Sprint 1 Conclusion - SkyLedge

Group 4 - EAT40005 progress update for Week 7

Khoa – Barsat – Dale – Sadman

Data Collection Metrics

- Data Collected via OBD-II Interface
  - Engine & Performance Metrics *(e.g., Engine RPM, Vehicle Speed Sensor, Engine Load, Throttle Position, Intake Manifold Pressure)*
  - Fuel & Emissions Data *(e.g., Short-Term & Long-Term Fuel Trim, Fuel Rail Pressure, Oxygen Sensor Readings, Fuel Tank Level Input)*
  - Diagnostics & Maintenance Indicators *(e.g., Diagnostic Trouble Codes (DTCs), Battery Voltage, Misfire Count, Readiness Monitors, Freeze Frame Data)*

Understanding Raw OBD-II Data

Data Format

Hexadecimal stream outputs CAN frames with ID, bytes, timing, and flags.

- Hard for visual interpretation
- Not a Machine Learning – friendly data type

=> Raw data collection will have to be converted to CSV format

Decoding Tools

- OBD Raw Data Parser (Python middleware)

Source: https://github.com/rakshitbharat/obd-raw-data-parser

- CAN Decoder by CSS Electronics

Source: https://github.com/CSS-Electronics/can_decoder

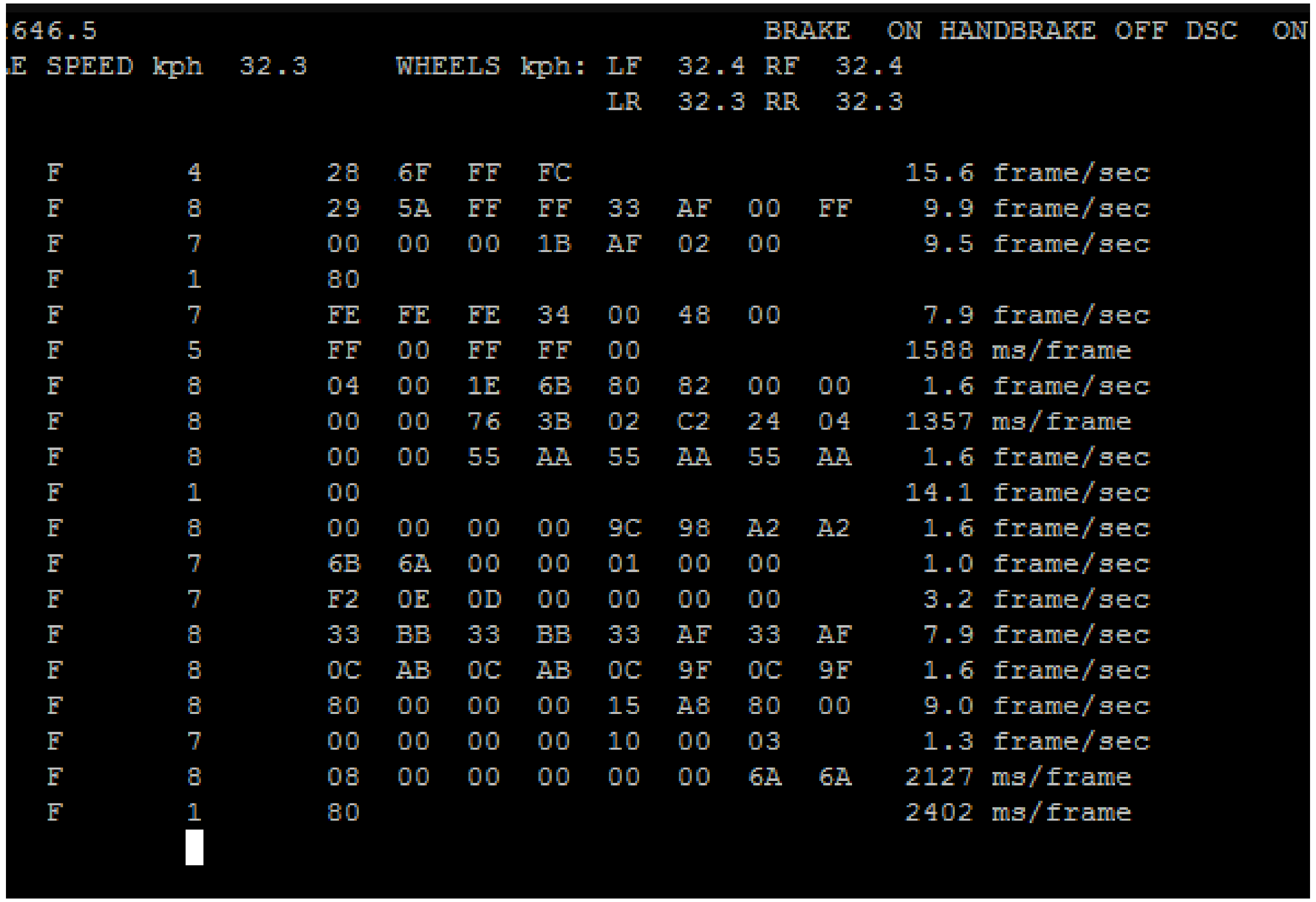


Figure 1. Example OBD-II raw data come straight from the ECU (Electronic Control Unit)

Raw Data Sources & Decoding Tools

1

Raw Data Sources

- ECUPrint Dataset [1]
- VED – Vehicle Energy Dataset [2]

2

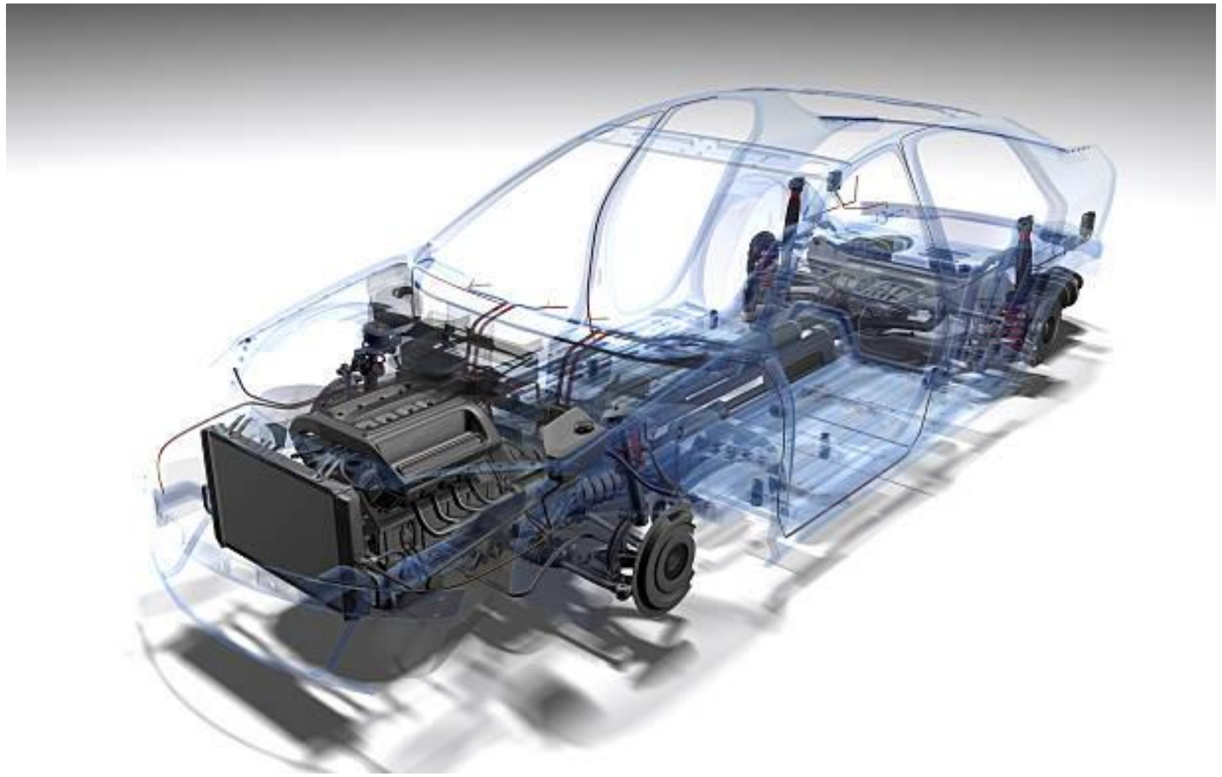
Decoding Tools

- OBD Raw Data Parser
- CAN Decoder

3

Highlights

Hex CAN frames decoded to timestamped CSV for ML pipeline and visual interpretation.



[1] https://github.com/LucianPopaLP/ECUPrint
[2] https://github.com/gsoh/VED

Sprint 1: Review

Week 1: Project Plan and Initial Architecture Blueprint

Finalised project plan, including architecture blueprint

Week 2: External Datasets & ML Research

Light modelling and classification of sample data

Week 3: Raw dataset & parsing plans

Gathered understanding of OBD Sensor data format and possible parsing options.

				Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12
1	**Planning Stage**														
1.1	Client Review and Product Research	All	1 weeks												
1.2	Potential Solution Brainstorming	All	1 weeks												
1.3	Project Planning and Literature Review	All	2 weeks												
1.4	Risk Registry and Risk Mitigation Plan	All	1 weeks												
2	**Research**														
2.1	Solution and AI Model Research	Dang + Dale	2 Weeks												
2.2	Project, Model, and Solution Constraints Investigation	Dang + Sadman	2 weeks												
2.3	Architecture Research	Sadman + Barsat	2 Weeks												
2.4	External Dataset Research	Dang + Dale	2 Weeks												
2.5	Ethical Data Collection and Consent Research (OBD-II + GPS)	Barsat	2 Weeks												
3	**Data collection**														
3.1	Physical Data Collection with OBD-II Sensor Device	All	4 weeks												

Legend
- In Progress
- Completed

Final Takeaways & Next Steps

- Explored and cleaned external OBD-II datasets
- Understood raw OBD-II data formats
- Prepared pipeline for logging and ML readiness

Next: Live data logging and testing with real vehicles.

Aim: Train anomaly detection by Sprint 2 end.

Set Threshold Alerts: In your data pipeline, flag when parameters exceed safe ranges (e.g., coolant >105 °C, fuel trim >±10 %) for immediate inspection.

Trend Analysis: Aggregate data by trip or day and plot rolling averages to spot creeping changes in temperature, voltage, or fuel trims.

Maintenance Scoring: Combine weighted metrics—e.g., 40 % fuel-trim drift, 30 % temp rise, 30 % voltage drop—into a “health index” that triggers service when it crosses a threshold.

Predictive Maintenance Indicators

1. Gradual Trends in Coolant Temperature

   •Rising baseline: A slow upward drift in coolant temp across similar driving cycles can indicate radiator clogging, failing water pump, or head gasket wear .

2. Fuel Trim Drift Over Time

   •LTFT trending positive or negative: A steady change in long-term fuel trim (e.g., moving from 0 % toward +15 % over weeks) predicts issues like airflow restrictions or injector degradation

3. Engine Load & Throttle Response Changes

   •Increasing ENGINE_LOAD at constant throttle: May reflect engine inefficiencies (e.g., carbon buildup), predicting maintenance needs for cleaning or tune-up.

4. Voltage Sag Patterns

   •Control Module Voltage decline under load (e.g., during cranking or accessories on) foreshadows battery or alternator deterioration .

5. Accumulated Engine Hours & MIL-On Time

   •TIME_RUN_WITH_MIL_ON & ENGINE_RUN_TIME: Correlating total engine-on hours with MIL-on durations helps model the likelihood of component wear and schedule preventive service before failures

Exploratory of External Raw Data in CSV

Automotive OBD-II Dataset – Karlsruhe Institute of Technology (KIT)

2.6M+ OBD-II rows cleaned for anomalies and missing values. [1]

OBD-II Dataset – Toyota Etios 2014

Five data types merged and labeled with consistent PID structure. [2]

LEVIN Vehicle Telematics Data – Yuñ Solutions

7M+ entries with missing motion sensor data and encoded loss values. [3]

Details can be found from Jupyter Notebook [4]

[1]https://radar.kit.edu/radar/en/dataset/bCtGxdTklQlfQcAq
[2]https://github.com/eron93br/carOBD
[3]https://github.com/YunSolutions/levin-openData
[4]https://colab.research.google.com/drive/1AtO5rNQdNKPHW

Data Cleaning & Feature Engineering Process

🔍 Objective: Prepare real-world OBD-II data for ML pipelines through systematic cleaning and feature preparations.

🧹 1. Data Cleaning

- **Standardization:**
  - Normalized column names (e.g., remove Â, [°], and brackets)
  - Unified datetime parsing for Time column
- **Missing Values:**
  - Handled using forward-fill; missing data rate was minimal (<0.03% fmost)
  - Example: Accelerator Pedal Position E [%] had 351 missing rows (~0.028%)
- **Anomaly Filtering:**
  - RPM > 8000 or < 0
  - Speed > 250 km/h
  - Temperatures beyond sensor design (e.g., < -40°C or > 130°C)

```python
# 1. Clean and standardize column names
df.columns = df.columns.str.strip() \
    .str.replace('Â', '') \
    .str.replace('°', '°C') \
    .str.replace('[\[\]]', '', regex=True) \
    .str.replace('°CC', '°C')  # Fix double encoding issue

# 2. Convert 'Time' column to datetime (if not already)
df['Time'] = pd.to_datetime(df['Time'], format='%H:%M:%S.%f', errors='coerce')

# 3. Handle missing values (ensure <0.03% across all relevant columns)
print("Missing Data (%):\n", df.isnull().mean() * 100)
df.ffill(inplace=True)  # Forward fill

# 4. Remove outliers or physically invalid values
df = df[
    (df['Engine Coolant Temperature °C'] >= -40) & (df['Engine Coolant Temperature °C'] <= 130) &
    (df['Vehicle Speed Sensor km/h'] >= 0) & (df['Vehicle Speed Sensor km/h'] <= 250) &
    (df['Engine RPM RPM'] >= 0) & (df['Engine RPM RPM'] <= 8000)
]

# 5. Reset index after cleaning
df.reset_index(drop=True, inplace=True)
```

Figure 3. Example Python pipeline for data cleaning on KIT dataset

```python
for file in kit_files:
    print(f"\n📄 File: {os.path.basename(file)}")
    df = pd.read_csv(file)
    print("✅ Loaded successfully.")
    print("🔢 Shape:", df.shape)
    print("📌 Columns:", df.columns.tolist())
    print("📊 Data Types:\n", df.dtypes)
    print("❗ Missing Values:\n", df.isnull().sum())
    print("📈 Basic Statistics:\n", df.describe(include='all').transpose())
    print("🔍 Sample Rows:\n", df.sample(min(3, len(df))))
```

Figure 2. Python script for raw data exploration from KIT dataset

```python
# Step 1: Group files by identical column sets
schema_groups = defaultdict(list)
for file in kit_files:
    file_path = os.path.join(kit_path, file)
    try:
        df = pd.read_csv(file_path, nrows=1) # Read first row (features)
        col_signature = tuple(df.columns)
        schema_groups[col_signature].append(file)
    except Exception as e:
        print(f"❌ Failed to read {file}: {e}")
print(f"✅ Found {len(schema_groups)} unique schema groups.")

# Step 2: Merge files with identical schema
output_dir = kit_path  # Output back to same folder
for i, (schema, files) in enumerate(schema_groups.items(), start=1):
    if len(files) > 1:
        print(f"\n🔄 Merging group {i} with files: {files}")
        combined = []
        for file in files:
            file_path = os.path.join(kit_path, file)
            df = pd.read_csv(file_path)
            df['source_file'] = file
            combined.append(df)
        merged_df = pd.concat(combined, ignore_index=True)
        output_path = os.path.join(output_dir, f"combined_{i}.csv")
        merged_df.to_csv(output_path, index=False)
        print(f"✅ Saved: {output_path}, shape: {merged_df.shape}")
```

Figure 4. Python script for grouping different logs with identical schema on KIT dataset.

Insight Summary & Learning Outcome

🎯 Purpose of This Exploratory Work => Demonstrate real-world OBD-II data handling and highlight typical challenges in preparing datasets for ML.

📌 Key Insights:

- **Raw vehicle telemetry data is rarely ML-ready**:
  - Inconsistent formats, corrupted symbols, missing timestamps, and unit saturation issues
- **Feature engineering improves clarity**:
  - Merging data based on session type adds contextual metadata for analysis
- **Data quality affects model outcomes**:
  - Uncleaned sensor caps (e.g., mass air flow = 255) can bias models

🧠 Takeaways for the Team:

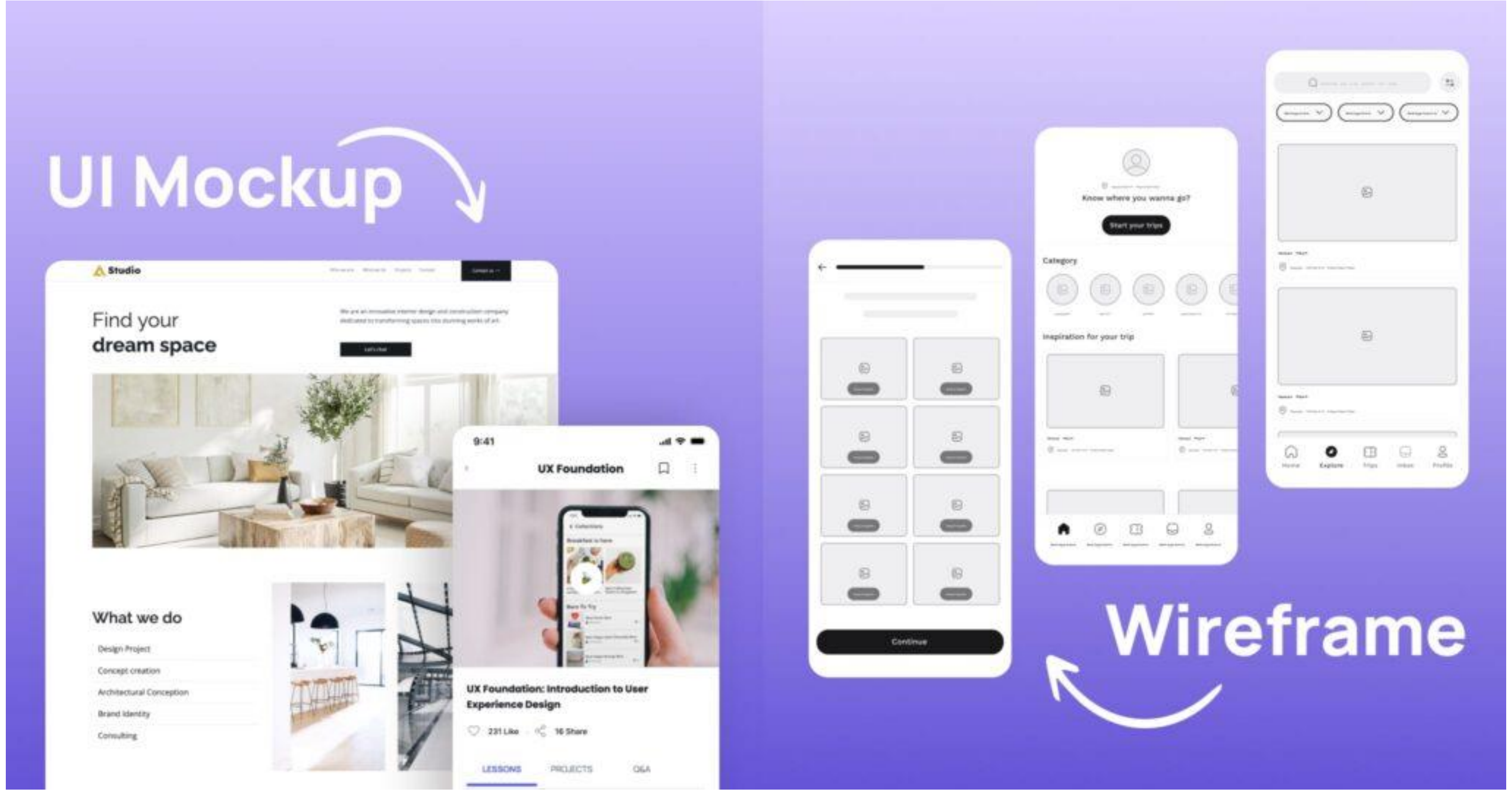
Real-world sensor data requires **thorough preprocessing** before being viable for training

A **repeatable cleaning pipeline** is essential for telemetry-based analytics

This process builds the foundation for **predictive maintenance**, **driver performance modeling**, and **fleet optimization**

- MIL Status & DTCs
  - Check Engine light = stored Diagnostic Trouble Codes (e.g., misfires, lean/rich mix)
  - Use DTCs to pinpoint critical issues (P0300, P0171, etc.)
- Cooling System Warnings
  - Engine coolant > 105 °C signals overheating
  - Spikes in temperature → possible thermostat or EGR failure
- Fuel Trim Extremes
  - STFT or LTFT > ±10% indicates fuel delivery or sensor issues
- Voltage & Sensor Anomalies
  - Control module voltage <12.5 V → battery/alternator issue
  - Slow O2 sensor response → aging catalyst or sensor

Possible UI MOCK UP



Sky Ledge Fleet Monitoring

Fleet Overview | Maintenance Alerts

Fleet Health Overview

Cargo Master 5000
Good

ID: TRK-001
Driver: John Smith

87%
Health Score

Good condition

Heavy Hauler 3000
Critical

ID: TRK-002
Driver: Sarah Johnson

42%
Health Score

Requires immediate maintenance

Road Runner XL
Warning

ID: TRK-003
Driver: Mike Wilson

68%
Health Score

May require maintenance soon

Freight Liner Pro
Good

ID: TRK-004
Driver: Robert Chen

92%
Health Score

Good condition

Cargo Express 2000
Critical

ID: TRK-005
Driver: Lisa Thompson

35%
Health Score

Requires immediate maintenance

Sky Ledge Fleet Monitoring

To exit full screen, press and hold Esc

Fleet Overview | Maintenance Alerts

← Back to Overview

Heavy Hauler 3000 Details

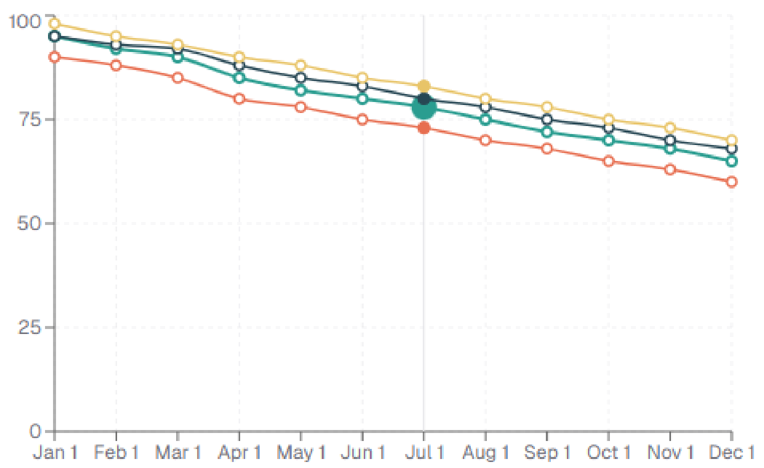
Overall Health

42%

Critical

Requires immediate maintenance

Health History



System Health | Maintenance History | Truck Details

Engine Cooling System — Critical

112 °C

Coolant temperature critically high, possible radiator clogging

Fuel System — Warning

14 %

Fuel trim drift detected, possible injector degradation

Electrical System — Warning

11.2 V

Battery voltage below optimal range

Engine Load — Warning

85 %

Engine load higher than expected, possible carbon buildup

Sky Ledge Fleet Monitoring

To exit full screen, press and hold Esc

Fleet Overview | Maintenance Alerts

Maintenance Alerts

Critical 2
Trucks requiring immediate maintenance

Warning 1
Trucks that may need maintenance soon

Healthy 2
Trucks in good condition

Critical Alerts

Heavy Hauler 3000 42%

Driver: Sarah Johnson

View Details →

Issues:

- Coolant temperature critically high, possible radiator clogging

Cargo Express 2000 35%

Driver: Lisa Thompson

View Details →

Issues:

- Coolant temperature dangerously high, possible head gasket issue
- Severe fuel trim drift, injector failure likely
- Battery voltage critically low, alternator may be failing
- Engine load excessive, immediate service required

Warning Alerts

Road Runner XL 68%

Driver: Mike Wilson

View Details →

Issues:

- Coolant temperature trending upward, monitor closely
- Fuel trim showing slight drift, possible air flow restriction