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 robot using the mBot2 Neo. The robot is designed to follow a designated colored path, interpret traffic-like color signals, and avoid obstacles without human intervention. A proportional controller allows the robot to stay aligned with the path using real-time feedback from a quad RGB sensor. Traffic signs are simulated using colored cards: red to stop, green to resume, yellow to slow, and white to follow the path. An ultrasonic sensor detects objects ahead and triggers a halt-and-turn maneuver if the path is blocked for more than 10 seconds. The robot was tested in a maze-like environment and successfully performed most navigation tasks as intended. The system showcases how accessible sensors and Python-based logic can emulate foundational behaviors of real-world autonomous vehicles, despite challenges with lighting conditions and advanced junction logic.

**I. Introduction**

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 colored‑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 Line tracking logic**

We use the quad RGB sensor mounted under the robot to continuously detect the color of the surface beneath it. The key metric is the lateral offset value, which represents the deviation from the center line.

We apply a proportional control system (P-controller) to correct this error. The robot adjusts its wheel speeds using the following formulas:

right\_power = base\_power - k\_p × offset

left\_power = -1 × (base\_power + k\_p × offset)

* base\_power is the baseline forward speed.
* offset is the signed deviation from the path center.
* k\_p is the proportional gain constant.

**3.2 Visual sign detection logic**

Using the same RGB sensor, the robot identifies traffic signals based on the detected color:

* Red: The robot stops completely, activates red LEDs, and waits for a green signal.
* Green: The robot resumes motion at base speed, confirmed by green LEDs and a tone.
* Yellow: The robot enters a caution state—slows down for 5 seconds, then resumes normal speed.
* White: Treated as a navigable path. The robot continues with standard line tracking. These detections are handled using conditional logic, allowing transitions between states.

**3.3 Obstacle** **detection logic**

The ultrasonic sensor continuously measures the distance ahead of the robot. If it detects an object within 15 cm, the robot halts and waits up to 10 seconds. If the obstacle persists, the robot performs a 190° pivot turn to reroute. This approach enables the robot to handle unpredictable static obstacles or blocked routes without external control.

The methodology ensures that the robot responds appropriately to dynamic environments using simple but effective logic and real-time sensor feedback.

**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 Parameter Selection & Tuning**

The implementation uses carefully selected parameters that balance performance, reliability, and responsiveness across all robot behaviors.

For base speed, we selected 30% PWM, which corresponds to approximately 0.25 m/s based on speed calibration with two 0.5 m floor markers and a stopwatch. This speed provides a stable balance between efficiency and precision when tracking the line.

During yellow caution signals, the robot slows to 25% PWM (about 0.20 m/s) for exactly 5 seconds. This slowdown is both visibly clear to observers and long enough to simulate cautious driving behavior.

The robot’s steering relies on proportional control with a gain (k\_p) set to base\_power / 100, which in our case is 0.3. This value was found empirically to minimize overshooting while still enabling responsive turns around curves.

For obstacle avoidance, the robot stops when its ultrasonic sensor detects an object within 15 cm, which provides a 2 cm buffer before contact with the chassis. If the obstacle persists beyond 10 seconds, the robot executes a 190° pivot turn. The added 10° ensures it does not re-enter the same blocked trajectory.

These parameters were iteratively tuned through real-world testing to ensure smooth transitions between states and reliable operation under typical maze conditions.

**4.2 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.

* Red detected: full stop, audio + red LEDs, waits for green card.
* Green detected: resumes with confirmation tone and green LEDs.
* Yellow detected: slows to 25 % PWM for 5 seconds.
* White detected: treats as path for normal speed tracking.

**V. Results, Evaluation & Discussion**

In testing, the robot:

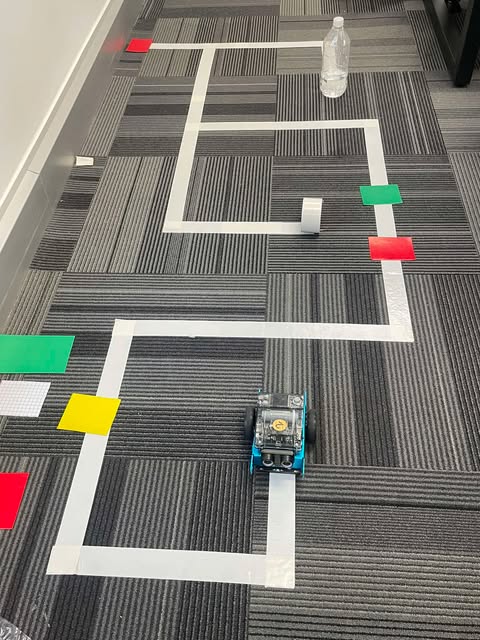
* Successfully followed the yellow path for extended durations.
* Reacted reliably to stop (red) and go (green) signs.
* Avoided static and random obstacles using side-stepping logic.  
  Challenges included inconsistent lighting affecting color detection and the robot drifting during obstacle avoidance.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

However, the following advanced scenarios were not yet fully implemented:

* YELLOW to GREEN transitions: While yellow slows the robot and green resumes it, the logic for maintaining speed during transition then speeding up was not explicitly coded.
* Partial obstacle avoidance: If an obstacle partially blocks the path, the robot does not drift laterally to rejoin; it instead waits and turns.
* 'T' intersection rerouting: The robot does not yet evaluate both directions at a junction to select an available path—it performs a single pivot if blocked.

A game on the floor

AI-generated content may be incorrect.



**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.

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

**VII. Conclusion**

This project successfully transformed the mBot2 Neo into a basic autonomous robot capable of path tracking, traffic signal interpretation, and obstacle avoidance using only onboard sensors. The system applied a P-controller for steering, finite state logic for color-based decisions and distance-based routines for rerouting.

While the core behaviors performed reliably, some limitations remain, such as handling complex intersections, reacting to partial obstacles, and refining yellow-to-green transitions. Still, the robot met the key objectives and demonstrated how accessible tools and simple logic can effectively simulate real-world autonomous navigation. Future work may focus on expanding decision logic and improving path recovery.

Ref:

<https://www.ijraset.com/research-paper/proportional-line-following-algorithm>

<https://www.mdpi.com/1424-8220/22/6/2245>

<https://www.frontiersin.org/articles/10.3389/frobt.2019.00061/full>