Swarthmore College
Department of Engineering

Hydrogen Fuel Cell Vehicle

ENGR 090 Final Report

Authors
Alexander Vasquez-Jaffe Kimaru Boruett

Advisor
Prof. Carr Everbach

May 11, 2026

Abstract

This project details the development of a prototype hydrogen vehicle for the Shell Eco-Marathon (SEM) Americas, focusing equally on its physical powertrain and software-driven race strategy. The drivetrain centers on a Horizon H-1000XP proton exchange membrane (PEM) fuel cell, a heavily regulated hydrogen delivery system, and an electric hub motor. Initial physical testing revealed critical torque limitations, prompting an immediate upgrade to a 36-volt motor controller to enable basic self-propelled movement. To resolve subsequent rough commutation and prevent system stress, the drivetrain was adapted to utilize a field-oriented controller capable of matching the fuel cell's variable voltage output. In parallel, a comprehensive telemetry platform was designed to optimize the vehicle's traction energy. A predictive longitudinal force simulation generated a burn-and-coast operating plan, successfully validating the vehicle's theoretical ability to complete the 15,400 m course within the 2,100 s limit. Real-time performance monitoring was achieved via a Flask and Socket.IO server parsing serial packets into a live web dashboard. While full physical deployment was constrained by mechanical timelines, the core software architecture and dynamic driver cues were successfully validated on a bench-scale Arduino prototype, with the entire software system completed for $672.

Contents

1 Introduction

1.1 Shell Eco-Marathon (SEM)

SEM is a global competition hosted by Shell (the British multinational oil and gas company) in which participants compete to design and race the most fuel efficient vehicle. There are three energy categories, hydrogen, internal combustion, and battery electric, and two vehicle categories for each energy category, prototype and urban concept. For this project, we competed in the prototype hydrogen division in the SEM Americas race. Because it was Swarthmore College Engineering's first time competing in the SEM since 2018, and our first time competing with the current vehicle, our primary goal for the race was to be able to travel to the competition, pass the technical inspection and complete a race.

1.2 Hydrogen Powered Drivetrain

The heart of the drivetrain for our vehicle is the Horizon Fuel Cells H-1000XP proton exchange membrane (PEM) hydrogen fuel cell. This fuel cell generates electrical which drives the hub motor used to drive the vehicle. Hydrogen for the fuel cell is stored in a gas cylinder and directed to the fuel cell using Teflon and Swagelok tubing. The goals were to build a propulsion system with hydrogen as the fuel source that is able to pass the stringent SEM technical inspection and then drive the vehicle around the track.

1.3 Electrical System

The electrical system in the vehicle can be broken up into two categories; the high voltage system which is the power outputted by the fuel cell and used to drive the hub motor, and the 12 volt system which covers the 12 volt accessory battery, safety system, telemetry, and fuel cell controller. The goals were to build a robust electrical system capable of safely being able to deliver enough power to the hub motor while also being to power all other accessory. It also must pass the stringent SEM technical inspection.

1.4 Telemetry System

Hydrogen fuel-cell prototype vehicles in the Shell-Eco Marathon(SEM) competition operate under a constrained objective: complete a fixed course within a time limit while minimizing energy use. That means the car cannot simply drive as slowly as possible. Instead, the team needs a strategy that determines where the vehicle should accelerate, where it should coast, and how much margin should be held relative to the required average speed. At the same time, the team needs telemetry that can show whether the car is behaving as expected during testing and, eventually, during live operation.

We proposed a unified software system to support both problems. The first part is a race strategy platform that converts measured track data into a simulation-ready route, estimates the effect of drag, rolling resistance, grade, and cornering constraints, and generates a **burn-and-coast** operating plan. The second part is a telemetry platform that reads fuel-cell data

from serial packets, decodes and logs the packets, and presents the results on a live dashboard in addition to vehicle dynamics such as acceleration, speed and driver inputs.

The proposed work had four main goals:

1. build a track-aware race strategy model for the hydrogen vehicle;
2. create an interactive interface for exploring strategy parameters and comparing track scenarios;
3. build a real-time telemetry dashboard for fuel-cell and vehicle-state monitoring; and
4. prepare the software so it can later be integrated with the full vehicle, including additional sensors and the mechanical and electrical subsystems documented by the rest of the team.

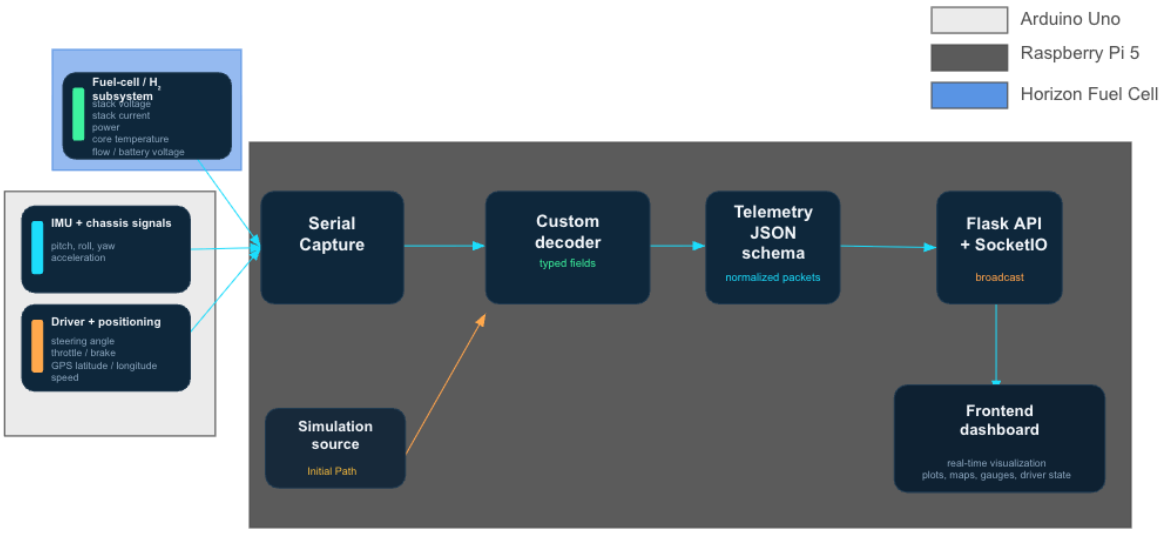


Figure 1: Architecture of the integrated telemetry and data acquisition system. Sensor data from the Horizon fuel-cell subsystem, IMU, and driver inputs are captured via serial communication on a Raspberry Pi 5, decoded into a normalized JSON schema, and broadcast via Flask API and SocketIO to a real-time frontend dashboard for live performance monitoring and strategy validation.

This integrated workflow transforms competition preparation from subjective estimations into a rigorous engineering methodology. By unifying predictive simulation with real-time analytics, the project facilitates both data-informed design and a data-informed race.

1.5 Chassis, Body, Steering, and Braking

The chassis, body, steering, and braking systems are beyond the scope of this project. However, they were still extremely important parts of the vehicle so that we could compete in the competition. The chassis was designed over multiple summer and semester as a research project by Nora Jiang and Mathew Shiley. Nora Jiang designed and installed the body

of the vehicle, which is made of carbon fiber paneling. It is a carbon fiber chassis built with quarter-inch carbon fiber tubing assembled using custom machined aluminum linkages. Both the steering and braking systems were designed and built by Jasper Mosley. The steering uses a rack an pinion steering system mounted to the vehicle with custom 3D printed mounts. The braking system has front and rear brakes which are isolated from each other. Both systems use hydraulic brakes.

2 Technical Discussion

2.1 Hydrogen Powered Drivetrain

2.1.1 PEM Fuel Cell

The fuel cell used is the H-1000 XP from Horizon Fuel Cell Technologies. This fuel cell was chosen for the vehicle because of our familiarity with Horizon fuel cells (before the current fuel cell, we had a Horizon H-1000 fuel cell), ease of use, and because the H-1000 XP is designed for the SEM. –add how it works

2.1.2 Hydrogen Delivery System

The SEM have very strict rules on the components that must be included in the hydrogen delivery system. There must be a cylinder, pressure gauge (for pressure in the system), a solenoid valve, a pressure relief valve, and a flow meter (provided by the competition) before hydrogen reaches the fuel cell. The SEM rules require that the solenoid valve be placed immediately after the regulator and the flow meter be placed immediately before the fuel cell. In the system we designed, hydrogen is stored in a 20CF gas cylinder located in the rear of the vehicle. It is the beginning of the hydrogen delivery system. Attached to the outlet of the tank is a Harris 425 single stage regulator which allows us to control output pressure from the gas cylinder. It has two gauges, a tank pressure gauge and an outflow pressure gauge. This regulator must be set to 6.5 to 8 psi. Next, there is a solenoid valve whose operation is controlled by the safety circuit and following the solenoid valve is the SEM issued flow meter. Finally, the hydrogen reaches the fuel cell.

The pressure relief valve used is a Swagelok SS-RL3S4 Low Pressure Proportional Relief Valve. The minimum relief pressure that this valve can be set to is 10psi, and since our system runs at a maximum of 8.5psi, we must set the pressure relief valve to its lowest relief pressure. This is done by screwing the valve to the pressure relief valve in as far as possible.

The tubing used is Swagelok quarter inch stainless steel tubing with Swagelok tube fittings (double ferrule compression type). The tubing entering the fuel cell is quarter inch teflon tubing connected to the fuel cell with quick-connect connectors. The teflon tubing and connectors came with the fuel cell from Horizon.

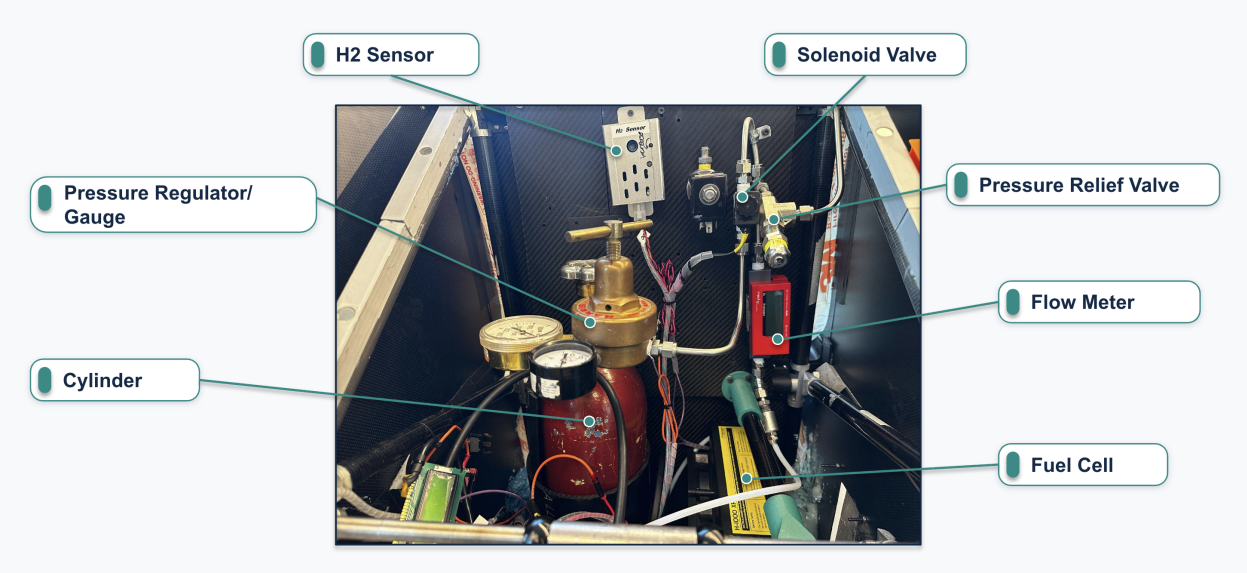


Figure 2: Diagram of Hydrogen System in Vehicle

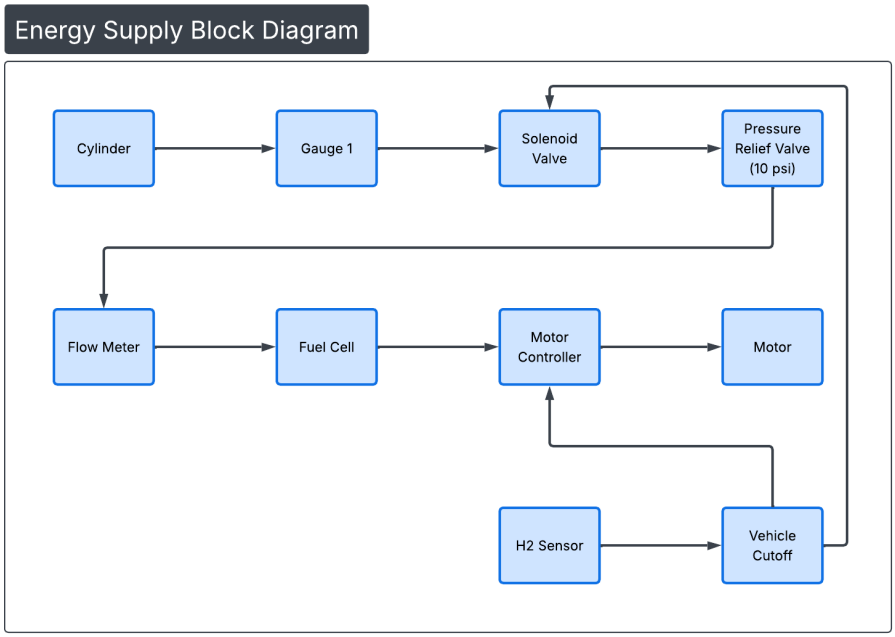


Figure 3: Energy Supply Block Diagram submitted to SEM as part of the required documentation

2.1.3 Electric Drive System

To maintain simplicity, a 1000w hub motor is used to drive the vehicle. It is connected to a 36 volt, 25 amp motor controller. The controller is capable of handling the varying voltage of the fuel cell as it is within its tolerance band.

2.1.4 Electrical System

SEM also has strict rules on the electrical system and what must be included in the safety system. For the safety system, there must be two latching emergency shutdown buttons, one in the cabin and one outside the vehicle. There must be a hydrogen sensor to detect hydrogen leaks and cause an emergency shutdown when hydrogen is detected.

Safety Circuits
The safety circuits are directly integrated into the Horizon fuel cell. The fuel cell comes with a latching emergency shutdown button connected to the fuel cell controller with a positive and negative wire in a loop. Since the competition requires two emergency shutdown buttons, a second button was simply added to that loop, and no other changes were made. The Horizon fuel cell also comes with an integrated normally-closed solenoid valve that is wired directly to the fuel cell controller. The electronics for this solenoid valve were simply left as is.

Hydrogen Sensor
A hydrogen sensor must be installed in the vehicle to detect hydrogen leaks and must initiate an emergency shutdown when it is tripped. The Horizon fuel cell has an integrated hydrogen sensor. Unlike previous hydrogen sensors used in earlier safety circuit iterations, where hydrogen detection de-energized a relay coil and opened a safety loop to trigger an emergency shutdown, this sensor outputs a signal to the fuel cell controller, which then initiates the shutdown.

The hydrogen sensor must be must be tuned to 25% of the LEL (Lower Explosive Limit) of hydrogen, i.e. 1% of hydrogen in air. To tune the sensor, there is an adjustment knob that can be adjusted with a mini flathead screwdriver on the circuit board of the hydrogen sensor board. However, to access that screw, the casing has to be taken apart. To have easier access to the adjustment knob, a small hole was drilled into the case, allowing a small flathead to slip into the case and turn the knob. When the knob is turned counter-clockwise, the sensitivity is increased, and vise versa. When the car is driving, the vibration causes the knob to shift by itself, so it is important to retune the sensor to 25% of the LEL before technical inspection at the competition. The best way to tune the sensor is to adjust the knob to a very high sensitivity (until the hydrogen in the air triggers the sensor), and then turn the knob 3-4 turns the other direction. Then, using a can of hydrogen gas that contains a concentration of hydrogen of 25% of the LEL, spray that on the sensor to see if it triggers. If the sensor is triggered, that means that it is set to the proper sensitivity.

Figure 4: Hydrogen Sensor

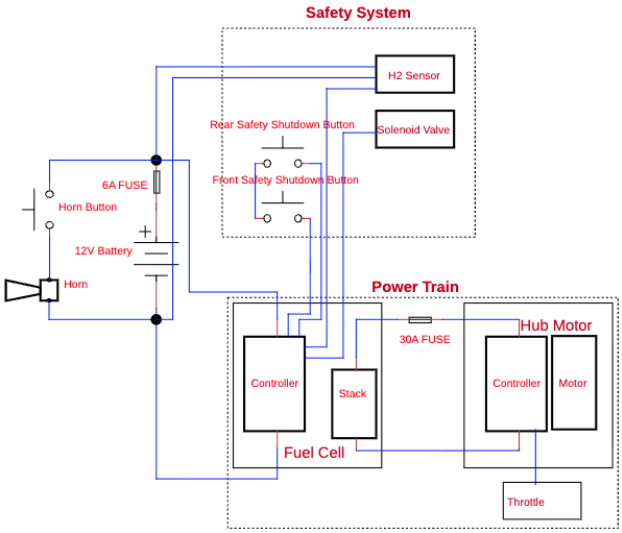


Figure 5: Wiring Diagram of Vehicle

Super Capacitor

Although the super capacitor was not used during the competition because we did not have the bandwidth for the added complexity, it was still tested on the vehicle. The specs for the super capacitor are not exactly known and a new one will have to be purchased if the super capacitor is going to be used in competition because the current one is likely very old and SEM requires teams to have documentation about the capacitor. Using the capacitor will help smooth out the voltage coming from the fuel cell and could help with getting more power to the hub motor. It will also help protect the fuel cell by smoothing any current spikes caused by the hub motor.

On the fuel cell controller, there is a place to directly plug it into the controller. Anderson connectors have been fitted to the capacitor for an easy connection to the fuel cell controller.

If the capacitor has not been used for a long period of time, or if it has been discharged, it must be charged before being connected to the fuel cell controller. If the capacitor is connected while discharged, it can draw a very large inrush current from the fuel cell and behave like a near-short circuit, which may damage the fuel cell. Follow the steps below to charge the capacitor before connecting it to the controller and then fit it to the controller.

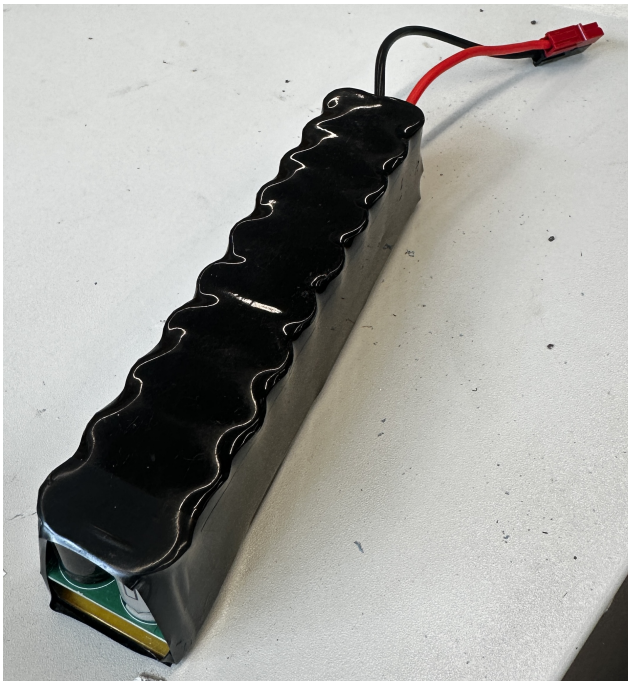


Figure 6: Super Capacitor

1. Set a power supply to about 12 V and set a current limit on it to about 5 A.
2. Connect the capacitor to the power supply and turn the power supply on
3. Once the capacitor stops pulling power (no current is flowing), disconnect the capacitor. Be very careful not to touch the leads on the capacitor as it could discharge very rapidly into your body causing a potentially deadly electric shock
4. Plug the capacitor into the fuel cell (it is a direct plug-and-play fit)
5. Operate the fuel cell like normal!
6. If the capacitor needs to be removed and discharged proceed with the following steps.
7. Plug the capacitor into the spare harness that consists of a red and black Anderson connector pair on one side and two female connectors on the other side (this harness just makes it easier to connect the capacitor to the resistor and power supply)
8. Take the large green resistor and touch both leads of the capacitor to each end of resistor, again being careful not to touch the capacitor leads. This will discharge the capacitor.

9. After discharging the capacitor, to plug it back into the fuel cell you must recharge it first

Figure 7: Resistor for Discharging Capacitor

2.1.5 Buck Converter

A buck converter can be installed between the fuel cell and the fuel cell controller to step down and regulate the fuel cell output voltage, allowing the controller to be powered directly by the fuel cell rather than by the accessory battery. It is a direct plug-and-play fit to the fuel cell, just like the super capacitor. We opted to not use the buck converter as it adds another layer of complexity, which we were not prepared for for this year's competition. Additionally, the buck converter does not have to be used even if the super capacitor is being used.

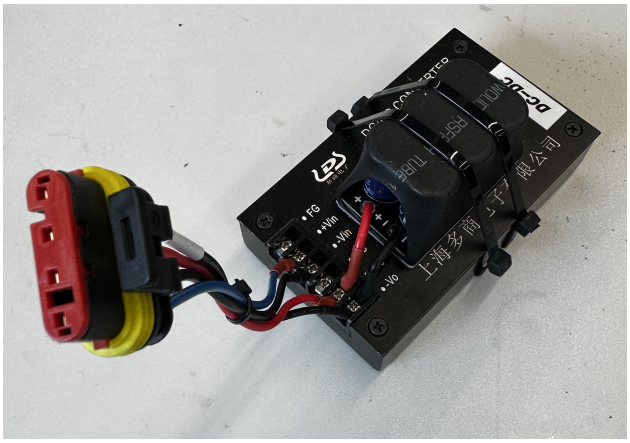


Figure 8: Buck Converter

2.1.6 Mounting Components in Rear

Mounts were designed for multiple components in the rear of the fuel cell. They were all placed on a 1/4" piece of carbon fiber that was cut to put on the frame rails and could support the weight of everything.

Fuel Cell Aluminum brackets were designed to match the footprint of the fuel cell, allowing it to remain in the same position on the floor. Carbon fiber tubing was then fitted around the fuel cell to secure the fuel cell to the chassis. This prevented the fuel cell from moving any other direction. Holes were drilled through the tubes, and screws were inserted through the frame and fastened directly into the fuel cell's mounting holes. The carbon fiber frame was secured to the chassis using 3D-printed fittings designed by Jeffrey Liang.

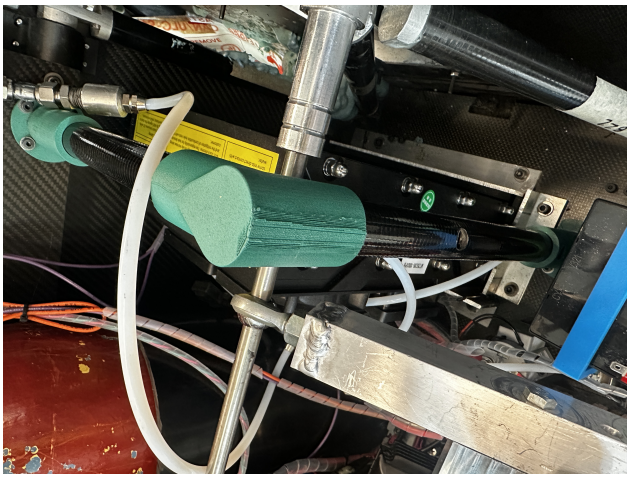


Figure 9: Aluminum Bracket (bottom) and carbon fiber frame

Fuel Cell Controller Mounts for the fuel cell controller were designed and 3d printed by Jeffrey Liang. They keep the motor controller upright allowing easy access to its wiring and minimizing the space used by the controller.

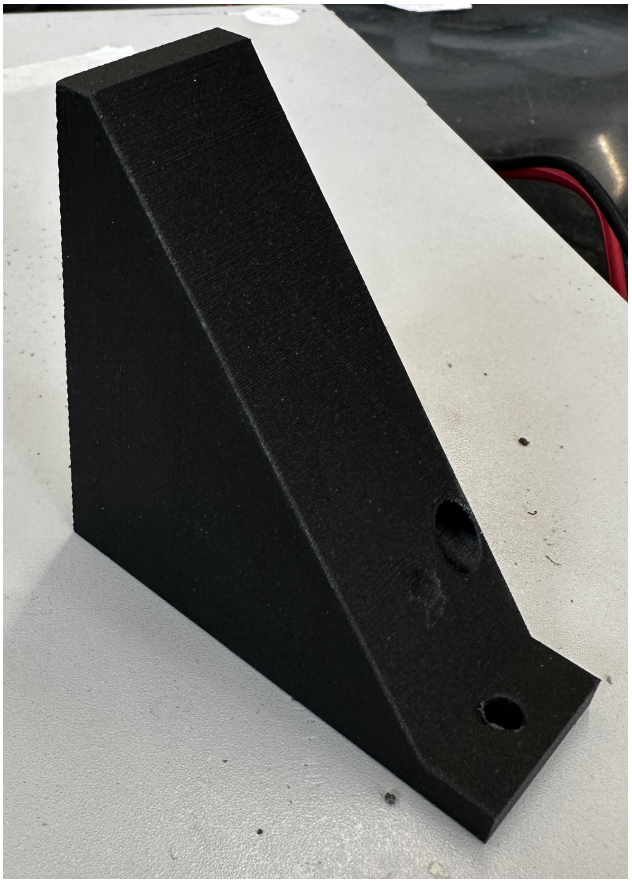


Figure 10: Fuel Cell Controller Mount

Hydrogen Gas Cylinder The hydrogen gas cylinder is secured in a plastic tube. This plastic tube is mounted to the rear floor with a 3d printed base. It was designed by Jeffrey Liang and Nathan Swartz.

12 V Accessory Battery The accessory battery is secured to the rear of the vehicle also with a 3d printer mount designed by Jeffrey Liang.

Figure 11: Accesory Battery in Battery Mount

Other Components Many other miscellaneous components, such as the motor controller and power bus, are securely mounted in the rear of the vehicle. They are all mounted to the floor with screws in their original mounting holes. The motor controller has a block of aluminum under it to act as a heat sink because the motor controller tend to get very hot.

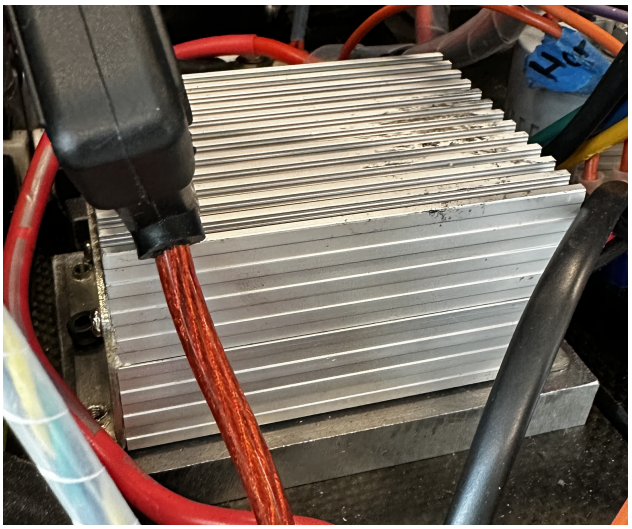


Figure 12: Motor Controller with Block of Aluminum Underneath

2.2 Telemetry System

2.2.1 Part 1: Predictive Race Strategy & Design Synthesis

The project combines race strategy modeling, track processing, and telemetry visualization for a hydrogen fuel-cell prototype vehicle. The main goal was to determine whether the vehicle could complete the required event distance within the allowed time while minimizing fuel-cell traction energy. The software framework converts real track data into a simulation-ready format, estimates grade and curvature, applies a burn-and-coast driving strategy, and produces time-series outputs for speed, acceleration, fuel-cell power, fuel-cell energy, and control mode.

The model is intentionally simple and interpretable. Rather than attempting to simulate every detailed physical effect in the vehicle, it captures the main effects needed for race planning: longitudinal resistance, limited motor and fuel-cell power, curvature-based cornering limits, and the timing requirement. This made the model practical for rapid iteration and useful for future integration with live telemetry.

Mathematical Model

The simulation represents the track in the spatial domain using cumulative distance s. Each track point contains elevation, grade, curvature, latitude, and longitude. The raw CSV track data is resampled onto a uniform distance grid, allowing the vehicle state to be updated from one distance step to the next.

The track grade is estimated from the change in elevation over distance,

$$\text{grade}(s) = \frac{dh}{ds}$$

and the corresponding road angle is

$$\theta(s) = \arctan(\text{grade}(s)).$$

The vehicle is modeled using a one-dimensional longitudinal force balance:

$$ma = F_{\text{drive}} - F_{\text{res}},$$

where m is the vehicle mass, a is acceleration, F_{drive} is the applied tractive force, and F_{res} is the total resistive force. The resistive force combines aerodynamic drag, rolling resistance, and grade resistance respectively:

$$F_{\text{res}} = \frac{1}{2}\rho C_d A v^2 + mgC_{rr}\cos\theta + mg\sin\theta.$$

The resulting acceleration is therefore

$$a = \frac{F_{\text{drive}} - \left(\frac{1}{2}\rho C_d A v^2 + mgC_{rr}\cos\theta + mg\sin\theta\right)}{m}.$$

Because the simulation advances by distance rather than by a fixed time step, the next speed is calculated using

$$v_{i+1} = \sqrt{\max\left(0,\ v_i^2 + 2a_i ds_i\right)}.$$

The drive force is limited by both available power and maximum tractive force:

$$F_{\text{drive}}(v) = \min\left(F_{\max}, \frac{P_{\text{drive,max}}}{\max(v, v_\epsilon)}\right),$$

where

$$P_{\text{drive,max}} = \min\left(P_{\text{motor,max}}, P_{\text{fc,max}}\right).$$

This captures the fact that the vehicle may be force-limited at low speed and power-limited at higher speed.

The track curvature imposes a local cornering speed limit. For curvature $\kappa(s)$, lateral acceleration is

$$a_y = v^2\kappa.$$

Constraining this to a maximum allowable lateral acceleration gives

$$v_{\text{corner}}(s) = \sqrt{\frac{a_{y,\max}}{\kappa(s)}}.$$

In the implementation, this value is clipped by the absolute speed cap of the vehicle. The simulated vehicle speed is then constrained so that it does not exceed the local cornering speed limit.

The fuel-cell traction power is computed from mechanical wheel power and drivetrain efficiency:

$$P_{\text{mech}} = F_{\text{drive}} v,$$

$$P_{\text{fc}} = \frac{P_{\text{mech}}}{\eta_{\text{drive}}}.$$

When the vehicle is coasting, $F_{\text{drive}} = 0$, so the fuel-cell traction power is set to zero. The cumulative fuel-cell traction energy is then computed as

$$E_{\text{fc}} \approx \sum_{i=1}^{N} P_{\text{fc},i} \Delta t_i,$$

where

$$\Delta t_i = \frac{ds_i}{\max(v_i, v_\epsilon)}.$$

The race timing requirement is checked using

$$v_{\text{avg,req}} = \frac{L_{\text{event}}}{T_{\max}}.$$

For the current event setup,

$$L_{\text{event}} = 15\,400\,\text{m}, \qquad T_{\max} = 2100\,\text{s},$$

so

$$v_{\text{avg,req}} = \frac{15400}{2100} = 7.33\,\text{m/s} \approx 26.4\,\text{km/h}.$$

The simulated strategy satisfies the timing constraint if

$$T_{\text{sim}} \leq T_{\max}.$$

The burn-and-coast controller uses speed thresholds and corner lookahead. The control mode is represented as

$$u_i = \begin{cases} 1, & \text{burn mode}, \\ 0, & \text{coast mode}. \end{cases}$$

A simplified version of the switching behavior is

$$u_i = \begin{cases} 0, & v_i > v_{\text{corner,lookahead}} + v_{\text{buffer}}, \\ 1, & v_i < v_{\text{burn,on}}, \\ 0, & v_i > v_{\text{burn,off}}, \\ u_{i-1}, & \text{otherwise}. \end{cases}$$

This logic makes the strategy interpretable: the vehicle burns when it is below the target speed, coasts when it has sufficient speed, and avoids wasting energy before upcoming corners.

2.2.2 Part 2: Live Telemetry & Dynamic Monitoring

The real-time dashboard was developed to make the vehicle's operating state visible during testing and future race operation. It displays fuel-cell status, vehicle dynamics, GPS position, driver inputs, and environmental data through a browser-based interface. The dashboard is built around a Flask server and Socket.IO, allowing telemetry updates to be pushed continuously to connected clients instead of requiring manual page refreshes.

The system supports both simulated and external telemetry data. When no physical data source is connected, the server can generate simulated payloads for dashboard testing and race-strategy visualization. When external data is available, a separate telemetry push script sends JSON packets to the server through the `/api/telemetry` endpoint. The server then broadcasts the received data to the dashboard. This allows the same interface to be used for simulation, logged fuel-cell data, and future live vehicle data.

The telemetry payload is organized into categories such as `h2_status`, `dynamics`, `mpu`, `driver`, `gps`, and `weather`. This structure makes the dashboard extensible: additional sensors can be added later without redesigning the entire interface. In the current implementation, the dashboard also interfaces with the race-strategy layer, so future versions can combine live vehicle data with planned speed targets, track progress, and burn-and-coast guidance.

3 Requirements and Constraints

3.1 Hydrogen Drivetrain

The hydrogen drivetrain part of this project was designed to satisfy the following requirements:

Table 1: Hydrogen Drivetrain Functional Requirements and Completion Status.

Requirement	**Description**	**Status**
Electrical System	Design and build circuits that would allow the vehicle to pass technical inspection and go on the track	Met
Hydrogen System	Design and build a system that would integrate the Horizon fuel cell into the vehicle and could power the vehicle	Partially Met

Table 2: SEM Competition Functional Requirements and Completion Status.

Requirement	Description	Status
Vehicle complete in time for competition	Have all parts of the vehicle complete and working before it has to be loaded in the van and driven to the competition	Partially Met
Transportation, lodging, and meals	Organize transportation, lodging, and meals for all of the team member that are attending the competition and transportation for the vehicle	Met
SEM Registration	Complete all SEM registration steps such as online portal registration, document submission, and correspondence with SEM	Met

3.2 Shell Eco-Marathon Competition

The hydrogen drivetrain part of this project was designed to satisfy the following requirements:

3.3 Telemetry System

3.3.1 Functional Requirements

The software portion of the project was designed to satisfy the following requirements:

Table 3: Functional Requirements and Completion Status.

Requirement	Description	Status
Track processing	Import real track CSV data and convert it into simulation-ready geometry	Met
Track visualization	Display a map of the track and support track-aware analysis	Met
Burn-and-coast simulation	Model acceleration, coasting, speed limits, and energy use	Met
Timing constraint evaluation	Estimate whether the vehicle can complete the event within the required time	Met in simulation
Fuel-cell data plotting	Plot fuel-cell data in real time or near real time	Met
Parameter tuning	Allow vehicle and strategy parameters to be adjusted	Met
Sensor integration	Incorporate all physical sensors into the live dashboard	Partially met
Raspberry Pi deployment	Prepare the system for eventual Raspberry Pi-based operation	Partially met
Full vehicle validation	Test the complete software and telemetry system on the actual race vehicle	Not fully met

3.3.2 Engineering Constraints

The project was constrained by time, available hardware integration progress, and the need to coordinate with other vehicle subsystems. The software needed to remain modular so that simulation data, fuel-cell data, and future live sensor data could be swapped without redesigning the entire dashboard. This was especially important because the mechanical, electrical, and fuel-cell systems were being developed in parallel.

Another constraint was that the official fuel-cell software had overhead and platform limitations. We therefore began developing a custom data-reading approach that could parse available fuel-cell communication data and display it through our own interface.

3.3.3 Cost

The direct additional cost of the software implementation was low because the system was built primarily with existing computers, open-source software tools, and hardware already available to the team. The project cost should therefore be reported in two categories: software development cost and full system-level vehicle cost.

Table 4: Telemetry System Project Cost Table.

Item	Estimated Cost
Software tools and libraries	$ 0
Existing development laptops	$ 0
Raspberry Pi 5 8gb RAM	$ 95
CAN boards and sensor interfaces	$ 60
Low Latency GPS Module	$ 80
12v Accessories Battery	$ 22
dc to dc converter 12v to 5v	$ 30
dc to dc converter 12v to 3.3v	$ 30
5G Pi Hat	$ 262
128gb SD Card	$ 48
Total Telemetry Cost	**$ 672**

4 Methods

4.1 Hydrogen Drivetrain

4.1.1 Safety System

The safety system was designed and built according to the rules provided by SEM. It was then tested by triggering the safety system in various ways to ensure the emergency shutdown system worked properly. This included pressing the emergency shutdown buttons and spraying small amounts of hydrogen on the hydrogen sensors to trigger the sensor.

4.1.2 Hydrogen Leak Detection

To test for hydrogen leaks while at the college, a spray bottle with soapy water was used. This mixture was sprayed on the connection points of the hydrogen tubing and then was allowed to sit. If bubbles began to form, that meant there was a leak and the fittings had to be tightened or adjusted. Paper towel is always held below the fitting that is being tested to prevent fluid from landing on areas that should not come in contact with fluid. At the competition, SEM provided leak detection fluid was used. It is difficult to leak test some of the harder to reach areas without spilling fluid on sensitive areas such as the fuel cell. To test these areas at the competition a hydrogen leak detector was borrowed from another team.

4.1.3 Running the Fuel Cell

There are very specific steps that must be taken to start the fuel cell.

Start-up

1. Open the gas cylinder knob
2. Set regular pressure to 6.0-7.5 psi. NOTE: On the current regulator, 0 psi is 10.0 psi so the regulator must be set to 16.0-17.5 psi. To increasing the pressure, tighten the regulator knob, to decrease the pressure loosen the regulator knob.
3. Turn on the fuel cell by pressing and holding the power button until it beeps

Shutdown

1. Turn off the fuel cell by pressing and holding the button until it beeps
2. Set regulator pressure to 0.0 psi. Zero is when the regulator knob is very loose.
3. Close the gas cylinder knob

4.2 Telemetry System

4.2.1 Part 1: Predictive Race Strategy & Design Synthesis

The race strategy model was implemented as a Python-based simulation pipeline. The first step was to process track data from CSV files containing latitude, longitude, and, when available, elevation. The track-processing script converted the raw GPS coordinates into cumulative distance, resampled the track onto a uniform spatial grid, and estimated grade and curvature. The resulting track representation allowed the simulation to evaluate how the vehicle would behave at each point along the course.

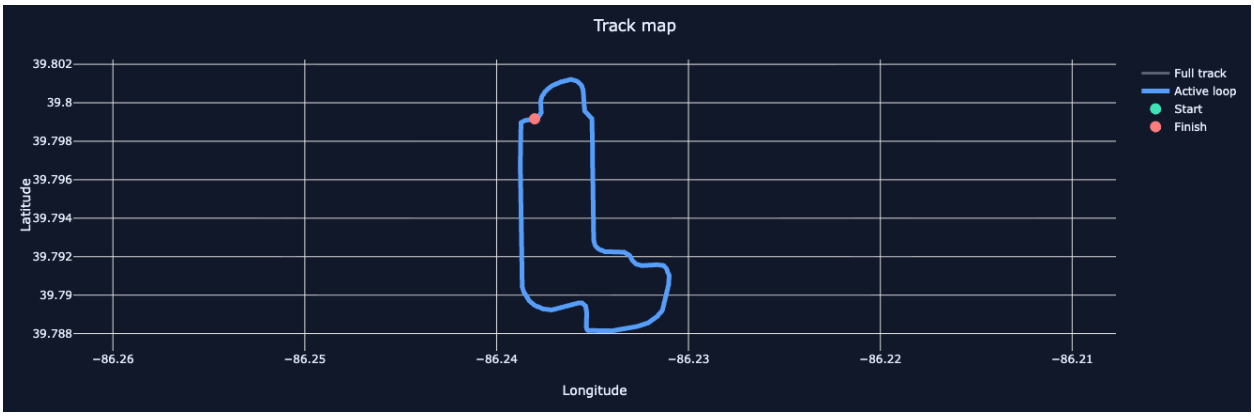


Figure 13: Raw Track Data

The vehicle model used a longitudinal force balance that accounted for aerodynamic drag, rolling resistance, grade resistance, motor force limits, motor power limits, fuel-cell usable power limits, and a curvature-based cornering speed constraint. The track was tiled to match the full race distance, and the vehicle was advanced through the course using a distance-domain update. At each step, the simulation calculated acceleration, speed, elapsed time, fuel-cell power, cumulative fuel-cell energy, and burn/coast state. The technical details and math for this is described in the technical discussion section above.

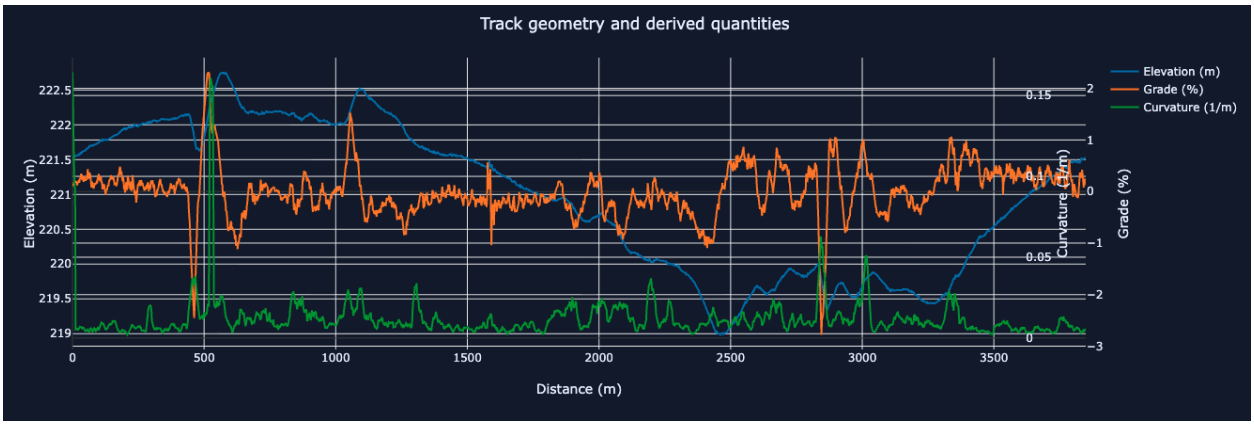


Figure 14: Track Grade, Elevation and Curvature

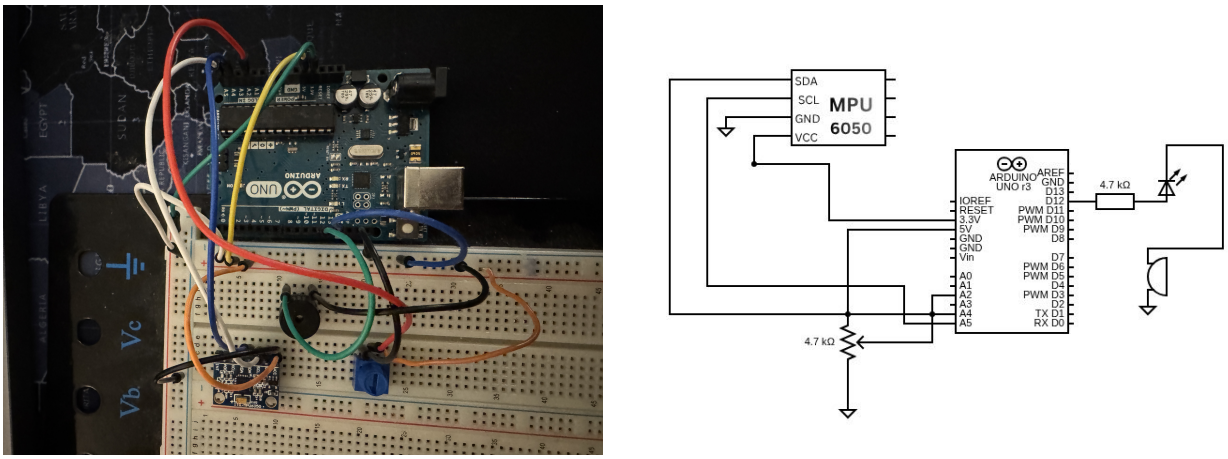
A rule-based burn-and-coast controller was used to represent the intended driving strategy. The controller commanded a burn phase when the vehicle was below the target speed and commanded coasting when the vehicle had sufficient speed or was approaching a corner. This approach was selected because it is interpretable and can be translated into driver cues. The simulation output was then exported as a time-series file containing distance, speed, time, acceleration, fuel-cell power, cumulative fuel-cell energy, burn/coast flag, grade, curvature, and cornering speed limit.

4.2.2 Part 2: Live Telemetry & Dynamic Monitoring

The live telemetry system was designed to provide a real-time dashboard for monitoring the vehicle during testing and eventual race operation. The dashboard was built around a Flask server and Socket.IO, which allowed telemetry values to be streamed continuously to a browser interface. The telemetry payload was organized into categories including fuel-cell status, vehicle dynamics, IMU data, driver inputs, GPS position, and weather information.

The telemetry architecture was designed to support both simulated and external data sources. During software development, the server could generate simulated telemetry for dashboard testing. When external data was available, a separate telemetry push script sent JSON packets to the server through an API endpoint. This made it possible to use the same dashboard for simulation data, logged fuel-cell data, and future live vehicle data.

Because the full telemetry stack was not completely deployed on the car within the available project timeline, a miniaturized Arduino-based prototype was constructed and tested. This prototype served as a bench-scale validation of the dashboard communication path and driver-cue concept. The Arduino system included a potentiometer to emulate steering angle, an IMU interface for dynamic sensing, and a buzzer for burn-and-coast cues. The firmware transmitted newline-delimited JSON frames over serial at a fixed sampling interval. These frames included steering position, gyroscope values, IMU status, and raw potentiometer readings.



(a) Arduino Prototype View

(b) Arduino Prototype Circuit Diagram

Figure 15: Miniaturized Arduino-based telemetry and driver-cue prototype used to test dashboard communication and burn/coast signaling before full vehicle deployment.

The buzzer was used to represent driver guidance from the race strategy layer. In the intended vehicle implementation, a burn cue would instruct the driver to apply propulsion, while a coast cue would instruct the driver to stop applying power and conserve energy. The tested prototype used audible cues through the buzzer, with the future goal of adding an LED indicator where the light is on during burn mode and off during coast mode. This would give the driver a simple visual cue that complements the buzzer which may be muffled when the driver has the helmet on.

5 Results

5.1 SEM Results

5.1.1 Transportation, Lodging, and Meals

Transportation

This year, SEM was at Indianapolis Motor Speedway in Indianapolis, Indiana. We decided to rent a large 7 passenger Crystler Pacifica mini-van and drive the car from Swarthmore to Indianapolis because the drive can be completed easily in a single day. Additionally, Jano (Alexander) is from Indianapolis and had completed this drive several times already and felt comfortable with the team making this drive. This van was selected because it allowed us to fold all of the rear seats into the floor leaving enough room to roll the entire vehicle into the van without having to disassemble any of it, and still allow room for our tools and spare parts. Because with the vehicle in the van there were only two available seats, most of the team flew to Indianapolis. Jano and Kimaru drove the van to Indianapolis, leaving on April 6th around 6:30 am and arriving around 7:30 pm after stopping in Pittsburgh to pick up the pressure relief valve from Swagelok and taking a pit stop at the grand opening of the Huber Heights, Ohio Buc-ee's. The rest of the team arrived in the evening of April 6th. Eric

and Kaw drove the vehicle back from Indianapolis to Swarthmore. They left on April 12th around 9am and arrived to Philadelphia around 10pm. The rest of the team left on April 11th in the afternoon. Overall, transportation to SEM was uneventful and everything went smoothly.

Lodging
The team's lodging was at the Speedway Holiday Inn Express, with 2-3 people per room. Jano stayed at his parents house during the competition week.

Meals
Meals were paid for by the Swarthmore College Office of Student Engagement. Funding was requested for meals through the Student Budgeting Committee. We received a per diem for meals of $15 for breakfast, $18 for lunch, and $20 for dinner. This per diem was deposited directly into our bank accounts by the college after we returned to Swarthmore and was based on the number of meals we were away from the college for.

5.1.2 Competition Schedule

Day 1 – April 7

- **1:30 PM**: Arrive at competition when doors open
- Complete participant and technical registration

Day 2 – April 8 (Arrive by 7:30 AM)

- **8:00 AM**: Mandatory Drivers' Briefing (Student Lounge)
  Compulsory for Driver(s) and Team Manager (Jano and Izzy required)
- **11:30 AM**: Track Walk
- **1:00 PM**: Family Photo
- **2:30–5:00 PM**: Practice Race

Day 3 – April 9 (Arrive by 7:30 AM)

- **8:00 AM**: Mandatory Drivers' Briefing (Student Lounge)
  Compulsory for Driver(s) and Team Manager (Jano and Izzy required)
- **11:00 AM–1:00 PM**: Practice Race *(Did not attend)*
- **4:00–6:30 PM**: Competition Prototype
  Last start: 5:55 PM (Did not attend)

Day 4 – April 10 (Arrive by 7:30 AM)

- **8:00 AM**: Mandatory Drivers' Briefing (Student Lounge)
  Compulsory for Driver(s) and Team Manager (Jano and Izzy required)
- **10:30 AM–1:00 PM**: Competition Prototype
  Last start: 12:25 PM (Did not attend)

- **4:00–6:30 PM**: Competition Prototype
  Last start: 5:55 PM (Did not attend)
- After competition: Dinner at Jano's house (food provided by his parents)

Day 5 – April 11 (Arrive by 7:30 AM)

- **Note**: If flying to Swarthmore today, check out of hotel and place all bags in the car
- **8:00 AM**: Mandatory Drivers' Briefing (Student Lounge)
  Compulsory for Driver(s) and Team Manager (Jano and Izzy required)
- **10:30 AM–1:00 PM**: Competition Prototype
  Last start: 12:25 PM (Attended)
- **1:30 PM**: Depart for airport (if flying to Swarthmore)
- **4:30–5:30 PM**: Awards Ceremony (Student Lounge)

5.1.3 Competition Results

At the end of the competition, we had passed the technical inspection and were able to put our vehicle onto the track. Our vehicle made it about halfway around the first lap before it unfortunately turned off. It appears that the fuel cell encountered an error and turned off, but we are not sure what the error was because there is not good error logging. In the hydrogen prototype division, there were 5 teams. Two teams completed the race, our team completed the technical inspection and entered the track, and the last two teams were unable to enter the track. Despite having the third best record in our division, we unfortunately did not place because we did not complete an entire race on the track.

5.2 Hydrogen Drivetrain

5.2.1 Electric Drive System

Due to being behind schedule, we were unable to test the vehicle driving under its own power on the ground with the driver in it until the night before we had to depart for the race. When we attempted to drive the car, we found that it was hardly able to move under its own weight. Initially, a 24 volt, 25 amp motor controller was being used but because we did not have enough power to drive the car we switched to a 36 volt motor controller. This switch allowed the car to drive under its own power, but very only very slowly. For both motor controllers used, there was a lot of chatter coming from them. After the competition, a new V6 Phaserunner MT controller was purchased from GRIN Technologies. This motor controller is ideal for the vehicle because it can be programmed to match the variable voltage output of the fuel cell and the power demands of the hub motor, allowing for smoother and more efficient motor control. The older motor controllers likely produced a lot of chatter because they used rougher commutation and were not well matched to the load motor, which can cause noisy and jerky operation when the motor is trying to make torque. The V6 Phaserunner MT uses field-oriented control and adjustable current and throttle settings,

which helps the motor run much more smoothly and quietly while also reducing sudden current spikes that can stress the fuel cell.

5.2.2 Electrical System

The final electrical system worked very well. However, there were many iterations and major changes made to its design before we raced.

Right before the competition, our hydrogen sensor became damaged and when it detected hydrogen it would not trip the safety circuit. We narrowed the issue down to a trace that appeared to be burnt up and a faulty relay on the circuit board. Because there was not enough time to order a new sensor before the start of the competition, we decided to order a replacement relay and manually replace the burnt trace with 30 gauge wire. The direct replacement for the relay was a Hamlin HE3621A0510 relay. However, these relays are not produced anymore, so instead a Meder Electronic SIL05-1A72-71D relay was used. Unfortunately, the replacement relay did not solve the issue and the sensor still did not work.

Once at the competition, we asked other teams if they had spare sensors we could purchase from them. A team we asked showed us that using the hydrogen sensor integrated in the Horizon fuel cell would be sufficient to pass SEM technical inspection, which led to a whole host of other discoveries.

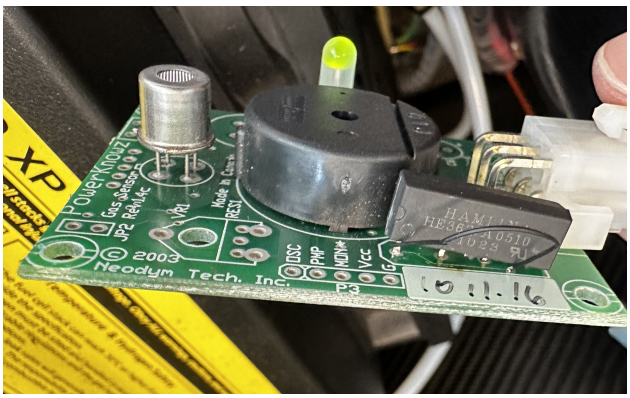


Figure 16: Original Relay with Replacement Trace Hand Soldered onto the Board

Initially, we were not aware that we were able to use the safety system which was integrated with the Horizon fuel cell (emergency stop switches, hydrogen sensor, solenoid valve), Therefore, a much more complex circuit was designed and implemented as shown in Figure 7. This system worked as intended and met all SEM specifications. However, nearly all of the components were redundant. After speaking with the other team about the hydrogen sensor, we realized that we could simplify the entire circuit and use the hydrogen sensor that was integrated with the fuel cell along with the other safety components. The new circuit designed and implemented after this was the final circuit used in the vehicle.

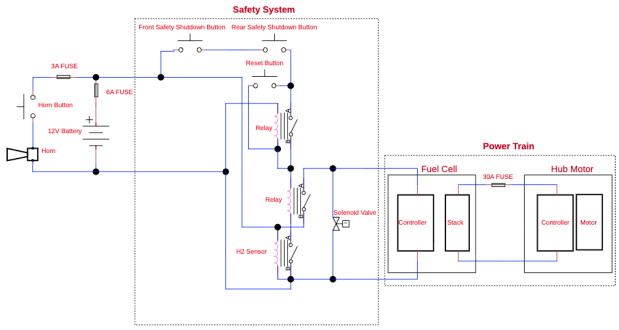


Figure 17: Circuit Diagram of Initial Circuit used in Vehicle

5.2.3 Hydrogen Delivery System

We realized that we had the wrong type of pressure relief valve only a few days before we were slated to depart for the competition. To get the part we needed, we decided to call and email Swagelok stores on the way to Indianapolis so that we could pick one up in person on the way. Luckily, Swagelok Pittsburgh had the pressure relief valves we needed in stock and we stopped in Pittsburgh on the way to Indianapolis to purchase them. Once we got to the competition, we successfully installed the new pressure relief valve.

The hydrogen delivery worked as expected during the competition. However, once at the competition we did have to rebuild parts of it due to failing parts of the technical inspection. Initially, we had an extra solenoid valve that was left over from the first interaction of the safety circuit, so that solenoid valve had to be removed because it was a normally-closed solenoid valve and leaving it in would restrict hydrogen flow. In addition, the initial design had the solenoid valve as the last component of the hydrogen delivery system before the fuel cell. This was against the SEM rules as the solenoid valve had to be immediately after the first pressure regulator. Luckily, moving the location of the solenoid valve was a quick fix. Finally, the other point where we failed the technical inspection was that we did not use teflon tape to seal the conical fitting. In the SEM rules, it states that the use of teflon or other sealing tape is not allowed, but if there is a conical fitting, then you must use teflon tape. We had not added teflon tape to the conical fitting but luckily this was an easy fix.

After fixing all of the issues that were pointed out to us at the first technical inspection, we then failed the second technical inspection due to hydrogen leaks. We had forgotten to leak test the fittings that were connected directly to the fuel cell and one of them was leaking badly. Once this fitting was tightened, the leak issue ended and we passed technical inspection.

5.3 Telemetry System

5.3.1 Part 1: Predictive Race Strategy & Design Synthesis

The predictive race strategy simulation produced a working race plan under the current model assumptions. The simulation successfully processed the track data, applied the longitudinal vehicle model, enforced cornering speed constraints, and generated burn-and-coast

behavior over the event distance. The resulting strategy met the race timing requirement in simulation while limiting fuel-cell traction energy by allowing the vehicle to coast during selected portions of the course.

The primary simulation outputs are the vehicle speed profile, cornering speed limit, burn/-coast state, fuel-cell power demand, cumulative fuel-cell energy, acceleration profile, and track map colored by speed. These plots demonstrate that the model is not simply producing an average-speed estimate; it is responding to track geometry, power limits, and strategy thresholds. Note that these figures represent a single lap to make the values stand out. A full four-lap race will condense the plots, the dashboard is, however, fully interactive and allows users to zoom in for finer details.

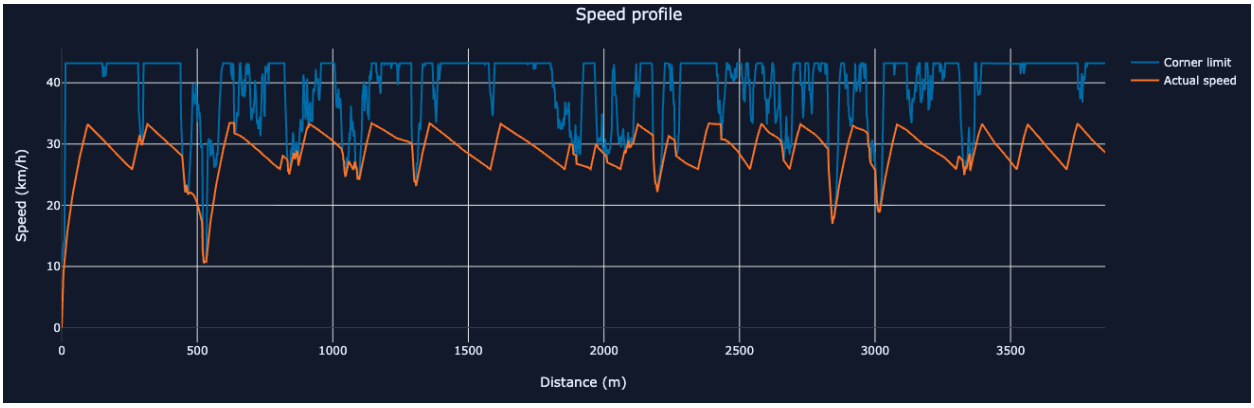


Figure 18: Simulated speed profile compared with the curvature-based cornering speed limit.

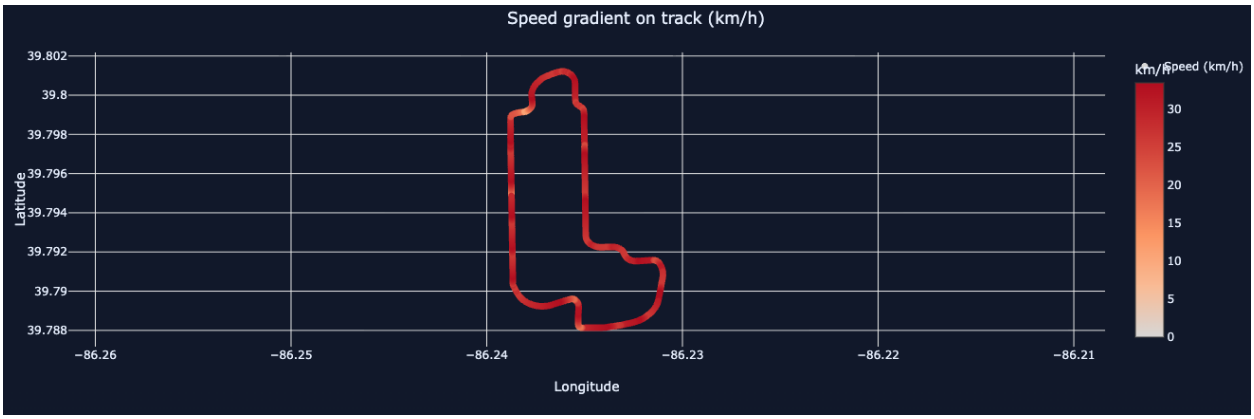


Figure 19: Speed Gradient on Track in km/h

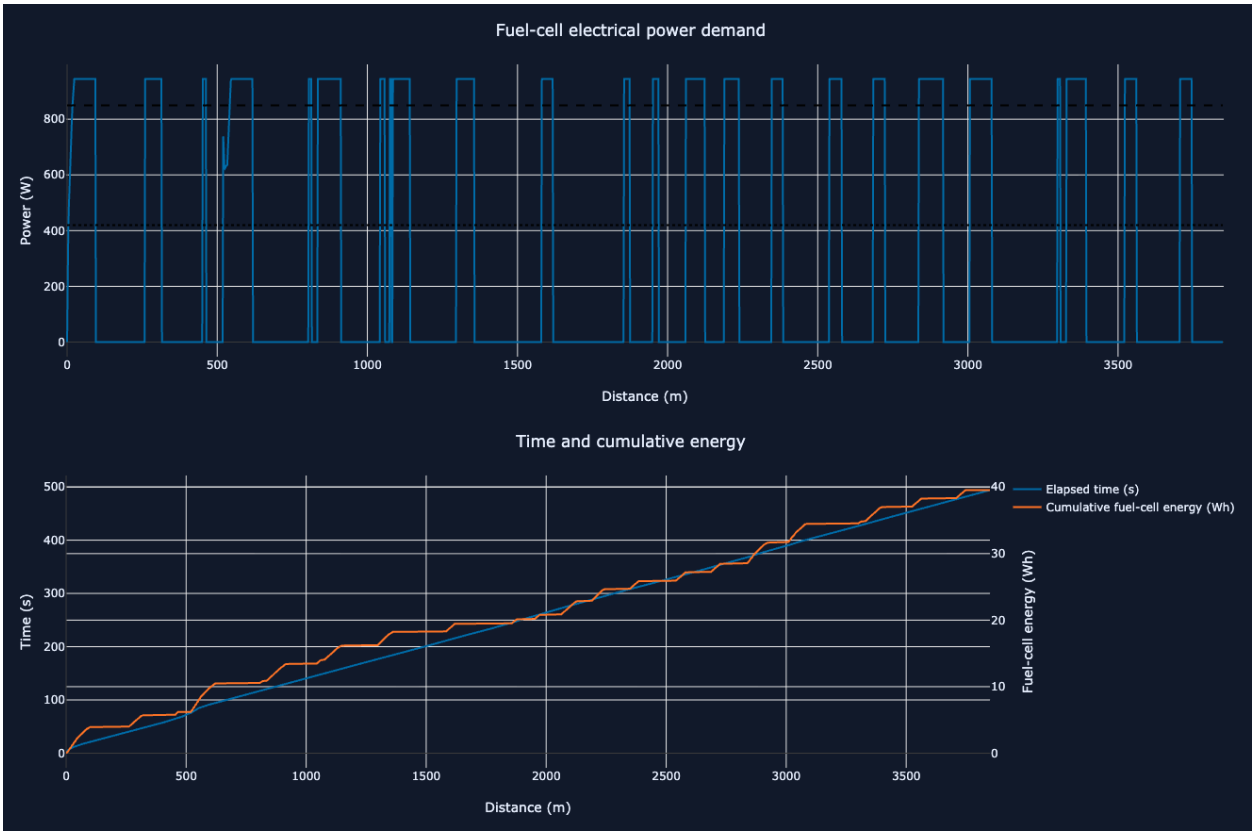


Figure 20: Fuel-cell power demand and cumulative traction energy from the race strategy simulation. Both plots show the burn-and-coast state determined by the controller.

Table 5: Vehicle and Strategy Simulation Parameters

Parameter	**Value**	**Unit**
Vehicle & Drivetrain		
Mass (m)	185	kg
Aerodynamic Drag Area (C_dA)	0.19	m^2
Rolling Resistance (C_{rr})	0.0045	-
Drivetrain Efficiency (η_{drive})	0.90	-
Motor Max Power	1000	W
Fuel Cell Peak Power	850	W
Fuel Cell Preferred Power	420	W
Track & Strategy limits		
Track Name	sem_2023_us	-
Base Track Length	3850	m
Total Event Distance	15,400	m
Event Time Limit	2100	s
Absolute Speed Cap	12.0	m/s
Max Lateral Acceleration	1.4	m/s^2
Burn-On Speed Threshold	7.2	m/s
Burn-Off Speed Threshold	9.2	m/s
Corner Lookahead Distance	55	m
Corner Speed Buffer	0.8	m/s
Pace Bias	1.0	-

Using these defined vehicle constraints and strategy thresholds, the predictive simulation yields the core performance and timing results detailed in the table below.

Table 6: Essential Race Strategy Simulation Results

Parameter / Metric	**Simulated Value**	**Unit**
Total Track Distance	15,400	m
Total Elapsed Time (T_{sim})	*1943.32*	s
Finish Margin	*156.68*	s
Average Speed	*7.93*	m/s

Constraint Satisfaction

The simulation confirms that the current vehicle model successfully meets the Shell Eco-Marathon (SEM) operational constraints. By completing the $15,400$ m event distance in under the $2,100$ s limit ($T_{sim} \leq 2100$ s), the burn-and-coast controller successfully maintains an average speed safely above the 7.33 m/s requirement without violating local track curvature limits (v_{corner}).

5.3.2 Part 2: Live Telemetry & Dynamic Monitoring

The real-time telemetry and dynamic monitoring system operated successfully as independent development elements, validating the core software architecture even though the complete system was not fully deployed on the final vehicle. The dashboard correctly displayed simulated race-strategy behavior (including the predictive burn-and-coast strategy) and accepted externally pushed telemetry through the API pathway.

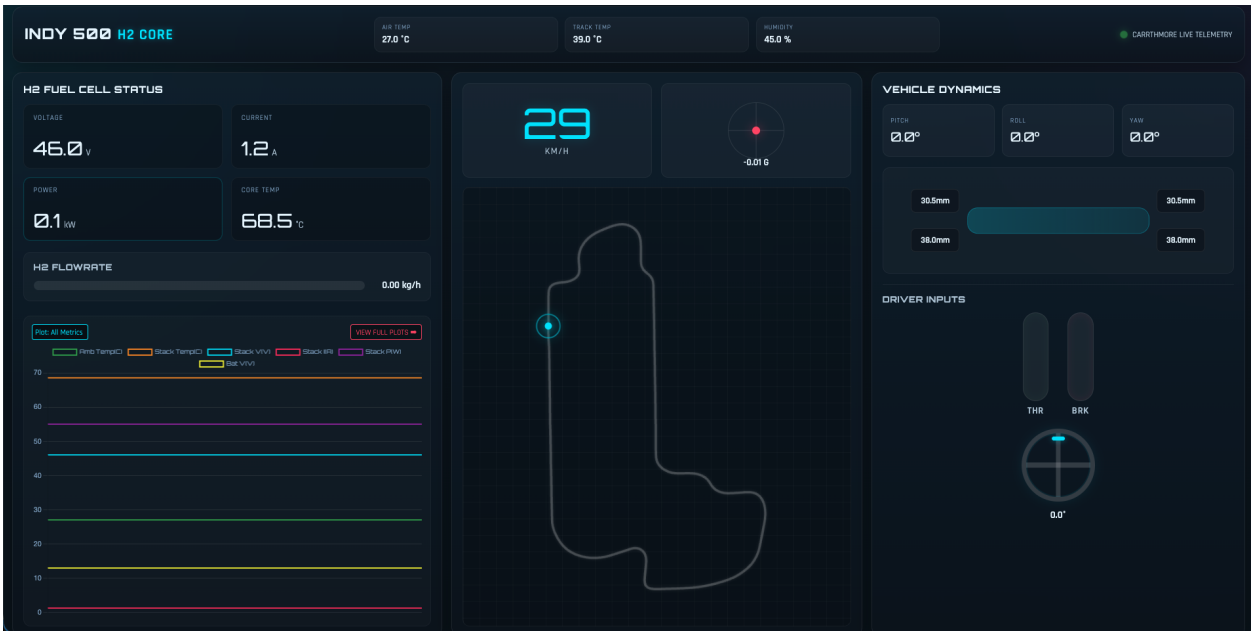


Figure 21: Race Dashboard Containing Fuel Cell Current Status, Vehicles Track Position, Speed and Dynamics

A major empirical success was the collection and real-time visualization of actual fuel-cell data through the RS232-to-USB connection. The dashboard successfully received and plotted real measurements from the hydrogen fuel-cell system, including stack temperature, stack voltage, stack current, stack power, ambient temperature, and battery voltage. Furthermore, the miniaturized Arduino-based prototype served as a successful bench-scale validation platform, successfully collecting MPU6050 sensor data, receiving potentiometer steering inputs, and triggering audio-based burn-and-coast driver cues.

Figure 22: Dedicated Fuel Cell Data Plot Showing Ambient and Stack Temp, Stack Voltage, Stack Current, Stack Power, and Battery Voltage

5.3.3 Requirements Evaluation and System Cost

While the core simulation, data visualization, and bench-scale prototype requirements were met, full vehicle validation and comprehensive sensor integration were only partially achieved. Specifically, live GPS integration, throttle and brake mapping, full CAN-based sensor integration, and direct flow-meter integration were not completed. These requirements were ultimately unmet due to a combination of strict time constraints, software driver compatibility issues, fluid changes to the physical vehicle hardware (specifically the brake design), and limited documentation for certain proprietary components.

Despite these physical integration hurdles, the standalone telemetry and simulation software was completed efficiently. By utilizing existing development laptops and open-source software tools, the final total cost of the telemetry implementation was kept to $672.

6 Discussion

6.1 Systems Engineering and Future Design

The development of the hydrogen vehicle's telemetry system highlighted the critical realities of systems engineering and hardware-software co-design. While the project successfully validated its real-time data visualization and predictive strategy modeling through a miniaturized Arduino prototype, the inability to fully deploy the system on the vehicle underscored the necessity of adaptable scoping and early integration planning. Future E90 teams must co-define mechanical, electrical, and telemetry requirements collaboratively from day one

rather than treating data acquisition as a secondary layer bolted onto a finished chassis. Ultimately, alongside early sensor integration, the telemetry simulation data strongly indicates that future iterations of the team should prioritize a more aerodynamic vehicle body design alongside powertrain optimization to maximize overall efficiency gains.

6.2 Recommendation for Next Competition

It is highly recommended that a new regulator is purchased as 0 pressure for the current regulator reads 11 psi rather than zero, and there seems to be no way to adjust this. In addition, to have better control over the outflow pressure, a dual stage regulator should be purchased rather than a single stage. It is also recommended to purchase extra L-shaped quick-connection connectors that are attached directly to the fuel cell. We discovered that several teams at the competition were having issues with these connectors and were unable to pass technical inspection due to leaks that stemmed from these. The fittings are unusual sizes, and it is nearly impossible to find new ones apart from buying them directly from Horizon Fuel Cells. However, an alternative could be to not use the quick-connectors and teflon tubing at all and use stainless steel tubing for the entire system.

7 Conclusion

The primary objective of this project was to design, build, and successfully race a prototype hydrogen vehicle in the Shell Eco-Marathon (SEM) Americas competition, marking Swarthmore College Engineering's first appearance in the event since 2018. To accomplish this, the team developed and integrated several critical subsystems, including a proton exchange membrane (PEM) hydrogen fuel cell drivetrain, a robust electrical and safety system, and a custom telemetry platform.

While the project faced significant physical integration and hardware challenges, the team achieved several vital milestones and identified clear paths for future improvement:

- **Drivetrain Adaptations:** After discovering the vehicle struggled to move under its own weight, the team successfully upgraded to a 36-volt motor controller to enable driving under its own power. To resolve rough commutation and motor chatter, a V6 Phaserunner MT controller was purchased post-competition to better match the fuel cell's variable voltage output.
- **Telemetry and Race Strategy Validation:** Developed using existing tools to keep costs low at $672, the standalone telemetry and predictive simulation software was a major success. The model successfully generated a track-aware, burn-and-coast operating plan capable of completing the 15,400 m event distance in under the 2,100 s time limit.
- **Data Visualization Prototyping:** Although full sensor deployment on the physical vehicle was hindered by time constraints and software compatibility issues, a miniaturized Arduino prototype successfully validated the real-time data dashboard, live fuel-cell monitoring, and driver-cue systems.

Ultimately, the development of this hydrogen fuel cell vehicle highlighted the critical necessity of hardware-software co-design. For future SEM competitions, it is strongly recommended that teams collaboratively define mechanical, electrical, and data acquisition requirements from the very beginning of the design process. By coupling these early integration strategies with specific hardware improvements—such as transitioning to dual-stage regulators, eliminating leak-prone quick-connectors in favor of continuous stainless steel tubing, and optimizing the aerodynamic body design—future iterations of the vehicle will be well-positioned to maximize fuel efficiency and competitive success.

Acknowledgements

We would like to extend our sincere gratitude to Matthew Shiley, Nora Jiang, Eric Chen, Kaw Moo, Kinley Zangmo, Isabella Hemler, Jeff Liang, Nathan Schwartz, Kayhaan Mohammed, and Magnus Julin for their invaluable assistance with the chassis design and the assembly of the vehicle's brake system, as well as for their wonderful camaraderie throughout the process. We are also deeply appreciative of the SBC for funding the club that spearheaded this project, and supporting our trip to Indianapolis for the competition.

References

[1] Schmid Elektronik, Altair, and Southwest Research Institute, "Improving Teams On-Track Performance through data, MVP and Knowledge-Driven Methods," presented at the Shell Eco-marathon Virtual Learning Sessions, Feb. 2022.

[2] M. Schmid, "Data & Telemetry Bootcamp Part 1/2: Craft Your Winning Edge: Simple Strategies for Instant Impact!," presented at the Shell Eco-marathon Data & Telemetry Bootcamp, Feb. 2025.

[3] A. Ostergaard, J. Taylor, and B. Wood, "Burn and coast method timing optimization for BYU Supermileage vehicle," Brigham Young University, Tech. Rep. 575.

A Representative Simulation Implementation

This appendix includes selected code excerpts from the race strategy simulator. The snippets are included to show how the mathematical model described in the report was implemented in software.

A.1 Vehicle and Strategy Parameters

The simulator stores the main vehicle and race-strategy assumptions in dataclasses. These parameters define the mass, drag area, rolling resistance, drivetrain efficiency, motor limits, fuel-cell power limit, cornering limit, race distance, time limit, and burn-and-coast thresholds.

Listing 1: Vehicle and strategy parameters used by the simulator.

```python
@dataclass
class VehicleParams:
    mass_kg: float = 185.0
    rho_air: float = 1.225
    crr: float = 0.0045
    cda_m2: float = 0.19
    drive_efficiency: float = 0.90
    motor_power_max_w: float = 1000.0
    motor_force_max_n: float = 190.0
    accel_max_mps2: float = 0.42
    brake_max_mps2: float = 0.90
    lat_accel_limit_mps2: float = 1.40
    abs_speed_cap_mps: float = 12.0
    fuel_cell_peak_usable_power_w: float = 850.0
    g: float = 9.81


@dataclass
class StrategyParams:
    event_distance_m: float = 15400.0
    event_time_limit_s: float = 2100.0
    burn_on_threshold_mps: float = 7.2
    burn_off_threshold_mps: float = 9.2
    corner_lookahead_m: float = 55.0
    corner_buffer_mps: float = 0.8
    pace_bias: float = 1.0
```

A.2 Resistive Force Model

The simulator calculates the total resistive force as the sum of aerodynamic drag, rolling resistance, and grade resistance. This directly implements the force balance used in the mathematical model.

Listing 2: Resistive force calculation used in the longitudinal model.

```python
def resistive_force(v: float, grade: float, p: VehicleParams) ->
   float:
    theta = np.arctan(grade)

    f_aero = 0.5 * p.rho_air * p.cda_m2 * v * v
    f_roll = p.mass_kg * p.g * p.crr * np.cos(theta)
    f_grade = p.mass_kg * p.g * np.sin(theta)

    return float(f_aero + f_roll + f_grade)
```

A.3 Cornering Speed Limit

The simulator uses track curvature to impose a local speed limit. This prevents the simulated vehicle from unrealistically carrying too much speed through tight corners.

Listing 3: Cornering speed limit from track curvature.

```
kappa = track['curvature_1pm']

v_corner = np.where(
    kappa > 1e-6,
    np.sqrt(np.maximum(0.0, p.lat_accel_limit_mps2 / kappa)),
    p.abs_speed_cap_mps
)

v_corner = np.clip(v_corner, 0.0, p.abs_speed_cap_mps)
```

A.4 Burn-and-Coast Simulation Loop

The main loop applies the burn-and-coast controller, computes drive force, updates acceleration and speed, and integrates fuel-cell traction energy.

Listing 4: Main simulation loop for burn-and-coast race strategy.

```
target_avg = (sp.event_distance_m / sp.event_time_limit_s) * sp.
    pace_bias
burning = True

for i in range(n - 1):
    dsi = max(ds[i + 1], 1e-3)

    look_idx = min(
        n - 1,
        i + max(1, int(sp.corner_lookahead_m / dsi))
    )
    future_corner_limit = np.min(v_corner[i:look_idx + 1])

    if v[i] > future_corner_limit + sp.corner_buffer_mps:
        burning = False
    elif v[i] < min(sp.burn_on_threshold_mps, target_avg + 0.2):
        burning = True
    elif v[i] > max(sp.burn_off_threshold_mps, target_avg + 0.8):
        burning = False

    f_res = resistive_force(v[i], track['grade'][i], p)

    if burning:
        drive_power = min(
            p.motor_power_max_w,
```

```
            p.fuel_cell_peak_usable_power_w
        )
        f_drv = min(
            p.motor_force_max_n,
            drive_power / max(v[i], 0.5)
        )
    else:
        f_drv = 0.0

    a = np.clip(
        (f_drv - f_res) / p.mass_kg,
        -p.brake_max_mps2,
        p.accel_max_mps2
    )

    v_next = np.sqrt(max(0.0, v[i] ** 2 + 2 * a * dsi))
    v_next = min(v_next, v_corner[i + 1], p.abs_speed_cap_mps)

    dt = dsi / max(0.15, 0.5 * (v[i] + v_next))
    mech = max(0.0, f_drv * v[i])
    elec = mech / p.drive_efficiency if mech > 0 else 0.0

    v[i + 1] = v_next
    t[i + 1] = t[i] + dt
    p_elec[i] = elec
    e_j[i + 1] = e_j[i] + elec * dt
    accel[i] = a
    burn[i] = int(burning)
```

A.5 Simulation Outputs

The simulation exports the key quantities used in the results plots: speed, time, fuel-cell power, cumulative fuel-cell energy, acceleration, burn/coast flag, cornering speed limit, grade, and curvature.

Listing 5: Time-series outputs exported by the simulator.

```
df = pd.DataFrame({
    'distance_m': s,
    'speed_mps': v,
    'time_s': t,
    'fuel_cell_power_w': p_elec,
    'fuel_cell_energy_j': e_j,
    'accel_mps2': accel,
    'burn_flag': burn,
    'corner_speed_limit_mps': v_corner,
    'grade': track['grade'],
    'curvature_1pm': track['curvature_1pm'],
```

```
12 })
```