Dynamic Motion and Vibration Telemetry System

Theory and Design for Mechanical Sensing and Measurements

Dr. Wei Li

December 7, 2025

1. Introduction

1.1 Background

Telemetry is central to modern motorsports, robotics, and vehicle dynamics. Professional systems combine inertial, GNSS, and thermal sensing to diagnose performance, validate models, and support closed-loop control. However, research-grade data loggers are expensive and often overkill for student projects. We set out to build a compact, low-cost version that still provides synchronized multi-sensor measurements suitable for structured system identification.

1.2 Motivation/Problem Definition

Longhorn Racing does not currently have a lightweight, low-cost telemetry setup that can record synchronized IMU, GNSS, and temperature data during vehicle tests. Existing tools are either too limited (single-sensor) or too expensive to modify for student experimentation. We needed a system that not only collects data, but also supports signal analysis, uncertainty evaluation, and basic detection/classification. We created a portable measurement system that can be mounted on LHR vehicles, stream multi-sensor data to a laptop in real time, and produce datasets suitable for studying dynamic motion, thermal behavior, and turning response.

This problem ties directly to the course requirements:

1. **Hardware design/selection:** choosing and integrating an IMU, GNSS module, and temperature probes
2. **Data acquisition & filtering:** streaming all channels at ~20 Hz and applying smoothing to reduce noise
3. **Experimental design:** running structured tests—accel/coast/brake, coastdowns, and CW/CCW circles
4. **Detection/classification:** identifying maneuvers such as braking, accelerating, and turning direction

In short, the core problem was to build a flexible, low-cost telemetry system that Longhorn Racing can use for repeatable dynamic testing while demonstrating the full measurement-system pipeline required.

1.3 Project Objectives

1. Build an integrated multi-sensor telemetry package
2. Acquire synchronized data at 20 Hz
3. Process & filter signals for dynamics analysis
4. Perform thermal, GNSS, and dynamic tests
5. Conduct event classification
6. Provide reproducible hardware/software documentation

2. System Architecture

2.1 Hardware Overview

Our telemetry system consisted of the following components:

- 1x Arduino Nano ESP32
- 1x BNO055 IMU
- 1x MAX-M10S GNSS module with magnetic antenna
- 4x DS18B20 temperature sensors
- 1x Basic breadboard and wiring

A photo of our system can be found in Appendix A.

2.2 Component Rationale

Our component selection focused on creating a simple, reliable, and low-cost telemetry stack that could measure motion, position, and temperature during various vehicle tests.

- Our Arduino Nano ESP32 was chosen as the central controller because it provides fast serial output, built-in wireless capability, and enough processing headroom to stream IMU, GNSS, and temperature data simultaneously.
- The BNO055 IMU was used for all motion and dynamics measurements. The BNO055 outputs fused orientation, acceleration, and angular velocity directly on-board, removing the need for custom filtering on the microcontroller and providing clean features for event detection.
- The MAX-M10S GNSS Module + Magnetic Antenna were selected to provide absolute position, velocity, and heading for outdoor driving. The magnetic antenna ensured stable satellite reception on the kart and Tesla.
- The DS18B20 Temperature Sensors were selected to capture tire, engine, and ambient temperatures on the kart. The waterproof housings made them suitable for direct contact with tires and hot surfaces, enabling thermal response analysis.

2.3 Mechanical Integration

On the go-kart, the temperature sensors were positioned approximately one inch from each front tire to capture load-dependent heating during dynamic maneuvers. A third sensor was mounted at the front of the kart to record ambient temperature, and the fourth was placed inside the engine block to track thermal behavior under repeated runs. The IMU and GNSS module were mounted at the front-center of the kart, just below the steering wheel and close to the vehicle’s center of gravity, ensuring representative motion measurements. For the Tesla Model S tests, no temperature sensors were used. The IMU was mounted on the center console to capture vehicle dynamics, while the GNSS antenna was positioned near the front windshield for clear satellite visibility. Photos of all sensor mounting locations for both platforms are provided in Appendix B.

3. Data Acquisition & Processing

3.1 Data Logging Configuration and Time Synchronization

Our data acquisition architecture is built around the Arduino Nano ESP32, which serves as the master clock and logger for all sensors. Early tests revealed that using GNSS UTC timestamps caused large gaps in the data whenever satellite lock was lost. To eliminate this problem, we switched to the Arduino's internal millisecond timer (t_ms) as the canonical sampling clock.

The logging loop was executed at 20 Hz, driven by a fixed 50 ms interval. All measurements, including IMU orientation, gyroscope, linear acceleration, GNSS latitude/longitude, and temperature, were stamped using this common time base. This provides consistent temporal alignment across sensors with different update rates:

- IMU: 20 Hz (synchronous with loop)
- GNSS: 10 Hz (interpolated within 20 Hz logging)
- Temperature sensors: 1 Hz (only update once per second, held constant otherwise)

The data were streamed over USB at 230,400 baud and written directly to CSV files using a Python logging script. Each row in the dataset corresponds to one 20 Hz frame and contains: t_ms, fix, lat, lon, euler_x, euler_y, euler_z, gyro_x, gyro_y, gyro_z,accel_x, accel_y, accel_z, temp0, temp1, temp2, temp3.

This format ensures that all subsequent processing, speed estimation, filtering, and event classification operate on a unified, evenly spaced time grid.

3.2 Calibration Procedures

IMU Calibration

- We calibrated the BNO055 [1] using the manufacturer's recommended routine. A custom Arduino calibration script displayed the system, gyro, accelerometer, and magnetometer calibration flags (0–3). The IMU was slowly rotated about each axis and moved in figure-8 patterns until all four subsystems reported a calibration level of 3. Once completed, we extracted the hard-iron and soft-iron offset structure using getSensorOffsets() and stored these offsets in the final firmware to ensure consistent performance across test sessions. This calibration mitigates orientation drift and improves the accuracy of the linear acceleration signal used for classification.

GNSS Calibration and Validity

- The MAX-M10S GNSS module [2] requires a warm-up period to acquire a stable 3D fix. During this time, the fix flag remains below 3, and the latitude/longitude readings show large jitter. All data collected before achieving a 3D fix were automatically trimmed during preprocessing. For all test runs, we ensured that satellite visibility was sufficient and that the fix was stable before any maneuvers began.

Temperature Sensor Verification

- The DS18B20 [3] sensors require approximately 0.5–1.0 s to stabilize after startup. Before each session, the four probes were allowed to reach ambient equilibrium. Their readings were compared, and any sensor deviating more than ±0.5 °C from the others was flagged. Because DS18B20 devices have slow thermal response (large time constants), no additional dynamic calibration was applied.[6]

OneWire Temperature Sensor Interface

- The four DS18B20 temperature sensors were interfaced using the OneWire digital bus protocol, which allows multiple devices to share a single GPIO pin while maintaining unique digital addresses for each sensor. Unlike analog thermistors, the DS18B20 sensors perform digital temperature conversion internally and communicate over a single data line using a time-critical signaling scheme. To ensure reliable data acquisition, each sensor was assigned to its own OneWire bus rather than sharing a single line. This strategy avoids bus contention, reduces timing conflicts, and simplifies sensor identification during polling. In the Arduino firmware, each OneWire bus is initialized with the DallasTemperature library[7], which handles device addressing, conversion timing, and resolution settings. Because OneWire conversions require approximately 94 ms at 9-bit resolution, the temperature sensors were intentionally sampled at 1 Hz, consistent with their dynamic limitations and thermal response time. This design ensured stable, low-noise temperature readings without interfering with the 20 Hz IMU and GNSS sampling loop.

3.3 Filtering and Preprocessing

Real-world telemetry exhibits significant noise, particularly in low-cost IMU and GNSS systems. To ensure reliable analysis, we applied several preprocessing steps.

GNSS Smoothing and Differentiation

GPS-derived speed is computed by differentiating position, which strongly amplifies noise. To address this, we applied a rolling-average smoothing filter (window size = 20 samples, equivalent to 1 s). This filter preserves large-scale motion trends (acceleration, coasting, braking) while suppressing jitter caused by satellite multipath and random measurement noise.

IMU Noise Reduction

The gyroscope and linear acceleration signals were smoothed using a smaller window (5–10 samples) to preserve dynamic behavior. Stationary tests revealed the IMU noise floor:

- Gyro noise ≈ ± 0.5–1 deg/s
- Linear accel noise ≈ ± 0.05–0.15 m/s^2

An example of raw vs. filtered IMU signals is available in Appendix B.

Data Trimming

The first 1–2 seconds of every dataset were removed to eliminate boot transients, GNSS warm-up, and unstable IMU readings. Rows with invalid GNSS readings (fix < 3) were excluded from path analysis but retained for IMU-only classification tasks.

Outlier Handling

Any IMU spikes beyond physically reasonable bounds (e.g., $> \pm 8$ g acceleration, $> \pm 500$ deg/s angular rate) were treated as serial or electromagnetic interference artifacts and replaced with local interpolated values.

3.4 Feature Extraction

After filtering, we extracted the following features for dynamics analysis, classification, and system identification.

Longitudinal Features

- Forward linear acceleration (accel_x): Used to identify accelerating, coasting, and braking states. Peaks and average slopes provided estimates of launch acceleration and braking deceleration.
- Speed from GNSS: Used to validate IMU accel_x and identify coastdown regions for drag-model estimation.

Lateral/Turning Features

- Yaw rate (gyro_z): Primary feature for turning classification. The sign of gyro_z directly indicates the turning direction.
- Lateral acceleration (accel_y): Used as a secondary indicator of turning magnitude.
- Signed area of GNSS path: Used for CW/CCW detection and turning radius estimation.
- Instantaneous turning radius: Computed using the relationship $R = \frac{v}{\dot{\psi}}$ where v is the smoothed speed and $\dot{\psi}$ is the yaw rate.

Thermal Features

- Tire heating rate (°C/s) during acceleration
- Engine temperature gradient over repeated runs
- The temperature difference between the inside and outside tires during circular motion

These features confirmed that the temperature sensors were sensitive enough to detect load transfer and long-term heating effects.

Event Timing Features

- Duration of acceleration, coasting, braking
- Duration and consistency of CW/CCW turn laps
- Transition detection between maneuver types

4. Experimental Design

4.1 Test Platforms

Our experiments were designed to validate the telemetry system, characterize vehicle dynamics, and support event classification. We performed the same three tests on two platforms: a go-kart and a Tesla Model S, in order to compare system behavior across different vehicle dynamics and mounting conditions.

The three tests were:

- Skidpad: Driving in a constant-radius circle with steady steering input, performed in both clockwise and counterclockwise directions
- Straight-line runs: Accelerating from rest to approximately 40 mph, then coasting for about five seconds, followed by braking
- Normal driving: A mixed driving scenario with varying steering and speed to verify real-world performance

Each test consisted of three trials with three runs per trial, which provided sufficient repeatability to evaluate consistency and classification accuracy.

The go-kart served as the initial test platform because its open layout made sensor mounting and debugging straightforward. We collected IMU, GNSS, heading, and temperature data, with all sensors placed near the center of gravity or at key thermal locations. Temperature probes measured ambient air, left and right front tire temperatures, and engine temperature. Complete diagrams and photos showing the exact placement of each sensor are provided in Appendix C. These tests validated the thermal sensors and provided early IMU and GNSS data, but they also revealed limitations such as inconsistent GNSS readings caused by metal interference, low roll and pitch variance, and general platform unreliability.

To obtain cleaner dynamics data, we repeated all tests on a Tesla Model S. This platform is more representative of an FSAE-style vehicle and provides suspension, predictable roll and pitch behavior, and more stable steering inputs. The IMU and GNSS module were mounted close to the vehicle center of gravity, with the GNSS antenna placed near the windshield for maximum satellite visibility. Sensor placement for the Tesla configuration is also shown in Appendix C. Temperature sensors were not used on the Tesla since they had already been validated on the go-kart.

Together, these tests produced comprehensive datasets covering longitudinal events, lateral maneuvers, temperature effects, and mixed driving conditions. This was sufficient to evaluate the telemetry system and support the classification and analysis tasks in the project.

5. Results

5.1 Temperature Results

One of the key metrics we were attempting to use to help classify various aspects of the vehicle performance was the Temperature data from the four different sensors placed throughout the kart, as seen in Appendix C Figure 1. Despite some potential flaws in our data acquisition, the temperature data proved to be very accurate and easy to interpret, allowing us to use it to classify vehicle performance. First, we were able to identify periods of intense acceleration through the amount of heat put into the tires as seen in the Coastdown tests, where there is a spike in tire temperature, followed by a gradual decline after the acceleration event has ended and the kart is coasting. Additional information can be extracted from the temperature data recorded. As seen in Appendix D, Figures 3 and 4, it is possible to determine the direction of travel based on the temperature of the tires. A clockwise circular motion greatly heated up the front right tire, and a counterclockwise circle heated the front left tire. This data suggests that the thermal data collected can be accurately used to reconstruct the tire conditions and the vehicle conditions.

5.2 GNSS Path Results

The next goal of some of our tests was to confirm that we could identify the path taken by a vehicle and dynamically compare this path to an idealized line. We are able to recreate the paths taken by both the kart and the Tesla very accurately. Even being able to effectively map the latitude and longitude onto a real map to reconstruct the actual driven path through Austin, as seen in Appendix D Figures 7 and 8. This feature was also used to simulate the existence of a racing line and compare our driving tests to this theoretical one. Appendix D Figures 5 and 6 demonstrate that we can faithfully reconstruct the driven paths from the GNSS sensor. From both of our attempted path recreations, we have less than 15 percent error between the ideal circle and our driven circles. This deviation is likely due to a combination of driver error and the GPS drift found within the data.

5.3 IMU & Dynamic Results

The final data extracted from our sensor suite was the IMU data, which we used to help classify our acceleration, braking, and turning segments. By overlaying the specific acceleration data and gyro outputs, we can reconstruct the vehicle's performance. By observing our gyroscope rate in the z-axis, we can easily filter the turning events, and through our X and Y acceleration, we can identify cornering and acceleration events.

Ultimately, through all of our data collection, we were able to use this to perform a classification and detection algorithm, and recreate the stated goals of the project. Despite some shortcomings of the data logging and sensor errors, we have been able to effectively utilize all of the data in our analysis.

6. Event Detection & Classification

6.1 Classification Performance

For the vehicle classification portion of our project, we identified two different system responses that needed to be classified. The two were the longitudinal and lateral classification systems. Every vehicle has three states for each of these, for longitudinal, a vehicle can be accelerating, coasting or braking. For lateral, the vehicle can be turning left, turning right, or going straight. The goal for this portion of the project was to use the data to identify which of these types were occurring at a specific point in time for the vehicle. The classification was only performed from interpreting the data collected from the telemetry box. This posed a few challenges as the collected data had to be carefully refined through post-processing. If this were not performed, the thresholds set would be incorrect, and it is critical to keep the thresholds at the same points.

The thresholds for longitudinal acceleration used the X acceleration data from the IMU. Based on our configuration, if the X acceleration data was below -0.75 m/s^2, then the code would understand it as an acceleration event. If the X acceleration was greater than 0.25 m/s^,2 then it would be considered a braking event. If the X acceleration did not fall in one of these two buckets, then it was considered coasting. For the acceleration event, we determined this as there was a pretty clear line between what was noisy coasting versus accelerating. The braking event threshold was a little more difficult to define as the line was more vague. We ended up deciding on 0.25 m/s^2 as the maximum coasting value, even with regenerative braking, which was below this deceleration value. From these thresholds, we were able to define on a GPS tracking graph video what state the vehicle was in longitudinally.

The threshold for lateral acceleration was slightly more straightforward when the data was processed. We used mainly the Z Gyro data, which represents the yaw rate of the vehicle. Lateral acceleration (Y acceleration) was also considered, as it is typically easier to interpret. However, random environmental factors can affect this reading, as it means the acceleration from side to side. Whereas the yaw rate is the actual representation of the turning of the vehicle about the center of gravity's vertical axis. The thresholds set were more than +10 deg/s of yaw to identify a left turn, while less than -10 deg/s was classified as a right turn. Any other value was considered straight. An example of the results can be seen in Appendix E. This would be greatly improved by implementing a steering column encoder to collect the steering angle. However, this was out of the scope of our project, and the results seemed to turn out fairly accurate with cleaned data.

7. Discussion and Conclusion

7.1 Key Findings

The telemetry system produced synchronized and interpretable IMU, GNSS, and temperature data across both test platforms. The IMU provided clear longitudinal and lateral dynamic signatures that allowed us to classify acceleration, braking, and turning events with simple threshold rules. The temperature sensors captured meaningful thermal trends, including tire heating during acceleration and load-dependent heating during clockwise and counterclockwise circles. GNSS measurements were accurate enough to reconstruct vehicle paths, estimate turning radii, and compare driven skidpad circles to idealized curves. Overall, the combined sensor suite provided a complete view of vehicle motion and thermal behavior and supported the classification and analysis tasks required for the project.

7.2 Limitations

Several factors limited the accuracy of the system. Small errors in sensor alignment and mounting stiffness produced minor discrepancies in IMU signals, especially in lateral acceleration and yaw rate. GNSS drift and satellite multipath introduced noise in position data and contributed to radius estimation errors. The metal structure of the go-kart reduced GNSS reliability, and the limited roll and pitch dynamics in the kart restricted the evaluation of those IMU channels. Temperature sensors could not be installed on the Tesla, limiting cross-platform comparison. The 20 Hz logging rate restricted high-frequency analysis and prevented detailed vibration studies. Despite these limitations, the data quality remained sufficient for the event classification and dynamic analysis performed.

7.3 Future Improvements

Future versions of the telemetry system would benefit from several technical upgrades. Increasing the data rate and using a dedicated logging solution would support higher fidelity IMU and GNSS data. A rigid mounting bracket or enclosure would improve sensor alignment and reduce noise caused by vibration. A higher-grade GNSS receiver or an RTK-capable module would reduce drift and improve path reconstruction. Integrating temperature sensors on the full vehicle platform would allow direct thermal comparison across systems. Adding a steering angle sensor or wheel speed sensors would greatly enhance lateral dynamics analysis and improve classification accuracy. These improvements would bring the system closer to professional motorsport telemetry capability while remaining low-cost and student accessible.

8. References

[1] Bosch Sensortec GmbH, BNO055 Intelligent 9-Axis Absolute Orientation Sensor Datasheet, Rev. 1.4, 2020.

[2] u-blox AG, MAX-M10S Ultra Low Power GNSS Module Datasheet, UBX-22008808, 2022.

[3] Maxim Integrated, DS18B20 Programmable Resolution 1-Wire Digital Thermometer Datasheet, Rev. 6, 2015.

[4] Espressif Systems, Arduino Nano ESP32 Technical Reference and Pinout Guide, 2023. Available: https://docs.arduino.cc/hardware/nano-esp32

[5] Adafruit Industries, Adafruit BNO055 Library Documentation, 2023. Available: https://github.com/adafruit/Adafruit_BNO055

[6] SparkFun Electronics, SparkFun u-blox GNSS Arduino Library Documentation, 2023. Available: https://github.com/sparkfun/SparkFun_u-blox_GNSS_Arduino_Library

[7] DallasTemperature Library, Arduino OneWire and Temperature Sensor Interface Documentation, 2023. Available: https://github.com/milesburton/Arduino-Temperature-Control-Library

[8] ME372C Course Notes, Measurement Systems, Filtering, Uncertainty, and Signal Processing, The University of Texas at Austin, 2025.

9. Appendices

Appendix A: Photo of System, Wiring Diagram, and Electrical Pinout

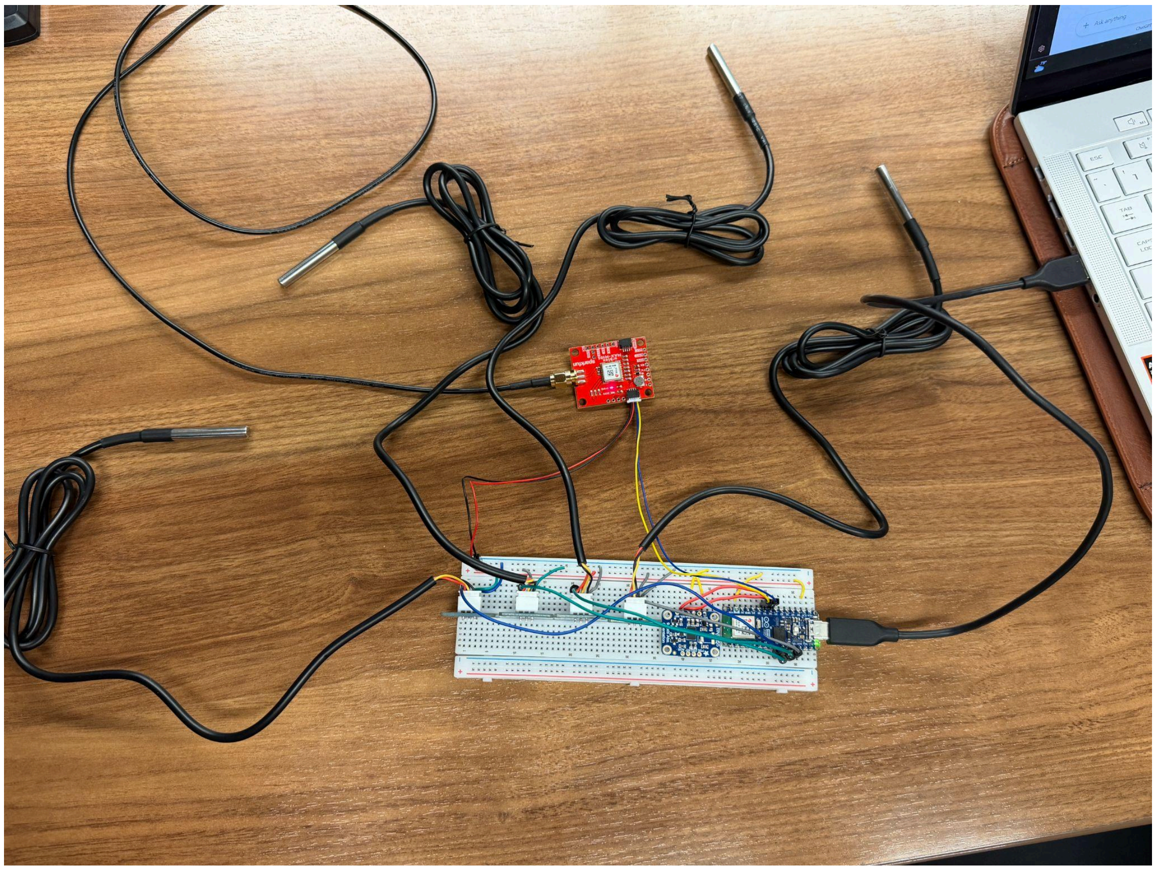

Figure 1: Telemetry System connected to laptop

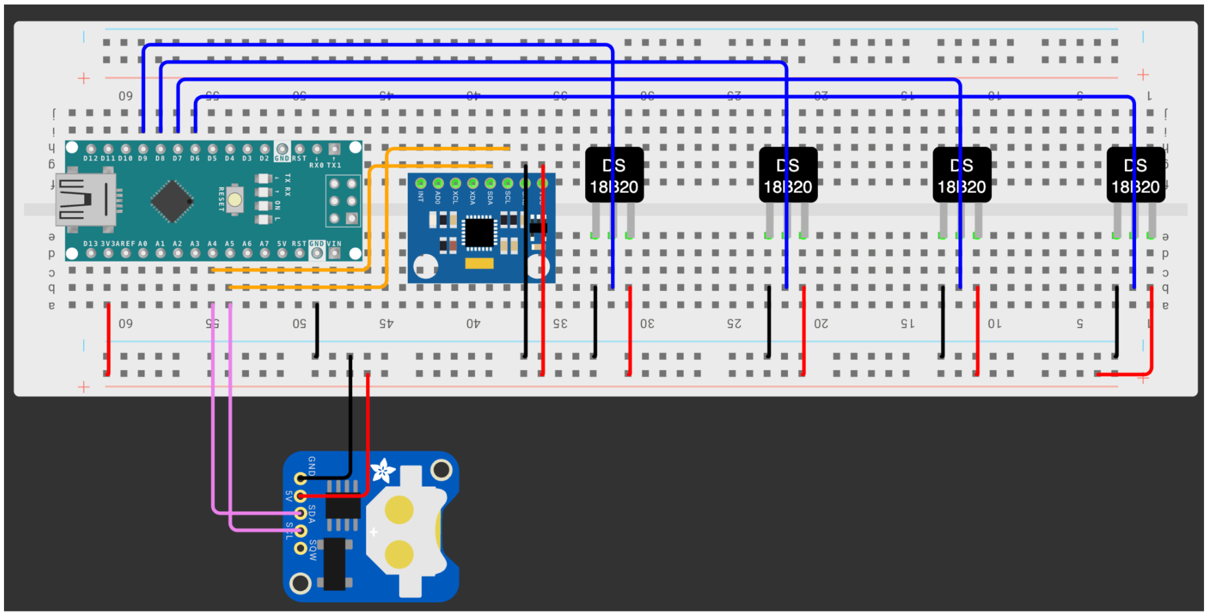

Figure 2: Wiring Diagram of Telemetry System

Table 1: Electrical Pin Out

Device	**Pin on Device**	**Connected To**	**Notes**
Nano ESP32	3.3V	Breadboard + rail	Powers all sensors
Nano ESP32	GND	Breadboard – rail	Common ground
Nano ESP32	D6	DS18B20 #1 SIG	Temp sensor (e.g., FL tire)
Nano ESP32	D7	DS18B20 #2 SIG	Temp sensor (e.g., FR tire)
Nano ESP32	D8	DS18B20 #3 SIG	Temp sensor (engine)
Nano ESP32	D9	DS18B20 #4 SIG	Temp sensor (ambient)
Nano ESP32	A4 (SDA)	BNO055 SDA, MAX-M10S SDA	Shared I²C data line
Nano ESP32	A5 (SCL)	BNO055 SCL, MAX-M10S SCL	Shared I²C clock line
BNO055 IMU	VIN	Breadboard + rail	3.3 V supply
BNO055 IMU	GND	Breadboard – rail	Ground
MAX-M10S GNSS	VIN	Breadboard + rail	3.3 V supply
MAX-M10S GNSS	GND	Breadboard – rail	Ground
DS18B20 (all)	VCC	Breadboard + rail	3.3 V supply
DS18B20 (all)	GND	Breadboard – rail	Ground

Appendix B: Sensor Locations

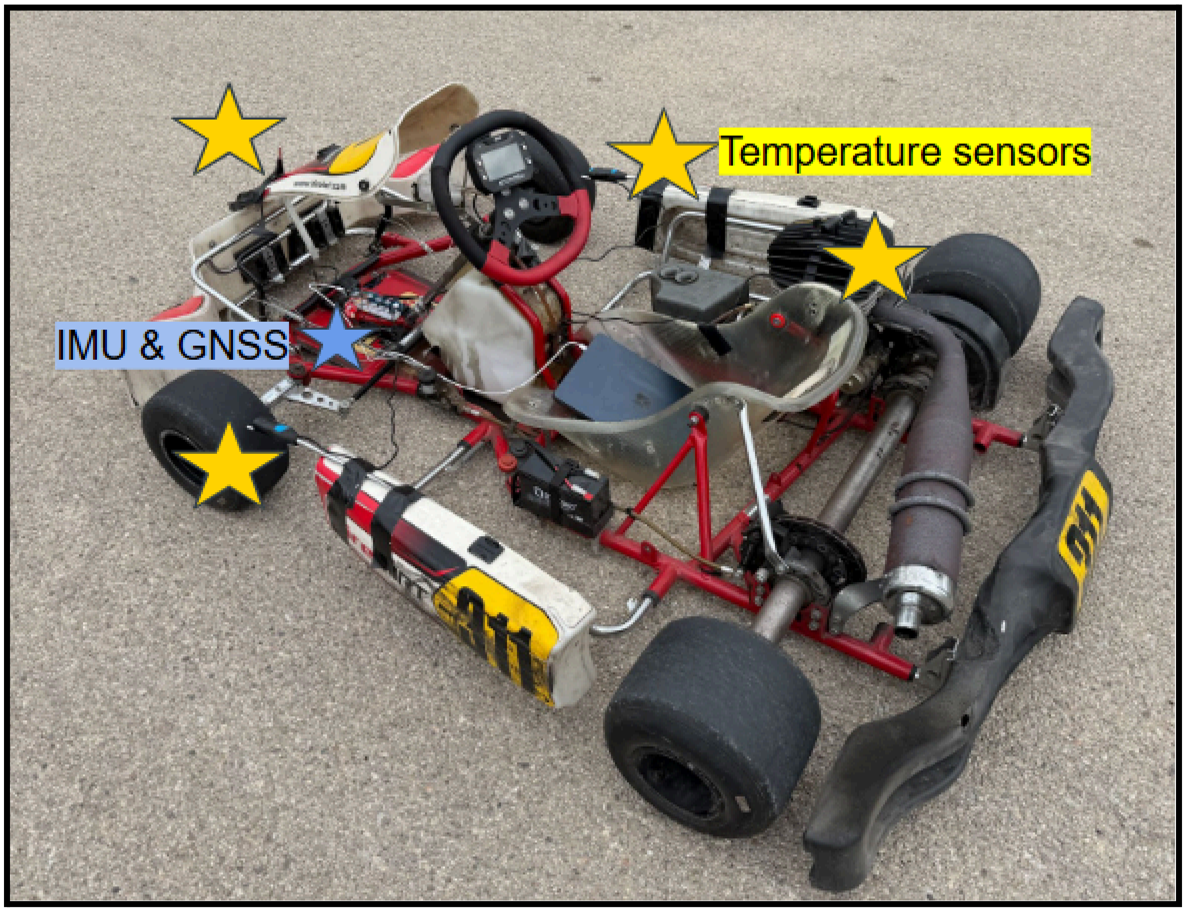

Figure 1: Go Kart Sensor Placement

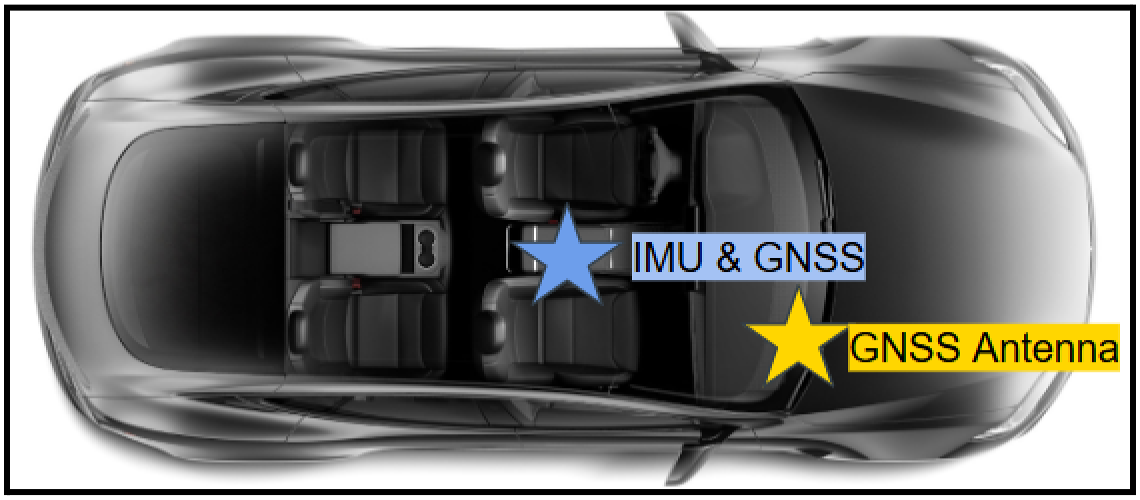

Figure 2: Tesla Model S Sensor Placement

Appendix C: Python Data Processing

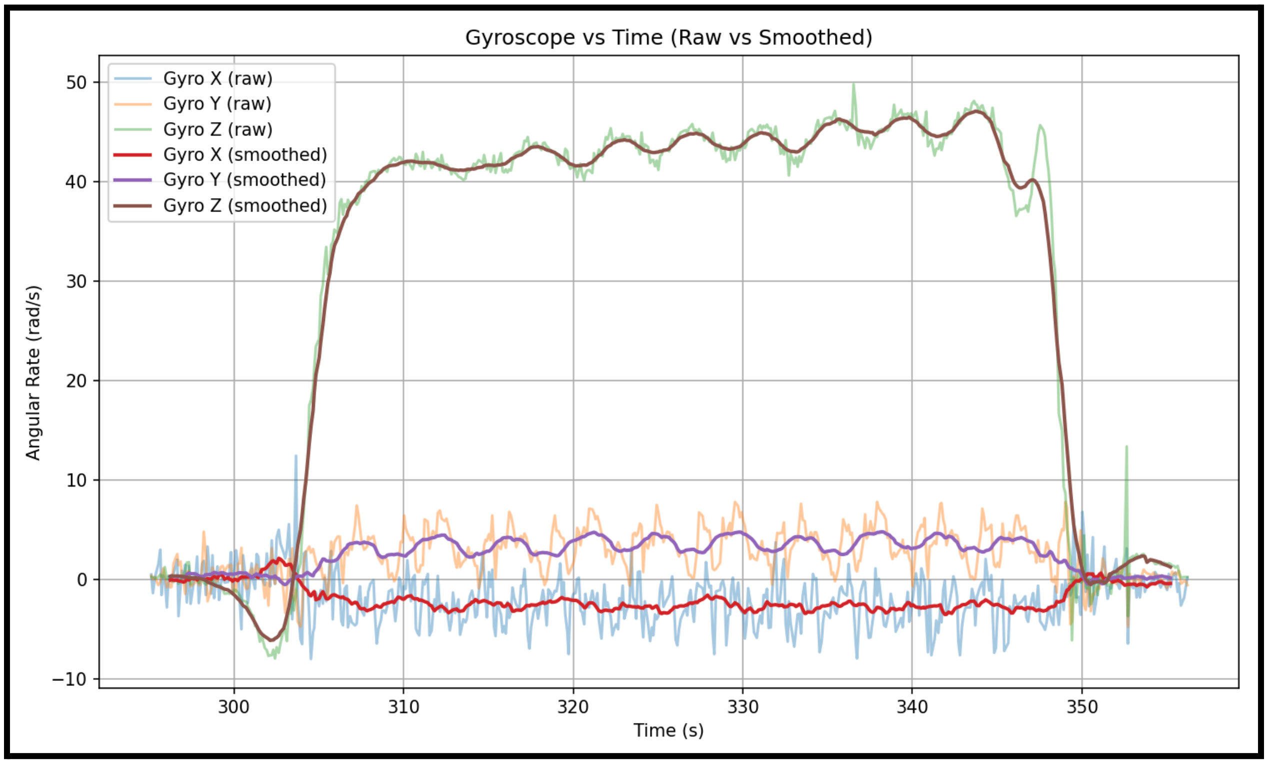

Figure 1: Raw vs. filtered IMU signals (Rolling Average Filter)

Appendix D: Results Plots

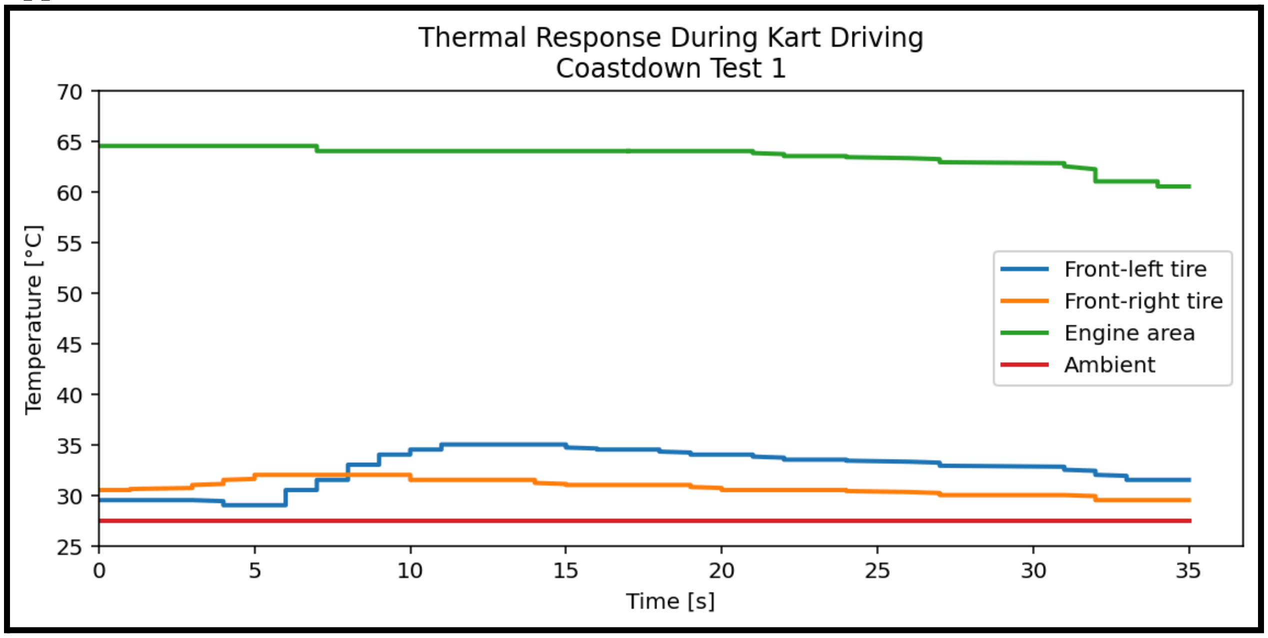

Figure 1: Thermal Response Coastdown 1

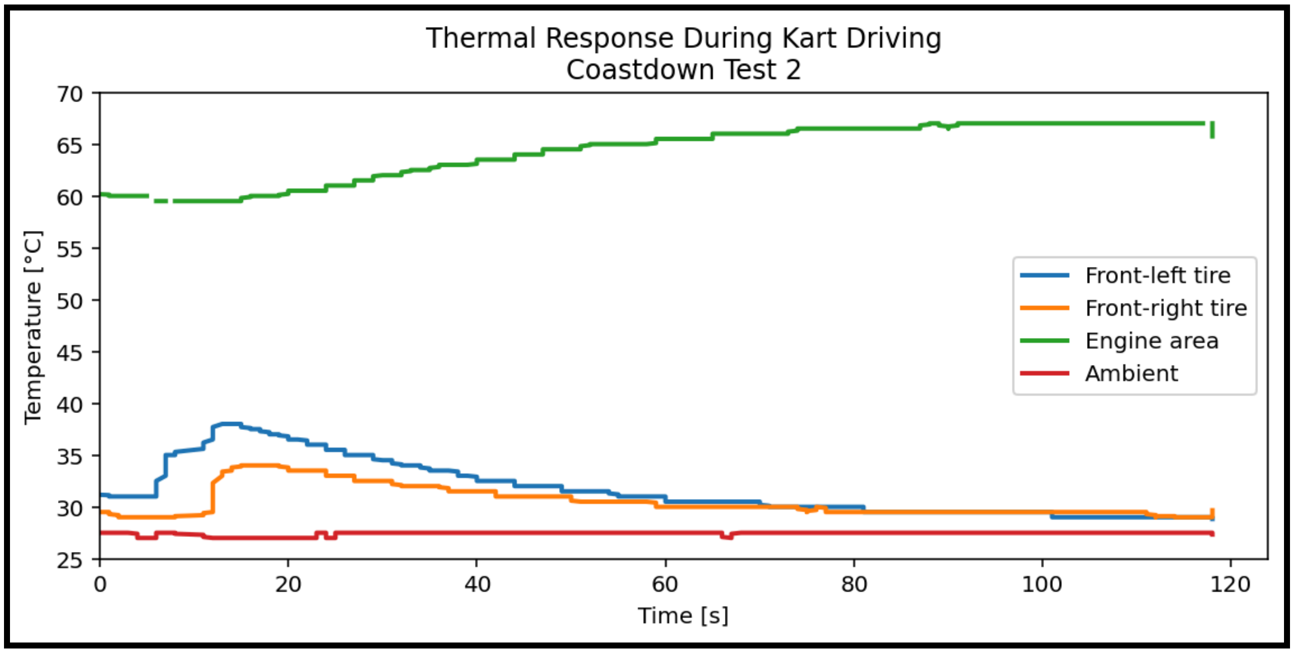

Figure 2: Thermal Response Coastdown 2

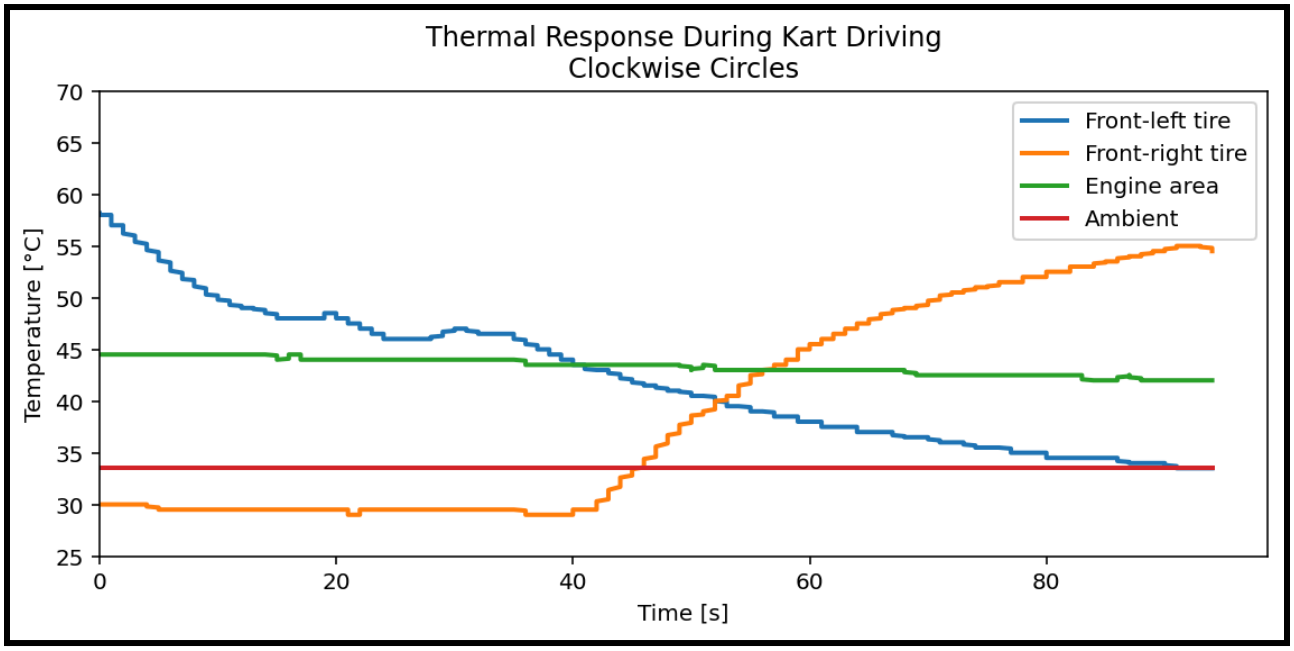

Figure 3: Thermal Response Clockwise

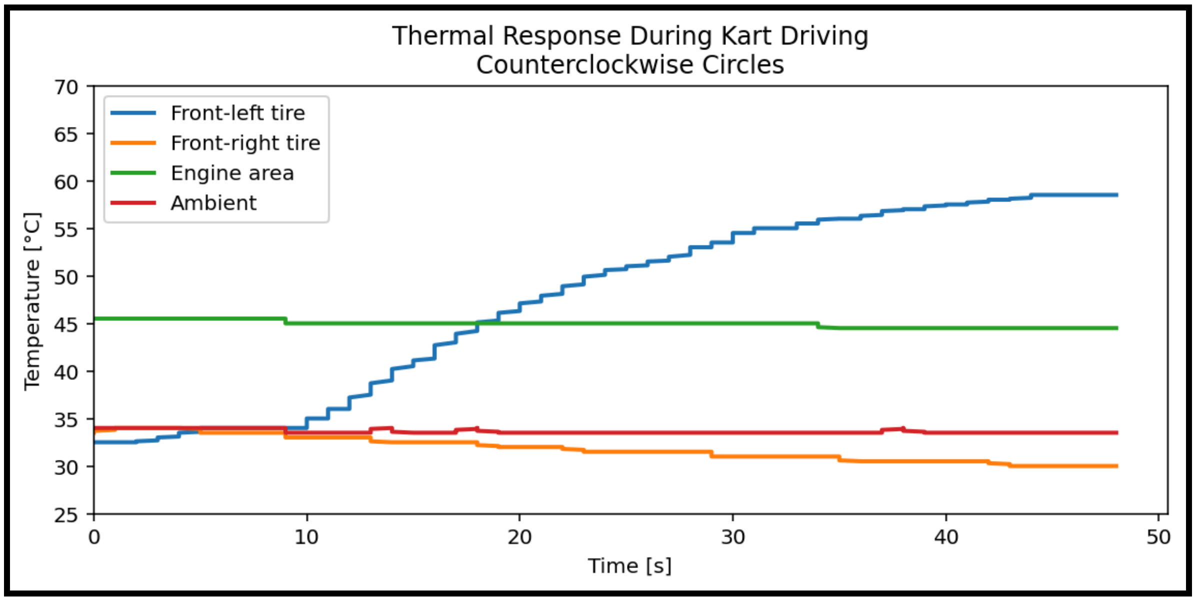

Figure 4: Thermal Response Counter Clockwise

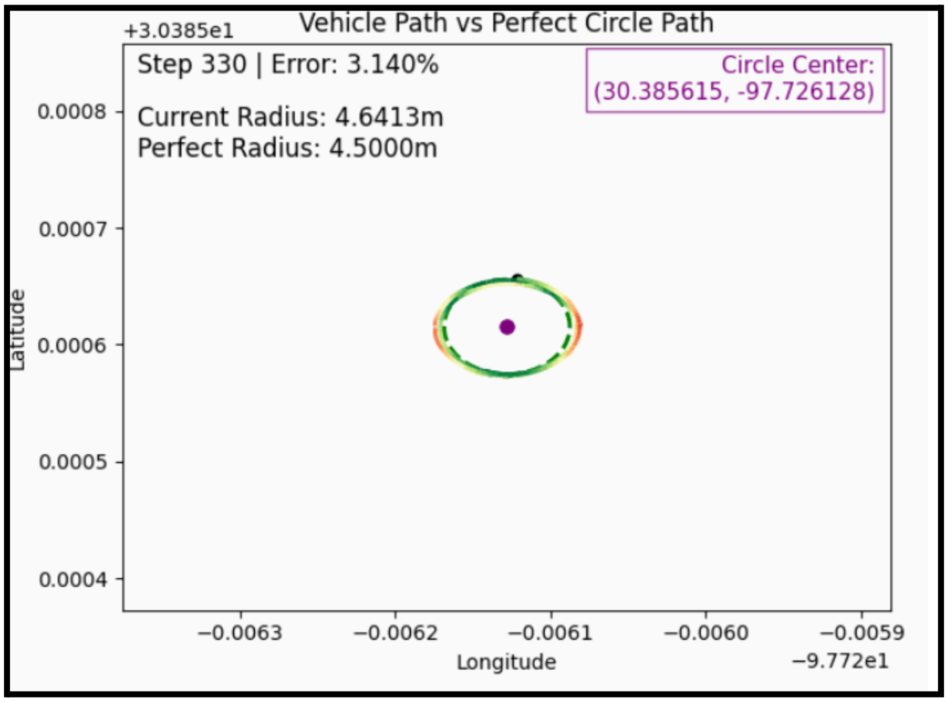

Figure 5: GNSS Path Reconstruction with a 4.5-meter circle and error

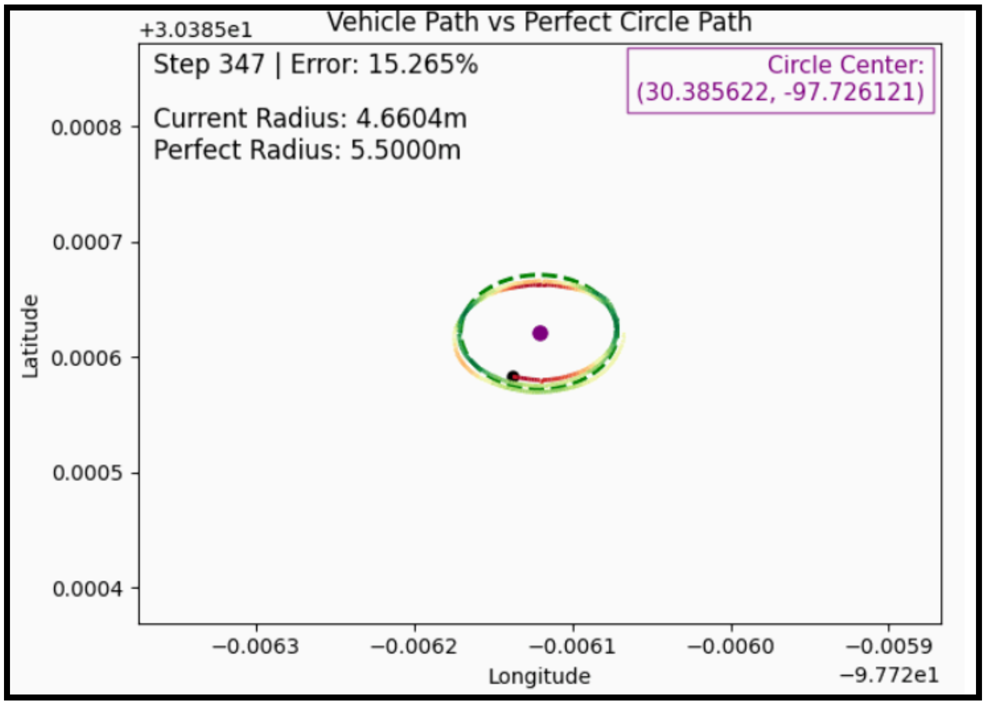

Figure 6: GNSS Path Reconstruction with a 5-meter circle and error

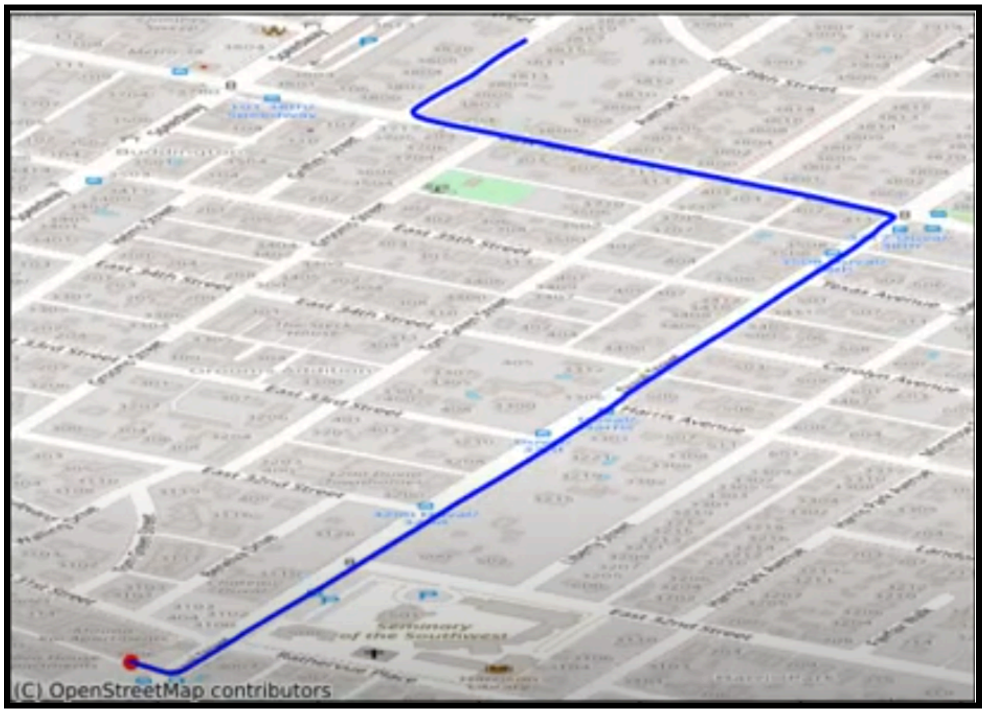

Figure 7: GNSS Path reconstruction driving on surface streets

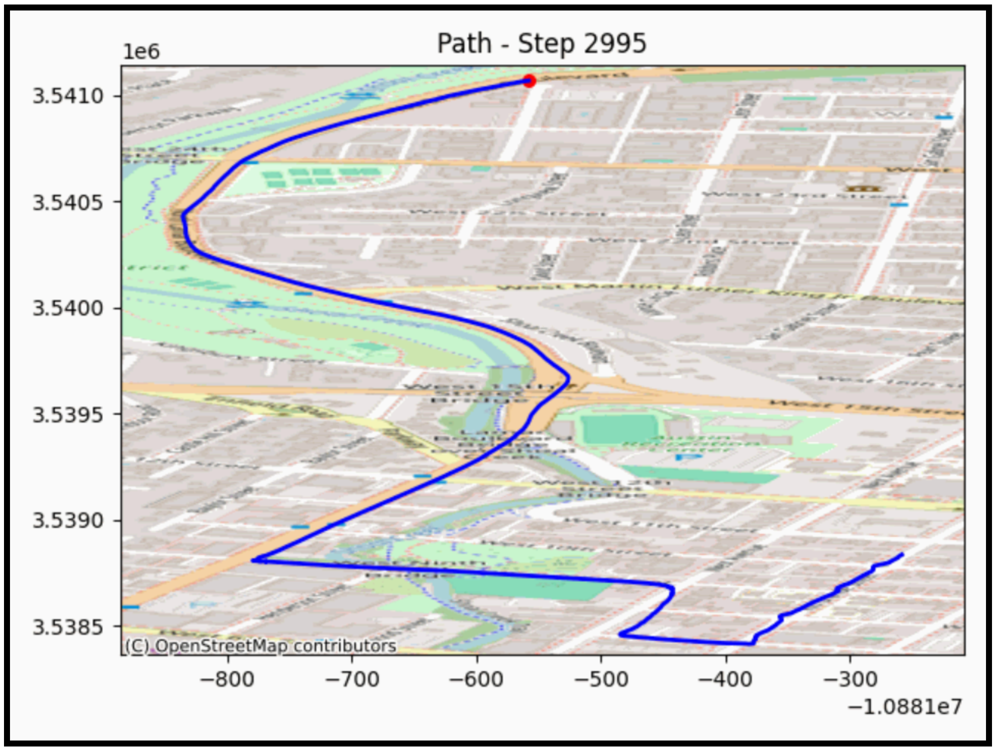

Figure 8: GNSS Path reconstruction driving on Lamar

Appendix E: Additional Plots

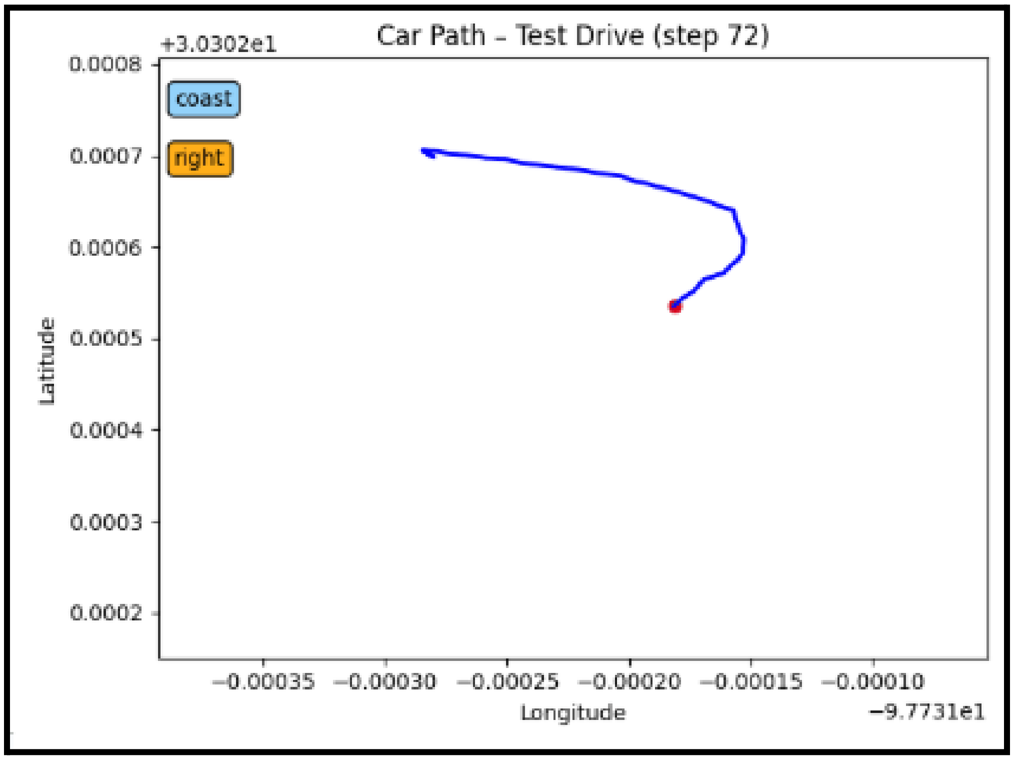

Figure 1: Tesla Vehicle Path with Classification