L&T Technology Services

ENGINEERING **THE CHANGE**

8th Edition TECHgium® - Finale

PID No.3271

Presentation Round

Purposeful.
Agile.
Innovation.

Automatic compensation of torque gaps for an electrical vehicle on a rough terrain

Janani K.S

Ujjwal Raj

Vedhava Krishna

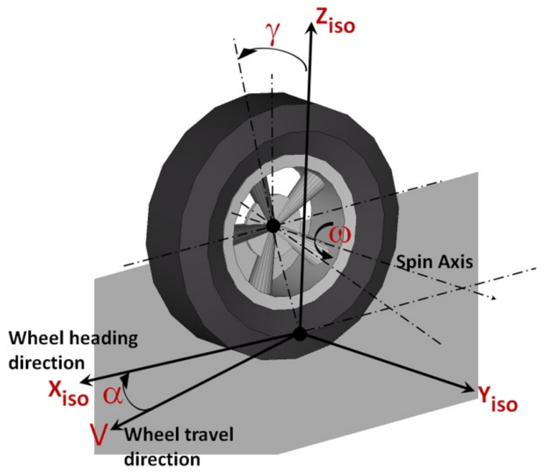
May 2025

TECHgium®
Engineering Innovators for Tomorrow

Indian Institute of Information Technology, Design and Manufacturing, Kancheepuram, Tamil Nadu

Challenge Statement

- ➢Torque gaps in electric vehicles (EVs) due to gear shifts and motor speed variations compromise stability, reduce ride comfort, and hinder performance on rough terrains.
- ➢Develop an AI-driven model that learns torque variation patterns and automatically compensates for required motor torque in a 4-wheel drive system.
- ➢Integrate real-time stability visualization using vehicle plane plots to monitor and enhance performance across diverse terrains.



Wheel Slip Condition

Instability due to wheel slip and improper distribution of toques to the wheels

Concept / Scope of solution

❖ **Concept:**

Our project focuses on an **AI-driven adaptive torque allocation system** for electric vehicles (EVs) operating on rough terrains. By leveraging **real-time IMU sensor data**, we dynamically distribute torque to each wheel based on **slip angles and terrain conditions**, ensuring **optimal stability, control, and energy efficiency**.

❖ **Solution:**

- **AI-Driven Torque Optimization:** A neural network predicts the optimal **duty cycle** for each wheel in real time, using **IMU-based slip estimation** to ensure better traction and stability.
- **Real-Time Adaptive Control:** The AI model dynamically adjusts **torque distribution** based on terrain conditions, preventing excessive power usage and reducing energy consumption.
- **Seamless Hardware Integration:** The system runs on a **Raspberry Pi**, controlling four independent DC motors through **L298N H-Bridge motor drivers**, with additional integration of an **ultrasonic sensor**, **LCD display**, and **buzzer** for obstacle detection and collision prevention—automatically halting the vehicle when danger is detected.
- **Smart Monitoring & Control:** A **Flask web app** provides real-time **stability visualization**, allowing both manual and AI-assisted control for better off-road performance.

Feedback from Previous Rounds

❖Key Feedback from Jury (Presentation Round):

Initially, our project focused on a **digital twin simulation** in Gazebo, using **sensor data** to train an AI model for **adaptive torque control and stability visualization**. However, the jury recommended developing a **hardware prototype (RC car)** for real-world validation.

❖Key Insights from Mentoring Session:

- **Use simulation data for AI training** and real-time hardware data for testing.
- **Test progressively**, starting in normal conditions before rough terrain.
- **Stability visualization (Roll vs. Time, Pitch vs. Time)** was validated as a strong performance metric.
- Suggested incorporating **a safety feature** to prevent collisions and enhance vehicle reliability during autonomous operation.

❖Enhancements to PoC:

The feedback helped us to incorporate a structured transition from **simulation to hardware** with a clear testing strategy for **adaptive torque allocation**. Also added a **safety feature** using an ultrasonic sensor, LCD, and buzzer to detect nearby obstacles, alert the user, and stop the vehicle automatically within a 10 cm threshold.

SWOT Analysis

STRENGTH

- **Real-Time Adaptive Torque Control & Stability Monitoring –** Ensures optimal traction and improved vehicle stability on varying terrains.
- **Integrated Simulation-to-Reality Pipeline –** Uses **Gazebo simulation and real-world hardware** for robust AI training and testing.
- **User-Friendly Interface with Safety Alerts –** Flask-based control panel with AI/manual toggle, direction control, live stability plots, and real-time obstacle detection via ultrasonic sensor.
- **Efficient & Transparent Operation –** Optimizes energy use, shows real-time torque distribution to all four wheels, and prevents collisions with automatic stop within 10 cm.

WEAKNESS

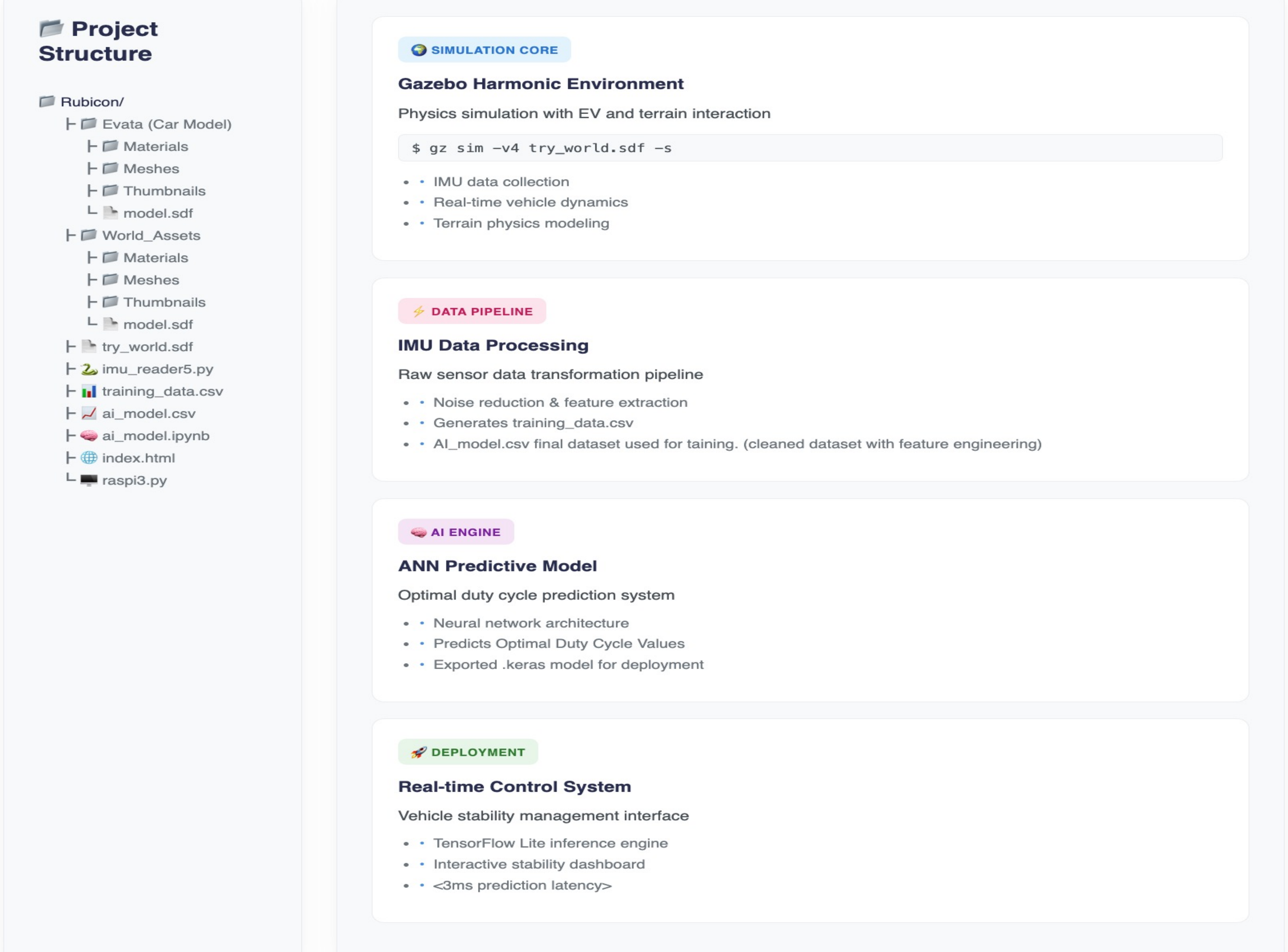
- **Raspberry Pi Stability Issues** – If not handled properly, it may **crash** during real-time torque allocation.
- **Sensor Fusion Required** – **MPU6050 alone is not enough**; additional sensors (LiDAR, encoders) are needed for better accuracy.
- **Latency in AI Predictions** – Limited computational power of Raspberry Pi might **delay duty cycle adjustments.**
- **Simulation-to-Reality Gap** – Gazebo training data may **not perfectly replicate** real-world terrain variations.
- **Limited Perception Without Vision** – Adding a camera with CNN-based road condition detection could significantly enhance safety and decision-making.

OPPORTUNITIES

- **Scalability to Real EVs –** Can be extended beyond RC cars to **full-scale electric vehicles** for off-road applications.
- **Autonomous Vehicle Integration** – Forms a foundation for self-driving EVs with **terrain-aware torque control.**
- **Industry Adoption –** Potential use in automotive R&D, motorsports, and defence vehicles for **improved traction control and energy efficiency.**
- **Energy Optimization –** The system can be integrated with **regenerative braking** and **intelligent power management**, reducing energy waste and **improving EV range**.

THREATS

- **Market Competition** – Established **automotive manufacturers and AI-driven powertrain systems** could make adoption **challenging**.
- **Hardware Reliability** – Sensor **failures or motor inefficiencies** could disrupt **real-time torque allocation**, affecting vehicle stability.
- **Environmental Variability** – **Unpredictable terrain and weather** conditions may impact the **AI model's accuracy**, leading to suboptimal torque distribution.
- **Cost & Feasibility** – Implementing **AI-driven adaptive torque allocation** in commercial EVs requires **cost-effective and scalable** hardware solutions.



📁 Project Structure

```
📁 Rubicon/
  ├ 📁 Evata (Car Model)
  │  ├ 📁 Materials
  │  ├ 📁 Meshes
  │  ├ 📁 Thumbnails
  │  └ 📄 model.sdf
  ├ 📁 World_Assets
  │  ├ 📁 Materials
  │  ├ 📁 Meshes
  │  ├ 📁 Thumbnails
  │  └ 📄 model.sdf
  ├ 📄 try_world.sdf
  ├ 🐍 imu_reader5.py
  ├ 📊 training_data.csv
  ├ 📈 ai_model.csv
  ├ 🧠 ai_model.ipynb
  ├ 🌐 index.html
  └ 🖥️ raspi3.py
```

🌍 SIMULATION CORE

Gazebo Harmonic Environment

Physics simulation with EV and terrain interaction

```
$ gz sim -v4 try_world.sdf -s
```

- • IMU data collection
- • Real-time vehicle dynamics
- • Terrain physics modeling

⚡ DATA PIPELINE

IMU Data Processing

Raw sensor data transformation pipeline

- • Noise reduction & feature extraction
- • Generates training_data.csv
- • AI_model.csv final dataset used for taining. (cleaned dataset with feature engineering)

🧠 AI ENGINE

ANN Predictive Model

Optimal duty cycle prediction system

- • Neural network architecture
- • Predicts Optimal Duty Cycle Values
- • Exported .keras model for deployment

🚀 DEPLOYMENT

Real-time Control System

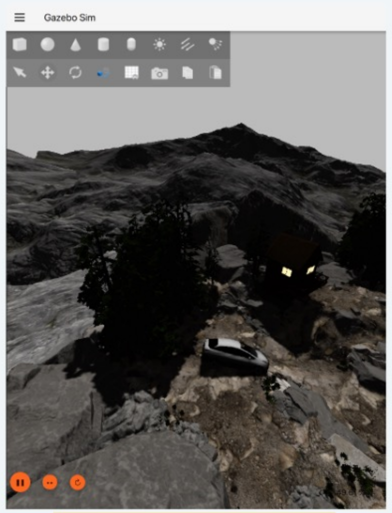
Vehicle stability management interface

- • TensorFlow Lite inference engine
- • Interactive stability dashboard
- • <3ms prediction latency>

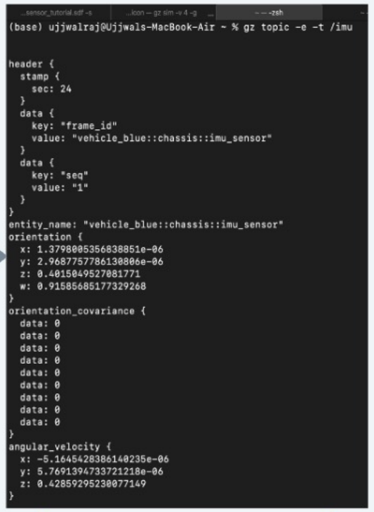
TECHpium®

- ➢ The project uses Gazebo Harmonic for simulating the car in rough terrain and collecting IMU data.
- ➢ The AI model predicts torque values, deployed on Raspberry Pi for real time vehicle control.
- ➢ The system features a web interface for remote control and stability visualization.

Implementation (1/2)

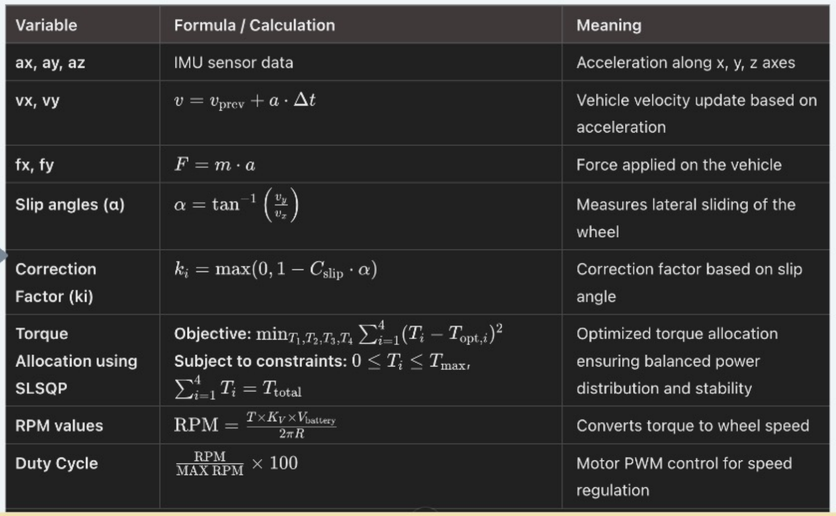


Gazebo Simulation



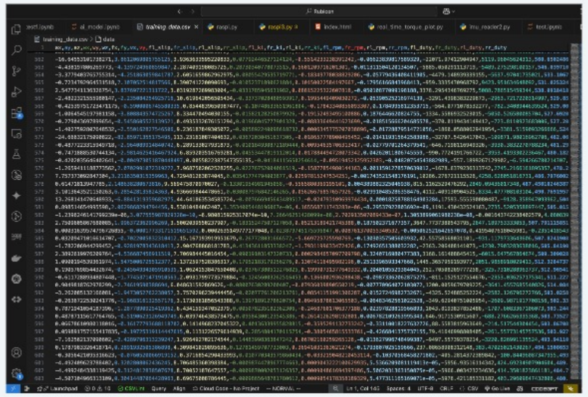
```
(base) ujjwalraj@Ujjwals-MacBook-Air ~ % gz topic -e -t /imu


header {
  stamp {
    sec: 24
  }
  data {
    key: "frame_id"
    value: "vehicle_blue::chassis::imu_sensor"
  }
  data {
    key: "seq"
    value: "1"
  }
}
entity_name: "vehicle_blue::chassis::imu_sensor"
orientation {
  x: 1.3798005356838851e-06
  y: 2.9687757786130806e-06
  z: 0.4015049527081771
  w: 0.91585685177329268
}
orientation_covariance {
  data: 0
  data: 0
  data: 0
  data: 0
  data: 0
  data: 0
  data: 0
  data: 0
  data: 0
}
angular_velocity {
  x: -5.1645428386140235e-06
  y: 5.7691394733721218e-06
  z: 0.42859295230077149
}
```

Real-Time IMU Data from Gazebo



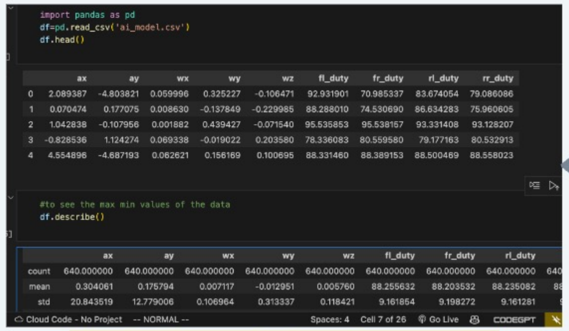
Variable	Formula / Calculation	Meaning
ax, ay, az	IMU sensor data	Acceleration along x, y, z axes
vx, vy	$v = v_{\text{prev}} + a \cdot \Delta t$	Vehicle velocity update based on acceleration
fx, fy	$F = m \cdot a$	Force applied on the vehicle
Slip angles (α)	$\alpha = \tan^{-1}\left(\frac{v_y}{v_x}\right)$	Measures lateral sliding of the wheel
Correction Factor (ki)	$k_i = \max(0, 1 - C_{\text{slip}} \cdot \alpha)$	Correction factor based on slip angle
Torque Allocation using SLSQP	Objective: $\min_{T_1,T_2,T_3,T_4} \sum_{i=1}^{4} (T_i - T_{\text{opt},i})^2$ Subject to constraints: $0 \le T_i \le T_{\max}$, $\sum_{i=1}^{4} T_i = T_{\text{total}}$	Optimized torque allocation ensuring balanced power distribution and stability
RPM values	$\text{RPM} = \frac{T \times K_V \times V_{\text{battery}}}{2\pi R}$	Converts torque to wheel speed
Duty Cycle	$\frac{\text{RPM}}{\text{MAX RPM}} \times 100$	Motor PWM control for speed regulation

Data Processing – Formulas Used for Dataset Preparation



```
training_data.csv > data
ax,ay,az,wx,wy,wz,fx,fy,vx,vy,fl_slip,fr_slip,rl_slip,rr_slip,fl_ki,fr_ki,rl_ki,rr_ki,fl_rpm,fr_rpm,rl_rpm,rr_rpm,fl_duty,fr_duty,rl_duty,rr_duty
[illegible]
```

Dataset with all feature



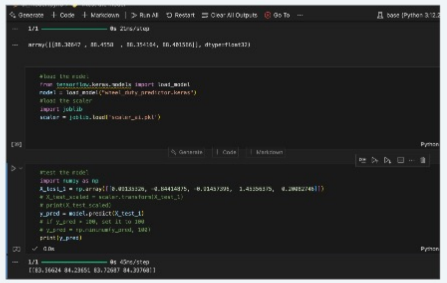
```
import pandas as pd
df=pd.read_csv('ai_model.csv')
df.head()
```

```
         ax         ay        wx         wy         wz    fl_duty    fr_duty    rl_duty    rr_duty
0  2.089387  -4.803821  0.059996   0.325227  -0.106471  92.931901  70.985337  83.674054  79.086086
1  0.070474   0.177075  0.008630  -0.137849  -0.229985  88.288010  74.530690  86.634283  75.960605
2  1.042838  -0.107956  0.001882   0.439427  -0.071540  95.535853  95.538157  93.331408  93.128207
3 -0.828536   1.124274  0.069338  -0.019022   0.203580  78.336083  80.559580  79.177163  80.532913
4  4.554896  -4.687193  0.062621   0.156169   0.100695  88.331460  88.389153  88.500469  88.558023
```

```
#to see the max min values of the data
df.describe()
```

```
               ax          ay          wx          wy          wz     fl_duty     fr_duty     rl_duty
count  640.000000  640.000000  640.000000  640.000000  640.000000  640.000000  640.000000  640.000000  640
mean     0.304061    0.175794    0.007117   -0.012951    0.005760   88.255632   88.203532   88.235082   88
std     20.843519   12.779006    0.106964    0.313337    0.118421    9.161854    9.198272    9.161281
```

Final cleaned data for model training



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
array([[88.38647 , 88.4558  , 88.354104, 88.401586]], dtype=float32)
```

```
#load the model
from tensorflow.keras.models import load_model
model = load_model("wheel_duty_predictor.keras")
#load the scaler
import joblib
scaler = joblib.load('scaler_ai.pkl')
```

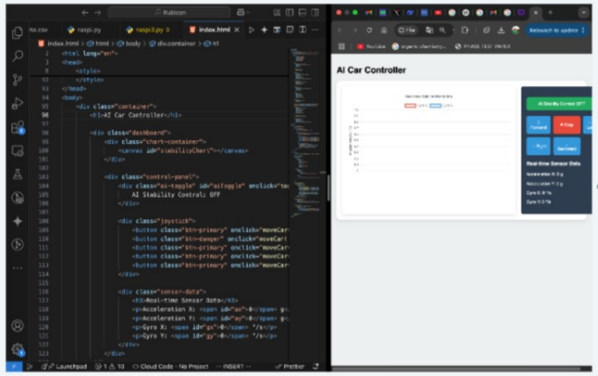
```
#test the model
import numpy as np
X_test_1 = np.array([[0.08135326, -0.84414875, -0.01457395,  1.45356375,  0.20052746]])
# X_test_scaled = scaler.transform(X_test_1)
# print(X_test_scaled)
y_pred = model.predict(X_test_1)
# if y_pred > 100, set it to 100
# y_pred = np.minimum(y_pred, 100)
print(y_pred)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
[[83.56624 84.23651 83.72687 84.39768]]
```

AI model predicting Duty Cycle

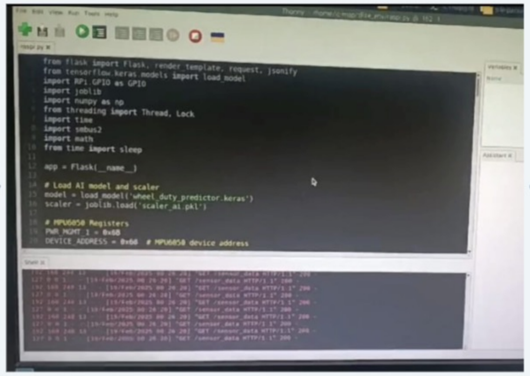
- **Gazebo Simulation & IMU Data Collection**
  Simulated a rough terrain world with an EV model in Gazebo, generating real-time IMU sensor data during vehicle movement.
- **Dataset Preparation & Torque Optimization**
  Processed IMU data to extract essential features and applied the SLSQP algorithm to determine optimal torque values for each timestamp.
- **Feature Engineering & Data Cleaning**
  Refined the dataset by selecting key features for model training, ensuring accuracy and consistency in predictions.
- **Neural Network Model for Duty Cycle Prediction**
  Trained an AI model with inputs (ax, ay, wx, wy, wz) to predict optimal duty cycle values (fl_duty, fr_duty, rl_duty, rr_duty) for real-time torque allocation.

Implementation (2/2)



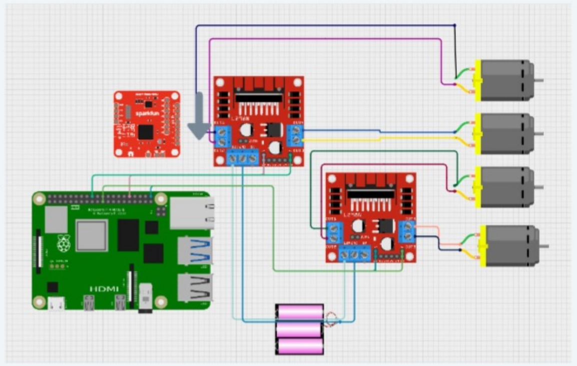
```
<html lang="en">
<head>
    <style>
    </style>
</head>
<body>
    <div class="container">
        <h1>AI Car Controller</h1>

        <div class="dashboard">
            <div class="chart-container">
                <canvas id="stabilityChart"></canvas>
            </div>

            <div class="control-panel">
                <div class="ai-toggle" id="aiToggle" onclick="to
                    AI Stability Control: OFF
                </div>

                <div class="joystick">
                    <button class="btn-primary" onclick="moveCar
                    <button class="btn-danger" onclick="moveCar(
                    <button class="btn-primary" onclick="moveCar
                    <button class="btn-primary" onclick="moveCar
                    <button class="btn-primary" onclick="moveCar
                </div>

                <div class="sensor-data">
                    <h3>Real-time Sensor Data</h3>
                    <p>Acceleration X: <span id="ax">0</span> g<
                    <p>Acceleration Y: <span id="ay">0</span> g<
                    <p>Gyro X: <span id="gx">0</span> °/s</p>
                    <p>Gyro Y: <span id="gy">0</span> °/s</p>
                </div>
            </div>
```

Flask App Code

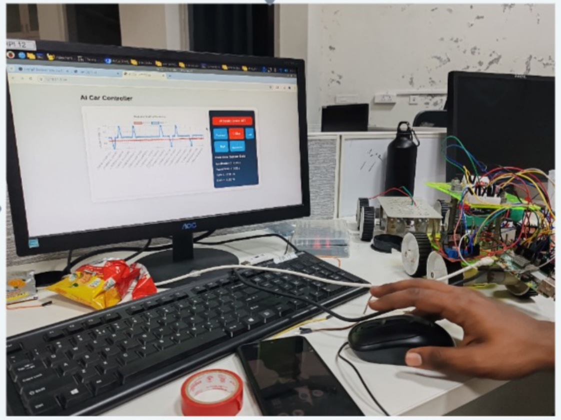


```
from flask import Flask, render_template, request, jsonify
from tensorflow.keras.models import load_model
import RPi.GPIO as GPIO
import joblib
import numpy as np
from threading import Thread, Lock
import time
import smbus2
import math
from time import sleep

app = Flask(__name__)

# Load AI model and scaler
model = load_model('wheel_duty_predictor.keras')
scaler = joblib.load('scaler_ai.pkl')

# MPU6050 Registers
PWR_MGMT_1 = 0x6B
DEVICE_ADDRESS = 0x68  # MPU6050 device address
```

Raspberry Pi Code

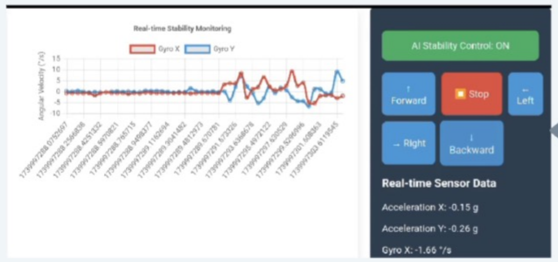


Circuit Diagram



Prototyping and testing

Car ready to run on rough surface



Real-time Sensor Data

Acceleration X: -0.15 g

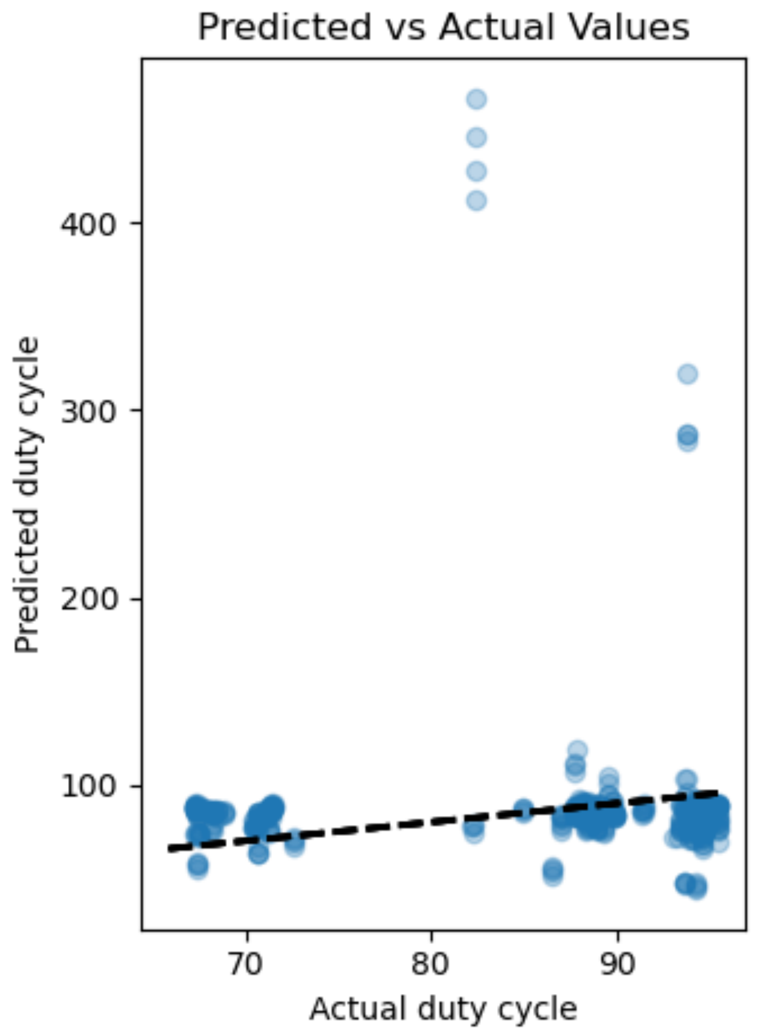
Acceleration Y: -0.26 g

Gyro X: -1.66 °/s

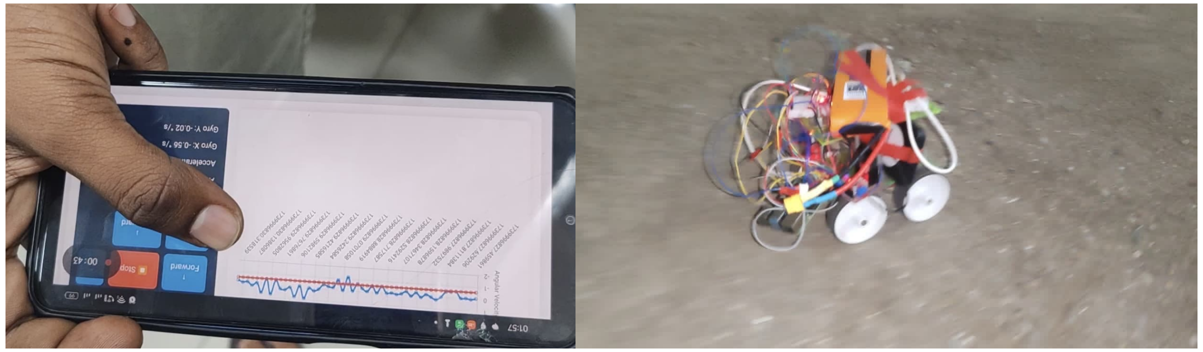
Direction Control and Stability Monitoring

- **Hardware Assembly & Prototyping**
  Integrated Raspberry Pi, MPU6050 sensor, ultrasonic sensor with LCD and buzzer, L298N motor drivers, RF 500 TB DC motors, battery, wheels, and chassis for real-world testing and safety-enabled navigation.
- **Flask App for Wireless Control**
  Developed a Flask-based interface to switch AI mode on/off, control car direction, and start/stop the vehicle remotely with real-time torque distribution display and obstacle detection alerts.
- **Raspberry Pi & AI Integration**
  Implemented AI-driven Raspberry Pi code to predict optimal duty cycles for L298N motor drivers using real-time MPU6050 sensor data.
- **Testing & Real-Time Stability Monitoring**
  Conducted rigorous testing across varying terrains, validating real-time torque allocation, visualizing roll and pitch stability plots, and verifying safety stop functionality through the ultrasonic system.

Testing / Analysis (1/2)

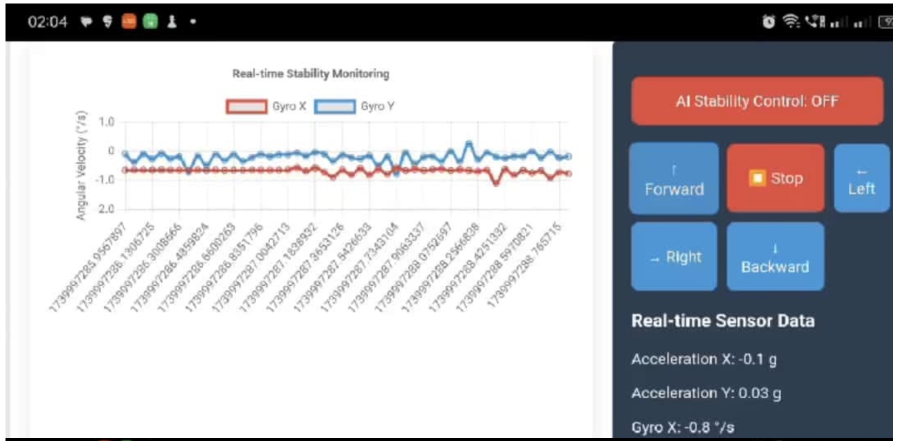


- ➢ Our AI model predicts the duty cycle well in most terrains, but in extreme conditions like steep slopes or rocky roads, it sometimes predicts very high values. This happens because the model tries to apply more torque to overcome tough terrain. The graph shows this behaviour, confirming the model's response to challenging situations.



- ➢ We successfully tested our EV car prototype on rough terrain, using a mobile device as the remote. The Raspberry Pi and mobile were connected to a common IP, enabling seamless signal transmission and command execution.

Testing / Analysis (2/2)

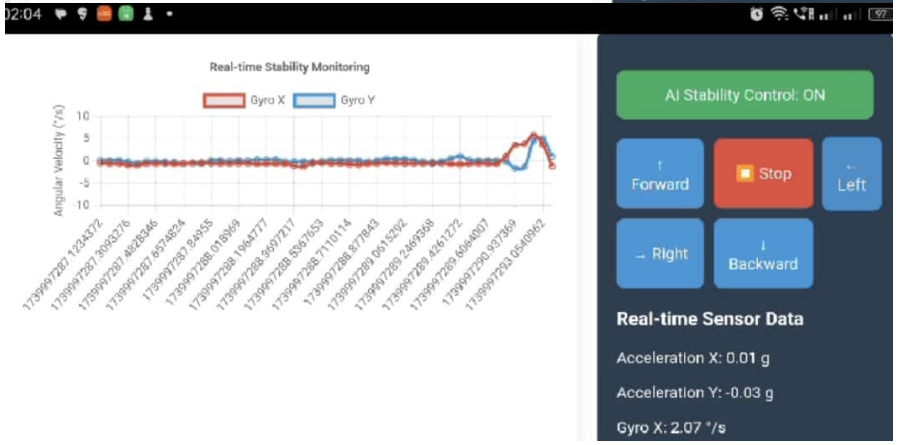


Real-time Sensor Data

Acceleration X: -0.1 g

Acceleration Y: 0.03 g

Gyro X: -0.8 °/s



Real-time Sensor Data

Acceleration X: 0.01 g

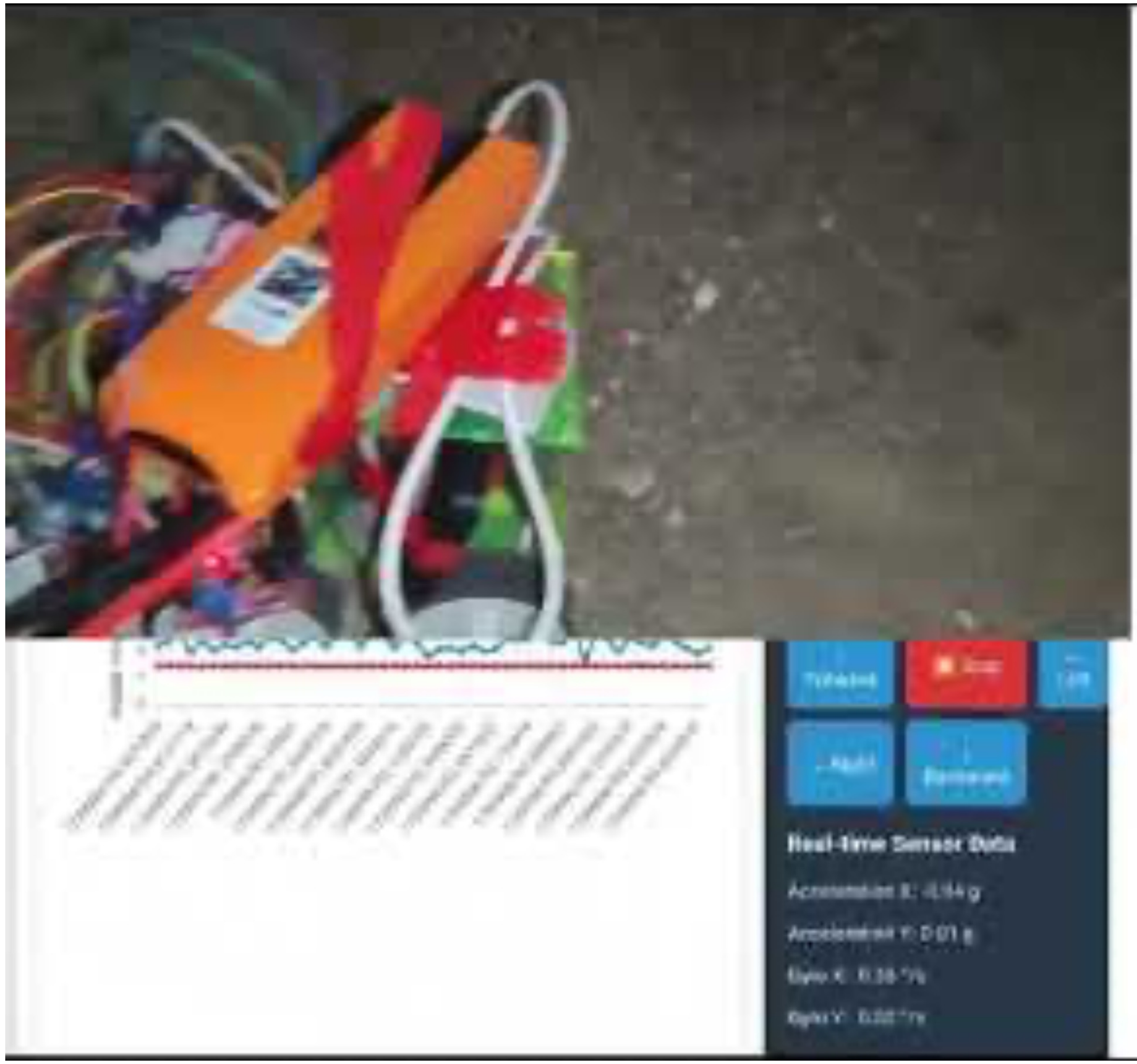
Acceleration Y: -0.03 g

Gyro X: 2.07 °/s

- **The red line shows roll vs. time, and the blue line shows pitch vs. time. When AI stability control is off, the graphs fluctuate more, indicating instability.**

- **With AI stability control on, fluctuations reduce significantly, proving that our model adapts torque in real-time by adjusting the duty cycle, improving stability on rough terrain.**

POC demo



Real-time Sensor Data

Acceleration X: -0.64 g

Acceleration Y: 0.01 g

Gyro X: 0.38 °/s

Gyro Y: 0.02 °/s

WITHOUT AI STABILITY CONTROL – HIGH FLUCTUATIONS IN ROLL AND PITCH OVER TIME INDICATE INSTABILITY.

High Level plan for converting POC to MVP

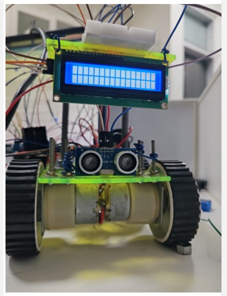
- **Upgrading to Automotive-Grade Components** – Replace toy-grade parts with industrial-grade motors, sensors, and controllers, ensuring durability and compliance through partnerships with automotive suppliers.
- **Real-World Testing & AI Model Improvement** – Train and validate the AI on actual roads, hills, and rough terrain using real-world IMU and slip data for enhanced performance in different conditions.
- **Integration with Vehicle Safety Systems** – Connect the AI to the vehicle's ECU for real-time torque control, braking, and stability management, ensuring compatibility with ABS and traction control.
- **Developing a Driver Monitoring System** – Implement a dashboard to display real-time stability, road conditions, and warnings through vibrations and audio alerts for better driver awareness.
- **Regulatory Compliance & Deployment Strategy** – Conduct extensive safety and certification testing, starting with commercial vehicles (e.g., delivery vans) before scaling to broader automotive applications.

Cost for POC vs MVP

Cost for POC

A small-scale prototype using hobby-grade components to test AI-based stability control.

- **Computing & Control:** Raspberry Pi 4 ₹8,000, Power Bank ₹2,000, Motor Drivers (x2) ₹202; **Total: ₹10,202**
- **Motors & Wheels:** DC Motors (x4) ₹1,600, Wheels (x4) ₹400 ; **Total: ₹2,000**
- **Sensors:** MPU6050 IMU Sensor ₹400, Ultrasonic Sensor ₹150, LCD Display ₹250, Buzzer ₹100 ; **Total: ₹900**
- **Power System:** Li-Po Battery **₹5,000**
- **Chassis & Wiring:** Chassis ₹500, Breadboard + Jumper Wires ₹200 ; **Total: ₹700**
- **POC (Proof of Concept) total cost – ₹18,802**



Cost for MVP

A full-scale automotive system with industry-standard hardware, AI integration, and regulatory compliance.

- **Core Electronics:** ECU ₹50,000, Rugged IMU Sensors (x2) ₹16,000, Fail-Safe Circuitry ₹30,000 ; **Total: ₹96,000**
- **Motors & Controllers:** BLDC Motors (x4) ₹2,00,000, Motor Controllers (x4) ₹60,000 ; **Total: ₹2,60,000**
- **Power & Chassis:** 48V Battery Pack ₹1,20,000, Waterproof Chassis ₹25,000, Wiring Harness ₹15,000 ; **Total: ₹1,60,000**
- **Software & AI Development:** Embedded System & AI Optimization ₹6,50,000
- **Testing & Compliance:** Safety Certifications & Field Testing ₹5,00,000
- **MVP (Minimum Viable Product) total cost – ₹16,66,000**

Result / Conclusion

The **Proof of Concept (POC)** successfully validates our AI-driven **adaptive torque allocation** system for electric vehicles. Using a **Raspberry Pi-based prototype**, we demonstrated real-time stability control on rough terrain, with AI effectively adjusting motor duty cycles to minimize roll and pitch fluctuations.

Key outcomes from our POC:

✓ **AI Stability Control Validated**: The system significantly improved vehicle stability by dynamically optimizing torque distribution using IMU data.

✓ **Real-Time Obstacle Avoidance Added**: Integration of an ultrasonic sensor with LCD and buzzer allowed the car to detect nearby obstacles and stop automatically, enhancing safety.

✓ **Seamless Remote Control & Connectivity**: The Raspberry Pi and Flask-based interface enabled smooth wireless control and monitoring over a shared network.

✓ **Successful Hardware-Software Integration**: Motors, sensors (IMU + ultrasonic), and AI logic worked in sync to execute real-time, data-driven corrections.

✓ **Scalability Potential Confirmed**: Although built with low-cost components, the validated concept is ready for industrial-grade expansion.

This PoC lays a strong foundation for developing a full-scale **MVP** equipped with **automotive-grade components**, **advanced safety feature**s, and **regulatory compliance** — paving the way for real-world deployment and commercial scalability.

References

- An Energy-Saving Torque Vectoring Control Strategy for Electric Vehicles Considering Handling Stability Under Extreme Conditions
- Data-Driven Adaptive Torque Allocation for Electric Vehicles
- Torque Distribution Strategies for Energy-Efficient Electric Vehicles With Multiple Drivetrains
- How do Ev works Video
- Fuzzy logic torque control system in four-wheel-drive electric vehicles for active damping vibration control
- Four Motor Drive and Torque Vectoring

L&T Technology Services

ENGINEERING **THE CHANGE**

Purposeful.
Agile.
Innovation.

Thank You

May 2025

TECHgium®
Engineering Innovators for Tomorrow