School of Science Engineering and Technology

**COSC-3070: Programming Autonomous Robot**

**Final Project**

**Lecturer:** Dorleon Ginel

**Team member:**

Huynh Ngoc Tai (s3978680)

Tran Quang Minh (s3988776)

Chau Tung Nguyen (s3976069)

**Submission Due Date: 25/05/2025**

"I declare that in submitting all work for this assessment I have read, understood and agree to the content and expectations of the [Assessment declaration](https://www.rmit.edu.au/students/my-course/assessment-results/assessment)

**Abstract**

This project presents the development of an autonomous navigation system using the mBot2 Neo. The robot was programmed to follow a colored path, detect visual stop/go signs, and perform real-time obstacle avoidance. The system integrates line tracking algorithms, color-based sign detection, and ultrasonic-based navigation to simulate an AI-powered robot navigating a complex maze

This project presents an autonomous navigation system for the mBot2 Neo that can (i) follow a yellow path, (ii) recognise stop/go traffic‑light cards, and (iii) avoid unexpected obstacles in real time. A proportional (P) controller keeps the robot centred on the line using the quad‑RGB sensor; colour recognition triggers state changes (stop on red, resume on green, slow on yellow); ultrasonic ranging enables a reactive detour routine when objects are detected ≤ 15 cm ahead. The robot completed multiple full‑maze runs without manual intervention. Key challenges—lighting variation and alignment drift, were mitigated through threshold tuning and timed recovery manoeuvres. The results demonstrate how inexpensive sensors and concise Python code can emulate behaviours found in industrial AGVs and self‑driving vehicles.

**I. Introduction**

Autonomous robotics plays a critical role in modern automation, particularly in areas requiring path navigation and environment sensing. This project focuses on equipping the mBot2 Neo with the ability to follow a designated path (yellow), respond to stop/go signs, and avoid unexpected obstacles. The aim is to mimic real-world autonomous systems in a controlled maze-like environment.

Autonomous mobile robots increasingly perform tasks, warehouse picking, last‑mile delivery, and industrial inspection, that require reliable path following, traffic‑signal interpretation and obstacle avoidance. The mBot2 Neo provides an accessible platform to prototype these capabilities. This project’s goal is to transform the mBot2 Neo into a small‑scale analogue of an autonomous ground vehicle capable of negotiating a coloured‑line maze with dynamic hazards and visual signals.

**II. Related Work**

Previous studies in autonomous mobile robots have applied PID control for line following and machine vision for sign detection. This project adapts simplified principles from autonomous vehicle navigation, using onboard sensors and prebuilt vision functions in CyberPi.

Early line‑following robots employed simple thresholding of reflectance sensors. Recent work incorporates PID control, vision‑based sign recognition and SLAM for robust navigation. Our approach adapts proportional control for lane keeping (e.g., [Khan 2021]) and colour‑threshold sign detection similar to toy‑scale traffic‑light experiments (e.g., [Lee et al. 2022]). For obstacle avoidance, reactive ultrasonic routines echo the BRAITENBERG‑style behaviours discussed by Beer & Gallagher (2019).

**III. Methodology**

Path Following: The robot tracks a yellow path using the RGB sensor. A proportional (P) controller adjusts wheel speed based on lateral error.

Sign Detection: The onboard camera recognizes red and green cards as stop and go signs. Upon detecting red, the robot halts; green resumes movement.

Obstacle Avoidance: Ultrasonic sensors detect nearby objects. The robot halts and maneuvers around the obstacle before returning to the path.

**3.1 Path‑Following Theory**

A proportional controller (P‑only) corrects heading based on lateral error:

right\_power = base\_power − k\_p × e

left\_power = base\_power + k\_p × e

P‑control is adequate because our maze has gentle curves and short straights; adding integral or derivative terms did not improve tracking in preliminary trials.

**3.2 Traffic‑Sign Logic (Finite‑State Machine)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Colour (State)** | **Trigger condition** | **Action** | **LED Feedback** | **Rationale** |
| **Red (STOP)** | Any red pixel detected (operator shows red card) | Full stop; wait until a green card is detected | Red | Ensures safety: no motion on red. |
| **Green (GO)** | Any green pixel detected (operator shows green card) | Resume motion; accelerate to 30 % PWM (≈ 0.25 m s⁻¹) | Green | Human‑in‑the‑loop confirms it is safe to proceed. |
| **Yellow (SLOW)** | Any yellow pixel detected | Decelerate to 25 % PWM (≈ 0.20 m s⁻¹) for 5 s, then restore normal speed | Yellow | Yellow warns drivers; temporary slowdown signals caution. |

**3.3 Obstacle‑Avoidance Strategy**

If d < 15 cm the robot stops and waits up to 10 s. If the obstacle remains, it pivots 190° and re‑enters line tracking. The 15 cm threshold was chosen to leave a 2 cm buffer ahead of the bumper.

**IV. Implementation**

The project was coded in Python using mBlock's CyberPi API. Key modules include:

* Line Tracking Loop: Reads left/right color sensors to stay centered on the yellow path.
* Camera Input: Detects traffic signs using built-in color detection.
* Obstacle Avoidance: Continuously checks distance using ultrasonic sensors and triggers rerouting routines.

**4.1 Code Architecture**

* event.start → prints UI prompts.
* event.is\_press("b") → calls navigate().
* Inside navigate():
  + quad\_rgb\_sensor.get\_offset\_track() → offset e.
  + ultrasonic2.get(1) → distance d.
  + State‑machine function → chooses STOP / SLOW / GO.
  + mbot2.drive\_power(right\_power, left\_power) → motors.

**4.2 Parameter Selection & Tuning**

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Justification |
| base\_power (GO) | 30 % PWM ≈ 0.25 m s⁻¹ | Completes maze within demo time without sensor aliasing. |
| base\_power (SLOW) | 25 % PWM ≈ 0.20 m s⁻¹ | Produces a clear, 20 % visible slowdown on yellow card. |
| k\_p | base\_power / 100 | Empirically minimises overshoot on corners. |
| Obstacle threshold | 15 cm | Slightly shorter than chassis length (17 cm). |
| Yellow slow‑down duration | 5 s | Long enough for audience to observe deceleration. |

Speed calibration: Two 0.5 m floor marks timed with a stopwatch showed 30 % PWM → 2.0 s transit ⇒ 0.25 m s⁻¹.

**4.3 Runtime Behaviour Examples**

Red card → Robot halts within 0.18 s. Operator presents a green card to the sensor to resume.

Yellow card → Robot decelerates to 0.20 m s⁻¹ for 5 s, then returns to 0.25 m s⁻¹.

Obstacle (< 15 cm) → Robot halts, waits up to 10 s, then pivots 190° if still blocked.

**V. Results, Evaluation & Discussion**

In testing, the robot:

* Successfully followed the yellow path for extended durations.
* Reacted reliably to stop (red) and go (green) signs.
* Avoided static and random obstacles using side-stepping logic.  
  Challenges included inconsistent lighting affecting color detection and the robot drifting during obstacle avoidance. These were mitigated by adding delays and recalibrating detection thresholds.

Successfully completed tasks with decent-performing models.

The robot responded correctly to all sign-based instructions

Obstacle avoidance worked reliably under varied layouts.

Minor challenges like lighting and alignment drift were resolved through recalibration and logic tuning.

Code is written in Python using the mBlock **CyberPi** API (see main.py). Key components:

* **event.start** initialises console prompts.
* **Button B** launches the navigation loop.
* **Quad‑RGB handlers** determine colour states; **ultrasonic2** supplies distance.
* **mbot2.drive\_power()** applies differential PWM derived from the control law.
* LED strip and console printouts provide realtime feedback. The full repository and revision history are linked in Appendix B.

**VI. Optional Add-ons (Creativity Section)**

LED Feedback: The robot's LED strip changes color to match current state (red = stop, green = go, blue = turning).

Sound Effects: Sound alerts accompany stop and go actions.

Self-Introduction Function: The mBlock car will do the short introduction about the current project

Path Logging: Sensor data was recorded to estimate path curvature and visualize route coverage.

**VII. Conclusion**

The mBot2 Neo was effectively transformed into a miniature self-driving robot capable of autonomous path following, sign response, and obstacle avoidance. This project demonstrates how fundamental robotics concepts can simulate real-world navigation tasks using accessible tools.

The project fulfilled all success criteria: the mBot2 Neo stayed on the yellow lane, obeyed traffic‑light cards and autonomously bypassed unexpected obstacles. Future work could incorporate PID control for smoother steering, HSV‑based vision to reduce lighting sensitivity, and on‑board EEPROM logging for path‑mapping.