Tech Expo, Fall 2025 – Team 20

Formula SAE Dyno Controller

Purdue University
Polytechnic Institute

Team: Matt Skrzyszowski

Mentors: Dr. Frederick Berry

Problem Statement

The Purdue Formula SAE Team wants to improve their engine development program to improve future competition vehicle performance. The team currently has a custom water-brake engine dynamometer that was developed specifically for use with Formula SAE compatible engines, however the current control solution is making the control of the brake inconsistent. The team is asking for a custom controller that integrates with all components on the dynamometer.

Customer Background

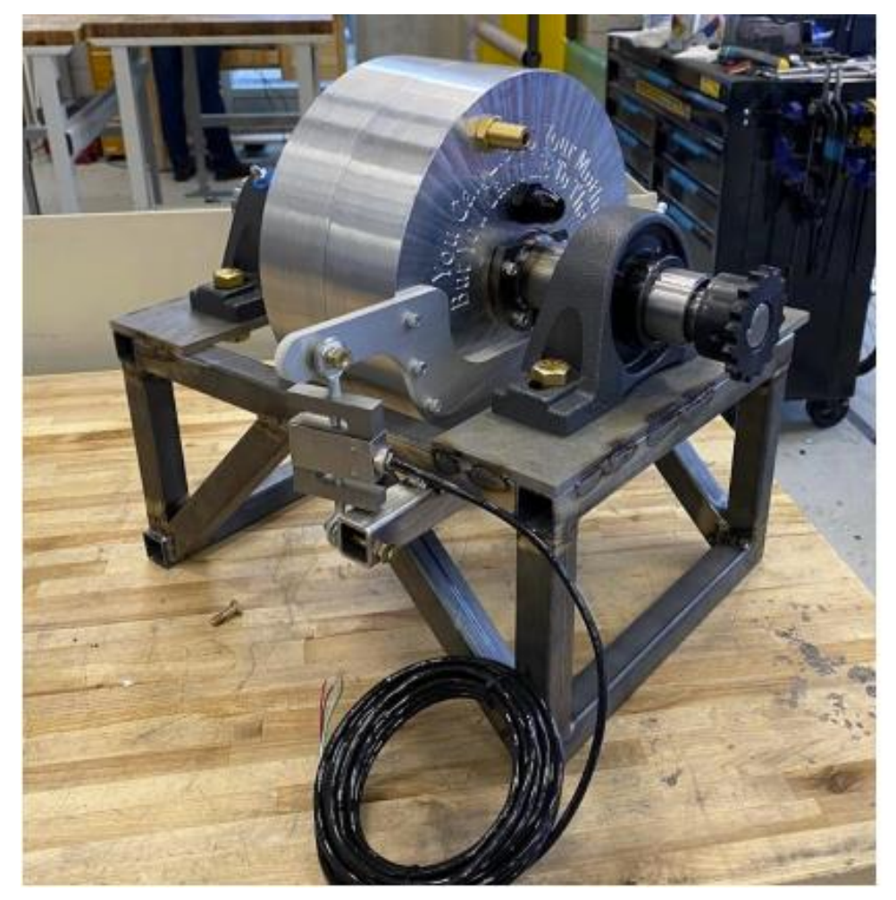
The Purdue Formula SAE team competes in the annual SAE (Society of Automotive Engineers) competition. The team designs, builds, and tests a formula/open-wheel style racing car that competes against 119 other teams from all around the world at the Michigan International Speedway. Almost all components are designed and manufactured in-house, including the carbon-fiber aerodynamics package and powertrain peripheral components. The team has recently had a string of top 10 results in competition, cementing their position as one of the best teams in the world.

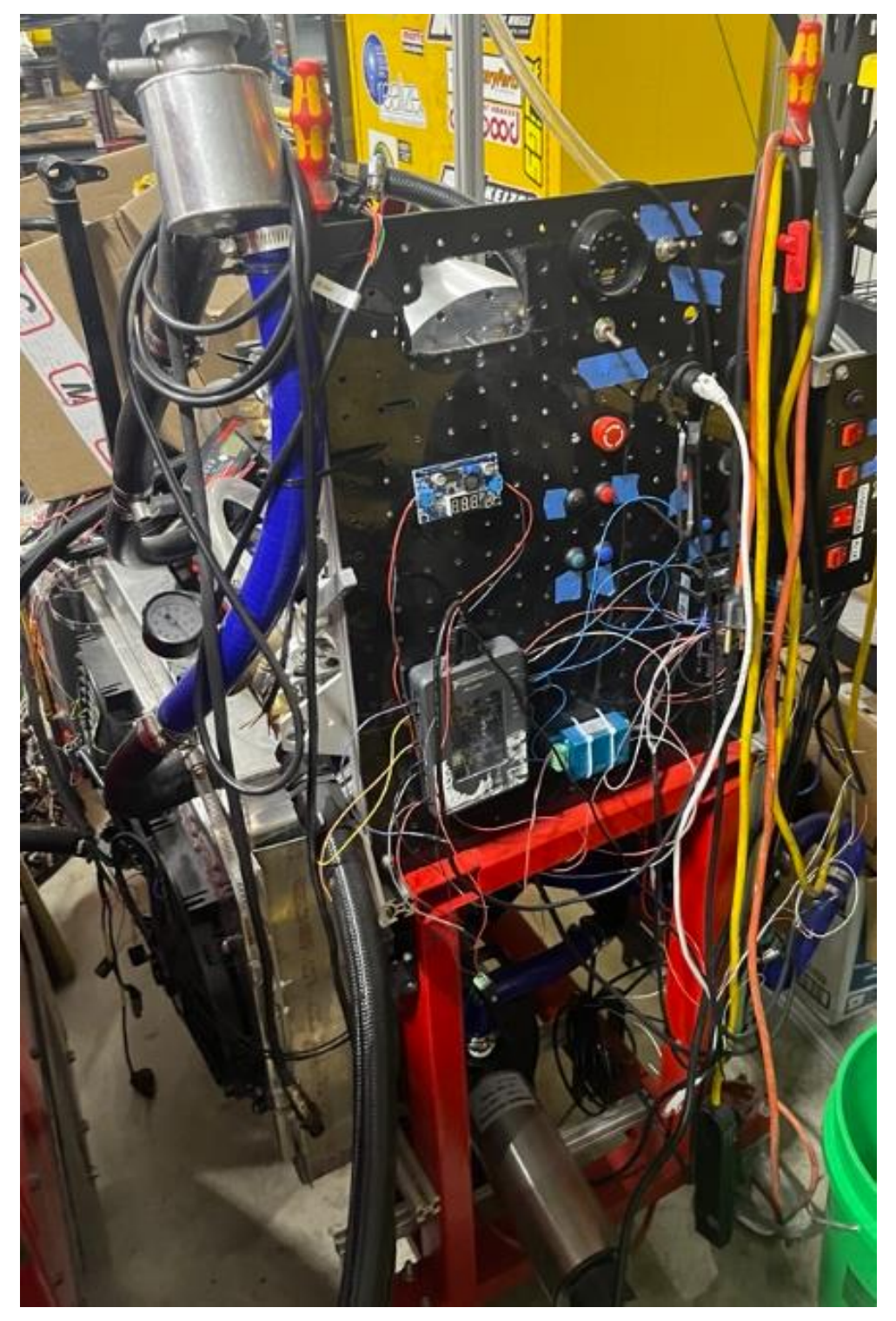
Requirements

Design Requirements	Design Targets	Validation
Rationale		
Stepper motor can be controlled (2 PWM outputs, one for step, one for direction)	Minimum control resolution is one step, control does not drift, can be controlled continuously for two hours	Stepper starting position marked, rotated 90 degrees back and forth for two hours, then final position checked relative to starting position
This is the primary function of the controller and is also the part that created the most problems in the previous solution.[3]		
Microcontroller can communicate with the Engine Control Unit (ECU)	Can send status messages and receive all CAN messages on the CAN bus for two hours	No communication errors for two hours
This is how the controller will receive information relative to the status of the engine. It is how it will receive control instructions as well.[3]		
Controller can read 4 analog voltage sensors	Analog voltage signals read with 0.01V accuracy	Power supply will power controller pin, and a multimeter will read the voltage at the controller pin. The microcontroller will then output the reading over CAN.
The controller may make adjustments in the control of the dynamometer based on inputs from analog sensors like temperature sensors.[3]		
Controller can read 2 digital frequency signals	Pulse width and frequency read withing .1% accuracy	A function generator will produce a signal at the controller pin. The microcontroller will then output the reading over CAN.
The controller may take in digital signals (like an RPM signal) to increase the accuracy and response of the controller rather than relying on the CAN communications with the ECU.[3]		
Controller must withstand the harsh environment of the dynamometer	The controller casing must be vibration and water resistant	The casing will be dunked under water for 10 minutes, dried, and then opened. If there is no moisture on the inside of the casing then this requirement is met.
The controller should not fail due to liquids being spilled on it or from the vibrations of the engine.[3]		
Controller must have three different operation modes: Manual, Automatic, and Sweep	Manual mode just moves the stepper to desired position, Automatic mode uses a PID function dependent on engine RPM and target RPM to control the stepper position, and finally Sweep mode sweeps the position of the stepper from closed to open over a desired amount of time	Manual mode must move to the exact desired position within .5% accuracy, Automatic mode must keep engine RPM ±100 RPM of target, and Sweep mode moves from full close to full open within 100ms of desired time
The function of the controller may be different for different engine development strategies and therefore should support all the different functions.[3]		

Experimentation and Concepts

The primary discussion for concepts was whether a custom solution for a PCB was required. The conclusion reached was that a custom PCB outweighed a purchased solution, as this allowed flexibility for future improvements and additions to the dynamometer solution as well as the ability to interface with the team's components reliably without additional tool purchases.

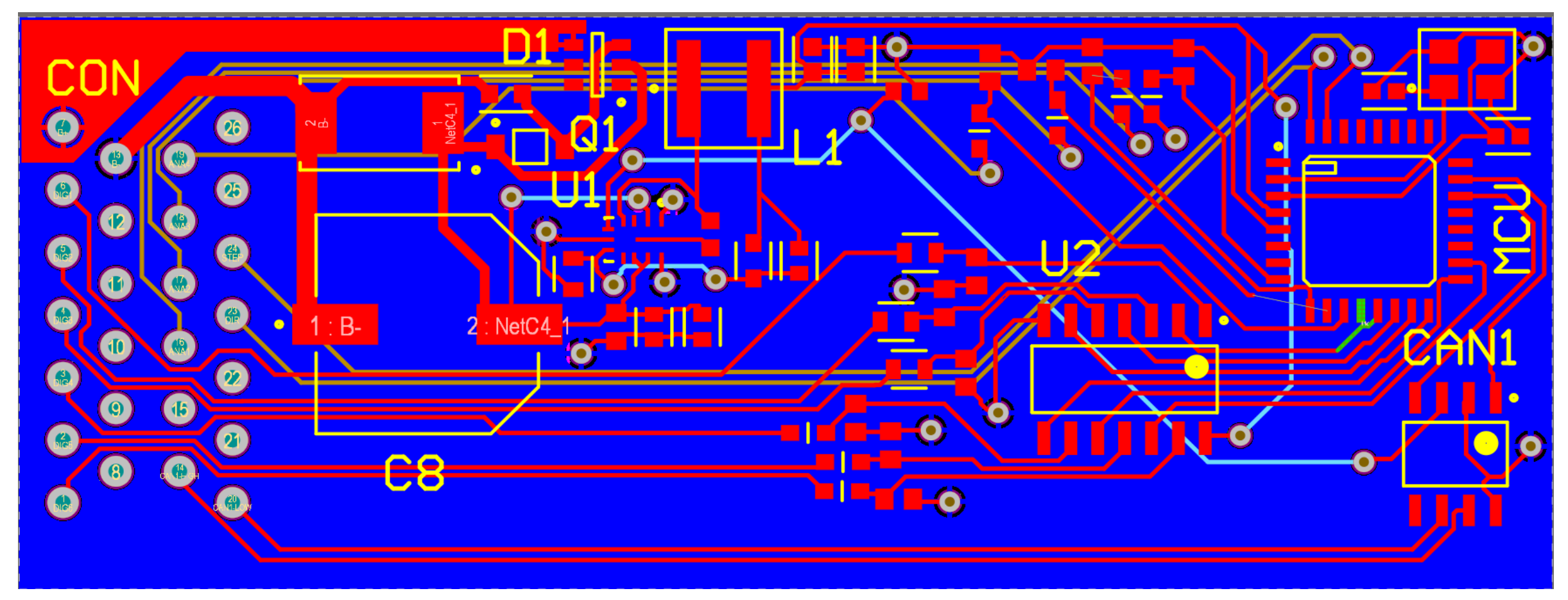


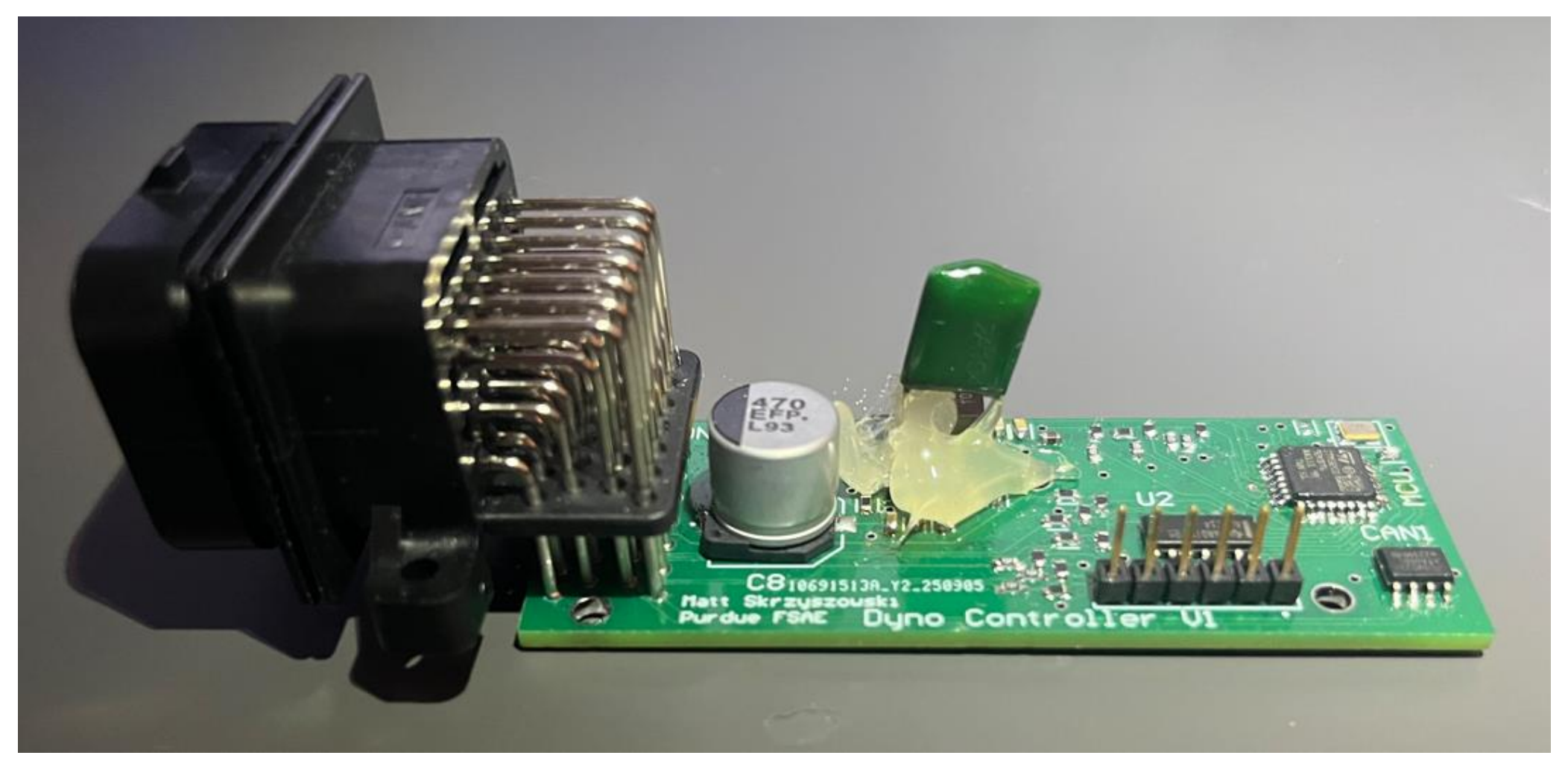


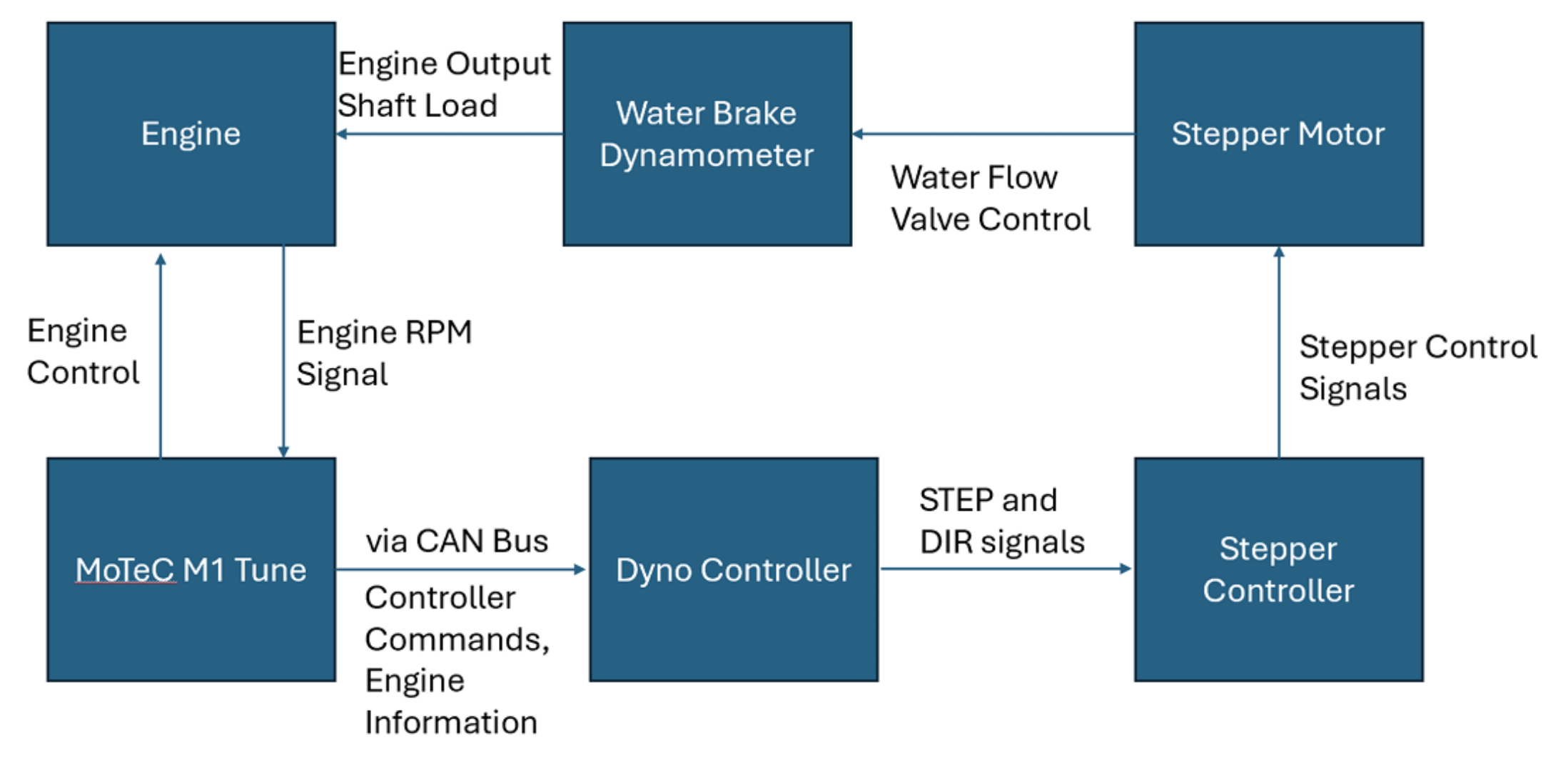
A previous solution for the dyno control included using a TI myRIO. This required many different devices wired to the primary device in order for this to function. There was also no interface with the CAN bus for communication with the engine ECU, which made user input difficult with two different software packages as well as added sensors for closed-loop control.

Final Design

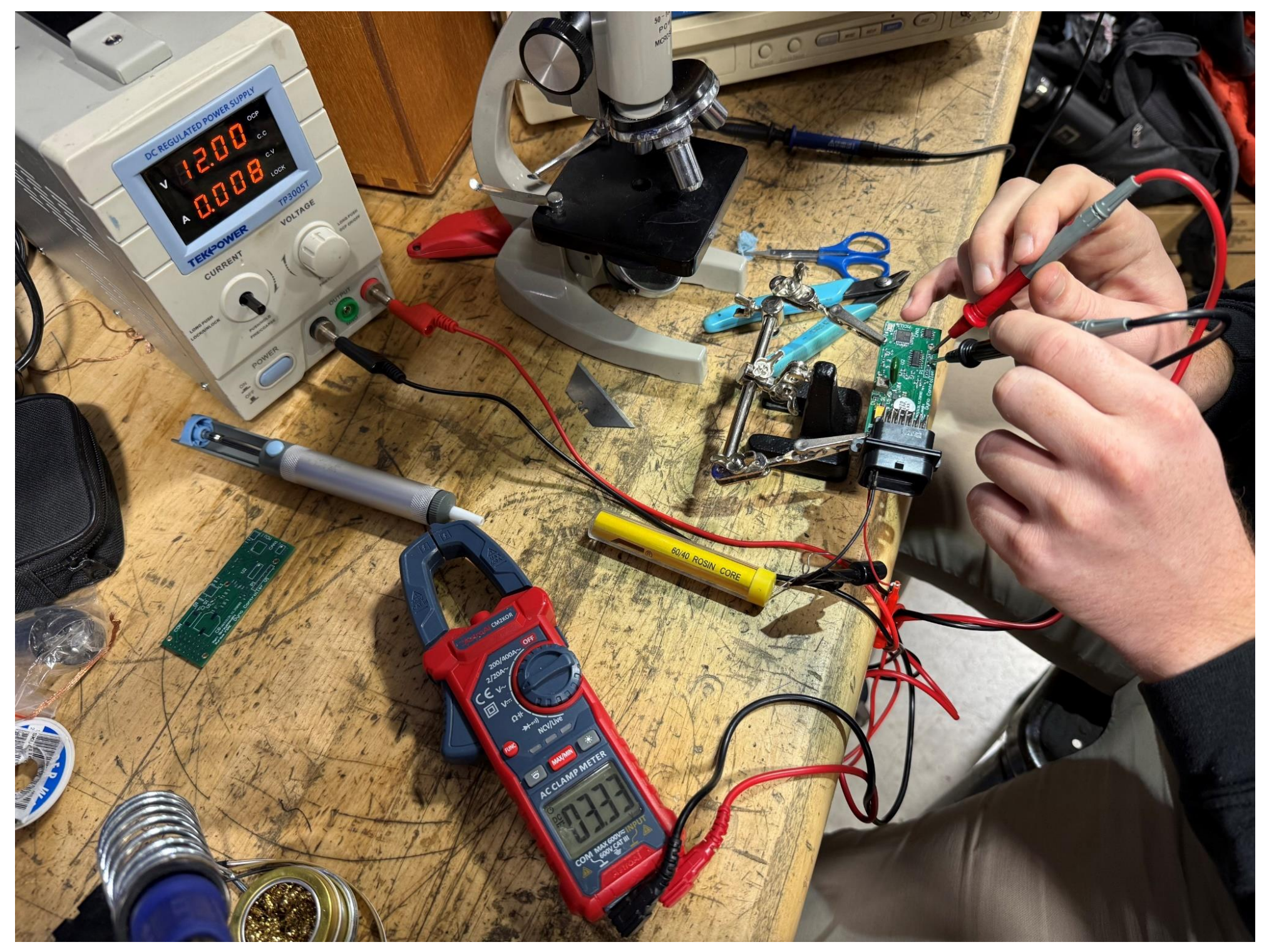
The final design was a PCB designed in Altium. This PCB uses an STM32 C-series microprocessor, has CAN bus communication via a CAN transceiver, has digital stepper control pins for step and direction, has 6x 12V tolerant digital input pins using a hex inverting buffer, and has 4x 5V analog sensors pins for additional analog sensor inputs. The PCB also has an onboard power supply circuit that accepts a wide range of external power voltages, as well as overvoltage and reverse polarity protection. The PCB was manufactured by JLCPCB and assembled in-house. The firmware package included control via CAN communication that would allow the user to select multiple different operation modes, including manual and automatic control. Automatic control was implemented by using closed-loop PID control.



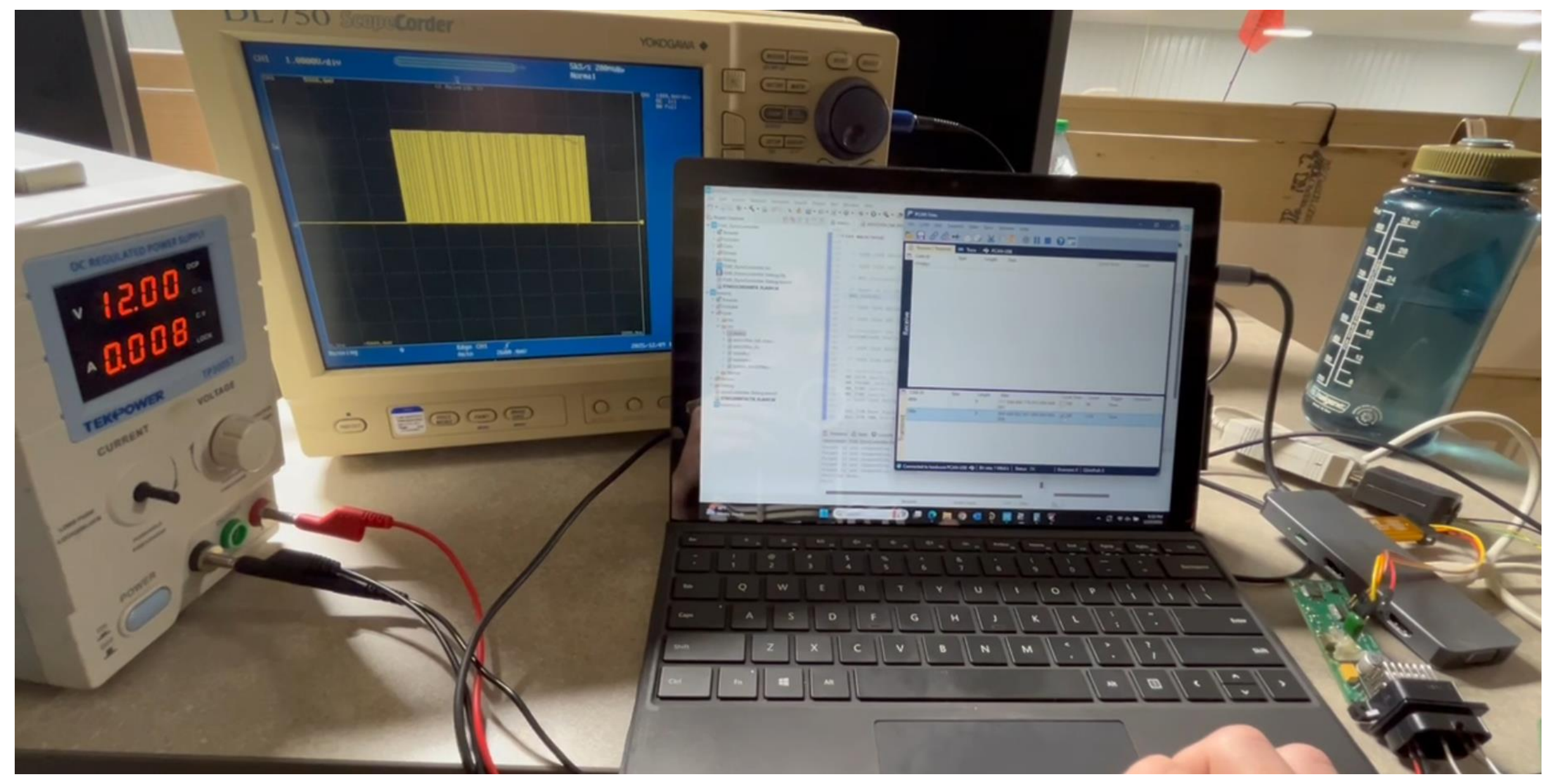




Testing and Results



Following manufacturing the electrical functions of the circuit were tested. The power supply circuit function was verified by applying a power via power supply and using a multimeter to verify 3.3V and 5V rail voltages. During this test, an error was found with the buck converter pin connections. This was resolved by cutting/exposing and soldering to top layer traces. Hex inverter inputs were verified by applying a 12V voltage to the input pins and using a multimeter to verify output voltage behavior. To ensure MCU functionality, a blank project was flashed to the board. There was initially an issue with the debugger's connection to the board, however this turned out to be incorrect pin connection to the debugger as well as the MCU needing to be in a reset state. CAN communication and stepper control output signals were verified using basic code, an oscilloscope, and a PCAN CAN bus analyzer device. The stepper control signal frequency and duty cycle were measured via oscilloscope, and CAN message receive and transmit was verified using the PCAN device.



Unfortunately, the Purdue Formula SAE team was not prepared for dyno operation so only bench testing was used to prove functionality. CAN messages between engine ECU and controller were simulated via PCAN to control stepper output. Oscilloscope was used to verify appropriate response from controller.