# Q Learning Metro Bot using Arduino

Harsh, Harsh Choudhary, Soham Jadhav

November 8, 2024

**Abstract**

This project explores the application of reinforcement learning, specifically Q-learning, in developing an autonomous Arduino-based bot capable of simulating a train that stops at designated stations. The bot utilizes an ultrasonic sensor to detect these stations and employs infrared (IR) sensors to follow a pre-defined path. The Q-learning algorithm is used to train the bot, allowing it to autonomously pause at each station for a pre-defined interval and resume movement. This project investigates how the integration of reinforcement learning techniques with low-cost hardware, such as Arduino and basic sensors, can contribute to educational platforms, offering students hands-on experience in autonomous navigation, decision-making processes, and machine learning principles. Through iterative training and evaluation, this project demonstrates the feasibility and effectiveness of combining Q-learning with sensor feedback to enhance robotic navigation tasks. The findings suggest that Q-learning can significantly improve the bot's accuracy in following paths and stopping at stations, providing an accessible approach to introduce students to concepts in artificial intelligence and robotics within an educational setting.

## 1 Introduction

The field of robotics and artificial intelligence (AI) is rapidly advancing, with applications ranging from industrial automation to autonomous vehicles. In educational settings, these technologies offer a compelling way to introduce students to complex concepts such as machine learning and reinforcement learning. One fundamental reinforcement learning algorithm, Q-learning, provides a framework for training agents to make sequential decisions by rewarding them for favorable actions and penalizing unfavorable ones. This paper investigates the development and training of a Q-learning bot designed to perform autonomous navigation tasks on an Arduino platform.

Our project centers around a bot that simulates a train, using ultrasonic and infrared (IR) sensors to navigate a track and autonomously stop at designated "stations." The ultrasonic sensor allows the bot to detect stations, while IR sensors enable it to follow a specified path. By implementing Q-learning, the bot learns to optimize its behavior for stopping at each station accurately and avoiding potential obstacles on the track.

This study aims to provide insights into how reinforcement learning can be practically applied to low-cost, accessible hardware like Arduino, making it an effective educational tool. By combining hardware and software, we create an interactive learning experience that fosters students' understanding of AI, robotics, and programming. The results demonstrate how reinforcement learning principles, when integrated with basic sensor technology, can be used to perform complex tasks with high accuracy and reliability.

## 2 Circuit

The circuit for the Q-learning Arduino bot involves connecting various sensors and actuators to the Arduino Uno. The primary components include an ultrasonic sensor for station detection, infrared (IR) sensors for line-following, and motors with motor driver module for movement. The ultrasonic sensors are mounted on the sides of the bot to detect stations, while two IR sensors are positioned on the bottom of the bot to help it follow a pre-defined path. The Arduino Uno serves as the central control unit, processing sensor data and making decisions based on the Q-learning algorithm to stop at stations and resume movement.

Following is the circuit diagram for the Q-learning Arduino bot, which shows the connections between the Arduino and the various components:

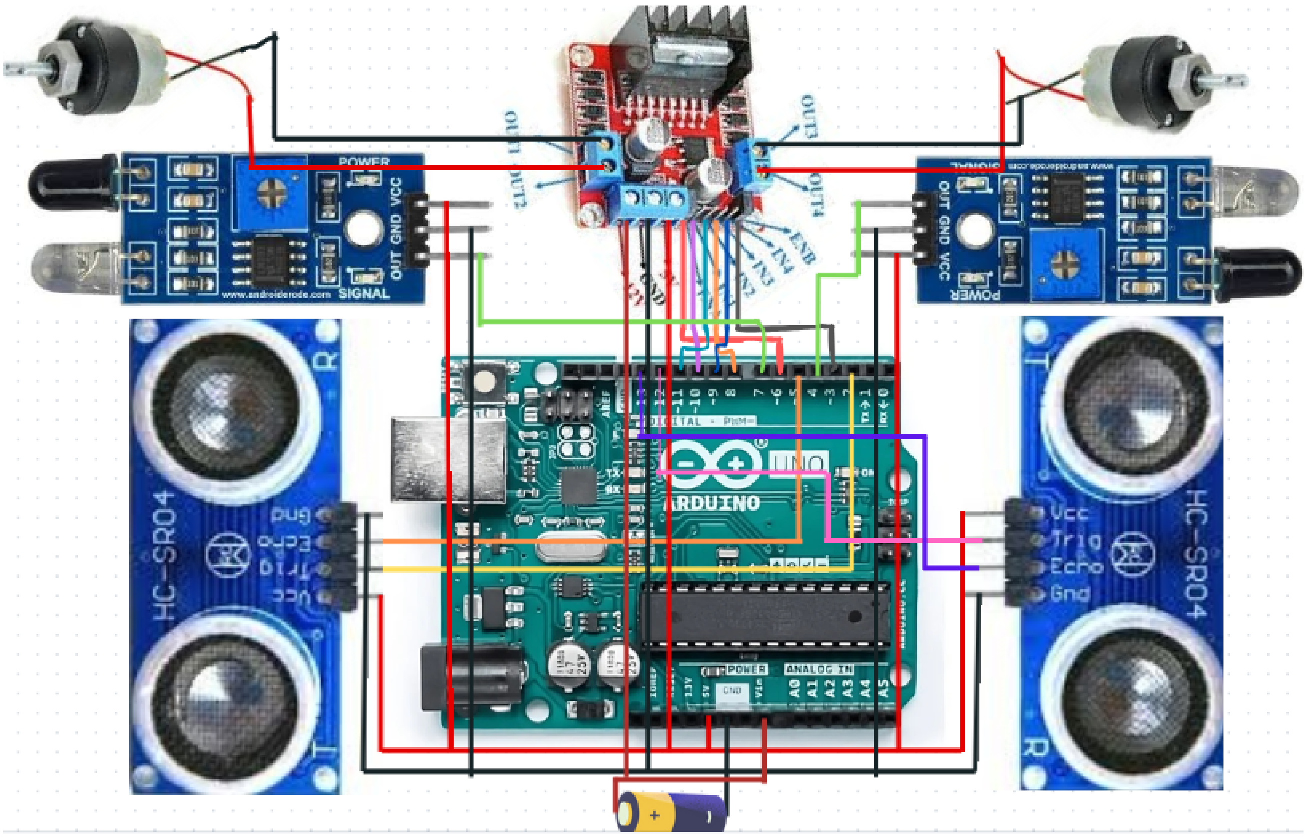

Figure 1: Circuit diagram for Arduino Metro bot

The table below summarizes the pin connections between the Arduino and the sensors/motors to ensure accurate wiring for proper functionality:

Table 1: Pin connections between Arduino and components

| Component | Arduino Pin | Description |
|---|---|---|
| Ultrasonic Sensor (Left) - VCC | 5V | Power supply for left ultrasonic sensor |
| Ultrasonic Sensor (Left) - GND | GND | Ground connection |
| Ultrasonic Sensor (Left) - Trig | D2 | Trigger pin for sending pulse |
| Ultrasonic Sensor (Left) - Echo | D5 | Echo pin for receiving pulse |
| Ultrasonic Sensor (Right) - VCC | 5V | Power supply for right ultrasonic sensor |
| Ultrasonic Sensor (Right) - GND | GND | Ground connection |
| Ultrasonic Sensor (Right) - Trig | D12 | Trigger pin for sending pulse |

| Component | Arduino Pin | Description |
|---|---|---|
| Ultrasonic Sensor (Right) - Echo | D13 | Echo pin for receiving pulse |
| IR Sensor (Left) - VCC | 5V | Power supply for left IR sensor |
| IR Sensor (Left) - GND | GND | Ground for left IR sensor |
| IR Sensor (Left) - Out | D7 | Signal output from left IR sensor |
| IR Sensor (Right) - VCC | 5V | Power supply for right IR sensor |
| IR Sensor (Right) - GND | GND | Ground for right IR sensor |
| IR Sensor (Right) - Out | D4 | Signal output from right IR sensor |
| Motor Driver - IN1 | D10 | Control pin 1 for motor direction |
| Motor Driver - IN2 | D11 | Control pin 2 for motor direction |
| Motor Driver - ENA | D6 | Enable pin for motor speed control |
| Motor Driver - IN3 | D8 | Control pin 3 for motor direction |
| Motor Driver - IN4 | D9 | Control pin 4 for motor direction |
| Motor Driver - ENB | D3 | Enable pin for motor speed control |

## 3 Underlying MDP

The Q-learning algorithm used in this project is based on the concept of a Markov Decision Process (MDP), which is a mathematical framework for modeling decision-making problems where outcomes are partly random and partly under the control of a decision maker (the agent). The agent interacts with an environment and makes decisions at each state based on available actions, aiming to maximize a cumulative reward over time.

In this project, the agent simulates the environment using it's ultrasonic sensors. The state space for the MDP model consists of tuple *station arrived*,*moved after stop*,*stopped for*. The purpose and calculation of these state parameters is as follows:-

1. *station arrived*: This takes two values- 0 and 1. The value of this is calculated using the distance reading from ultrasonic sensor. The distances are divided into bins, and for our project the no. of bins is two only. So if the reading is close, the station is arrived, otherwise not. The environment provides the values for this using a fixed threshold but the agent is unaware of the threshold itself and tries to learn the best actions based on the values.

2. *moved after stop*: This parameter again takes values 0 and 1 indicating whether or not the bot has moved after stopping at the station. The purpose is to penalize the agent if it stops again after moving, thus making re-stopping at same station a bad policy.

3. *stopped for*: This is the no. of seconds the bot has stopped at the station without moving. We wanted the agent to wait for 5s at each station, so the environment gives reward according to that, however, the bot itself is unaware of the desired time to stop and has to learn it based on the structure of the reward.

The actions are either to *stop*, or *move*. If the action is move, the line following behavior is taken care of by the infrared sensors using Finite State Machine logic and is not the part of Q learning.

The Markov decision process is shown in diagram below, for a 5 seconds stopping at each station. Since we are in the Q Learning framework, the agent is unaware of the true probabilities for each of the transition, and implies temporal difference method to learn online.

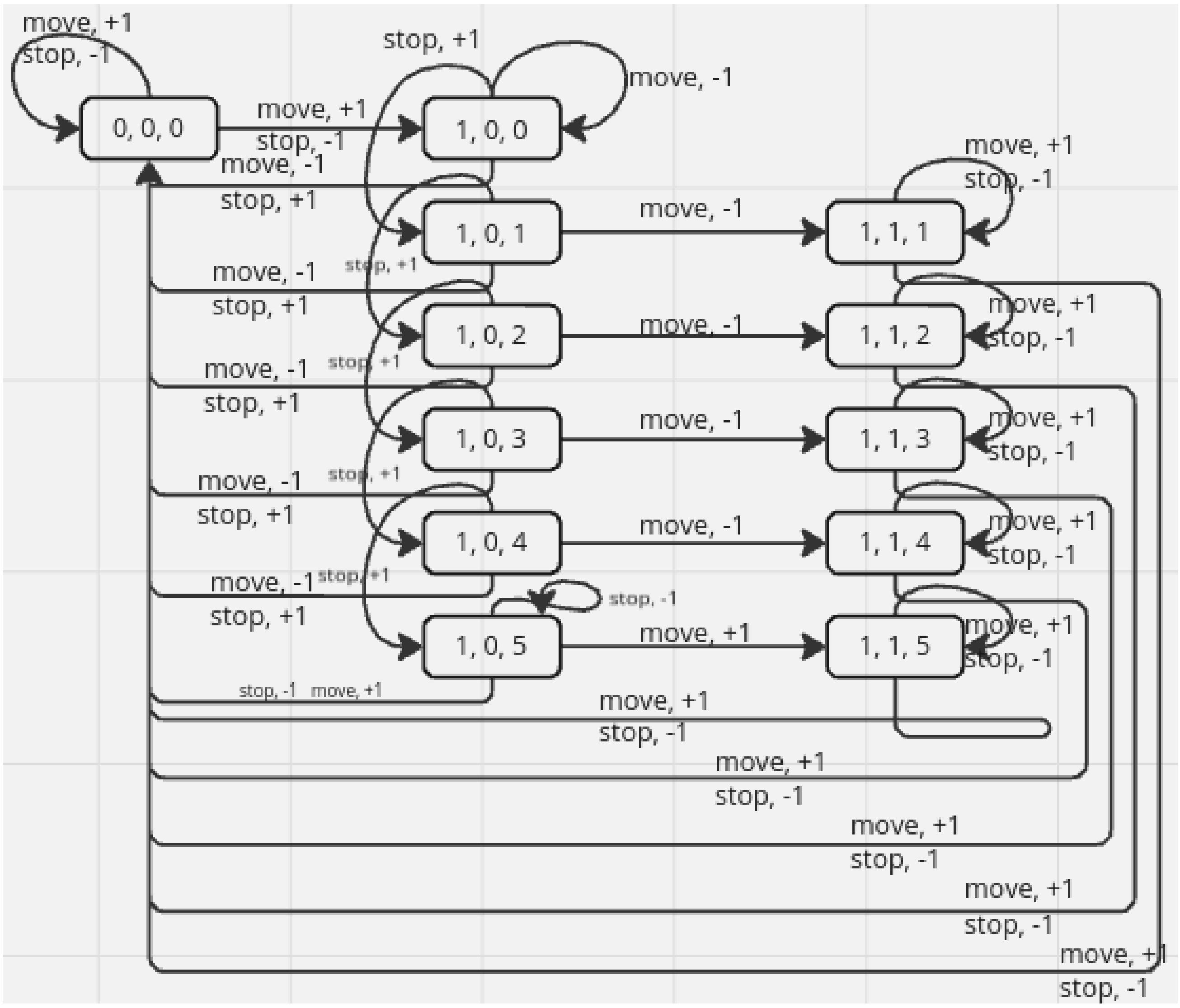

Figure 2: MDP for the metro bot problem

## 4 Arduino and C code

The complete Arduino and C code for this Q-learning bot project is available in a public GitHub repository. This repository contains all necessary code files for replicating and modifying the project. You can access it directly from the following link:

GitHub Repository - Q-learning Arduino Bot Code

## 5 Results

The bot is able to stop at the designated stations for 5 seconds successfully, only after one or two failures. The behavior to not re-stop at the same station was tricky to learn. The line following behavior was not smooth, and the bot would often skid off the track, especially at high speeds. However, at lower speed, the bot was able to traverse the path well. The recorded video demo is put here.

## 6 Conclusion

This project demonstrates the successful integration of Q-learning reinforcement learning principles with low-cost hardware to create an Arduino-based autonomous bot capable of line-following and station-stopping behaviors. By leveraging ultrasonic sensors for station detection and IR sensors for path following, the bot effectively simulates a train that autonomously stops at designated points, showcasing the practical application of machine learning in embedded systems.

Through iterative training and parameter adjustments, the bot achieved accurate performance in detecting and pausing at stations, highlighting the adaptability of Q-learning in dynamic environments. The project serves as an educational tool, illustrating the synergy between robotics and machine learning concepts. It offers an accessible platform for students and enthusiasts to explore AI-driven decision-making processes in real-world scenarios, bridging the gap between theoretical learning and hands-on experience.

In conclusion, the project demonstrates that Q-learning can be effectively applied to simple robotic systems, enabling autonomous decision-making with basic sensor feedback. This work not only advances the understanding of reinforcement learning in resource-constrained environments but also emphasizes the potential of Arduino-based systems in educational settings for teaching fundamental AI and robotics principles.

## 7 Further Contributions

Future developments for this project should focus on enhancing the bot's capabilities to accurately simulate a train station parking system. The primary objectives include improving the bot's ability to stop precisely at stations, implementing smooth acceleration and deceleration, which is very difficult to do with mere encoding techniques as the environment keeps changing (fog, light etc.).

To achieve precise station stopping, the bot should be trained to adjust its speed and braking in response to proximity data from the ultrasonic sensors. This refinement will allow the bot to make smoother, more gradual stops, mimicking the behavior of real-world trains approaching a platform. By implementing smooth acceleration and deceleration techniques, the bot can improve passenger comfort and system efficiency in a simulated environment, adding realism to the model.

This project's next stages should contribute to building an advanced, affordable model of a train parking system, making it suitable for educational and prototyping purposes in robotics, transportation systems, and artificial intelligence.