5.0 Requirements Specification

By Leo Dai, Lawrence Benitez, Aiden McDougald, Conner-Ryan Petersen, Evan Steinhoff

5.1 Introduction

Our software will be used by an electric formula SAE car to control the vehicle. This is similar to the ECU for existing electric cars without the consumer features. The software consists mainly of the Central Control Unit that communicates with sub-systems to change and manage car dynamics. There will also be an out-of-car database that records all the telemetry on the track. This telemetry will include tire temps, steering angle, pedal positions, active suspension settings, speed, GPS location, relative angle of the car, and more if we get the funding. The project uses a Raspberry Pi 5 (Central Control Unit), several Arduino Uno 4s (Sub-Systems), and another Raspberry Pi TBD (For the steering wheel UI). These units will control different aspects of the car such as active suspension, pedal telemetry, and steering wheel UI. Modules will be tied to their specific hardware applications. For example, the Active Suspension Unit will simulate suspension for the tire it is attached to, only communicating to the Central Control Unit for sensor data. The end user of our product is the Formula Race Club. This includes the driver in the car (we want them to be able to effectively control and monitor the car on the fly) and the team around the driver (which needs access to the latest telemetry to also effectively manage and coach the driver).

5.2. CSCI Component Breakdown

5.2.1 Trackside App CSCI

- The Trackside App (CSCI) is responsible for collecting, processing, and displaying data related to system performance and operations. It gathers information from various sensors and transmits it to a central location for monitoring and analysis.

5.2.1.1 Trackside GUI CSC

- The Trackside Graphical User Interface (GUI) Computer Software Component (CSC) provides the user interface for the telemetry application. It allows users to interact with the system by viewing live data, historical trends, and diagnostic information in a visually accessible way.

5.2.1.1.1 GUI Main Screen CSU

- The GUI Main Screen Computer Software Unit (CSU) is the primary screen displayed to users, serving as the central hub for accessing different telemetry data categories, controlling the system, and receiving real-time updates

5.2.1.1.1.1 GUI button module

- The GUI button module refers to a specific software unit that handles button functionalities within the telemetry GUI. It controls user inputs, such as navigation between different screens, activating system commands, or executing data refreshes based on user actions.

5.2.1.1.2 Telemetry Data Collection CSU

- This unit is responsible for gathering real-time data from all vehicle sensors, such as speed, engine metrics, tire pressure, suspension, and more. It interfaces directly with the car's CAN Bus system, collecting and forwarding telemetry data for processing and analysis.

5.2.1.1.2.1 Sensor Interface Module

- Interfaces with vehicle sensors (speed, engine, tire pressure, etc.) to gather real-time data.

5.2.1.1.2.2 CAN Bus Data Module

- Collects data from the car's internal CAN Bus system for use in telemetry.

5.2.1.1.2.3 Data Aggregation Module

- Aggregates data from multiple sources into a unified format for processing and analysis.

5.2.1.1.2.4 Error Detection & Handling Module

- Detects and manages errors or inconsistencies in sensor data.

5.2.1.1.3 GPS System CSU

- This unit handles the integration of GPS data, providing real-time tracking of the vehicle's position on the race track. It records location, speed, and acceleration/deceleration patterns, which can be used for strategy and performance evaluation.

5.2.1.1.3.1 GPS Data Collection Module

- Collects real-time GPS data, including vehicle position, speed, and heading.

5.2.1.1.3.2 Track Mapping Module

- Maps the GPS data onto a virtual representation of the race track to monitor position.

5.2.1.1.3.4 Lap Tracking Module

- Uses GPS data to calculate lap times and track the number of completed laps.

5.2.1.1.4 Data Visualization CSU

- This unit takes the data obtained through various sensors and other CSUs and creates graphs/visualizations that are easy to read and understand.

5.2.1.1.4.1 Real-Time Data Display Module

- Converts raw telemetry data into visual elements (graphs, charts, tables) that update in real time.

5.2.1.1.4.2 Customizable Dashboard Module

- Allows users to customize which data is displayed based on preferences or role (engineer, driver, etc.).

5.2.1.1.4.3 Historical Data Visualization Module

- Displays trends and comparisons over time, such as lap times and fuel consumption.

5.2.1.1.4.4 Visual Alert Display Module

- Provides visual indicators for alerts, such as flashing icons or highlighted data when thresholds are crossed.

5.2.1.1.5 Alarm & Notification System CSU

- This CSU generates alerts and notifications when specific telemetry data crosses pre-defined thresholds, such as high engine temperature, low energy, or tire pressure issues. Visual

and audio alerts ensure the driver and team are aware of potential issues in real time.

5.2.1.1.5.1 Threshold Monitoring Module

- Continuously monitors telemetry data for predefined thresholds, like temperature or tire pressure.

5.2.1.1.5.2 Visual Alarm Module

- Triggers visual warnings on the dashboard (flashing icons, colored indicators) when a threshold is exceeded.

5.2.1.1.5.3 Audio Alert Module

- Emits sound alarms for critical issues, like engine overheating or low fuel levels.

5.2.1.1.5.4 Alert Logging Module

- Logs when alarms are triggered and resolved, creating a historical record of all alerts.

5.2.1.1.6 Driver Performance Interface CSU

- This unit provides the interface through which spectators receive real-time telemetry data about the driver’s performance including lap times, speed, throttle, and brake inputs.

5.2.1.1.6.1 Lap Time Display Module

- Displays the driver's current, previous, and best lap times in real-time.

5.2.1.1.6.2 Throttle & Brake Input Module

- Tracks and displays the driver’s throttle and brake inputs for analysis.

5.2.1.1.6.3 Speed & Acceleration Data Module

- Shows real-time speed and acceleration data relevant to the driver’s performance.

5.2.1.1.6.4 Spectator Display Module

- Provides a simplified version of driver performance data designed for spectator viewing.

5.2.1.1.7 Race Strategy Tools CSU

- This CSU includes tools that help optimize race strategy based on real-time telemetry data, such as lap time prediction and energy consumption tracking. It analyzes current performance and suggests adjustments to race strategy.

5.2.1.1.7.1 Lap Time Prediction Module

- Analyzes current performance and predicts upcoming lap times based on telemetry data.

5.2.1.1.7.2 Energy Consumption Tracking Module

- Monitors real-time fuel or energy usage and estimates remaining levels throughout the race.

5.2.1.1.7.3 Pit Stop Optimization Module

- Recommends optimal times for pit stops based on tire wear, fuel/energy levels, and race conditions.

5.2.1.1.7.4 Performance Adjustment Module

- Suggests real-time adjustments (e.g., speed, fuel strategy) based on track conditions and competitor data.

5.2.1.2.10 Environmental Data Integration CSU

- This unit integrates external environmental data, such as weather conditions (temperature, wind, humidity), to optimize vehicle settings and strategy based on changing track and environmental conditions during the race.

5.2.1.2.10.1 Weather Data Collection Module

- Gather real-time weather data (temperature, humidity, wind speed) from external sources.

5.2.1.2.10.2 Track Condition Monitoring Module

- Analyzes weather data to estimate track conditions (e.g., dry, wet, or slippery).

5.2.1.2.10.3 Environmental Data Display Module

- Displays relevant environmental data on the telemetry dashboard.

5.2.1.2.10.4 Strategy Adjustment Module

- Suggests real-time strategy adjustments based on changes in weather or track conditions.

5.2.1.2.12 Calibration & Diagnostics CSU

- This unit provides tools for calibrating the sensors and the telemetry system before the race, ensuring accurate data collection. It also includes a diagnostics mode for pre-race checks and troubleshooting to confirm that all systems are functioning correctly.

5.2.1.2.12.1 Sensor Calibration Module

- Provides tools to calibrate vehicle sensors to ensure accurate telemetry data collection before each race.

5.2.1.2.12.2 Diagnostics Mode Module

- Runs pre-race diagnostic checks on sensors and systems to detect potential issues.

5.2.1.2.12.3 Self-Test Module

- Automates periodic self-tests of the telemetry system to ensure ongoing reliability during races.

5.2.1.2.12.4 Post-Race Diagnostics Module

- Reviews data and sensor performance after the race to detect any anomalies or performance issues during the event.

5.2.1.2 Database CSC

- The Database CSC is responsible for handling the data collection of telemetry data from the CCU and then storing the data in a parsable way for the Telemetry GUI to access.

5.2.1.2.4 Wireless Data Transmission CSU

- This CSU is in charge of transmitting telemetry data wirelessly from the racecar to the telemetry app during a race. It ensures low-latency communication via reliable wireless protocols (Wi-Fi, RF), enabling real-time monitoring from remote locations.

5.2.1.2.5 Data Logging CSU

- The data logging CSU handles real-time storage of telemetry data, both locally on the vehicle and remotely on cloud servers. This allows for both real-time analysis and post-race review, storing metrics such as lap times, vehicle diagnostics, and performance metrics.

5.2.2 Active Suspension CSCI

- The Active Suspension CSCI simulates the traditional mechanical suspension through a motor-driven bolt system. It also actively adjusts the suspension to maintain the traction of the attached wheel.

5.2.2.1 Suspension Motor Controller CSC

- The suspension motor controller controls the brushless DC motor attached to the active suspension CSCI. This component is also responsible for the precision motor control needed for the suspension to be as sharp as possible.

5.2.2.1.1 Motor Speed Controller CSU

- This unit is responsible for determining the velocity change of the motor attached to it. This unit receives data from the suspension data and determines the appropriate adjustments needed.

5.2.2.1.1.1 Actuator Output Estimater module

- This module takes in data from the Motor Rotation-Angle Sensor module to estimate/calculate the actuator output of the electromagnetic actuator.

5.2.2.1.1.2 Actuator Inertia Force Calculator module

- This module calculates the internal inertia force of the electromagnetic actuator based on the suspension stroke acceleration of the electromagnetic actuator.

5.2.2.1.1.3 Motor Output Calculator module

- This module calculates the motor output from the sum of the electromagnetic actuator output and the internal inertia force of the electromagnetic actuator

5.2.2.1.1.4 Required Motor Output Calculator module

- Using the current motor output calculated we can then adjust the motor output to match a predetermined map.

5.2.2.1.1.5 Motor Output Controller module

- After calculating the appropriate output of the motor required this module sends a signal to the motor to match it.

5.2.2.1.2 Suspension Data Receiver CSU

- This unit is responsible for receiving the data needed for the Suspension Motor Controller. This includes but is not limited to vertical acceleration data, acceleration data, motor speed, and wheel toe.

5.2.2.1.2.1 Level Sensor module

- Receives signals from a level sensor and parses through the data for the Motor Speed CSU

5.2.2.1.2.2 Motor Rotation-Angle Sensor module

- Receives signals from a motor rotation-angle sensor and then parses through the data for the Motor Speed CSU

`

5.2.2.1.2.3 CCU Data Parser module

- Receives data from the CCU for vehicle data such as vertical acceleration and parses through the data for the Motor Speed CSU

5.2.3 Steering/Pedal Unit CSCI

- The four sensors for the accelerator and brake capture analog signals (measuring voltage) and convert them into digital binary data, representing the corresponding range of input

5.2.3.1 Input Observer CSC

- Monitors the steering and pedal inputs and sends that data to the CCU. It also monitors for errors and send the appropriate messages to the CCU.

5.2.3.1.1 Steering Reader CSU

- converts analog signals from the steering system into digital data for precise interpretation of steering inputs. Send this data into CCU

5.2.3.1.1 Steering Angle Sensor module

- Gets sensor data about the steering angle of the wheel for the steering reader

5.2.3.1.2 Pedal Observer CSU

- The Pedal Observer converts analog signals from the accelerator and brake sensors into binary data. It ensures accurate interpretation of pedal inputs for the vehicle's control system.

5.2.3.1.1 Throttle Pedal Sensor module

- Gets sensor data from the throttle pedal to determine how hard the driver is pressing on the pedal.

5.2.3.1.2 Brake Pedal Sensor module

- Gets sensor data from the brake pedal to determine how hard the driver is pressing on the pedal.

5.2.3.1.3 CCU Informer CSU

- The CCU Informer sends that data about steering and pedal inputs to the CCU

5.2.3.1.3.1 Data Packager module

- Prepares the data about steering wheel angle, throttle pedal depression, and break pedal depression to send to the CCU

5.2.3.1.4 Pedal Safety CSU

- monitors the pedal sensor inputs for anomalies or faults. It ensures safe operation by triggering corrective actions or alerts if abnormal pedal behavior is detected.

5.2.3.1.4.1 Error Monitor module

- Puts the data from the Input Observer CSC through tests to determine if any of the inputs seem to deviate from

desired inputs. For example, if the throttle input and break inputs are both at 100% we know that something is wrong.

5.2.4 Steering Wheel Unit CSCI

- The Steering Wheel Unit is housed inside the vehicle's steering wheel. It conveys information to the driver through a Raspberry Pi connected to an LCD Display. The driver will be able to control the steering wheel unit through buttons and knobs connected to the steering wheel.

5.2.1.1 Graphics Interface CSC

- The Graphics Interface will be an LCD display connected to a Raspberry Pi.

5.2.1.2 Steering Wheel Unit Processor CSC

- The Processor will be a Raspberry Pi running Linux. The processor will be connected to the CCU through wifi. The CCU will send information to the Raspberry Pi, and the Pi will send inputs to the CCU. The processor will generate the graphics and information that is shown on the Graphics Components CSC.

5.2.5 Central Control Unit CSCI

- The Central Control Unit controls and manages all data and subsystems inside the car.

5.2.5.1 Data Management CSC

- Sends and receives data from the different CSCI

5.2.5.1.1 I2C Bus CSU

- Manages an I2C bus for the majority of sensors and makes sure to receive inputs and send it to the CCU

5.2.5.1.1.1 Round Robin module

- uses a round-robin approach to collect data from a variety of sensors

5.2.5.1.1.2 GPS receiver module

- Collects data from a GPS

5.2.5.1.1.3 Thermal camera module

- Collects thermal camera data

5.2.5.1.1.4 Gyroscope module

- Collects gyroscopic data from a rotational sensor

5.2.5.1.1.5 Acceleration sensor module

- Collects directional acceleration data from an accelerometer

5.2.5.2.1 Wireless CSU -- Manages wireless connection and upkeep

5.2.5.2.1.1 SWU Com module

- Send and receive signals from the Steering Wheel Unit CSCI.

5.2.5.2.1.2 Database Com module

- Sends telemetry data to the database

5.2.5.2 Vehicle Stability CSC

5.2.5.2.1 Traction Control CSU

- The Traction Control CSU will use data from the electric motors to determine if the drive wheels are slipping. If a drive wheel is slipping, the central control unit will limit the power sent to that wheel until the slipping stops. The limited power will be controllable by the driver through the steering wheel unit.

5.2.5.2.1.1 Loss of Traction Detector module

- The Loss of Traction Detector determines whether traction

5.2.5.2.2 Virtual Differential CSU

- The Virtual Differential CSC will use the data from the steering wheel angle and accelerometer to determine how much power to send to the rear left and right drive wheels independently. While the steering wheel is turned, more power will be given to the rear drive wheel on the outside of the turn, proportionally to the angle the steering wheel is turned.

5.2.5.2.2.1 Wheel Distance Calculator module

- Determines how much more a wheel travels based on the steering angle of the car. This also takes into account whether the car has lost traction or not.

5.2.5.2.2.2 Motor Power Distributor module

- Distributes the power of each motor based on the ratio of the Wheel Distance Calculator module.

5.2.5.2.3 Suspension Kinematics CSU

- The Suspension Kinematics CSU will control the vehicle's active suspension system. Using data from the accelerometer, the central control unit will calculate how stiff each wheel's suspension should be, and send this info to each active suspension system controller.

5.2.5.2.3.1 Stiff/Soft Determinator Module

- The Stiff/Soft Determinator will determine which parts of the suspension need to be stiff and which parts need to be soft based on the steering wheel input and the throttle and brake inputs.

5.2.5.2.3.2 PID Distributor Module

- The PID Distributor will send the proper PID values to each active suspension unit based on how stiff/soft it needs to be which is calculated by the Stiff/Soft Determinator

5.2.5.3 Safety Check CSC

- The Safety Check CSC will ensure that all systems are active and that we are receiving a constant data flow per allocated amount of time. It also makes sure to cut power to the electric drive motors when any faults are detected.

5.2.5.3.1 Data Flow Checker CSU

- The Data Flow Checker checks the last time each data from each sensor has been received to determine if there is constant data flow or not.

5.2.5.4 Power Draw Monitor CSC

- The Power Draw Monitor CSC will give us a precise number of how much power we are drawing for telemetry purposes.

5.2.5.4.1 Power Estimator CSU

- Estimates the power we are using the car as a whole. Also maintains the power usage for each component.

5.2.5.4.1.1 Motor Power Estimator module

- Estimates the power needed for each of the electric motors.

5.2.5.4.1.2 Hardware Power Estimator module

- Estimates the power needed for each of the hardware components of the car.

5.3 Functional Requirements by CSC

5.3.1 Telemetry App CSCI

5.3.1.1 Telemetry GUI CSC Functional Requirements

- 5.3.1.1.1 The Telemetry GUI shall display real-time telemetry data
  The Telemetry GUI must present real-time vehicle telemetry data, including speed, tire pressure, suspension status, GPS location, and other critical performance metrics, in a user-friendly and intuitive manner.
- 5.3.1.1.2 The Telemetry GUI shall allow customization of displayed data
  Users must have the ability to customize which metrics are displayed on the dashboard. This could include real-time speed, suspension data, throttle inputs, and lap times, based on their preferences or roles (driver, engineer, etc.).
- 5.3.1.1.3 The Telemetry GUI shall provide historical data visualization
  The system must be able to access and display historical telemetry data, enabling users to review and analyze past performance. This includes visual representation in the form of graphs and charts for trends like lap times and tire wear.
- 5.3.1.1.4 The Telemetry GUI shall support real-time alerts and notifications
  The GUI must include visual alerts that notify users when telemetry data exceeds predetermined thresholds, such as overheating, low tire pressure, or critical energy levels.

5.3.1.2 Database CSC Functional Requirements

- 5.3.1.2.1 The Database shall store real-time telemetry data
  The Database must securely store real-time telemetry data received from the vehicle, ensuring minimal data loss during transmission.
- 5.3.1.2.2 The Database shall support wireless data transmission
  The Database must facilitate reliable wireless transmission of telemetry data from the racecar to the Telemetry GUI using low-latency protocols (e.g., Wi-Fi, RF).
- 5.3.1.2.3 The Database shall allow data logging for post-race analysis
  The Database must log all telemetry data, allowing it to be accessed post-race for in-depth analysis, including metrics like lap times, driver performance, and vehicle diagnostics.
- 5.3.1.2.4 The Database shall ensure data security
  The Database must implement measures to secure the data from unauthorized access or tampering, ensuring data integrity both during transmission and storage.

5.3.2 Active Suspension CSCI

5.3.2.1 Suspension Motor Controller CSC Functional Requirements

- 5.3.2.1.1 The Suspension Motor Controller shall receive data from a level sensor
- 5.3.2.1.2 The Suspension Motor Controller shall receive data from a motor-rotation angle sensor
- 5.3.2.1.2 The Suspension Motor Controller shall receive data from the CCU
- 5.3.2.1.2 The Suspension Motor Controller shall control the motor of the suspension unit such that the suspension replicates a predetermined characteristic map

5.3.3 Steering/Pedal Unit CSCI

5.3.3.1 Input Observer CSC Functional Requirements

- 5.3.3.1.1 The Input Observer shall read inputs from the throttle pedal
- 5.3.3.1.2 The Input Observer shall read inputs from the brake pedal
- 5.3.3.1.3 The Input Observer shall convert analog signals from the steering system into digital data.
- 5.3.3.1.4 The Input Observer shall send the processed steering data to the Central Control Unit (CCU) for further action.
- 5.3.3.1.5 The Input Observer shall convert analog signals from the accelerator and brake sensors into binary data.
- 5.3.3.1.6 The Input Observer shall obtain sensor data from the throttle pedal to determine the driver's pressure on it.
- 5.3.3.1.7 The Input Observer shall obtain sensor data from the brake pedal to determine the driver's pressure on it.
- 5.3.3.1.8 The Input Observer shall send data about steering and pedal inputs to the CCU.
- 5.3.3.1.9 The Input Observer shall prepare data for transmission to the CCU.
- 5.3.3.1.10 The Input Observer shall monitor pedal sensor inputs for anomalies or faults to ensure safe operation.
- 5.3.3.1.11 The Input Observer shall test data from the Input Observer to identify any deviations from expected pedal inputs, such as both throttle and brake inputs being at 100%.

5.3.4 Steering Wheel Unit CSCI

5.3.4.1 Graphics Interface CSC Functional Requirements

- 5.3.4.1.1 The Graphics Interface shall receive data from a Raspberry Pi
- 5.3.4.1.2 The Graphics Interface shall clearly display data to the driver

5.3.4.2 Steering Wheel Unit Processor CSC Functional Requirements

- 5.3.4.2.1 The Steering Wheel Unit Processor shall receive data from the Central Control Unit
- 5.3.4.2.2 The Steering Wheel Unit Processor shall send driver inputs to the Central Control Unit
- 5.3.4.2.3 The Steering Wheel Unit Processor shall condense the data received into an easily interpretable graphic
- 5.3.4.2.4 The Steering Wheel Unit Processor shall send graphics to the Steering Wheel Unit Graphics Interface

5.3.5 Central Control Unit CSCI

5.3.5.1 Data Management CSC Functional Requirements

- 5.3.5.1.1 The Data Management CSC shall manage data communication between all CSCs in the system such that data integrity is kept in a timely manner.
- 5.3.5.1.2 The Data Management CSC shall receive sensor data from an I2C Bus.
- 5.3.5.1.3 The Data Management CSC shall receive GPS data.
- 5.3.5.1.4 The Data Management CSC shall receive thermal camera data.
- 5.3.5.1.5 The Data Management CSC shall gyroscopic data.
- 5.3.5.1.6 The Data Management CSC shall acceleration data.

5.3.5.2 Vehicle Stability CSC Functional Requirements

- 5.3.5.2.1 The Vehicle Stability CSC shall receive data from the accelerometer and electric drive motors
- 5.3.5.2.2 The Vehicle Stability CSC shall receive data from the accelerometer and electric drive motors to detect wheel slippage
- 5.3.5.2.3 The Vehicle Stability CSC shall limit the throttle when it detects loss of traction
- 5.3.5.2.4 The Vehicle Stability CSC shall calculate the amount of power needed to give to each electric motor such that it replicates a limited slip differential.
- 5.3.5.2.5 The Vehicle Stability CSC shall follow commands to change behaviors and settings
- 5.3.5.2.6 The Vehicle Stability CSC shall calculate the optimal stiffness for each active suspension unit
- 5.3.5.2.7 The Vehicle Stability CSC shall calculate the amount of power to give each active suspension unit to meet a certain stiffness

5.3.5.3 Safety Check CSC Functional Requirements

- 5.3.5.5.1 The Safety Check CSC shall make sure each sensor is giving constant data flow to the system

- 5.3.5.5.2 The Safety Check CSC shall cut power to the motors if any fault connections are detected.

5.3.5.4 Power Draw Monitor CSC Functional Requirements

- 5.3.5.6.1 The Power Draw Monitor shall estimate the power used by the car as a whole.
- 5.3.5.6.2 The Power Draw Monitor shall keep track of the power usage of each individual component.
- 5.3.5.6.3 The Power Draw Monitor shall send telemetry data on the power usage of each component to the database

5.4 Performance Requirements by CSC

5.4.1 Telemetry App CSCI

5.4.1.1 Telemetry GUI CSC

- 5.4.1.1.1 The Telemetry GUI shall display data with less than 100ms latency
    - The Telemetry GUI must update data with a maximum latency of 100 milliseconds to ensure real-time data visualization during a race.
- 5.4.1.1.2 The Telemetry GUI shall support simultaneous display of up to 10 data metrics
    - The GUI must allow for the simultaneous display of at least 10 different telemetry metrics, ensuring that all relevant information is available to the driver and pit crew without lag or display issues.

5.4.1.2 Database CSC

- 5.4.1.2.1 The Database shall handle data transmission with a latency of less than 50ms
    - The Database must transmit telemetry data wirelessly with a latency of no more than 50 milliseconds to maintain a real-time connection with the Telemetry GUI.
- 5.4.1.2.2 The Database shall store telemetry data at a rate of 10 samples per second
    - The Database must be capable of logging telemetry data at a minimum rate of 10 samples per second per metric to ensure high-resolution data for post-race analysis.

5.4.2 Active Suspension CSCI

5.4.2.1 Suspension Motor Controller CSC

- 5.4.2.1.1 Return required motor output results within 10ms on the arduino
    - The suspension motor controller should be able to be as fast and reactive as possible. This is necessary to simulate a smooth suspension system

- 5.4.2.1.2 Be able to adjust suspension depending on CCU requests
    - The active suspension system should be able to modulate how the suspension acts depending on the request of the driver. This means the suspension motor controller will have multiple characteristic graphs for each setting.

5.4.3 Steering/Pedal Unit CSCI

5.4.3.1 Input Observer CSC

- 5.4.3.1.1 Consistently be able to report steering and pedal inputs to the Central Control Unit

5.4.4 Steering Wheel Unit CSCI

5.4.4.1 Graphics Interface CSC

- 5.4.4.1.1 Consistently display all pertinent information to the driver through an easily navigable UI

5.4.4.2 Steering Wheel Unit Processor CSC

- 5.4.4.2.1 Be able to hold a consistent wifi connection to the Central Control Unit and communicate with it
- 5.4.4.2.2 Consistently generate display for the Graphics Interface

5.4.5 Central Control Unit CSCI

5.4.5.1 Data Management CSC

- 5.4.5.1.1 The Data Management CSC should be receiving the data within 10 ms
    - The Data Management CSC should be able to instantly receive sensor data to ensure the car is fast to react to situations.

5.4.5.2 Vehicle Stability CSC

- 5.4.5.2.1 Vehicle Stability CSC should be able to calculate when traction is lost within 10 ms
    - When traction is lost the car should be quick to react to reduce the effects of traction loss.

- 5.4.5.2.2 Vehicle Stability CSC should be able to calculate the motor power distribution within 50ms
    - The differential should be fast in order to replicated a mechanical limited slip differential
- 5.4.5.2.3 Vehicle Stability CSC should be able to calculate PID within 50 ms.
    - The suspension should also be fast to maintain tyre contact with the ground.

5.4.5.3 Safety Check CSC

- 5.4.5.5.1 Be able to ensure that all signals are within expected range and there are no errors or unexpected latency
    - If data isn't being transferred as expected there can be major malfunctions with physical consequences

5.4.5.4 Power Draw Monitor CSC

- 5.4.5.6.1 Consistently be able to report how much power the whole vehicle is using
    - If the power draw is not being recorded correctly it can cause major issues for all other systems

5.5 Project Environment Requirements

Following are the hardware requirements to run and test the FSAE Software:

Item	Use	How will be or was acquired
Arduino	Sensor Reporting and Controls	Spring 2024 SAFAB Funding
Raspberry Pi	Electronic Control Unit	Spring 2024 SAFAB Funding
Motor and Motor Controller	Electronic Suspension Compression Control	Spring 2024 SAFAB Funding
Spring Pack	Main Suspension Spring	Spring 2024 SAFAB Funding
Mounting Hardware	Suspension Unit Housing	Spring and Fall 2024 SAFAB Funding

A Raspberry Pi will act as the CCU (Central Control Unit) of the vehicle. Each active suspension unit will be controlled by an arduino that sends information to the CCU through a hard-wired connection. The CCU will then in turn send instructions to the motor controllers based on the information received.

Following are the software requirements for the FSAE Software:

Category	Requirement
Operating System	Linux
Compiler	GNU C++ Compiler
Input Handler	Arduino IDE

The Raspberry Pi will be running Linux. On it there will be a C++ program that takes input from all the connected arduino's, processes the information, and makes the motor controller react accordingly. This process needs to have as little latency as possible and must be robust, as the car's active suspension system will be relying on it.