WRO 2025 Engineering Journal – Future Engineers

Team & Project Overview
Team Name: **Nexus Dynamics**
Category: Future Engineers
Goal: Build a fully autonomous vehicle that can navigate the track, complete 3 laps in both clockwise and anticlockwise directions, handle corners precisely using PID wall-following, and park accurately in one of six designated zones

Our objective was to design, develop, and implement a fully autonomous robot capable of decision-making based on real-time sensor data. The robot needed to navigate a fixed environment, detect colored boxes (red and green), make directional decisions, and avoid obstacles, all while staying within the defined black border. This project required the integration of mechanical design, electronics, software, control systems, and precise tuning to meet the challenge criteria within a limited size and power budget.

Hardware Implementation (week 1)

The robot's foundation was built with a compact chassis designed for stability and modularity. We began by prototyping various layouts, ultimately settling on a symmetrical design optimized for balance and wiring convenience.

Key components:

SBC: Raspberry Pi 5 (8GB RAM) — selected for its high processing power.

Power Supply: 2-cell 2000mAh LiPo battery — lightweight, high current output, and good endurance, 10,000mAh powerbank and 2,500mAh powerbank.

Motor Driver: L298N — suitable for small DC motors and provided two-channel control.

Motors: Brushed DC motors with gearbox for torque (IN1: GPIO17, IN2: GPIO27, ENA: GPIO22).

Steering Mechanism: SG90 9g micro servo — chosen for compact size, sufficient torque, and precision.

Ultrasonic Sensors: HC-SR04 front (Trig: GPIO23, Echo: GPIO24) and left (Trig: GPIO25, Echo: GPIO8).

Camera: Raspberry Pi Camera Module v2 — enabled color detection for navigation logic.

The integration of components required careful layout planning to ensure minimum electromagnetic interference and optimized cable routing. We built custom mounts for the servo and ultrasonic sensors using 3D-printed parts, each tailored to our specific mounting angles and structural constraints.

Mobility Engineering (week 2)

Mobility was at the heart of this project. The robot had to steer accurately based on color and proximity decisions, maintain high speed during open travel, and make split-second directional changes in reaction to red or green box detection.

Turning Mechanism Design

We used a front-wheel steering mechanism actuated by a 9g SG90 servo. This design allowed sharper turning compared to differential steering, and also helped preserve speed during directional shifts. The servo was connected to a 3D-printed steering link that rotated the front axle.

We designed this turning mechanism in Fusion 360, performing Finite Element Analysis (FEA) to simulate stress distribution under servo load. Areas experiencing repeated stress (servo horn) were found to be prone to fatigue, especially with thin prints. We addressed this by:

Increasing the wall count in 3D prints to 5 perimeters (instead of default 2 wall count), resulting in more solid structural sections.

Reinforcing high-load bearing joints using additional PLA fillets and localized infill boosts.

Optimizing the linkage angle to reduce required servo travel (to stay within 0-180° limits without stalling).

Acceleration mechanism;

Initially, our mobility system was powered by a **single N20 micro gear motor**, chosen for its compact form factor and ease of 3D integration. However, during early testing phases, we encountered **significant performance issues**. The N20 motor, while lightweight and simple to mount, lacked the necessary torque to reliably move the entire robot chassis across without randomly stalling.
To solve this we replaced the N20 with a higher-torque brushed DC motor, which solved the power issues but introduced a new spatial challenge. Due to its larger physical footprint, the upgraded motor couldn't be mounted horizontally within our limited chassis dimensions. To overcome the size constraint, we engineered a custom 3D-printed bevel gear mechanism that allowed the motor to be mounted vertically while transferring motion to the horizontal drivetrain. This solved our problems and to enforce the bevel gears we used 100 percent infill.

Movement Testing

We tested our turning radius, servo actuation time, and motor response across various terrain types — plastic mats, cardboard, and polished tile. The robot maintained excellent directional control and avoided oversteering due to damping logic in code.

<u>Power Management System (week 3)</u>

initially, our power system relied solely on a 10k mah power bank, chosen for its portability and familiarity. While it was able to supply 5V, this setup quickly proved insufficient as our new motor required higher current and voltage levels than the power bank could deliver. Hence we used a step up, but this caused our robot to run out of power very quickly(20 mins)

We had two options to solve this problem

Purchase a larger, high-output power bank, which would increase bulk and limit placement.

Switch to a separate LiPo system to power the motor independently.

We chose the second route for flexibility and performance. We integrated a 3-cell (3S) 2000mAh LiPo battery to exclusively handle the drive motor, which inherently outputs 11.1v — much closer to our motor's requirements. This eliminated the need for a step-up voltage regulator, which would have added inefficiency and more wiring. The LiPo not only delivered sufficient current for the motor, but was easier to palace than a larger powerbank.

Mechanical and Sensor Systems (week 4)

Structural Choices

Instead of traditional ribbing for strength, we increased 3D print wall count to 5 and infill to 35%, prioritizing rigidity over lightweighting. This strategy led to fewer weak points during stress tests and better impact resistance during collisions.

Sensor Alignment and Mounting

Ultrasonics: Front-facing sensor calibrated to detect objects <20cm. Left/right -facing sensor positioned at 45° outward to scan wall distance (triggering turns).

Camera: Mounted at a slight downward angle. Python script extracted ROI (Region of Interest) to isolate box zones and border lines.

We also tested ultrasonic interference scenarios and added averaging logic to ignore single-sample anomalies (noise).

Open challenge coding(week 5)

Day 1: GPIO Mapping & Hardware Interface

- Mapped out all ultrasonic sensors (front, left, right) using RPi.GPIO.
- Assigned control pins to the L298N motor driver.
- Calibrated the SG90 steering servo through PWM on GPIO18.
- Initial issue: PWM signal interfered with Pi boot due to overlapping with UART.
  → **Solution**: Disabled serial console and reassigned safe PWM-compatible GPIO.

Day 2: Basic Motor & Servo Logic

- Implemented forward and reverse logic for DC motors.
- Developed left/right angle steering logic using angle-to-duty cycle conversion.
- Created functions: `move_forward()`, `move_backward()`, `turn_left()`, `turn_right()`.
- Early servo bug: Servo would jitter at idle.
  → **Fix**: Added `stop_servo()` function to disable PWM signal when idle.

Day 3: Wall-Following with PID

- Implemented a **PID control algorithm** for wall-following using ultrasonic sensors.
- Set distance setpoints to 15 cm (CW) and 17.5 cm (CCW).
- Initial Kp/Kd tuning caused oscillation.
  → Refined to: `Kp = 1.5`, `Kd = 0.2`, `Ki = 0`.

Day 4: Overturning Bug

- Problem: Robot would **oversteer** during sharp corners because it continued moving too long after wall disappearance.
- Root Cause: Ultrasonic polling rate was too low; delay between checks was 50ms.
- **Fix**: Reduced ultrasonic check interval to every 100 loops with median filtering. Improved corner detection significantly.

Day 5: Bidirectional Wall Following

- Implemented toggle for left/right wall tracking.
- Developed direction-specific PID setpoints.
- Bug: Switched directions mid-run due to variable overwrite.
  → Solution: Used persistent `direction_mode` flag.

Day6+ 7: Parking Logic + precision fix and final code arrangement

- Created logic for reverse parking into 6 zones.
- Issue: Robot would **crash into wall** due to lack of front feedback.
- Fix: Added **front ultrasonic sensor** (GPIO23/24) and used it to trigger stop at 12 cm threshold during parking.
- Built lap tracking logic using turn completion criteria.

- Turn was confirmed if:
    - Servo angle ≥ 20°
    - Ultrasonic detects greater than 100cm
    - Wall reacquired within0.45 seconds
- Replaced time-based counting with angle-distance hybrid logic for consistency.
- Split logic into clean files:
    - `pid_wall_following.py`
    - `obstacle_handler.py`
    - `green_detection.py`
    - `parking_handler.py`
    - `main.py`

Obstacle challenge(week 6)

Day 1: Green Object Detection Logic

Used Pi Camera + OpenCV to detect green and red boxes.

Problem: **Walls were being misclassified as green** under bright lighting.

Fix: Implemented **CLAHE (Contrast Limited Adaptive Histogram Equalization)** to improve color contrast before HSV filtering.

Applied color mask in specific **Region of Interest (ROI)** to avoid top/bottom edge noise.

Day 2: Added the movement logic for obstacle challenge

Once an obstacle is detected it determines it colour

- If red it turns the sevo -25 and moves forward until the wall is detected and pid follows the right wall until the next obstacle is detected
- If green it turns the sevo 25 and moves forward until the wall is detected and pid follows the left wall until the next obstacle is detected

Then this process is repeated until it detects the lowest pixel of the orange line is under a vertical Y-axis threshold (y > 160 pixels) then stops. And moves forward until it detects the font ultrasonic less than 20cm then moves back(for 0.5seconds) while the servo is set at 25. Then moves forward for 0.4 seconds and continues the process 12 times and stops.
Which marks the end of the obstacle challenge.

We used ultrasonic sensors (left, right) to maintain a consistent distance from the wall during navigation. The PID controller continuously adjusts the motor speeds based on the distance error from the side wall.

Parameters:

- Proportional gain (Kp): 0.9
- Integral gain (Ki): 0.03
- Derivative gain (Kd): 0.1

END of the engineering journal