CAN Transceivers	$30
Total	$670 +/- 10%

Table 13: Estimated Costs of Tools, Materials and Supplies

The team expects to develop standalone PCB circuits to test each of the modules independently before creating the final system PCB. This is to test each of the modules first and confirm that they are working before implementing them in the complete system circuit and PCB. This will help to mitigate the need to develop multiple versions of the entire PDM unit, but instead iterate each module until it is working properly. However, the final PDM unit circuit/PCB will be tested thoroughly and also may need iterations, so we account that we may need to create multiple versions as well as manufacture multiple PCBs to have a backup PDM unit.

Material/Supplies	**Estimated Cost**
4-layer PCB and Stencil	$100
MCU	$25
CAN Transciever	$10
Regulators	$25
eFuses	$90
Connectors	$40
Enclosure	$100
Supporting Electrical Components (Resistors, Capacitors, Inductors, Diodes, etc)	$80
Total	$470 +/- 10%

Table 14: Estimated Cost of the Final PDM Unit

The team has created an estimated bill of materials above with their price for the final standalone PDM unit, however in the future this could vary drastically based on any changes in the components selected due to the component shortages or changes in the design. Also, the prices of components have been increasing during the COVID-19 pandemic, so the cost of some of these components or alternatives could go up in the future. In general, the hardware component of the project will consume the most resources. Since the software side does not require much capital expenditure to complete the project.

Software

The software development team does not expect much expenditure as most of the deliverables require only time and access to a computer and specific open-source and licensed development software. For this project the open-source and licensed tools that will be used will be STM32 Cube IDE for the development of the firmware and pinout of the MCU, GitHub for version control of the software, Altium for PCB design which is provided under a license from the Schulich Formula Racing Team and JIRA and Microsoft Teams for project management and team communication, which is free to use under a student license.

Risk and Risk Mitigation Plans

Hardware

Some potential risks that are imposed by the current situation on our project deliverables is that our team may not be able to develop or perform extensive electrically testing on the hardware due to restrictive access to lab space and lab equipment as well as component shortages and material delays caused by the worldwide supply chain shortage during the pandemic.

To mitigate the lab access risk, the team has agreed that they are prepared to purchase any required lab equipment that may be necessary to complete the development and tests of hardware. Also, the Schulich racing team has agreed to allow us to use any of their lab equipment to develop and test our prototype. This gives us access to a power supply, a soldering station, a makeshift reflow oven, multimeter, and other testing equipment. The Schulich racing team has also given us permission to use this equipment at home in case access to the University and lab spaces becomes restricted.

To mitigate the supply chain issues, we will use websites such as trustedparts.com, to ensure we are sourcing components from distributors where they are in stock. Additionally, during the component selection phase we will ensure we select components that are relatively common where applicable and widely available. Also, we plan on being proactive in the prototyping of our hardware design by ordering electronic components early. Furthermore, we will make sure to order additional parts in case there is an issue with a specific component or there is a shortage of parts in the future. If we are unable to secure the specific components that we need due to the shortage such as for the appropriate microcontroller or eFuses as we have witnessed that these are in high demand, we plan to find alternatives wherever we can. To find usable alternatives we plan on looking at through-hole components instead of SMD components as well as over specifying parts.

If we are unable to perform any hardware prototyping due to that pandemic our team will ensure to demonstrate our design by showing a 3D model of our PCB design populated with the respective

PCB components that we plan on using as well as show the different circuit schematics and simulations for each module using circuit simulation software such as EasyEDA, LTSpice or Pspice.

Another potential risk is the manufacturing process of the PCBs as the PCBs espcially the final PCB will be quite complex and involve soldering tons of tiny components including capacitors and resistors, which may have soldering pads underneath the component. Many of the onboard components are very small and heat sensitive, so to mitigate this risk and simplify the process, the team is allowing us to use their custom reflow oven. We will also purchase a stencil and use it to reflow the on side of the PCB that includes the MCU, voltage regulators, eFuses, and other components to make sure that they are soldered in proper place.

Another huge and arguably the largest risk in the hardware design process is design errors. Unlike software, hardware errors are difficult to correct quickly and at a minimal expense. Creating and assembling prototypes can take up to multiple weeks due to the shipment time of components and PCBs, thus limiting the number of iterations in the hardware development that are realistically feasible within the Capstone time frame. To mitigate this risk our team plans on ordering and testing components/PCBs early and spending a significant amount of time in the planning phase, properly researching into and selecting components and creating the schematics to hopefully minimize the number of iterations needed in the hardware prototyping.

Software

Some potential risks that could impact the software development is that each member may not have access to each of the different standalone module PCBs as well as the final PCB module. To mitigate this risk and ensure that each team member working on the firmware development has access to a microcontroller and electrical components needed for testing, development boards with be purchased for the specific MCU that we plan to use. The development board will allow each user the ability to connect different peripherals as well as properly debug and run their code on the MCU.

To make sure the software team is progressing and completing different deadlines a minimum viable product (MVP) has been decided based on the must-have requirements to reduce the scope of the project and make it manageable for the team. If time permits, the team can keep working on other ideas that were not included in the MVP. During development, each developer will be utilizing GitHub to properly keep track and combine the changes that they are making, to ensure that their code will not be interfering with one another.

If our team is unable to perform any hardware prototyping due to that pandemic our team will ensure to demonstrate our firmware by simulating the software environment through the use of a development kit and LEDs and other indicators to display different control signals and changes to the unit.

Major Technical Tasks and Milestones

The major remaining technical milestones for the project have been grouped into these categories:

1. Hardware
2. Software Development
3. Enclosure

The remaining technical milestones are:

- **Hardware**
    - Power Regulation Module
        - Power Regulation Module Schematic designed in Altium
        - Power Regulation Module PCB created
        - Power Regulation Module Components Ordered
        - Power Regulation Module PCB Assembled Tested and Verified
    - Shutdown Circuit
        - Shutdown Circuit Schematic designed in Altium
        - Shutdown PCB created
        - Shutdown Circuit Components Ordered
        - Shutdown Circuit PCB Assembled Tested and Verified
    - Reverse Polarity Circuit
        - Reverse Polarity Circuit Schematic designed in Altium
        - Reverse Polarity PCB created
        - Reverse Polarity Circuit Components Ordered
        - Reverse Polarity PCB Assembled Tested and Verified

- eFuse Module
  - eFuse Module Circuit Schematic designed in Altium
  - eFuse Module PCB created
  - eFuse Module Components Ordered
  - eFuse Module Assembled Tested and Verified
- CAN Transceiver
  - CAN Transceiver Circuit Schematic designed in Altium
  - CAN Transceiver PCB created
  - CAN Transceiver Components Ordered
  - CAN Transceiver Assembled Tested and Verified
- MCU
  - MCU Circuit Schematic designed in Altium
  - MCU PCB created
  - MCU Components Ordered
  - MCU PCB Assembled Tested and Verified
- Diagnostics Module
  - Diagnostics Module Circuit Schematic designed in Altium
  - Diagnostics Module PCB created
  - Diagnostics Module Circuit Components Ordered
  - Diagnostics Module PCB Assembled Tested and Verified
- Completion of modular prototype PDM unit
- Final Component Selection and Ordering
- Design of Final PDM unit schematic
- Final PCB assembled and tested and validated

- **Firmware Development**
  - Scheduler implementation (setup and boiler plate)
  - ADC sampling schedule
  - Detection of faults in ADC data
  - CAN transmission of ADC data
  - PWM generation
  - Enabling/disabling of HSD’s

    - HSD start up procedure
- **Enclosure**
    - Design of Enclosure
    - 3D printing and manufacturing of the prototype enclosure
    - Final Enclosure manufacturing
    - Enclosure Testing and Validation

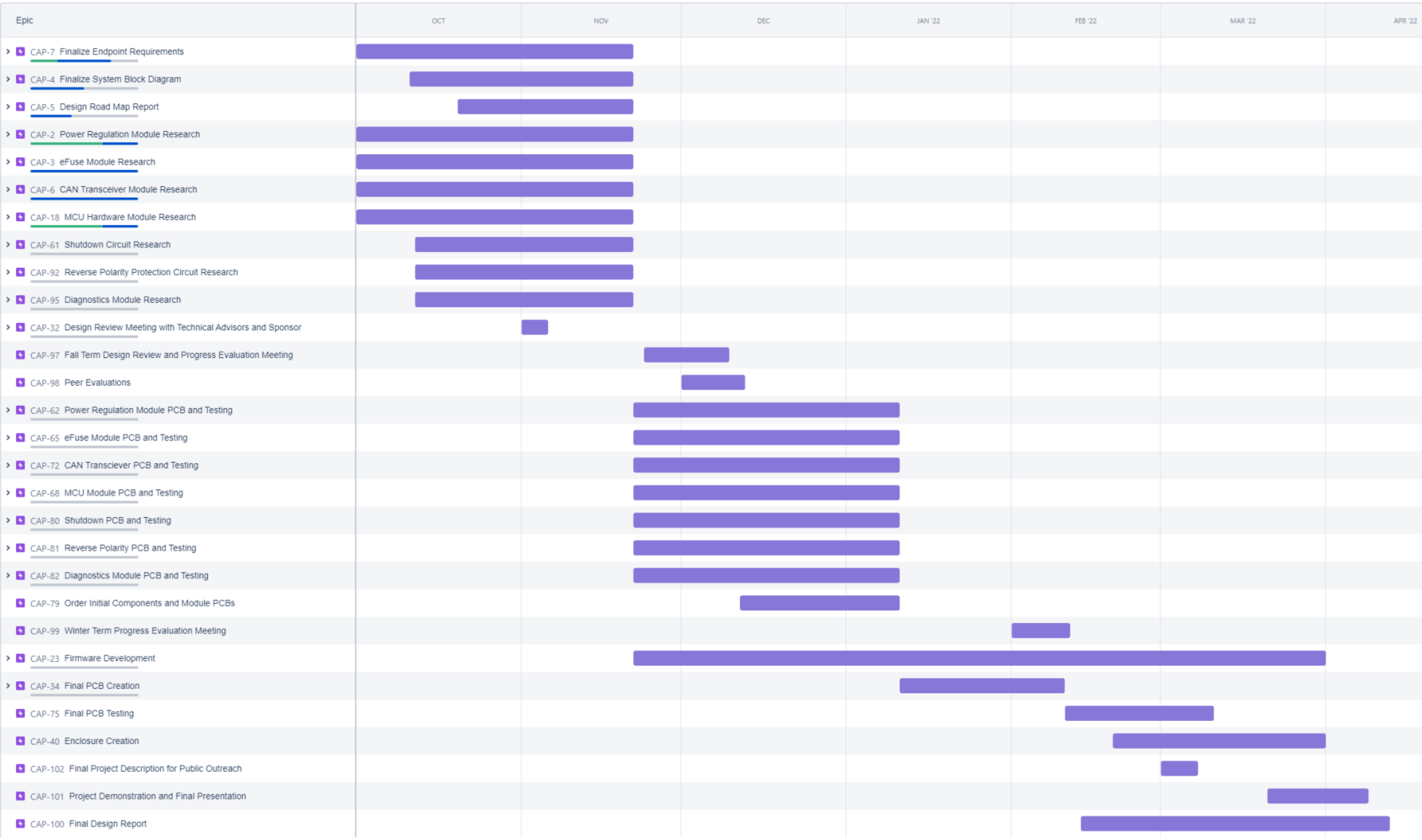


Figure 19: Gantt chart showing the Major Milestones and the Project Timeline

Roles and Responsibilities

Aidan

Aidan will be the project manager for the duration of the project which encompasses keeping the project and team on track through ensuring that each member has the resources and support to be successful. The project manager will also be responsible for keeping the team accountability for decisions and completing tasks. Aidan will also be helping in specifically the hardware design for the project as he has been working on the power regulation and diagnostics module.

Dylan

Dylan is the testing and validation lead and is responsible for overseeing all validation tests for the hardware and firmware systems. The testing and validation lead will work closely with each hardware module and with the firmware lead to ensure that all tests have been completed successfully and that the final product meets the requirements specified by the sponsor. If any tests are failed, the testing and validation lead should provide recommended fixes to the rest of the team and provide a detailed report of any troubleshooting conducted to try and resolve the issue.

Tyler

Tyler is the systems architecture lead, responsible for leading system-level design and ensuring that all aspects of the system are designed correctly to work together and ultimately meet the requirements for Schulich Racing. He is leading communications with the sponsor to clarify requirements and testing to ensure that what we design is compatible and optimal for the components the PDM interacts with. Additionally, Tyler will lead PCB design, setting up the work environment in Altium and guiding and directing PCB development and layout in order to ensure a cohesive system.

Adam

Adam is the hardware lead responsible for all circuit design deliverables and deadlines. The hardware lead will be responsible for tracking the progress of all the circuits and ensuring that they are all following a unified designs scheme. Having a unified design scheme will be integral in ensuring that all the individual hardware modules work together correctly. Furthermore, the hardware lead will run design reviews of all schematics to allow for the team to collaborate on circuit designs and help catch errors.

Guillaume

Guillaume has been designated as the firmware lead which encompasses the entirety of the code dedicated to ensuring the successful operation of the microcontroller unit in its interactions with the remainder of the PDM system. He is also responsible for programming any test functions which may be required by the remainder of the team to test components which fall within their domain of responsibilities. Additionally, Guillaume is designated as a support for the technical lead which is responsible for devising the overall technical vision of the project, which includes areas such as requirement determinations, PCB design and system level design.

RACI

R	Responsible
A	Accountable
C	Consulted
I	Informed

Table 15: RACI Chart Legend

Project Tasks	**Aidan**	**Tyler**	**Dylan**	**Adam**	**Guillaume**
Consult and Finalize Endpoint Requirements	C, I	R, A	C, I	C, I	C, I
Hardware System Architecture Design	R	A, C	C, I	C	C
Power Shutdown Circuit Design (Component Selection and Schematic)	C	R	I	A	I
Power Shutdown Circuit Tests	C	R	A	I	C
Power Regulation Circuit Design	R	C	I	A	I, C

(Component Selection and Schematics)					
Power Regulation Circuit Tests	R	C, I	A	C, I	C
eFuses Module Circuit Design (Component Selection and Schematics)	C	C	I	R, A	C
eFuses Module Circuit Tests	C, I	I	A	R	C
MCU Circuit Design (Component Selection and Schematics)	C, I	I	C	A	R
MCU Circuit Tests	C, I	C	A	C	R
CAN Transceiver Circuit Design (Component Selection and Schematics)	I	I	R	A	C, A
CAN Transceiver Circuit Tests	I	I	R. A	C	A
Firmware System Architecture Design	C, I	I	R	I	R, A
CAN Firmware Development	C, I	I	R	I	A, C
eFUSE Control Firmware	C, I	I	C, R	C	A, R
Pinout and Firmware Data Collection	C, I	C	R	C	A
Firmware Validation	C	C	R, A	C	A, R

Enclosure Design	C	R	R, A	R	I
Final PCB Design	C	R	C	A, R	I
Final PCB Assembly	C	R	C	A, R	I

Table 16: RACI Chart for the Team

Measures of Success and Validation Tests

Hardware

Power Regulation Module

The power regulation module must produce a: 5V, 8V and 3.3V rail with a +/ 5% accuracy especially for the 3.3V rail specifically powering the on-board components such as the microcontroller. To test each of the regulators the following tests will be done:

1. Voltage Rail Accuracy Test: For each produced voltage rail from their respective regulator circuit, they will be tested with a multimeter/oscilloscope on a standalone PCB connected to an electronic or normal load to confirm that the output voltage is within the +/- 5% range for different loads.
2. Current Draw Test: For each regulator circuit, the current draw will also be measured when a load or electronic load is connected to it, to validate that the regulator limits the proper current amount, and also provide enough current for all the endpoints.
3. Thermal Test: Each circuit and their connections (traces) will also be thermally tested to make sure that power dissipation is being properly addressed and is minimal. This will be tested for by measuring the temperature of the regulator as well as the PCB board while it is connected to the end points or electronic load.

eFuse Module

1. Overcurrent Test: For each eFuse circuit, connect its output to an electronic load (eLoad). Set the eLoad to the eFuse’s max trip threshold and run 10 trip tests at ambient room temperature. The eFuse should trip every time within the maximum expected trip time
2. Undervoltage Test: For each eFuse circuit, connect the supply voltage to a variable power supply. Set the power supply to the expected voltage for that eFuse circuit, and then begin lowering the voltage until reaching the under voltage lock out protection threshold. Check that at this threshold the eFuse shuts off and reports a fault.

3. Overvoltage Test: For applicable eFuses that support overvoltage, connect the output to a variable power supply. Raise the voltage from the expected voltage for that circuit until it reaches the overvoltage protection threshold. Ensure that eFuse shuts off and reports a fault at this threshold.
4. Current Monitor Test: For each eFuse circuit with current monitoring, connect the output to an eLoad and set the eLoad to draw current at the expected steady state value for that circuit. Read the output of the current monitor pin and ensure it matches the eLoad current within the error margins of that eFuse.

MCU Module

1. Power supply stability: The MCU is to be powered by a 3.3V source which may be slightly noisy. We must probe the power pins of the MCU to ensure that the voltage supply remains at this level +/-2% with the use of the decoupling capacitors.
2. SWD/JTAG interface: The MCU modules SWD interface must be able to connect and interface with an external computer using a SWD to USB converter device. The connected computer must be able to debug and flash new firmware onto the connected microcontroller.
3. Multiplexing unit: The multiplexing unit must be able to reliably switch between analog signals with minimal signal loss. The input and output of the multiplexing unit may be probed to analyze both signals on a signal oscilloscope. The multiplexing unit must respond to changes on its select pins.
4. Battery voltage scaling: The voltage divider circuit from the battery to the MCU must scale down the battery voltage signal to a level which allows the MCU's ADC to detect changes in its strength. The ADC reference voltage is 3.3V, thus the maximum typical battery voltage once scaled by the divider circuit must remain below this threshold. Conversely, the minimum typical battery voltage once scaled must be low enough for the ADC to detect meaningful changes in the battery voltage.

CAN Transceiver

1. CAN Message Sending: The CAN module must be able to reliably convert CAN signals from the MCU module to a CAN signal that meets the requirements of the CAN bus. To validate this operation, a signal with the same characteristics of the MCU's CAN_TX and

CAN_RX will be applied to the transceiver. An oscilloscope will be connected to the CAN_L and CAN_H lines to verify the transceiver is able to convert the CAN_TX and CAN_RX to a 2V differential on CAN_L and CAN_H for dominant bits and a 0V differential for recessive bits.

2. CAN Message Receipt: The CAN module must be able to reliably down-convert a signal from the CAN bus to a signal interpretable by the MCU module for reading via pins CAN_RX and CAN_TX. To validate this functionality, a square wave with a 2V amplitude and a frequency matching the CAN network's baud rate will be applied to the CAN_H and CAN_L pins. An oscilloscope will be connected to CAN_RX and CAN_RX to verify that the transceiver is able to down-convert the square wave into a signal readable by the MCU module.
3. Optimal R_s Resistor Selection: To minimize the level of noise on the CAN bus, the MCP2551 has a pin to connect a resistor. The level of resistance across this pin varies the slew rate which in turn varies the rise and fall times of CAN_H and CAN_L. A 2V square wave with a frequency equal to the baud rate of the CAN network will be applied to the CAN_L and CAN_H lines. An oscilloscope will be connected to both the CAN_H/CAN_L wires as well as the CAN_TX and CAN_RX pins. R_s will be varied from 1k and 10k and the resistor value that minimizes noise detected by the oscilloscope will be used in the final design.

Shutdown Circuit

At its core, the shutdown circuit only needs to do 2 things:

1. When the shutdown circuit is open, allow no power to pass to the ignition or fuel pump
2. When the shutdown circuit is closed, allow power to pass to the ignition and fuel pump

To test the basic functionality of the shutdown circuit, we need to assemble the circuit, including power and the ignition and fuel pump, and then observe its behavior in the above 2 conditions measuring voltage at the output of the PDM.

Additionally, efficiency/reliability testing is required to ensure that the circuit does not overheat or consume too much power. To test this, we will run the fuel pump and ignition or a load of equivalent power draw while measuring the temperature until the temperature reaches a steady state. At this point we will record the temperature, verifying it is below 150ºC, and the voltage drop across the PMOS transistor, ensuring it is less than 0.5V.

COVID-19 Hardware Testing Backup Plan

In the event the COVID-19 pandemic results in restricted access to the University of Calgary lab equipment, the team has developed a backup plan for performing hardware testing and validation. This plan has been discussed with the project sponsor and is in agreeance with the plan. For details on the backup plan please refer to the Risk Mitigation Plan section.

Firmware

1. HSD enable pin testing: The firmware must be able to set HSD connected GPIO pins high in order to enable current flow through the HSD's.
2. PWM generation: A PWM signal must be able to be generated with a specified duty cycle. An oscilloscope should be used to verify that the expected waveform is generated.
3. ADC sampling: Analog signals entering the ADC port must be able to be sampled and stored in memory at a rate of 20Hz with the required resolution. A reference signal may be fed into the microcontroller to simulate a diagnostic signal and ensure that the collected data matches closely with the reference.
4. Multiplexer control: GPIO pins must be able to interface with the multiplexer unit select pins in order to switch analog signal can be sampled at any one time. This test should include verifying that the firmware can switch the signal being sampled, then sample the signal with minimal delay.
5. CAN messaging: A CAN message should be able to be sent using the on-board CAN peripheral of the microcontroller in combination with the CAN transceiver.
6. Startup procedure: Using the RTOS scheduler, the microcontroller should be able to stagger the enabling of specified HSD circuits.
7. CAN messaging routine: Using the RTOS scheduler, the microcontroller should be able to execute a routine which sends out all diagnostic information at a rate of 20Hz.
8. ADC sampling routine: Using the RTOS scheduler, the microcontroller should be able to execute a routine which samples and stores all connected analog signals. The routine should utilize the multiplexer unit to switch the signal being sampled. This routine should execute at a rate of 20Hz. For this test, it is important to ensure that the storing of data is

quick enough such that our sampling speed is not affected, nor that other routines may have their execution time delayed significantly.

9. Data analysis routine: Using the RTOS scheduler, the firmware should be able to observe previously collected data and detect fault conditions. Fault conditions are typically marked by if a signal is above or below a certain threshold. This routine should be able to execute at a rate of 1Hz and be quick enough to not impede the execution of other routines.

The Team Logbook

The team maintains a logbook through multiple different sources:

1. The team keeps a log of each team members updates and results achieved after every meeting and workday through a OneNote notebook. Team members are also able to log their hours they spent working on specific tasks.
2. The team meets with the sponsor representative on a weekly basis on Saturdays at noon and all-important updates are logged as well through and distributed to the team after each meeting.
3. Each team member has a seat in the sponsor's GitHub team, all commits, pull requests, and other data tracked by GitHub is available to the Sponsor and the team.
4. Each team member has a seat in the sponsor's Jira account for project management. The team uses a kanban board to track what is started, in progress, and finished.
5. Microsoft Teams is used with specific channels to keep an organized record of historical messages.

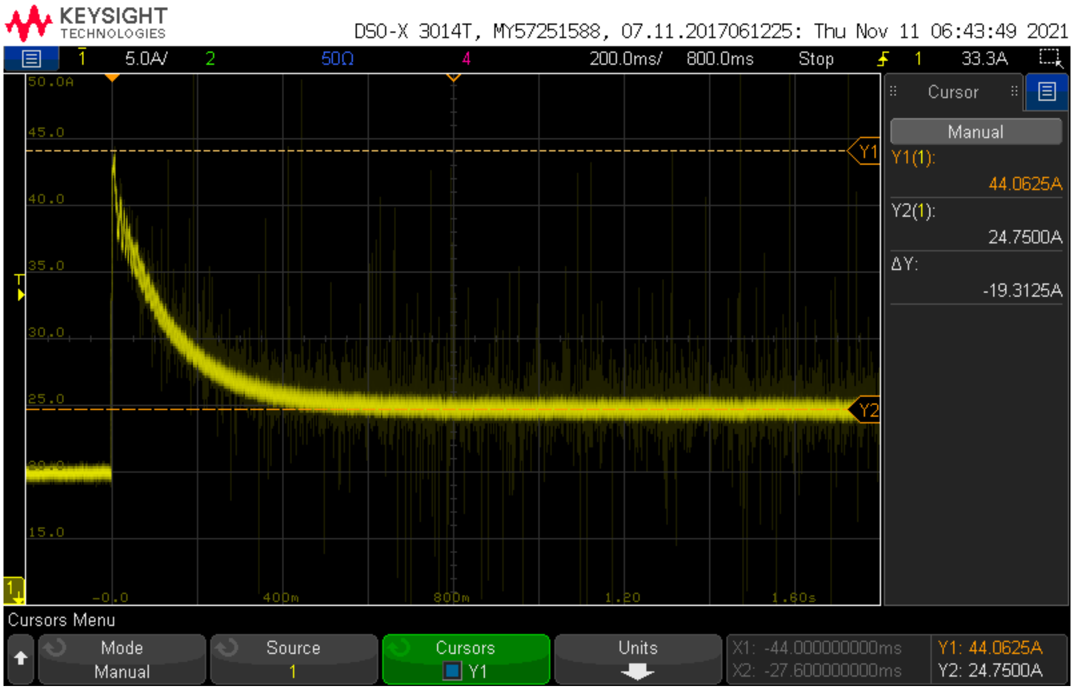
Appendix

Appendix A: High Power Device Oscilloscope Current Measurements

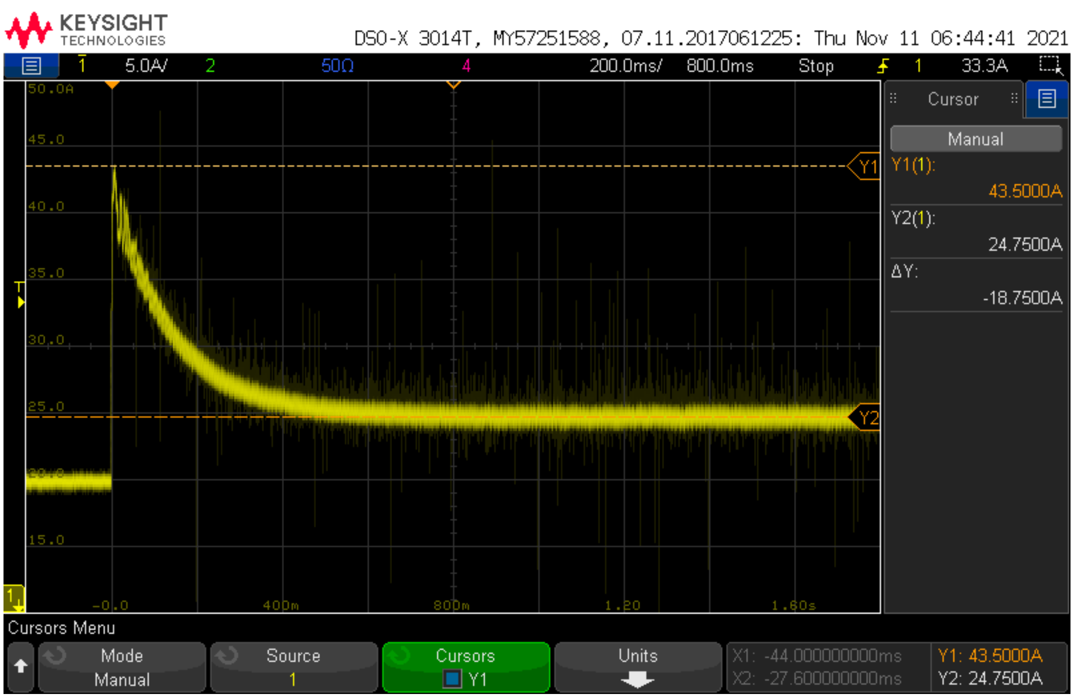
See below for the oscilloscope traces from the current measurements our team took of the fans and pumps Schulich Racing is considering putting on the car. Schulich Racing requested that we take measurements of multiple fans to get an idea of what the power draw is to inform their vehicle/cooling design and based on our measurements they are currently planning to use the SPAL 5.2" and the Mishimoto 8" fans (2 fans total).

SPAL 5.2" Fan

Face down on table

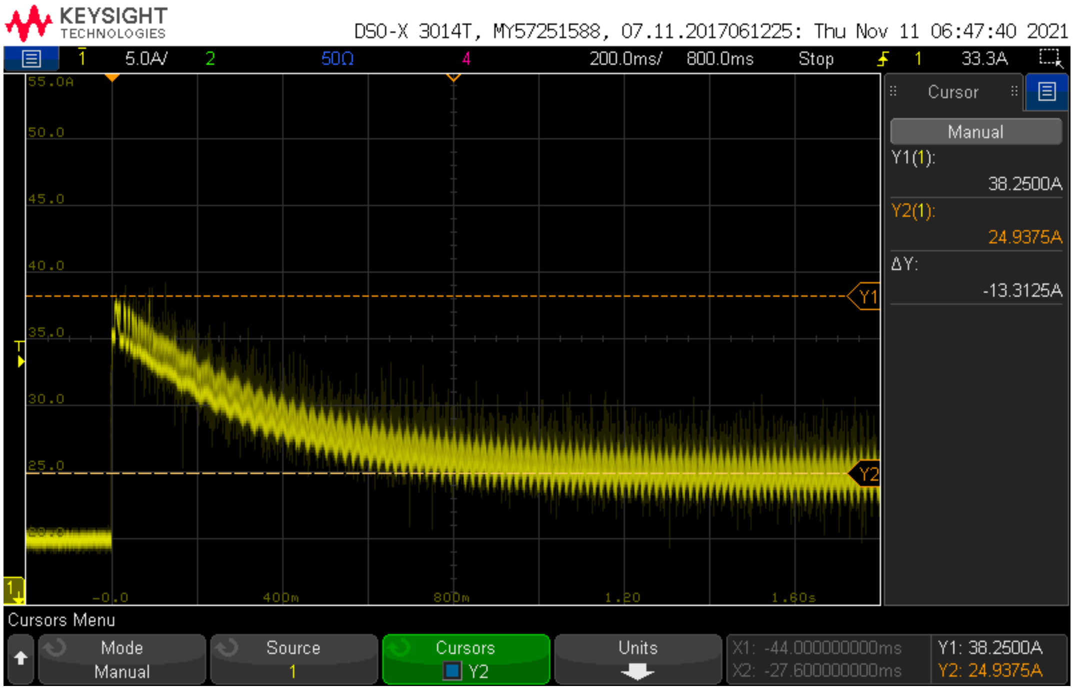


In open air

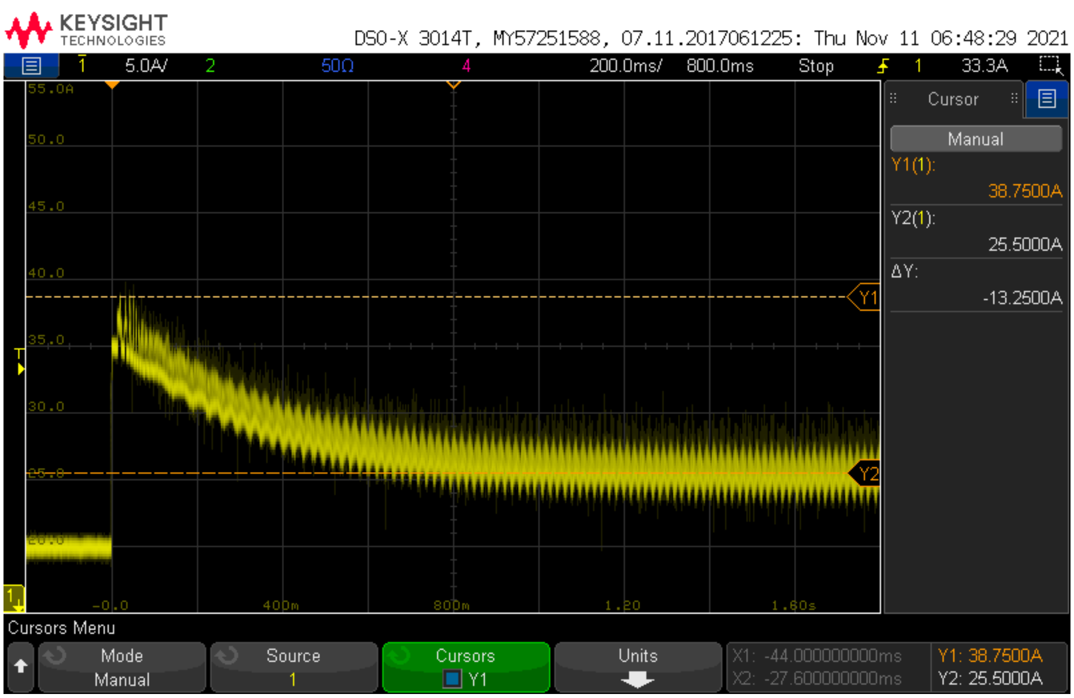


Mishimoto 8" Fan

Face down on table

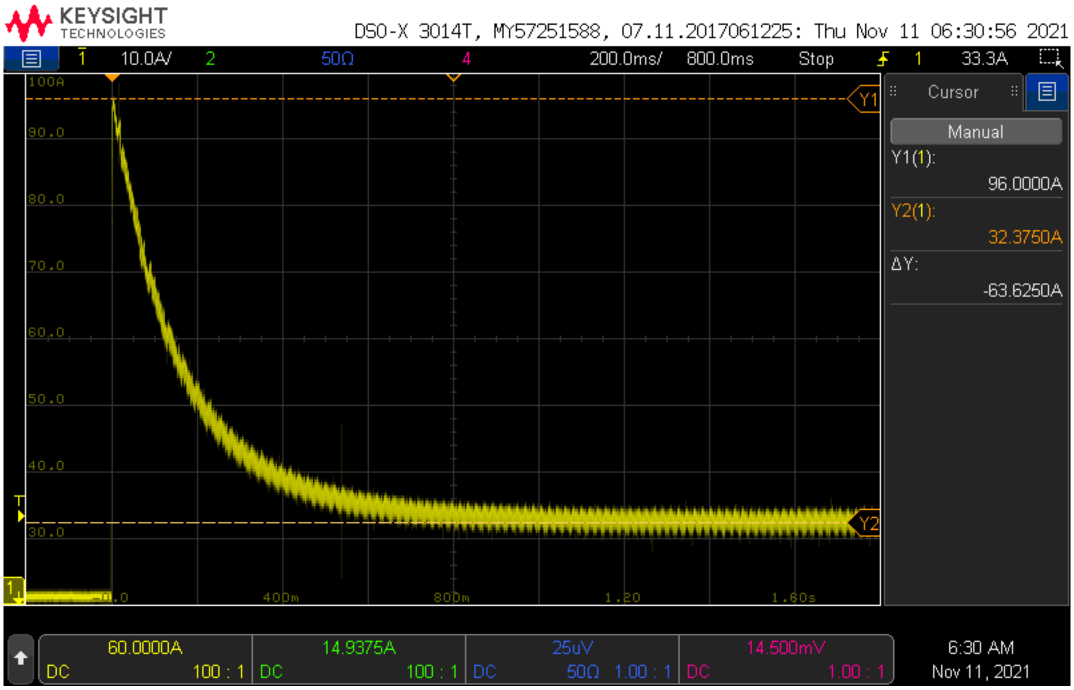


In open air

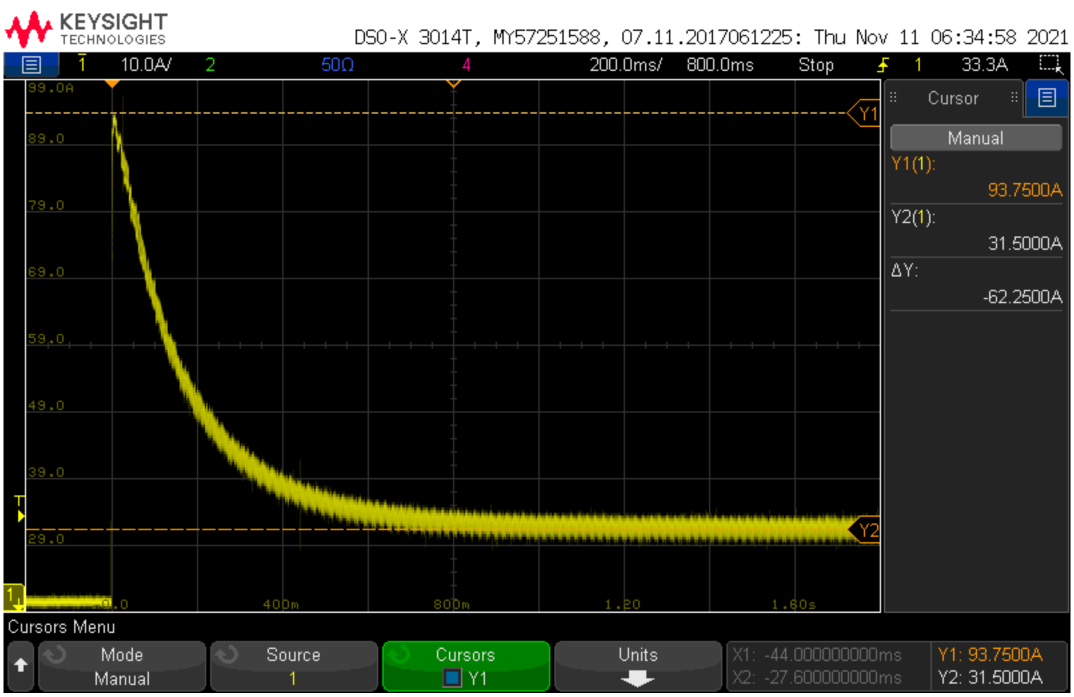


Mishimoto 11" fan

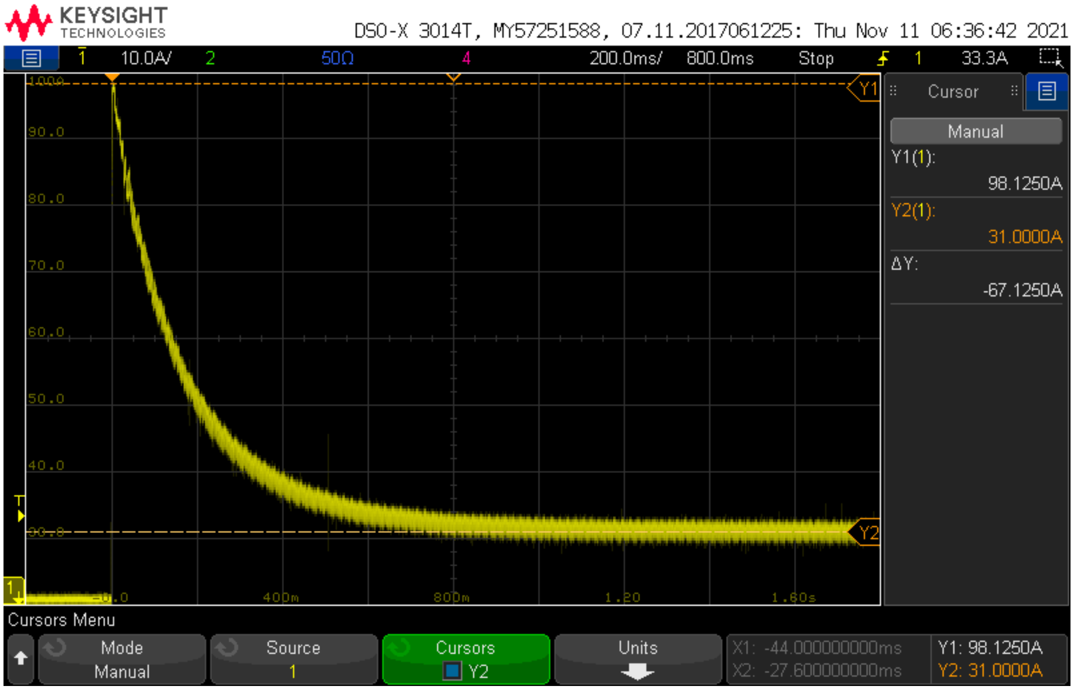
Face down on table



Face on radiator

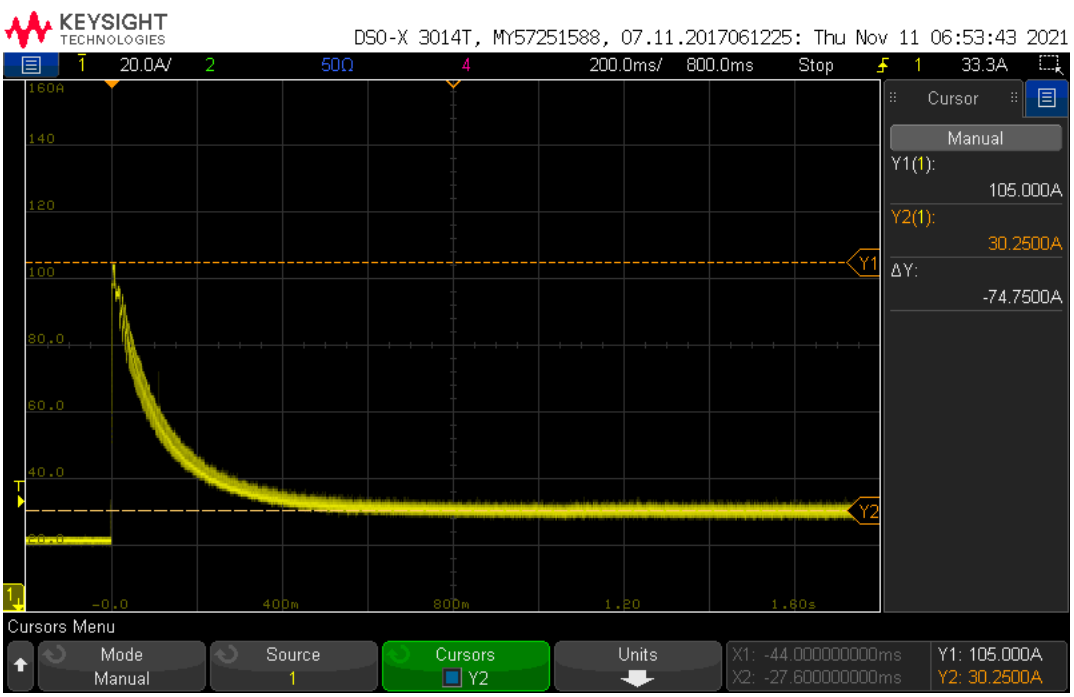


In open air



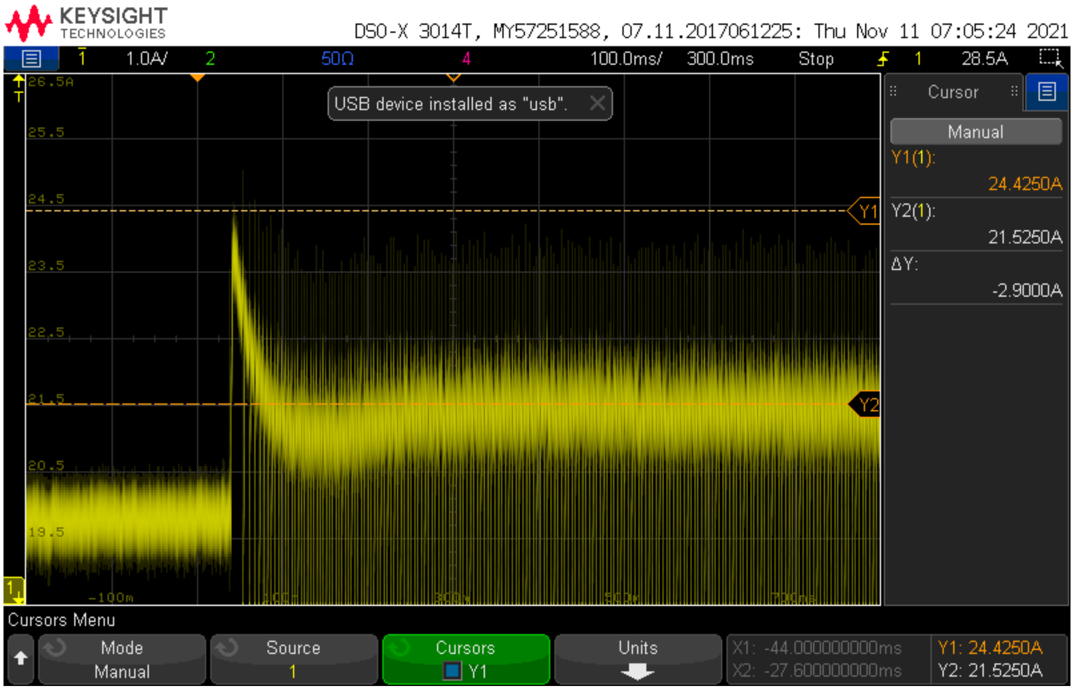
SPAL 11" Fan

Face on radiator

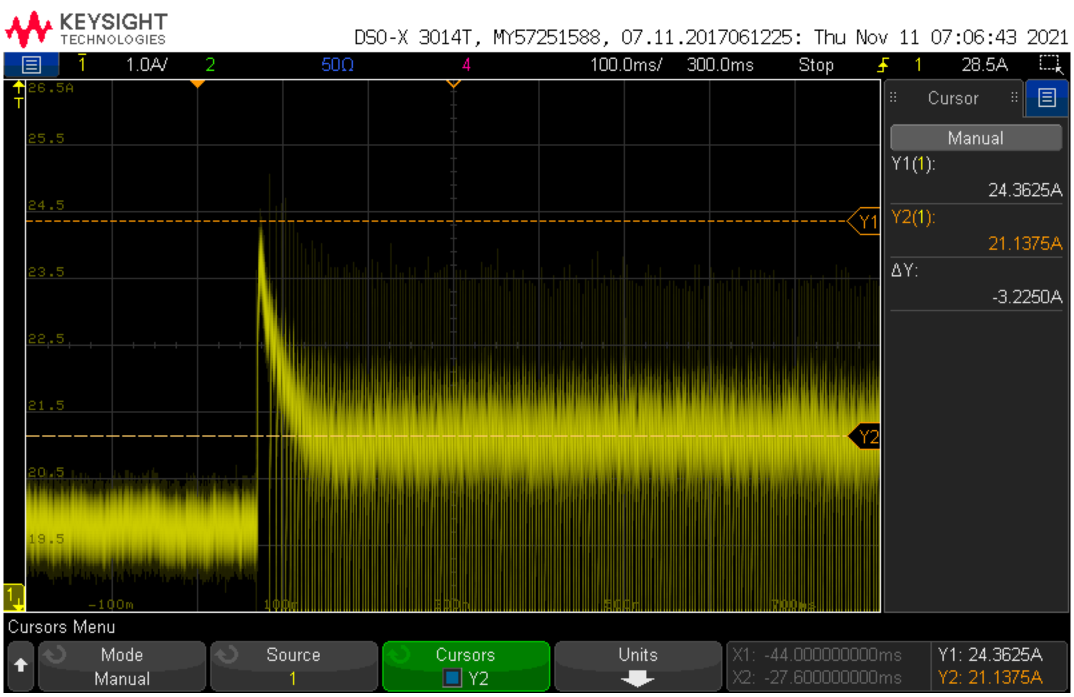


Davies Craig EBP 23 Water pump

Pumping water unrestricted

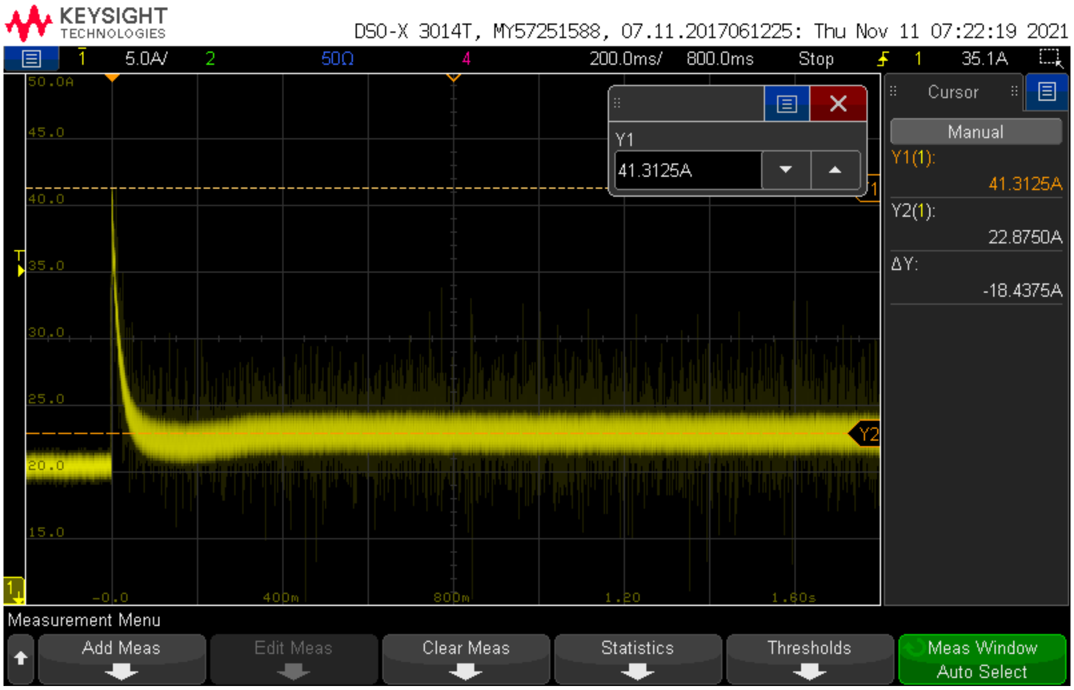


Pumping water with restriction

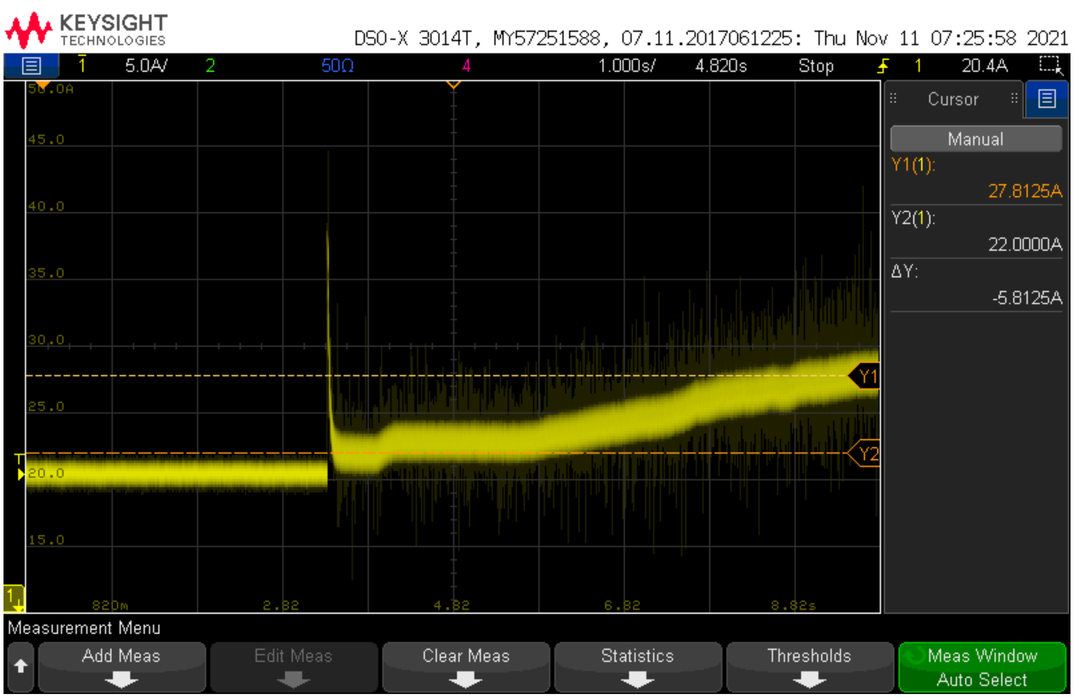


Fuel Pump

Pumping fuel unrestricted



Pumping fuel with increasing restriction



Appendix B: Shutdown Circuit FSAE Rules and Clarification Request

T.3.2 Brake Over Travel Switch - BOTS

T.3.2.1 The vehicle must have a Brake Over Travel Switch (BOTS). Brake pedal travel exceeding the normal range will actuate the switch

T.3.2.2 The BOTS must be a mechanical single pole, single throw (commonly known as a two position) switch (push-pull or flip type).

T.3.2.3 Operation of the BOTS to the OFF position must Open the Shutdown Circuit **IC.9.2.2** / **EV.8.2.2**

T.3.2.4 Repeated operation of the switch must not reset or restore power

T.3.2.5 The driver must not be able to reset the BOTS.

T.3.2.6 The switch must be implemented with analog components, and not using programmable logic controllers, engine control units, or similar functioning digital controllers.

IC.9.4 Cockpit Main Switch

IC.9.4.1 Configuration - The Cockpit Main Switch must:

a. Be a push-pull or push-rotate emergency switch (pushing the button is the OFF position)

b. Have a diameter of 24 mm minimum

IC.9.4.2 Location – The Cockpit Main Switch must be:

a. In easy reach of the driver when in a normal driving position wearing Harness

b. Adjacent to the Steering Wheel

c. Unobstructed by the Steering Wheel or any other part of the vehicle

IC.9.4.3 Function - the Cockpit Main Switch may act through a relay

17727

Edit Details

Using MOSFETs on PDM for Shutdown Circuit

Rule Numbers:

2022:IC.9.4.3
2022:T.3.2.6

Opened: 2021-11-17 22:29:46 ET

State: Open

Status: New

Team Member

2021-11-17 23:44:03 ET EST (GMT-5)

Tyler Sawatzky
Team Member for Univ of Calgary, Schulich Racing since 2020

Hello,

Our team is building a custom Power Distribution Module (PDM) for our car, and as part of this project we are considering implementing the Shutdown Circuit for the BOTS and Cockpit Main Switch into physical circuits on our PDM (previously we used relays for these functions). However, I would like to clarify whether our proposed implementation will fulfill the requirements for the Shutdown Circuit.

See the attached diagram for an idea of what we would like to implement - a circuit that uses MOSFETs to enable power to the fuel pump and ignition. A harness wire would go through both the BOTS and the Cockpit Main Switch, either of which being open circuit or shorted to ground would cut voltage to the NMOS transistor and ultimately kill power to the fuel pump and ignition.

What I would like to clarify specifically is:

T.3.2.6: The switch must be implemented with analog components, and not using programmable logic controllers, engine control units, or similar functioning digital controllers.
- Since MOSFETs can be considered analog components and there is no reliance on programmable logic, can you please confirm that this implementation will satisfy this rule?

IC.9.4.3: Function - the Cockpit Main Switch may act through a relay
- This rule states that relays are acceptable, but it is not clear on what is unacceptable. I believe that this implementation using MOSFETs is robust and reliable enough to satisfy the intent of this rule, would you be able to please confirm whether or not this is sufficient?

Thank you,
Tyler Sawatzky

- Shutdown Circuit Diagram.png

References

[1] R. Semiconductor, "Linear Regulator and Switching Regulator," ROHM Semiconductor, [Online]. Available: https://www.rohm.com/electronics-basics/dc-dc-converters/linear-vs-switching-regulators. [Accessed 2 November 2021].

[2] E. G. Emmanuel Odublade, "Different types of voltage regulators and working principle," Power & Beyond, 22 04 2020. [Online]. Available: https://www.power-and-beyond.com/different-types-of-voltage-regulators-and-working-principle-a-919791/. [Accessed 1 November 2021].

[3] T. Instrument, "TPS543x 3-A, Wide Input Range, Step-Down Converter," April 2017. [Online]. Available: https://www.ti.com/lit/ds/symlink/tps5431.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1637441521346&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com. [Accessed 10 November 2021].

[4] T. Instrument, "LM1575/LM2575/LM2575HV SIMPLE SWITCHER® 1A Step-Down Voltage Regulator," April 2013. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm2575-n.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1637442890305&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.co. [Accessed 5 November 2021].

[5] T. Instrument, "M3940 1-A Low-Dropout Regulator for 5-V to 3.3-V Conversion," Febraury 2015. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm3940.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1637443471115&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%. [Accessed 3 November 2021].

[6] T. Instruments, "Basics of eFuses," Texas Instruments, December 2016. [Online]. Available: https://www.ti.com/lit/an/slva862a/slva862a.pdf?ts=1637531210111&ref_url=https%253A%252F%252Fwww.google.com%252F#:~:text=An%20eFuse%20is%20an%20%E2%80%9Cactive,polarity%20and%20short%20circuit%20faults.. [Accessed 3 November 2021].

[7] Texas Instruments, "Adjustable Current Limit of Smart Power Switches (Rev. B)," Texas Instruments, November 2016. [Online]. Available: https://www.ti.com/lit/an/slva859b/slva859b.pdf?ts=1635511459821&ref_url=https%253A%252F%252Fwww.google.com%252F. [Accessed 3 November 2021].

[8] D. Knight, "Introduction to High-Side Load Switches," 4 February 2016. [Online]. Available: https://www.digikey.ca/en/maker/blogs/introduction-to-high-side-load-switches. [Accessed 3 November 2021].

[9] Rohm Semiconductor, "Automotive 2ch 45 mΩ High-Side Switch," Rohm Semiconductor, 15 March 2019. [Online]. Available: https://fscdn.rohm.com/en/products/databook/datasheet/ic/power/ipd/bv2hc045efu-c-e.pdf. [Accessed 3 November 2021].

[10] STMicroelectronics, "STM32 32-bit Arm Cortex MCUs," [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html. [Accessed 19 November 2021].

[11] STMicroelectronics, "Basic Extended Controller Area Network Interface," [Online]. Available: https://www.st.com/content/ccc/resource/training/technical/product_training/de/ee/37/c6/60/9c/45/a5/STM32L4_Peripheral_CAN.pdf/files/STM32L4_Peripheral_CAN.pdf/jcr:content/translations/en.STM32L4_Peripheral_CAN.pdf. [Accessed 19 November 2021].

[12] STM32-Base, "Guide: Connecting your debugger," [Online]. Available: https://stm32-base.org/guides/connecting-your-debugger.html. [Accessed 19 November 2021].

[13] STMicroelectronics, "Value-line Arm®-based 32-bit MCU with up to 256 KB Flash, timers,," [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f030cc.pdf. [Accessed 19 November 2021].

[14] Merriam-Webster, "Half Duplex Definition and Meaning," Merriam Webster, 03 Sept 2021. [Online]. Available: https://www.merriam-webster.com/dictionary/half%20duplex#:~:text=Definition%20of%20half%20duplex,at%20a%20time%20%E2%80%94%20compare%20duplex. [Accessed 20 Nov 2021].

[15] B. Kelechava, "Controller Area Netowrk Standards," The ANSI Blog, 4 June 2020. [Online]. Available: https://blog.ansi.org/2017/02/controller-area-network-can-standards-iso-11898/#gref. [Accessed 20 Nov 2021].

[16] S. Broyles, "A System Evaluation of CAN Transceivers," Texas Instruments, 01 Jul 2019. [Online]. Available: https://www.ti.com/lit/an/slla109a/slla109a.pdf?ts=1634418305215&ref_url=https%253A%252F%252Fwww.google.ca%252F#:~:text=The%20role%20of%20the%20transceiver,signal%20transmitted%20over%20the%20bus.&text=The%20transceiver%20distinguishes%20between%20two%20b. [Accessed 19 Nov 2021].

[17] Embitel, "ISO 11898 CAN Protocol Software Solution," Embitel, 20 Mar 2019. [Online]. Available: https://www.embitel.com/blog/embedded-blog/what-is-can-protocol-stack-why-its-critical-software-solution-for-ecu-communication. [Accessed 20 Nov 2021].

[18] SAE International, "2022 Formula SAE Rules," [Online]. Available: https://www.fsaeonline.com/cdsweb/gen/DocumentResources.aspx.

[19] Components101, "Design Guide - PMOS MOSFET for Reverse Voltage Polarity Protection," Components101, 28 May 2020. [Online]. Available: https://components101.com/articles/design-guide-pmos-mosfet-for-reverse-voltage-polarity-protection. [Accessed 10 November 2021].

[20] STMicroelectronics, "STM32Cube initialization code generator," [Online]. Available: https://www.st.com/en/development-tools/stm32cubemx.html. [Accessed 20 November 2021].

[21] STMicroelectronics, "Description of STM32F4 HAL and low-layer drivers," [Online]. Available: https://www.st.com/resource/en/user_manual/dm00105879-description-of-stm32f4-hal-and-ll-drivers-stmicroelectronics.pdf. [Accessed 20 November 2021].

[22] FreeRTOS, "The FreeRTOS™ Kernel," [Online]. Available: https://www.freertos.org/RTOS.html. [Accessed 20 November 2021].

[23] W. H. Paul Horowitz, The Art of Electronics (Third Edition), New York: Cambridge University Press, 2015.

[24] Onsemi, "NCV7800," November 2021. [Online]. Available: https://www.onsemi.com/pdf/datasheet/mc7800-d.pdf. [Accessed 2 November 2021].

[25] RICHTEK, "2A, 18V, 700kHz ACOTTM Synchronous Step-Down Converter," 2016. [Online]. Available: https://www.richtek.com/assets/product_file/RT7274=RT7279=RT7280=RT7281/DS7274_79_80_81-03.pdf. [Accessed 1 November 2021].

[26] Little Fuse, "Automotive Fuses," [Online]. Available: https://www.littelfuse.com/products/fuses.aspx. [Accessed 20 November 2021].

[27] B. Giovino, "IoT Microcontrollers Have ADCs, but Know When to Choose and Apply an External ADC," digikey, 21 November 2018. [Online]. Available: https://www.digikey.ca/en/articles/iot-microcontrollers-have-adcs-know-when-apply-an-external-adc. [Accessed 20 November 2021].

[28] Mbed, "Mbed OS Introduction to architecture," [Online]. Available: https://os.mbed.com/docs/mbed-os/v6.15/introduction/architecture.html. [Accessed 20 November 2021].