





**Department of Electrical  
& Computer Engineering**  
Faculty of Engineering & Architectural Science

<b>Course Title:</b>	Microprocessor Systems
<b>Course Section:</b>	COE538
<b>Semester/Year (e.g. F2017)</b>	F2024

<b>Instructor</b>	Dr. Lev Kirischian
-------------------	--------------------

<b>Assignment/Lab Title:</b>	Final Project Report
------------------------------	----------------------

<b>Submission Date:</b>	12:00, Nov.26th, 2024
<b>Due Date:</b>	12:00, Nov.26th, 2024

<b>Student LAST Name</b>	<b>Student FIRST Name</b>	<b>Student Number</b>	<b>Section</b>	<b>Signature*</b>
Muruganantham	Haran	501166674	04	
Shanmuganathan	Theeshiikan	501157664	04	

\*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a “0” on the work, an “F” in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at:

<http://www.ryerson.ca/senate/current/pol60.pdf>

**Table of Contents**

<b>Table of Contents</b>	<b>1</b>
<b>Project Overview</b>	<b>2</b>
<b>Project Challenges</b>	<b>3</b>
<b>Project Insights</b>	<b>3</b>

## **Project Overview**

This project aimed to develop a large software application that would guide an eebot along a predetermined maze using the assembly language. This maze consisted of only one start and end, various 3-way intersections, multiple paths with dead-ends, and an S curve along the path. To achieve this, we used an approach that would handle each aspect the bot needed to guide itself through the maze successfully. We broke this down into a component that would handle the guidance system and a component that would handle the navigation management. Then, in the end, we combined these two components to generate a functioning maze guidance program.

We started building the guidance system by creating a method that would allow us to understand better what the robot was doing at any given time. To do this, we displayed various indicators on the LCD, such as voltage, bumper, state, and sensor data. This real-time data display was crucial as it would allow us to ensure the program was running successfully, and if any issues were to come up, the data would allow us to debug them quickly. Next, we initialized the sensors and set them to track the path and notice any changes in the path so the required action could be taken. These sensors were set up to notice when the bot was on the black tape surface of the maze or the white paper surface of the surroundings. Finally, the last component of the guidance system was initializing the motors so that the bot could move. The code was generated so the bot could move front and back, perform turns in both directions, make successful u-turns, and perform various other actions to navigate the maze.

Once the guidance system was complete, the next task was the navigation management system. We designed the navigation management system so that the robot had some logic and processes for moving through the maze and tackling the scenarios it would come across. Our code was designed such that in an intersection, the bot would prefer to take a right turn instead of going straight or left, and in the case of a junction with only straight and left, it would prefer to go straight. If there is no obstacle in the path, the bot will continue until it reaches another junction or exits the maze. If there is an obstacle, the robot will hit the wall, which in turn will press its front bumper switch. The code is built such that when that switch is pressed, it will start backtracking to the junction and then take the alternate path.

## **Project Challenges**

Whilst programming the bot to navigate, we encountered several significant challenges. One of the primary issues was ensuring the sensors' accuracy and proper calibration, as even minor detection errors caused the bot to deviate from the intended path. The variability between individual bots further complicated matters, with slight differences in the threshold values and build quality of the bot led to inconsistent behavior during testing. Although we structured the code with comments for organization, the numerous subroutines added complexity which made debugging and optimization a time consuming process. Additionally, the limited lab time and shared testing spaces presented challenges. Finding enough time to run tests and implement adjustments was often difficult as the maze was almost always full of other bots. The crowded environment also introduced distractions, making it harder to focus solely on problem-solving. These challenges required us to adopt effective strategies for time management, troubleshooting, to ensure that the bot performed consistently and accurately.

## **Project Insights**

One significant insight we took away from this project was the value of breaking things down into smaller tasks and testing as we went. Focusing on one section at a time made it easier to troubleshoot and figure out where the issues were. The LCD displays were also really helpful, especially for giving quick updates about what the micro bot was doing in real-time. For future projects, we plan to stick with this structured approach since it helped us stay organized and improved the bot's overall reliability.