School of Science Engineering and Technology

RMIT UNIVERSITY

COSC-3070: Programming Autonomous Robot

Final Project Report

Lecturer: Dorleon Ginel

Team member:
Huynh Ngoc Tai (s3978680)
Tran Quang Minh (s3988776)
Chau Tung Nguyen (s3976069)

Submission Due Date: 25/05/2025

I. 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.

II. 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.

III. 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. Here are some recent work incorporates PID control, vision-based sign recognition and SLAM for robust navigation. Our approach adapts proportional control for lane keeping (Khan 2021) and colour-threshold sign detection similar to toy-scale traffic-light experiments (Lee et al. 2022). For obstacle avoidance, reactive ultrasonic routines echo the BRAITENBERG-style behaviours discussed by Beer & Gallagher in 2019 (Beer RD and Gallegher JC 2019).

IV. Methodology

This project implements three core robotic behaviors which are the path following, traffic sign recognition and obstacle avoidance that according to the structured sensor input and real-time decision logic. These behaviors are unified in a continuous control loop that guides the mBot2 go through a maze environment.

1. **Line tracking logic**

The robot uses a quad RGB sensor mounted on its underside to track the colored path, typically white or yellow. This sensor continuously detects the color beneath each of its four positions and calculates a lateral offset value, representing how far the robot has deviated from the path center.

This offset is used as the input for a Proportional (P) controller. The P-controller calculates correction values for the motor speeds to steer the robot back toward the center of the path. The robot's movement is governed by the following equations:

right_power = base_power - k_p × offset

left_power = -1 × (base_power + k_p × offset)

- base_power: defines the robot's standard forward speed.
- offset: is the real-time lateral deviation from the path center as reported by the sensor.
- k_p: is the proportional gain factor that determines how aggressively the robot steers back to the path.

The control logic ensures that when the robot drifts to the left (negative offset), the right wheel slows down and the left wheel speeds up, steering it back toward the line. The opposite occurs for positive offset. This allows for smooth and responsive corrections without oversteering, even on curved sections of the maze.

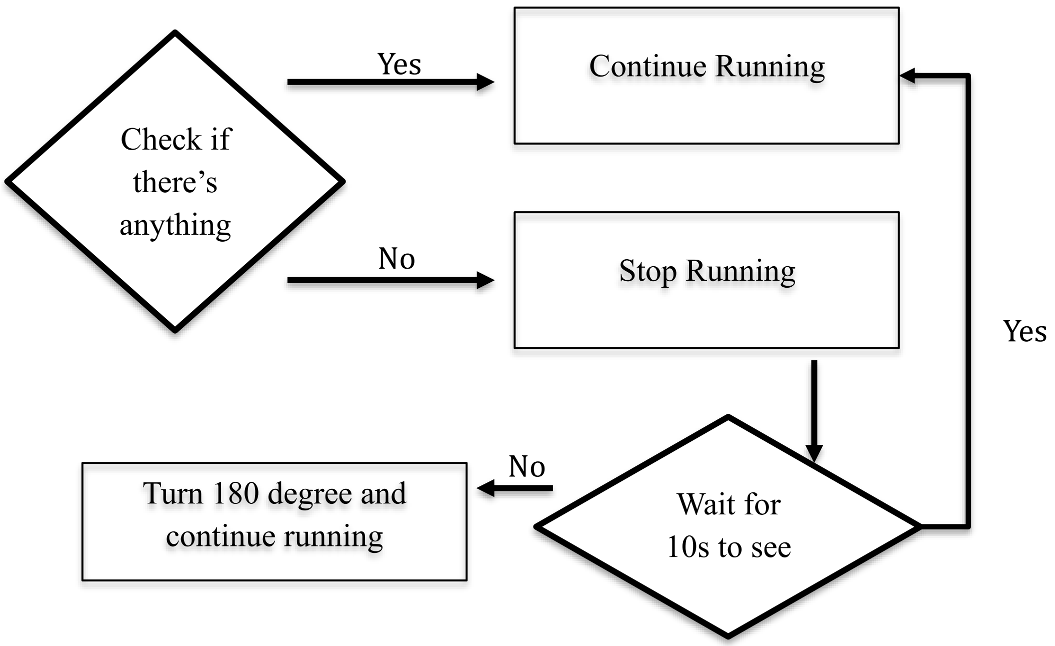
Combined with frequent sampling and modest base speed, this method enables the robot to maintain consistent and stable path-following behavior.

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 10 seconds or a green signal then turns back.
- Green: The robot resumes the motion at base speed, confirmed by green LEDs and a tone.
- Yellow: The robot enters a caution state and slows down the speed for 5 seconds, then go back to the normal speed base.
- 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. Obstacle detection logic


Check if there's anything
Yes
Continue Running
No
Stop Running
Yes
No
Turn 180 degree and continue running
Wait for 10s to see

The obstacle avoidance logic in our robot is designed as a reactive state machine that mirrors a simple but effective decision-making process. This system allows the robot to autonomously respond to unpredictable changes in its environment using the ultrasonic sensor.

At each iteration of the control loop, the robot checks for any obstruction in front using its ultrasonic sensor. If no obstacle is detected, then the robot continues to line following as normal. If an obstacle is detected, the robot immediately stops and enters a waiting state. This wait period gives time for temporary obstacles (such as a passing hand or moving object) to clear naturally.

During this pause, the robot continuously monitors the distance to see if the object has moved out of range. If the obstacle disappears before the timeout, the robot resumes navigation on its current path. However, if the object remains in place beyond the fixed time threshold, the robot interprets it as a static blockage. It then performs a pivot turn (in our case, 190°) to exit the current path and search for an alternate route.

This logical sequence of detect, stop, wait and reroute to ensures that the robot can react adaptively in dynamic environments without relying on external commands. The use of a wait period before turning minimizes unnecessary rerouting and mimics basic deliberative reasoning found in more complex autonomous systems.

V. Implementation

The robot was programmed in Python using the mBlock CyberPi API, with a modular structure that reflects each core navigation task. The code integrates three major functionalities:

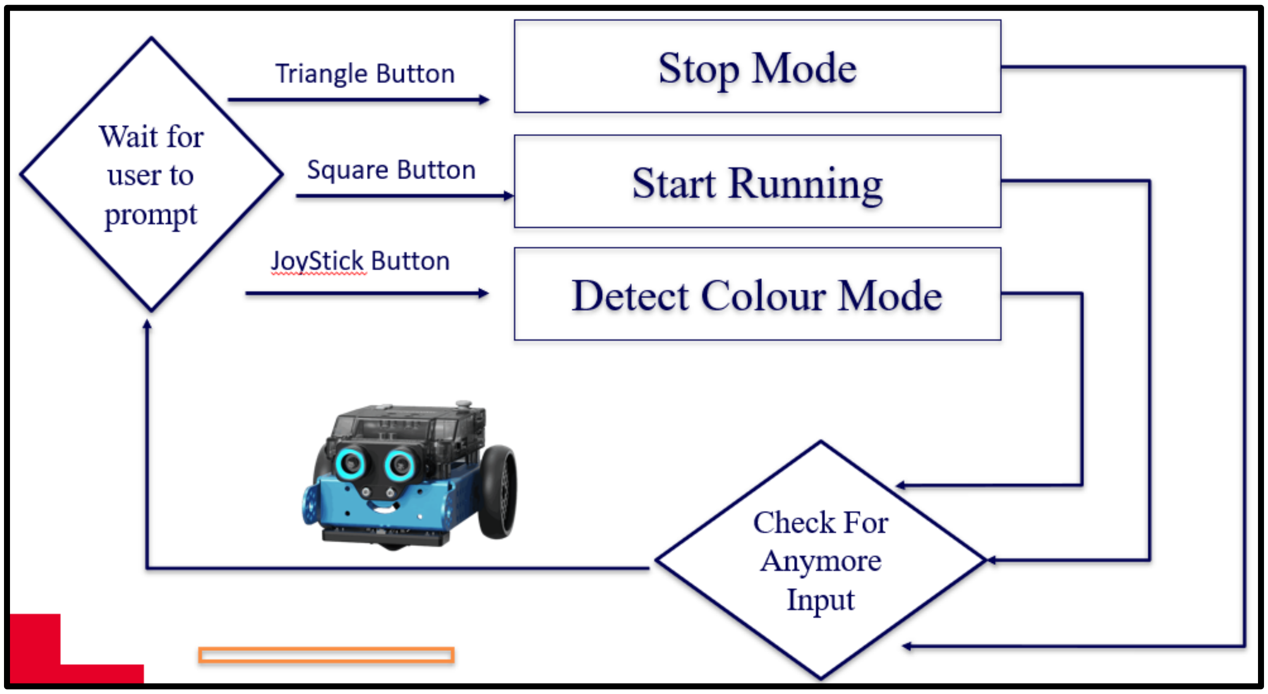
- Line Tracking Loop: Continuously reads RGB sensor offset values and applies a proportional controller to adjust motor speeds for center alignment.
- Color Sign Detection: Uses RGB sensor input to determine whether the robot should stop, go, or slow down based on traffic-like color cues (red, green, yellow, white).
- Obstacle Avoidance: Actively checks the frontal environment using the ultrasonic sensor and initiates stop-wait-reroute logic when objects are detected.-

The whole code is structured around event-driven triggers and a main navigation loop, using the CybePi event module to handle user interation through on device buttons and runtime behaviour which handle all the functionalities above

1. **System Initialization and Event Handling**

When the robot is powered on, the user is prompted to choose from the following operating modes:

- Stop Mode (Activated by pressing the Triangle button): Sets the base power in the proportional control formula to zero, effectively stopping the robot.
- Run Mode (Activated by pressing the Square button): Sets the base power to 30 and starts the main control loop, which includes line tracking, LED-based traffic signal responses and obstacle avoidance logic.
- Color Detection Mode (Activated by pressing the Joystick button): Uses the quad-color sensor to display the currently detected color on the console, allowing users to test, debug and calibrate color recognition.



2. Parameter Selection & Justification

To implement the control strategies described in our methodology, we selected specific numeric values that align each robot behavior with realistic performance expectations. These parameters were iteratively tested and tuned for stability, responsiveness and clarity of state transitions.

For line following, we applied a base movement speed of 30% PWM, equivalent to approximately 0.25 m/s. This speed enables the robot to maintain accurate path tracking while completing the maze in a reasonable duration without excessive overshoot.

The proportional gain constant (k_p) was defined as base_power / 100, yielding a value of 0.3. This gain level allowed smooth curvature correction based on real-time deviation measurements from the RGB sensor. Higher gains led to instability on bends, while lower gains made the robot sluggish.

Upon detecting a yellow caution signal, the robot slows down to 25% PWM for 5 seconds. This fixed duration provided a visible and realistic reaction to simulate safe slowdown behavior at intersections or warning zones.

For obstacle avoidance, we set the ultrasonic threshold to 15 cm, offering a 2 cm margin before potential collision with the chassis. If an obstacle remained for more than 10 seconds, the robot executed a 190° pivot turn to avoid re-entering a blocked zone. This extra 10° ensured clear redirection without looping or stalling.

These values were selected through a mix of theoretical estimation and trial-and-error refinement during physical tests. Collectively, they ensure consistent, repeatable behavior under all standard operating conditions.

3. Color-Specific Behavior in Runtime

The robot's behavior during runtime reflects the implementation of our line tracking, traffic signal and obstacle avoidance strategies. Each color detection triggers a defined response governed by real-time sensor input and control logic:

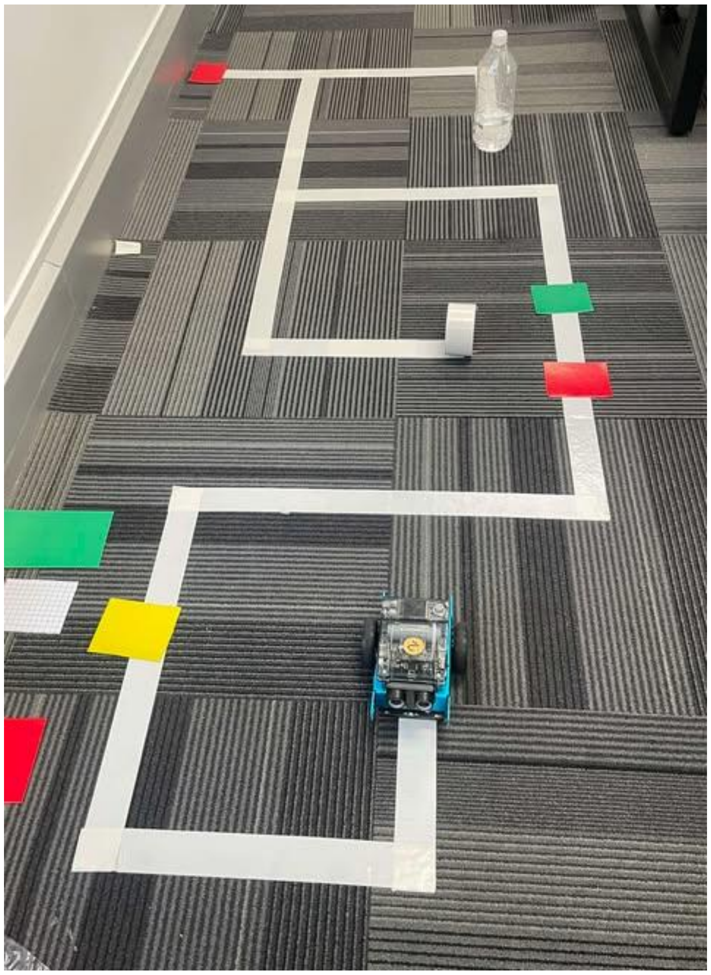
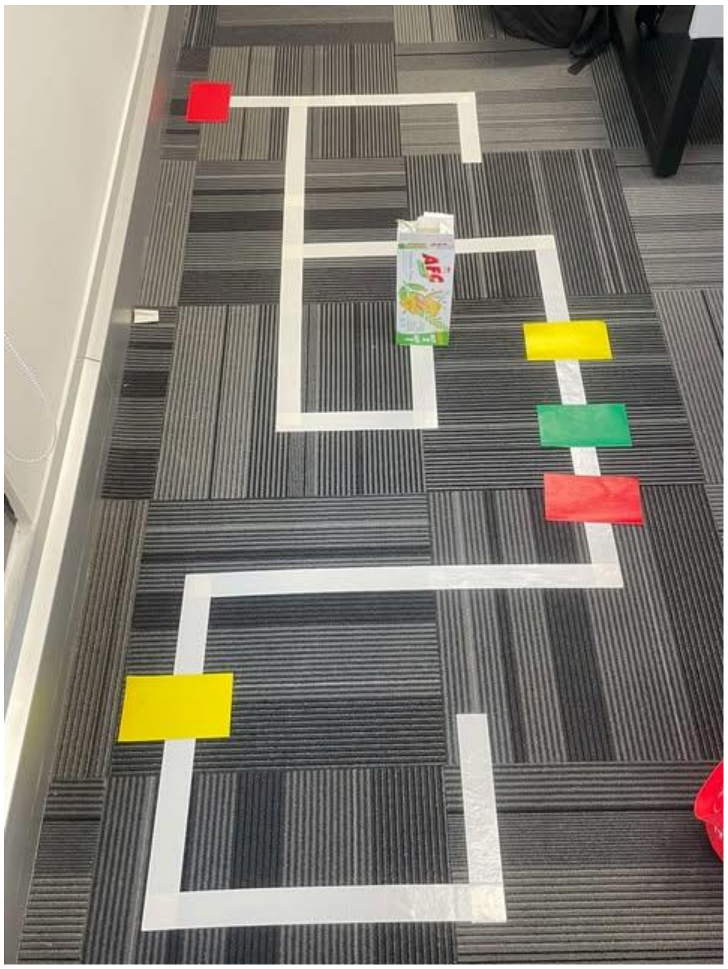
- Red detected: The robot stops immediately using mbot2.EM_stop(), activates red LEDs and plays a low-frequency tone at 261 Hz. It remains in this state until a green signal is detected or until 10 seconds pass, after which it executes a 180–190° U-turn to resume progress.
- Green detected: If the robot was previously stopped by a red signal, detecting green initiates forward motion at 30% PWM. This is accompanied by a confirmation tone at 659 Hz and green LED illumination to confirm the state change.
- Yellow detected: Upon identifying a yellow signal, the robot reduces its speed to 25% PWM and enters a 5-second caution state. This slowdown is maintained using the P-controller at a reduced speed, after which the robot automatically returns to normal

operation. A specific mid-range tone at 294 Hz and yellow LEDs indicates this transitional state.

- White detected: Treated as a default track signal, the robot continues line following at 30% PWM. This is the baseline behavior during unmarked navigation sections.

VI. Results, Evaluation & Discussion

In order to ensure that the robot is able to perform in the real demo, we decided to test the robot in the test environment which we tried to implement every scenario possible in a smaller scale maze as following.

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.
- Able to turn 180 degrees when the red sign and obstacle are ahead and not changed.
- Challenges included inconsistent lighting affecting color detection and the robot drifting during obstacle avoidance. These were mitigated by adding delays and recalibrating detection thresholds.

After the real demo, there are some 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 for the first GREEN transition then speeding up at the second GREEN was coded, but due to the short amount of time as per required for the yellow sign (5 seconds), which make it too slow for the robot to actually recognize the second Green sign to start working back.
- 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 will perform a single pivot if blocked.

VII. Creativity Section Add-ons

To enhance both usability and interactivity, several creative features were integrated into the robot's system:

- LED Feedback: The LED strip is used to indicate the robot's state in real time. Red LEDs activate when the robot stops, green LEDs illuminate during forward movement and blue LEDs flash during turns or rerouting actions. This visual feedback makes it easy for observers to interpret robot behavior at a glance.
- Sound Effects: Distinct tones were assigned to specific transitions. A low-frequency tone plays when the robot halts, a high-frequency tone indicates motion resumption (on green) and a mid-tone is triggered during yellow-based slowdowns. These sounds serve as an audible layer of feedback during testing and presentations.
- Path Logging (Conceptual): While full path logging was not implemented, sensor readings were monitored to illustrate how the robot could log line offset and obstacle distances over time. In future iterations, these logs could support visual analytics like path heatmaps or efficiency reports.

These enhancements not only improved the demonstration experience but also showcased the robot's decision-making process more clearly to users and audiences.

VIII. 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.

IX. References link

- Khan, M. (2021). *Proportional Line Following Algorithm Using Sensor Offset*. International Journal for Research in Applied Science and Engineering Technology (IJRASET).
Retrieved from: https://www.ijraset.com/research-paper/proportional-line-following-algorithm

- Lee, C., Kim, Y., & Park, J. (2022). *Color-Based Object Detection and Tracking for Autonomous Driving*. Sensors, 22(6), 2245.
https://doi.org/10.3390/s22062245
Direct link: https://www.mdpi.com/1424-8220/22/6/2245

- Beer, R. D., & Gallagher, J. C. (2019). *Braitenberg Vehicles as a Platform for Neuro-Robotics Research*. Frontiers in Robotics and AI, 6, Article 61.
https://doi.org/10.3389/frobt.2019.00061
Direct link: https://www.frontiersin.org/articles/10.3389/frobt.2019.00061/full

- Makeblock. (n.d.). *CyberPi API Documentation*. Makeblock Help Center.
Retrieved from: https://www.yuque.com/makeblock-help-center-en/mcode/cyberpi-api

- Makeblock. (n.d.). *CyberPi mBuild Modules API*. Makeblock Help Center.
Retrieved from: https://www.yuque.com/makeblock-help-center-en/mcode/cyberpi-api-mbuild

X. Github Code Repo Link

https://github.com/Cecil-B-Liv/Robot-mBlock-Code.git

https://github.com/Cecil-B-Liv/Robot-mBlock-Code