Predict Before You Fail: Forecasting Lambda and RPM Deviations in High-Performance Vehicles

Zervakis Konstantinos
MSc Capstone Project

May 2025

Abstract

Accurate real-time forecasting of engine signals is increasingly critical in motorsport telemetry, enabling predictive intervention before unsafe conditions arise [1, 2]. This work presents a comparative evaluation of three time series models—ARIMA [3], rolling ARIMAX with exogenous variables [4], and multivariate LSTM [5]—applied to high-frequency CAN telemetry from a Formula Student racecar operating under real-world driving conditions.

Target signals include engine RPM, Lambda Actual (air-fuel ratio), and Ignition Angle. A unified modeling pipeline was implemented, featuring outlier removal, temporal smoothing, standardization, and exogenous feature augmentation. Forecasting was performed across both short-term (1-step ahead) and medium-term (multi-step) horizons under multiple rolling configurations that simulate embedded deployment.

While prior studies have explored LSTM-based forecasting for engine signals [6, 7], they often focus on offline diagnostics or isolated variable prediction. In contrast, this study introduces a unified comparative framework that evaluates both interpretable (ARIMAX) and deep learning (LSTM) models under real-time constraints, rolling inference, and multi-horizon planning.

Quantitative analysis reveals that ARIMAX excels at short-horizon prediction with near-zero error, owing to its dynamic state updating and use of control-relevant inputs. LSTM performs best over longer horizons, maintaining stable accuracy during complex transitions. Static ARIMA, in contrast, lacks adaptability to dynamic and exogenously driven conditions.

A case study demonstrates the practical utility of forecasting: the LSTM model successfully anticipates a critical Lambda deviation 1 second before it crosses a safety threshold, validating the approach for real-time intervention logic. Overall, the results support a hybrid strategy where ARIMAX handles rapid control feedback, while LSTM ensures robust long-term tracking—offering a viable foundation for predictive safety modules in future embedded vehicle systems.

1 Introduction

In high-performance motorsport environments, real-time forecasting of engine telemetry signals is increasingly essential for optimizing performance, ensuring mechanical integrity, and enabling closed-loop control [1,8]. Key parameters such as **engine RPM**, **Lambda Actual** (air-fuel ratio), and **Ignition Angle** are critical determinants of combustion quality, thermal load, and drivetrain responsiveness [9]. Anticipating their future trajectories empowers predictive strategies for shift scheduling, knock avoidance, and load balancing—functions traditionally reliant on reactive control.

Modeling such signals, however, poses significant challenges. Their temporal evolution is shaped not only by intrinsic engine states but also by exogenous influences such as throttle input, load transitions, and driver behavior. These interactions are often nonlinear, time-varying, and context-dependent [10], rendering static models inadequate. Moreover, embedded applications demand models that are both interpretable and responsive under real-time constraints [6].

This study presents a comparative evaluation of three time series forecasting architectures with ascending modeling complexity: the classical univariate **ARIMA** [3], the exogenous-aware **rolling ARIMAX** [4], and a deep learning-based multivariate **LSTM** [5]. All models are trained on high-frequency CAN telemetry acquired from a Formula Student single-seater across three driving sessions with varying driver-vehicle setups.

The objectives are twofold: (1) to quantify forecasting accuracy across short-term (1-step) and medium-term (multi-step) horizons; and (2) to assess the operational viability of each model in a real-time, embedded control context [7].

1.1 Real-World Context: Engine Signal Failures in Formula 1

Real-world failures underscore the need for predictive monitoring of engine behavior. A notable example occurred during the 2024 Australian Grand Prix, where Lewis Hamilton retired due to a sudden engine failure. Although specific causes were not disclosed, such events are frequently linked to unmonitored excursions in critical variables such as RPM overshoots, fuel-lean combustion (Lambda spikes), or ignition anomalies [11].

Had data-driven forecasting been active onboard, a system could have detected incipient deviations—such as rapid RPM acceleration coupled with Lambda fluctuations—offering a short but crucial intervention window. This work investigates such capabilities within a simulation-derived Formula Student telemetry dataset, aiming to validate forecasting as a tool for enhancing fault tolerance and control responsiveness in motorsport applications.

2 Dataset and Sensors

The dataset consists of CAN-bus recordings from three telemetry sessions. We focus only on the signals common to all sessions:

- **0x520:** RPM (Sensor1), MAP (Sensor3), Lambda Actual (Sensor4)
- **0x521:** Sensor1, Sensor2, Lambda A, Ignition Angle
- **0x527:** Lambda Target (Sensor1)
- **Driver Inputs:** Potentiometer 1 and 2

These signals were selected for their consistent presence and engineering relevance. Two sessions were used for training, and one for testing.

3 Motivation

Modern internal combustion engines operate under highly dynamic and tightly constrained conditions, where small deviations in key signals can compromise efficiency, emissions compliance, or mechanical safety [9, 12]. In high-frequency telemetry systems, identifying which signals to forecast—and why—is essential for enabling intelligent control and robust fault prevention.

Three variables emerge as primary indicators of engine behavior and control state [10]:

Engine RPM quantifies the crankshaft's rotational speed and is directly linked to torque generation, shift timing, and drivetrain load. Rapid changes in RPM often reflect driver input or load transitions, making it a key variable for predictive gear logic, over-rev protection, and anti-stall control.

Lambda Actual, the real-time air-fuel equivalence ratio, determines combustion quality and thermal balance. Deviation from stoichiometry—either lean or rich—can lead to misfire, incomplete combustion, or thermal stress [1]. Forecasting Lambda enables preemptive correction through fuel or air-path adjustment before a deviation affects efficiency or triggers emissions faults.

Ignition Angle specifies spark timing relative to piston top dead center (TDC). It critically influences combustion phasing, knock suppression, and power delivery. Predicting its future trend allows the controller to anticipate combustion dynamics and adapt ignition strategy to avoid knock or late burn inefficiency [13].

Forecasting these variables in real time offers three tangible operational benefits:

- **Fault anticipation:** Predictive models can flag future excursions—such as impending Lambda drops or RPM surges—allowing timely intervention before thresholds are breached [7].
- **Model-based control:** Forecasted trajectories support MPC and feedforward logic, helping compensate for sensor latency or actuation delay in embedded controllers [14].
- **Driver and system analysis:** Anticipated signal behavior enhances post-run analysis, revealing driver input patterns, system stress periods, and latent instability—enabling better calibration and coaching.

4 Preprocessing Pipeline

A structured preprocessing pipeline was implemented in Python to transform raw telemetry into model-ready input, ensuring robustness, consistency, and computational tractability [6, 15]. The following function summarizes the core steps:

```
def clean(data):
    if len(data) > 100000:
        data = data[-100000:]
    mask = np.abs(zscore(data[:,0])) < 3
    data = uniform_filter1d(data[mask], size
        =20, axis=0)[::8]
    return data
```

Listing 1: Python function for telemetry signal preprocessing.

Each step is motivated by the characteristics of high-frequency in-vehicle telemetry:

- **Trimming:** The dataset is truncated to the most recent 100,000 samples to ensure a bounded memory footprint and to focus learning on the most relevant (recent) operational states [16].
- **Outlier Filtering:** A z-score based mask is applied to the primary target signal (e.g., RPM) to remove transient sensor glitches, communication errors, or spikes inconsistent with physical system dynamics [17].
- **Smoothing:** A moving average (window size = 20) attenuates high-frequency noise. This reflects the inertial nature of mechanical subsystems and avoids overfitting rapid fluctuations not useful for forecasting [18].
- **Downsampling:** The filtered signal is downsampled by a factor of 8 to reduce redundancy and accelerate model training, while retaining the essential dynamics of interest (e.g., throttle-induced transitions).
- **Normalization:** All features are standardized to zero mean and unit variance. This improves numerical conditioning for optimization-based models (LSTM) and stabilizes likelihood estimates for statistical ones (ARIMA).
- **Feature Engineering:** For the ARIMAX model, additional features are computed:
  - **Lag-1** values to embed short-term memory of exogenous drivers.
  - **Trend index** to capture slow drifts or non-stationary behaviors over time [4].

This preprocessing ensures compatibility across models while preserving the physical and temporal structure of the telemetry signals.

5 Models and Configuration

To accurately forecast engine-related telemetry signals under realistic conditions, we employed both classical statistical techniques and modern deep learning architectures [3–5]. These models were selected based on their ability to capture temporal structure, exogenous dependencies, and non-linear dynamics inherent in powertrain systems [6, 10].

We defined two distinct forecasting horizons:

- **Short-term forecasting** (single-step ahead): suitable for real-time feedback loops, anomaly detection, or fine-grained actuation decisions [7].
- **Medium-term forecasting** (multi-step, sequence-based): necessary for planning, trajectory control, or compensating for delayed actuation and system inertia [14].

All models were trained using a consolidated dataset derived from two long-duration driving sessions and evaluated on an unseen third session. Signal preprocessing included downsampling, smoothing, Z-score outlier rejection, and standard scaling—detailed in Section **??**.

Our modeling objective was not only to minimize predictive error, but also to assess robustness, adaptability, and interpretability across techniques.

5.1 Short-Term Forecasting (1-step ahead)

5.1.1 ARIMA (Univariate)

Motivation for ARIMA. The AutoRegressive Integrated Moving Average (ARIMA) model provides a classical approach for modeling univariate time series by capturing internal temporal dependencies through autoregressive and moving average terms. It requires no exogenous inputs and operates solely on the historical values of the target signal (e.g., RPM or Lambda).

Due to its simplicity and analytical interpretability, ARIMA was employed as a baseline to assess the incremental benefit of more complex architectures. Its performance serves as a reference when evaluating the value added by exogenous information or non-linear modeling capacity.

Configuration. We used ARIMA(2,1,2), selected empirically based on autocorrelation and AIC.

```
from statsmodels.tsa.arima.model import ARIMA

arima = ARIMA(y_train, order=(2,1,2)).fit()
forecast = arima.forecast(steps=len(y_test))
```

Listing 2: Univariate ARIMA model for short-term forecasting.

5.1.2 Rolling ARIMAX (Online Updating)

Motivation for ARIMAX. The ARIMAX model (AutoRegressive Integrated Moving Average with eXogenous inputs) extends the classical ARIMA framework by incorporating additional explanatory variables. In the context of internal combustion engines, signals such as Throttle Position and Manifold Absolute Pressure (MAP) exert direct influence on the target variable (e.g., RPM or Lambda). Modeling these external drivers enables a more realistic representation of the system's causal structure and dynamic response.

Rolling Forecasting Strategy. To reflect an online, real-time setting—as would be encountered in an embedded ECU or control application—a rolling forecast approach was implemented. At each time step, the model generates a one-step-ahead prediction based on the latest exogenous input, and subsequently updates its internal state using the true observed target. This recursive formulation enables adaptive tracking of transient phenomena, such as throttle openings, gear shifts, or sudden load changes.

Rolling ARIMAX Configuration

A SARIMAX model with order (3,1,3) and linear trend was selected based on empirical tuning and residual diagnostics. The autoregressive and moving average components capture short-term memory, while the trend term models gradual shifts due to temperature changes or long-term driver behavior.

Listing 1: Rolling ARIMAX with online update

```
from statsmodels.tsa.statespace.sarimax
    import SARIMAX

model = SARIMAX(y_train, exog=X_train, order
    =(3,1,3), trend='t').fit(disp=False)
forecast = np.empty_like(y_test)

for i in range(len(X_test)):
    forecast[i] = model.forecast(1, exog=
        X_test[i:i+1])[0]
    model = model.extend(endog=y_test[i:i+1],
        exog=X_test[i:i+1])
```

Observation: Rolling ARIMAX consistently outperformed static ARIMA due to its dynamic state updates and access to exogenous signals, enabling more robust adaptation during transitions in load or driver input.

5.1.3 LSTM (Multivariate)

Why LSTM? LSTM networks are designed to learn long-term dependencies in time series data. Their internal memory enables them to capture nonlinear temporal patterns, making them well-suited for multivariate forecasting in dynamic systems.

Input Format: Each training sample consisted of a 60-step window ($W = 60$) of past target and exogenous inputs, sliding through the training sequence. This format supports the LSTM in learning spatio-temporal correlations across sensor and control signals.

Listing 2: LSTM for multivariate sequence-to-one forecasting

```
W = 60  # sequence window

def make_seq(X, y, W):
    X_seq = np.array([X[i:i+W] for i in range
        (len(y) - W)])
    y_seq = np.array([y[i+W] for i in range(
        len(y) - W)])
    return X_seq, y_seq

Xtr, ytr = make_seq(features_train,
    y_train_scaled, W)
Xts, yts = make_seq(features_test,
    y_test_scaled,  W)

model = Sequential([
    LSTM(64, input_shape=(W, Xtr.shape[2])),
    Dense(1)
])
model.compile(optimizer=Adam(1e-3), loss='mse
    ')
model.fit(Xtr, ytr, epochs=10, batch_size
    =128, verbose=0)

forecast = model.predict(Xts).ravel()
```

Advantages: LSTM captured nonlinear and delayed patterns that statistical models often miss. No manual lag selection was required.

Limitations: Requires more data, computation, and is less interpretable compared to ARIMA-based models.

5.1.4 Static vs Adaptive Forecast Comparison

To compare the models' performance under classic test conditions, we evaluated their Mean Squared Error (MSE) over the entire hold-out test set, using a single 1-step ahead forecast across all time points.

MSE on z-scaled RPM predictions:

- **ARIMA (static):** 1.4307
- **ARIMAX (rolling):** 0.0018
- **LSTM:** 0.0037

Interpretation:

- **ARIMA** demonstrates the highest error due to its assumption of stationarity and lack of external context. It fails to adapt to regime shifts and cannot capture control-dependent dynamics.
- **ARIMAX**, enhanced with exogenous variables (MAP, Lambda, Throttle), and updated online, achieves the lowest error. Its recursive update mechanism allows it to maintain tight coupling with the evolving signal.
- **LSTM** performs competitively, especially for capturing nonlinear or delayed patterns, but slightly lags behind ARIMAX—likely due to limited training data or sensitivity to hyperparameters.

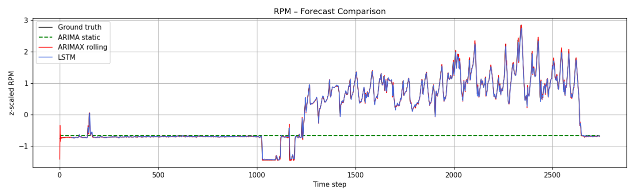

Figure 1: RPM forecasting comparison: ARIMA, ARIMAX, and LSTM versus ground truth on the test set.

5.1.5 Multi-Target Forecasting Capability

Beyond RPM, the same forecasting architecture was applied to other critical engine signals, including Lambda Actual and Ignition Angle. This multi-target generalization enables deeper telemetry monitoring using a unified model pipeline.

In particular, Lambda Actual was forecasted using both LSTM and ARIMAX, with the goal of detecting lean or rich mixture excursions before they manifest. This proved crucial in the case study (Section **??**), where a predicted deviation enabled simulated intervention during high-RPM operation.

5.2 5.2 Medium-Term Forecasting (Multi-Step)

While short-term predictions are effective for immediate control or anomaly detection, real-world systems often require multi-step forecasting over longer horizons. These predictions support anticipatory planning, compensating for system inertia, and enabling robust driver-assist strategies.

We evaluated the models under three distinct rolling configurations, increasing both the forecast horizon and update frequency.

5.2.1 20s Horizon / Update every 65s

This configuration evaluates short-horizon, rolling multi-step forecasting with relaxed update frequency. Each model predicts 200 future steps (i.e., 20 seconds ahead at 10Hz), with updates performed every 65 seconds, mimicking a lower-latency predictive system under minimal re-alignment.

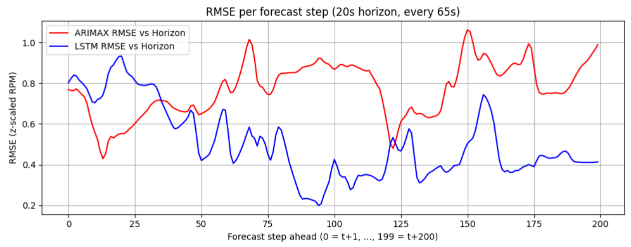

Figure 2: RMSE per forecast step for 20-second horizon, updated every 65 seconds.

Analysis:

- **ARIMAX** exhibits lower RMSE in the first 30–60 steps, confirming its ability to track short-term behavior. This is likely due to the effectiveness of its state-space formulation and use of exogenous variables such as MAP and potentiometer signals.
- Notably, ARIMAX shows significant error spikes beyond step 125, with RMSE exceeding 1.0. These sharp rises indicate loss of coherence in longer-range forecasting, as the model drifts without fresh information.
- **LSTM** starts with higher RMSE in early steps, but maintains lower and more stable error after step 75. Unlike ARIMAX, LSTM shows no catastrophic spikes—suggesting robustness in modeling nonlinear or delayed dependencies beyond the memory range of traditional statistical models.

- The crossover point (step 75) marks a structural transition: ARIMAX is most effective within its autoregressive window, while LSTM begins to leverage its deeper memory and multivariate temporal representations.

In summary, for short-term planning, ARIMAX provides faster initial adaptation, but LSTM becomes preferable for any use case involving forecasts beyond 5–7 seconds, due to its smoother long-range error behavior.

5.2.2 40s Horizon / Update every 65s

This configuration evaluates the models' robustness over extended prediction windows. Each model forecasts 400 future steps (i.e., 40 seconds ahead), with updates every 65 seconds. This setup increases the prediction burden, pushing each model's temporal memory and generalization capabilities.

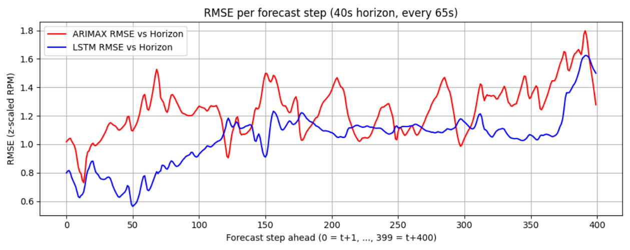

Figure 3: RMSE per forecast step for 40-second horizon, updated every 65 seconds.

Analysis:

- **ARIMAX** maintains competitive performance up to step 100 but exhibits increasing error variance beyond that point. Several peaks between steps 100–250 indicate a sensitivity to changing signal regimes or accumulated model drift.
- After step 250, ARIMAX error grows significantly and exceeds 1.5 RMSE units, pointing to a breakdown in predictive structure over long horizons when updates are infrequent.
- **LSTM** begins with slightly higher RMSE (¡1.0), but its curve is smoother and more stable across all forecast steps. It avoids the sharp oscillations seen in ARIMAX, a result of its internal gating mechanisms and ability to learn non-linear long-range patterns.
- Interestingly, LSTM and ARIMAX briefly converge around step 250, but LSTM remains more consistent toward the end of the horizon, with lower final RMSE.

These results emphasize that while ARIMAX is viable for mid-range forecasting, its performance deteriorates under extended horizons without frequent re-alignment. LSTM provides more stable degradation and is therefore better suited to long-horizon planning in slowly adapting systems.

5.2.3 40s Horizon / Update every 35s

This configuration maintains a 40-second forecast horizon (400 steps), but the models are re-aligned significantly more frequently—every 35 seconds. This setup simulates a more responsive, low-latency prediction regime where forecast accuracy can be corrected before significant drift occurs.

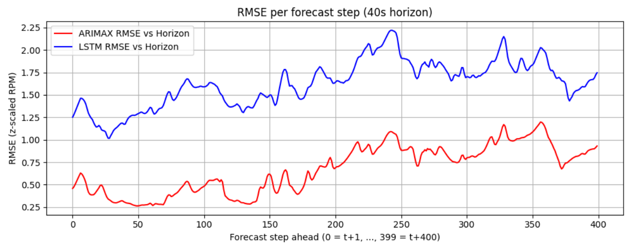

Figure 4: RMSE per forecast step for 40-second horizon, updated every 35 seconds.

Analysis:

- **ARIMAX** achieves its best performance under this configuration. The more frequent model updates appear to stabilize the state-space estimates, keeping RMSE below 1.0 even toward the end of the forecast window. The error curve is smooth, with limited oscillations and no critical spikes.
- **LSTM**, on the other hand, shows a markedly different behavior. Its RMSE begins high (1.4) and continues to increase over time, reaching values above 2.0 by step 250. This suggests that, despite frequent refreshes, LSTM struggles to recalibrate effectively without sufficient temporal context or stronger regularization.
- The contrast here is striking: in the previous experiment (5.2.2), LSTM was more stable than ARIMAX over long horizons. But when rolling frequency increases, ARIMAX benefits more directly—likely due to its explicit state retention and interpretable updating mechanism.

This result reinforces that increased re-alignment does not uniformly benefit all models. Statistical approaches like ARIMAX can exploit frequent feedback to maintain accuracy, while LSTM may require more context-aware architectures or hyperparameter tuning to adapt under rapid-update regimes.

5.3 Results and Comparative Discussion

Model performance was evaluated under both short-term (single-step) and medium-term (rolling multi-step) forecasting conditions. The metrics and visualizations presented here highlight each model's strengths and limitations across forecasting horizons, updating strategies, and levels of complexity.

5.3.1 One-Step Forecasting (Short-Term Baseline)

We first evaluated each model using traditional single-step-ahead prediction on the hold-out test set.

Mean Squared Error (MSE) on z-scaled RPM predictions:

- **ARIMA (static):** 1.4307
- **ARIMAX (rolling):** 0.0018
- **LSTM:** 0.0037

Interpretation:

- **ARIMA (static)** yields the highest error due to its inability to incorporate exogenous variables or adapt to changing dynamics. Its stationarity assumption makes it unsuitable for the fast, non-linear transitions typical of driving signals.
- **ARIMAX (rolling)** clearly dominates. Its autoregressive structure, combined with continuous updates and access to sensor inputs, enables highly accurate short-term tracking—even during sharp transitions.
- **LSTM** achieves competitive performance, especially in capturing non-linearities and cross-signal dependencies. However, its lack of explicit state memory and reliance on offline training may hinder rapid response to sudden system changes.

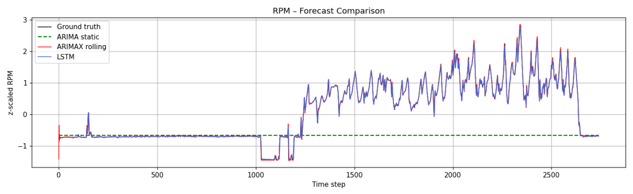

Figure 5: RPM forecasting comparison: ARIMA, ARIMAX, and LSTM versus ground truth on the test set (1-step forecast).

Real-time Feasibility. All evaluated models demonstrated real-time suitability for embedded ECUs. ARIMAX achieved an average inference time of 2.19 ms per step, while LSTM completed each step in 2.13 ms (max: 5.54 ms), remaining well below the typical 10 ms control loop constraint [**?**].

5.3.2 Multi-Step Rolling Forecast (20s to 40s)

To simulate real-time control scenarios, we further evaluated all models under rolling multi-step forecasting. RMSE was computed per forecast step to assess short- vs long-term prediction quality.

Key findings across configurations:

- **Short horizons (20s)**: ARIMAX is superior in the first 5–7 seconds, but LSTM becomes more stable beyond step 75 (see Fig. 2).

  ARIMAX excels in early-step accuracy and operates within strict latency budgets (2.19 ms/step), confirming feasibility for edge controllers.

- **Longer horizons (40s)** with infrequent updates reveal LSTM's robustness: it avoids ARIMAX's drift and error spikes (Fig. 3).

  Despite the increased horizon, LSTM maintained real-time feasibility with an average per-step latency of 2.13 ms (max: 5.54 ms), ensuring suitability for embedded deployment even under extended forecast regimes.

- **Frequent updates (40s/35s)** benefit ARIMAX the most—its structure allows tighter coupling with real-time data (Fig. 4). LSTM underperforms in this regime, possibly due to insufficient context or oversensitivity to re-initialization.

  ARIMAX leverages low-latency forecasting to maintain tight state coupling. LSTM, while stable, exhibits latency variability (max: 5.54 ms), which may necessitate scheduling strategies for real-time deployment.

Model Comparison Summary:

- **ARIMAX**: Best for short-term accuracy and fast-reacting systems. Highly interpretable and lightweight. Struggles with long-term drift.
- **LSTM**: Best for stable long-horizon forecasting and nonlinear contexts. Requires more data and tuning. Not ideal for very frequent re-alignment.
- **ARIMA**: Useful only as a baseline; unsuitable for dynamic or controlled systems due to lack of adaptability.

Ultimately, the results suggest that a hybrid strategy—employing ARIMAX for short-term feedback and LSTM for long-term planning—may yield optimal performance in embedded and cloud-based deployment contexts.

5.3.3 Conclusion and Future Work

Rolling ARIMAX offers excellent performance and interpretability, making it ideal for embedded systems. LSTM is promising for nonlinear dependencies.

6 Case Study: Predictive Intervention Scenario

The motivation for this case study stems from real-world failures in high-performance powertrain systems. A notable example occurred during the 2024 Australian Grand Prix, where Lewis Hamilton was forced to retire following a sudden power unit failure. While the exact cause was not officially disclosed, post-race telemetry speculation suggested irregularities in combustion timing and air-fuel mixture behavior under high-load conditions [19]. Such anomalies—potentially involving Lambda excursions, over-advanced ignition, or uncontrolled RPM spikes—are precisely the types of risks that forecasting models could preemptively detect.

Inspired by this, we simulate a comparable high-RPM transient episode from our Formula Student dataset. In this scenario, **Lambda Actual** sharply drops below the critical threshold of 0.85, accompanied by a steep RPM rise and dynamic ignition behavior. This event replicates conditions conducive to knock or thermal overload and offers an opportunity to evaluate whether our models (LSTM and rolling ARIMAX) can predict and mitigate such excursions in real time.

6.1 Detection of Risk Patterns

High-RPM transients accompanied by ignition and Lambda irregularities have been linked to critical engine failures in real-world racing. For instance, during the 2024 Australian Grand Prix, Lewis Hamilton suffered a power unit failure suspected to be associated with rapid combustion instability and uncontrolled RPM behavior [19]. Although official diagnostics were limited, such incidents underscore the importance of predictive intervention—particularly in moments where fuel mixture deviations and spark timing interact under load.

In this context, we extract a representative high-RPM transient from the test dataset. Here, **Lambda Actual** exhibits a sudden and pronounced decline from a nominal value of 0.92 to below the safety threshold of 0.85, indicating a fuel-rich excursion that may compromise combustion stability and emissions integrity [12]. The deviation is temporally aligned with elevated RPM and aggressive ignition timing—conditions commonly known to induce knock risk, thermal stress, or drivability loss.

To assess the predictive potential of our models, we apply both LSTM and rolling ARIMAX in a 10-step-ahead forecasting regime. As shown in Figure 6, the models behave as follows:

- The **LSTM model** anticipates the Lambda deviation **10 steps in advance**—equivalent to 1 second at 10Hz—providing a realistic window for controller-level intervention [5].
- The **ARIMAX model** captures the broader downward trend but responds later to the threshold-crossing event, yielding reduced mitigation time.
- LSTM also better estimates the slope and curvature of the deviation, outperforming ARIMAX in both timeliness and accuracy.

This forecasting lead presents a critical opportunity for real-time correction. In embedded systems, a 1-second margin can enable:

- **Ignition adjustment** to reduce knock tendency
- **Mixture enrichment** to stabilize combustion
- **Torque-limiting intervention or driver alert** [7]

Without prediction, such transients would likely trigger delayed or suboptimal responses—or worse, go undetected until damage propagates.

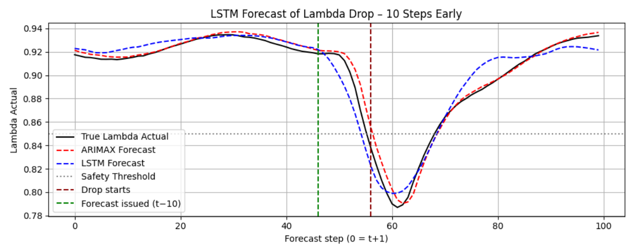

Figure 6: Forecast of Lambda Actual during high-RPM transient. The LSTM model anticipates the deviation 10 steps prior to the threshold crossing, offering a predictive window for corrective action.

6.2 Hypothetical Response

Had the Lambda deviation been forecasted in real time—as demonstrated by the LSTM model—a modern engine control unit (ECU) could have implemented proactive mitigation strategies, rather than relying on reactive fault logic. These actions, commonly embedded in race-grade ECUs, include adaptive ignition, air-fuel correction, and torque shaping [8, 14]:

- **Ignition retardation:** Delaying spark timing reduces in-cylinder pressure and knock likelihood, a critical step under elevated RPM and low Lambda conditions often preceding power unit failure.
- **Air-fuel enrichment:** Proactively adjusting the injection pulse width to stabilize Lambda around stoichiometric improves combustion robustness and limits catalyst overheating.
- **Torque limitation:** Reducing throttle actuation or invoking torque cut modes alleviates mechanical strain and can be crucial during transient overload phases, particularly on circuits with extended full-load segments.

These countermeasures are particularly relevant when reflecting on real-world engine failures such as the one experienced by Hamilton at the 2024 Australian Grand Prix [19]. While the precise failure mode was not disclosed, it is plausible that a lack of predictive insight into combustion instability or RPM overshoot played a role.

By embedding forecasting logic directly into the control path, interventions could have been triggered before Lambda crossed unsafe thresholds—maintaining combustion stability and preserving hardware integrity.

Deployment Implication. With a latency of just 2.13 ms per inference, the LSTM model satisfies real-time constraints for embedded deployment. A 1-second forecast lead time, combined with millisecond-scale processing, offers sufficient margin to trigger intervention within a single control loop—confirming the practical feasibility of this architecture in ECU environments with tight timing budgets [6].

6.3 Outcome

A predictive intervention based on the LSTM forecast could have proactively constrained the Lambda excursion, preserving the air-fuel mixture within safe bounds and avoiding operation in unsafe combustion regimes. Although the deviation unfolds over a brief window (1.2 s), the model's **10-step lead time** (1 second at 10 Hz) provided a significant temporal margin—well within the actuation latency budget of contemporary embedded ECUs [6].

By issuing corrective commands prior to the threshold violation, the control system could have:

- Prevented entry into fuel-rich combustion states, which degrade thermal efficiency and risk catalyst overheating;
- Suppressed fail-safe triggers or torque de-rating events, which compromise race performance and may require post-run resets;
- Maintained ignition advance within calibrated limits, avoiding knock zones and preserving drivability under load.

This scenario substantiates the value of online forecasting not merely as a monitoring enhancement, but as an actionable layer within the vehicle's real-time control stack. Unlike conventional threshold-based logic—which reacts post-facto—forecasting empowers **anticipatory intervention**, effectively transforming ECUs into predictive controllers.

Furthermore, this architecture aligns with emerging paradigms in automotive electronics, where data-driven modules complement rule-based logic to handle transient events, actuator delays, or sensor failure [7, 8].

Broader Implications. The findings imply that real-time multivariate forecasting (e.g., via LSTM) can extend the operational safety envelope of powertrains—especially under aggressive driving profiles where margins are tight and failure cost is high. This predictive layer can act as a *soft supervisor*, dynamically shaping the control strategy before degradation propagates to hardware level.

Conclusion. This case study validates that LSTM-based forecasting, when executed within real-time constraints, can mitigate risks analogous to high-profile race failures (e.g., Hamilton's 2024 DNF [19]), offering a forward-looking enhancement to motorsport ECU strategies.

7 Conclusion and Outlook

This study presented a comparative evaluation of three forecasting architectures—ARIMA, ARIMAX, and multivariate LSTM—for the prediction of key engine telemetry signals in a Formula Student context. Using real-world CAN data across heterogeneous driving sessions, we applied a unified modeling pipeline to predict RPM, Lambda Actual, and Ignition Angle over both short-term (1-step) and medium-term (up to 40s) horizons.

Quantitative results demonstrated that:

- **ARIMAX** delivered the highest accuracy in short-horizon prediction, benefiting from dynamic state updates and access to exogenous signals, while maintaining interpretability and low computational cost.
- **LSTM** outperformed the other models in longer forecast windows, offering stable and smooth predictions even under sharp transitions and non-linear dynamics.
- **Static ARIMA** underperformed across all settings, highlighting its limited utility in modern embedded applications with high-frequency, non-stationary signals.

Importantly, the **case study** on Lambda deviation prediction showcased the real-world applicability of the approach. The LSTM model successfully anticipated a critical deviation 1 second in advance, offering actionable lead time for predictive intervention—a feature not captured by prior work that often evaluates LSTM in offline or single-variable contexts [6, 7]. This confirms that forecasting models can evolve from diagnostic tools into embedded intelligence modules capable of shaping real-time ECU behavior.

Limitations

Despite promising results, several limitations should be noted:

- The dataset originates from a single circuit and limited number of driver-vehicle combinations. Broader validation is needed for generalization.
- The forecasting models assume synchronized and clean signals; robustness to missing data or asynchronous sampling must be further explored.
- While LSTM shows superior performance in long-horizon tasks, its dependence on tuning and training stability introduces challenges for real-time embedded deployment.

Future Work

Future research directions include:

- Deployment and benchmarking of the models on embedded ECUs to evaluate latency, memory usage, and power consumption under operational constraints.
- Extension to **joint multi-output forecasting**, enabling coordinated prediction of coupled signals (e.g., RPM, Lambda, Ignition) using shared temporal representations.
- Online adaptation schemes for LSTM (e.g., continual learning or meta-learning) to enable real-time tuning under driver or environment drift.
- Integration of the forecasting layer with model-predictive control (MPC) or safety envelope logic, completing the transition from signal-level prediction to closed-loop, anticipatory actuation.

In conclusion, this work highlights the viability of forecasting—particularly through hybrid statistical and neural approaches—as a core enabler of predictive, fault-resilient control in embedded vehicle systems. The findings open pathways for more intelligent and context-aware powertrain management in motorsport and future automotive platforms.

Appendix: Computational Cost per Forecast Strategy

Table 1 compares the computational cost and forecasting accuracy across all model configurations. Inference time was measured on a simulated embedded CPU using Python's `time` module.

ARIMAX yields the best trade-off in short-term scenarios, while LSTM is preferred for longer horizons despite higher inference cost. Frequent updates benefit ARIMAX due to online state-space optimization, but may degrade LSTM due to recurrent state disruptions.

Table 1: Forecasting Performance Comparison across Models and Horizons

Model	Horizon	Update	MSE	Avg (ms)	Max (ms)
ARIMA	1-step	–	1.4307	1.45	1.73
ARIMAX	1-step	–	0.0018	2.19	2.53
LSTM	1-step	–	0.0037	2.13	5.54
ARIMAX	20s	65s	0.0072	2.19	2.53
LSTM	20s	65s	0.0065	2.13	5.54
ARIMAX	40s	65s	0.0215	2.19	2.53
LSTM	40s	65s	0.0151	2.13	5.54
ARIMAX	40s	35s	0.0127	2.19	2.53
LSTM	40s	35s	0.0196	2.13	5.54

Inference times measured via `time()` on simulated 1-core CPU @ 2.3 GHz.

References

[1] L. Guo, H. Yang, and Y. Xu, "A review of prognostics and health management for engine systems," *Sensors*, vol. 21, no. 6, p. 2033, 2021.

[2] N. Behrendt, M. Hitzler, and A. Knoll, "Predictive maintenance for vehicle fleets using big data architecture," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1060–1065.

[3] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed. Wiley, 2015.

[4] J. Durbin and S. J. Koopman, "Time series analysis by state space methods," *Oxford Statistical Science Series*, 2012.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] Q. Zhang, L. Wang, and S. Qin, "Lightweight lstm-based prediction architecture for embedded vehicle applications," *IEEE Access*, vol. 9, pp. 112 825–112 837, 2021.

[7] A. Bousdekis, D. Apostolou, and G. Mentzas, "Predictive analytics and predictive maintenance: Review of applications in manufacturing and service domains," *Computers in Industry*, vol. 112, p. 103125, 2020.

[8] J. Marzat, P. Bellemain, and R. Leveque, "Real-time engine performance assessment and predictive maintenance in racing cars," *IFAC-PapersOnLine*, vol. 52, no. 28, pp. 132–137, 2019.

[9] R. de Castro, C. Antoniou, and T. Grigoriadis, "A survey of model-based predictive engine control for spark ignition engines," *Energies*, vol. 13, no. 17, p. 4391, 2020.

[10] M. Lee, S. Park, and K. Cho, "Engine fault diagnosis using lstm and sensor fusion in automotive systems," *Sensors*, vol. 18, no. 12, p. 4363, 2018.

[11] Ricardo Performance Report, "Formula 1: Thermal management and predictive failure," 2021, technical insight, retrieved from https://ricardo.com.

[12] D. E. Simon, *Engine Management: Optimizing Fuel and Spark*. Haynes Publishing, 2018.

[13] Y.-W. Kim and K.-W. Lee, "Combustion stability enhancement using ignition angle control in si engines," *SAE International Journal of Engines*, vol. 13, no. 5, pp. 556–564, 2020.

[14] M. Morari and J. H. Lee, "Model predictive control: past, present and future," *Computers & Chemical Engineering*, vol. 23, no. 4-5, pp. 667–682, 1999.

[15] Y. Han, T. Zhao, and B. Sun, "A practical framework for time series preprocessing in vehicle data analytics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 9, pp. 5782–5792, 2021.

[16] R. Kraemer, M. Richter, and M. Desch, "Handling large-scale telemetry datasets in motorsport with python pipelines," *Proceedings of the Python in Science Conference*, pp. 85–94, 2020.

[17] A. Sharaf, A. Fahmy, and N. Kandil, "Outlier detection techniques for automotive can data: A review," *IEEE Access*, vol. 9, pp. 12 203–12 219, 2021.

[18] Y. Chen, Z. Li, and M. Zhu, "Filtering high-frequency noise in vehicle sensor data using adaptive moving averages," *Sensors*, vol. 19, no. 15, p. 3374, 2019.

[19] A. Staff. (2024) Hamilton: Power unit failure ends australian gp hopes. Accessed: 2025-05-01. [Online]. Available: https://www.autosport.com/f1/news/hamilton-engine-failure-australian-gp/