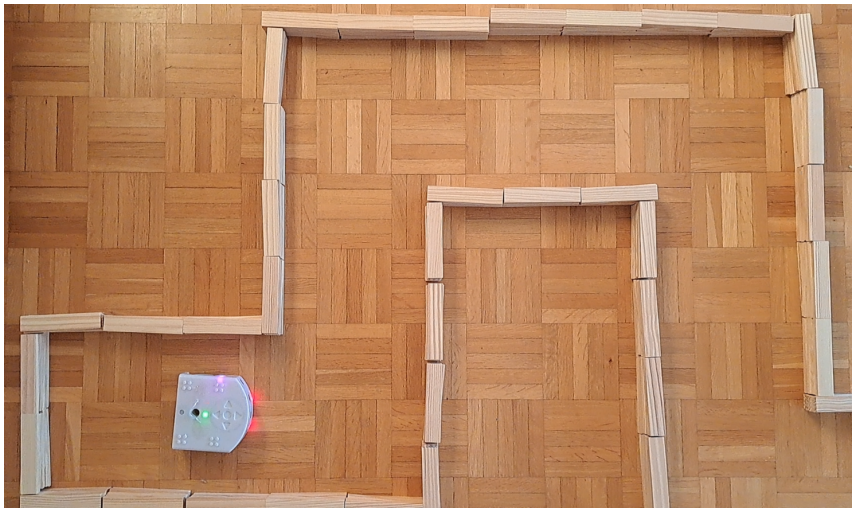
Reinforcement Learning for Thymio

Wall Following and Opening Detection

Amin Karbassi

January 18, 2026



1 Problem Statement

Autonomous navigation is a fundamental capability for mobile robots operating in indoor environments. In many practical scenarios, such as corridors, rooms, or structured hallways, robots must move safely along walls while identifying and exploiting openings such as doorways or intersections. Achieving this behavior becomes particularly challenging when the robot has limited sensing capabilities and no access to global localization or a predefined map.

Traditional rule-based approaches for wall-following and obstacle avoidance typically rely on manually tuned thresholds and heuristic decision rules. While effective in controlled settings, such controllers often suffer from poor generalization, sensitivity to sensor noise, and brittle behavior in corner cases. Small changes in environment geometry may lead to oscillations, collisions, or failure to detect openings.

The objective of this project is to investigate whether **reinforcement learning (RL)** can be used to enable a **Thymio robot** to autonomously learn wall-following behavior and reliably detect and enter openings on the left or right. The learning process relies exclusively on the robot's proximity sensors, without predefined navigation rules or global information.

A secondary objective is to train the robot in simulation and transfer the learned policy to a real physical Thymio robot with minimal modification, demonstrating the feasibility of sim-to-real deployment.

2 Thymio educational robot

Thymio is an open-source educational robot designed by researchers from EPFL, in collaboration with ECAL, and produced by Mobsya, a nonprofit association whose mission is to offer comprehensive, engaging STEAM journeys to learners of all ages.

3 Starting Point

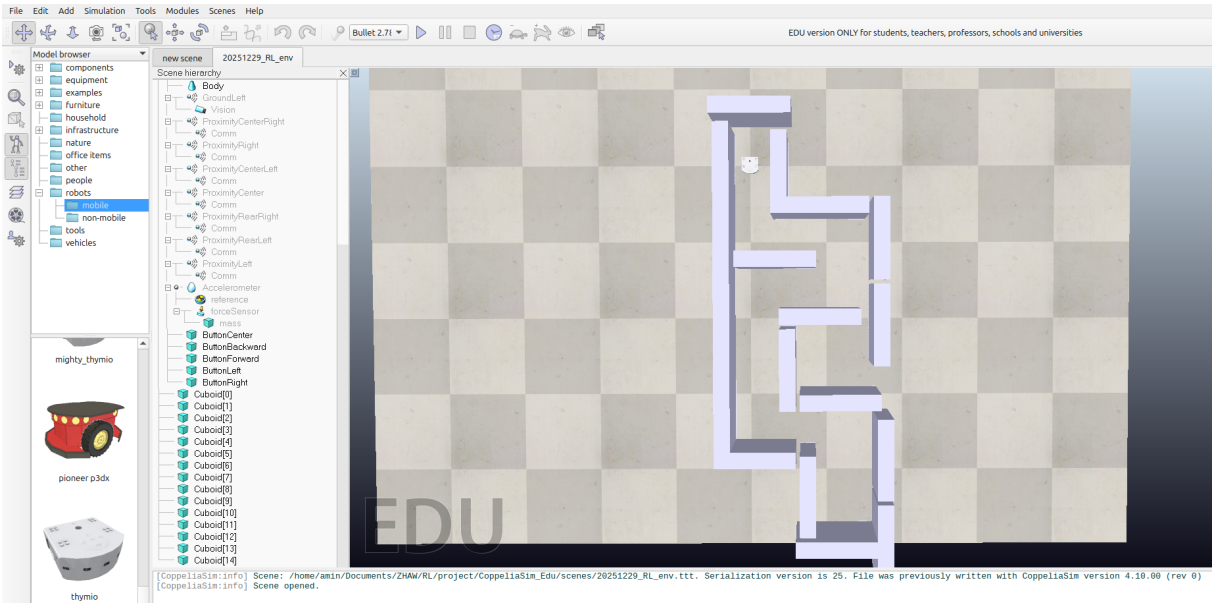
The project was conducted using the CoppeliaSim simulator, which provides a physics-based environment and a detailed Thymio robot model. Simulation was chosen as the initial development platform to allow rapid iteration, debugging, and visualization of learned behavior. A physical Thymio robot was available and used later for deployment experiments.

At the beginning of the project, the robot had:

- no prior knowledge of the environment,
- no map or localization capability,
- access only to five horizontal proximity sensors (left, center-left, center, center-right, right),
- control over two differential-drive wheel motors.

Initial experiments focused on validating basic communication between Python and CoppeliaSim via the ZMQ Remote API. Sensor readings and motor commands were tested independently to ensure correct interpretation and stable actuation.

Simple reactive obstacle-avoidance strategies were then implemented, such as turning away from obstacles when a front sensor exceeded a threshold. Although these rules prevented immediate collisions, they led to undesirable behaviors including oscillations near corners, excessive spinning, and failure to recognize side openings. These limitations motivated the use of reinforcement learning.



4 Reinforcement Learning Setup

4.1 Environment and Task Definition

The robot operates in a two-dimensional environment composed of walls forming corridors and paths with openings on the left and right. The task is not defined as reaching a specific goal location. Instead, the robot must continuously:

- move safely along walls,
- avoid frontal and lateral collisions,
- detect openings in the walls,
- and turn into those openings when encountered.

The task is formulated using only local sensor information, ensuring that the learned policy can be deployed on a real robot without reliance on simulation-specific state variables.

4.2 State Representation

The state representation is designed to balance expressiveness with learnability. Rather than using continuous positions or orientations, the state is derived entirely from proximity sensor readings. For tabular Q-learning, sensor values are discretized into symbolic categories representing distance to obstacles (e.g., far, medium, near). The final state consists of five components:

- left wall proximity,
- front wall proximity (aggregated from three sensors),
- right wall proximity,
- detection of a left-side opening,
- detection of a right-side opening.

Opening detection is achieved through short-term temporal memory. By analyzing recent side sensor measurements, the robot can determine whether it has transitioned from following a wall to encountering free space. This temporal abstraction allows the robot to distinguish genuine openings from transient sensor noise.

For the Deep Q-Network (DQN) approach, continuous or normalized sensor values are used directly as network inputs, allowing the neural network to learn internal feature representations automatically.

4.3 Action Space

The action space is intentionally minimal and compatible with the physical Thymio robot:

- move forward,
- turn left in place,
- turn right in place.

This restricted action set simplifies learning, reduces ambiguity in policy interpretation, and ensures safe execution on real hardware.

4.4 Reward Design

Reward shaping is central to achieving stable and purposeful navigation behavior. The reward function is designed to reflect multiple competing objectives: safety, forward progress, wall-following stability, and timely exploitation of openings. A base step penalty of -5.0 is applied at every timestep to discourage idle behavior and promote efficient motion. Collision avoidance is given highest priority. When the robot approaches a frontal wall, as indicated by the center-facing sensors, forward motion is penalized increasingly based on proximity. Moving forward when very close to a frontal obstacle incurs a strong penalty of -20.0, while turning away from the obstacle is mildly rewarded ($+1.0$) to encourage evasive actions. Approaching but not yet imminent frontal obstacles results in a smaller penalty of -10.0 for forward motion. Lateral safety is enforced by penalizing proximity to side walls. If either the left or right sensor indicates a very close distance to a wall, a penalty of -15.0 is applied to discourage wall scraping. To promote progress, forward motion is rewarded when the robot is sufficiently far from frontal obstacles. If the minimum front distance exceeds 0.35 m, moving forward yields a positive reward of $+15.0$. An additional centering bonus of $+3.0$ is granted when the robot maintains approximately equal distances from the left and right walls while moving forward in an open corridor. Opening detection is explicitly encouraged. When a left-side opening is detected through short-term memory and the robot turns left, a reward of $+30.0$ is applied; an equivalent reward is given for turning right into a detected right-side opening. Conversely, ignoring a detected opening while continuing forward in a constrained space incurs a penalty of -10.0. To avoid excessive spinning, a small turning cost of -2.0 is applied whenever the robot turns instead of moving forward. In dead-end situations, where the front is blocked and no openings are detected, turning actions are positively reinforced ($+5.0$) to promote escape behavior. This multi-term reward function was refined iteratively through empirical observation and proved essential for balancing safety, efficiency, and behavioral stability.

4.5 Learning Algorithms

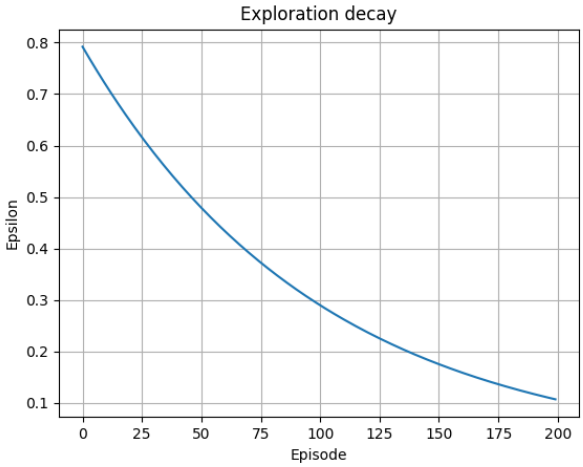
Two reinforcement learning methods were implemented and compared.

Tabular Q-Learning Tabular Q-learning was used as an initial baseline. The discretized state space enabled fast prototyping and clear interpretability of learned values. However, this approach is limited in scalability and sensitive to discretization choices.

Deep Q-Network (DQN) The Deep Q-Network (DQN) approach extends the tabular formulation by approximating the Q-function with a neural network operating on richer state representations. In addition to continuous or discretized proximity sensor inputs and opening detection flags, the DQN state includes the *previous action executed by the robot*. This action-history augmentation addresses a common failure mode observed during early experiments: rapid oscillation between left and right turning actions in symmetric or ambiguous situations. By explicitly encoding the last action in the state, the network can learn temporal dependencies between consecutive decisions. The reward function further penalizes oscillatory behavior by applying a penalty of -8.0 whenever the robot switches directly from a left turn to a right turn or vice versa. This discourages indecisive back-and-forth motion. Conversely, when no immediate frontal threat is present and the robot continues to move forward consistently, an additional reward of $+5.0$ is applied. This promotes stable corridor traversal and reduces unnecessary turning. Together, the inclusion of action history in the state and oscillation-aware reward shaping significantly improves behavioral smoothness and learning stability compared to tabular Q-learning.

4.6 Training Protocol

Training is organized into episodes. Each episode starts with the robot placed near a wall or within a corridor, terminates upon collision or after a fixed step limit, uses a decaying exploration rate.



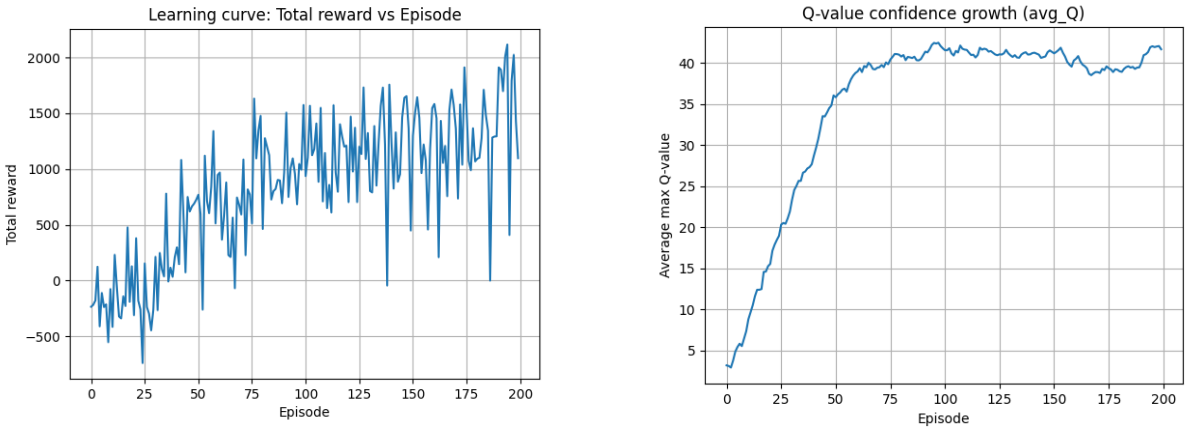
During training, multiple metrics are logged, including total reward, episode length, collision rate, and average Q-value, enabling detailed performance analysis.

5 Results and Analysis

5.1 Training Performance

Learning progress is evaluated using quantitative metrics logged during training.

- Total reward per episode
- Number of steps
- Collision rate
- Average Q-value growth



(a) Total reward per episode

(b) Average Q-value per episode

Figure 1: Training performance metrics over learning episodes.

Training curves show that early episodes are dominated by random motion and frequent collisions. As training progresses, the robot learns to maintain safer distances from walls and reduce

unnecessary turning. In later episodes, the robot consistently detects and enters openings.

5.2 Behavioral Observations

Qualitative evaluation through simulation videos reveals emergent behaviors such as stable wall-following, anticipatory turning near corners, and consistent entry into side openings. These behaviors emerge without explicit programming, driven solely by interaction and reward feedback.

5.3 Deployment

Learned policies are deployed in a policy-only mode without exploration or learning updates. This separation enables direct evaluation of trained behavior and facilitates transfer to the real robot.

Deployment experiments confirm reduced oscillations, improved stability, and reliable opening detection compared to early training stages.

6 Conclusion and Explanation of Own Work

This project demonstrates that reinforcement learning can successfully enable a small mobile robot with limited sensing to perform structured navigation tasks such as wall-following and opening detection.

Key contributions include:

- designing a compact state representation combining instantaneous perception with short-term memory,
- implementing and comparing tabular Q-learning and DQN methods,
- refining reward functions through systematic experimentation,
- developing a complete pipeline from simulation training to real-robot deployment.

A significant portion of the work involved diagnosing subtle issues related to sensor interpretation, simulator configuration, and learning dynamics. Many improvements were achieved empirically through observation and iterative refinement.

Supplementary Material

Simulation videos demonstrating the learned behavior on Thymioare available online:

- `Before-training video`
- `After-training video`