






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

<b>Course Title:</b>	<b>Microprocessor Systems</b>
<b>Course Number:</b>	<b>COE 538</b>
<b>Semester/Year (e.g.F2016)</b>	<b>Fall 2024</b>

<b>TA:</b>	<b>Mohsen Ensafjoo</b>
------------	------------------------

<i>Assignment/Lab Number:</i>	<b>COE 538 Final Project</b>
<i>Assignment/Lab Title:</i>	<b>COE 538 Final Project - Section 4</b>

<i>Submission Date</i>	<b>November 27, 2024</b>
<i>Due Date:</i>	<b>November 28, 2024</b>

<b>Student LAST Name</b>	<b>Student FIRST Name</b>	<b>Student Number</b>	<b>Section</b>	<b>Signature*</b>
Chin	Kyle	501118291	3	
Luu	Evan	501181521	4	
Su	Jason	501158090	3	

\*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>

**Description:**

The EEBOT guidance program coded in assembly language uses a multitude of subroutines in order to navigate through a maze. The maze consists of various turns and obstacles that the robot should be able to overcome autonomously. The first main block of code to explore is the code that controls the LCD. The LCD's purpose is to display the battery voltage and the EEBOT's state at all times while the maze is being navigated. To achieve this, the LCD is first cleared of any pre-existing content. After that, the battery voltage and current state are constantly displayed using a separate subroutine. The second key block of code is the state machine. The state machine is what controls the EEBOT's actions and processes based on the current state. For example, when the bot is in the forward state and the front bumper is triggered, the bot then switches to the reverse state and does a 180° rotation into the forward state to backtrack to the last T/L junction. Another function of the bot is to return to the ALL STOP state whenever the rear bumper is triggered. The third main block of code is the guider code. This code is in charge of keeping the EEBOT centered on the black line to guide itself through the maze. Furthermore, whenever a T/L junction is met, the guider code will sense this and take the left path each time. When our code was loaded onto the EEBOT, the EEBOT was able to successfully reach the end of the maze for every trial without fail.

**Problems:**

Whilst doing this project, as expected, our group encountered several problems that made completing the project slightly more challenging. The first problem that was encountered was that the bot would sometimes not be able to recognize the black line on the maze. The bot would sometimes completely ignore the line, thus leading to it not being able to correct itself. This error would also make the bot sometimes not recognize a junction and therefore miss the turn. To fix this problem, the speed of the bot needed to be slower to give the sensors adequate time to sense the line. However, this solution was short-lived due to the next error. During the next session, the bot would behave differently than how it did in the previous session. Eventually, we figured out that the issue of this was likely since we used a different bot than last time. Using the same bot, would decrease the amount of variables and ensure that the only thing that could be wrong was the code itself. However, this was when we ran into the realization that there could be issues other than the code and the bot. At one point, we tested the bot and it worked successfully. Then, the same code was uploaded into the bot where nothing was changed at all, and there were several syntax errors in the code and none of our previously working codes would work either. The software was restarted, and there was still the same error. Finally, the whole computer was restarted and it worked fine afterwards.

**What We Would Do Differently:**

If we were to do the project again, to make the process easier, we would start by eliminating any variables that could potentially pose an obstacle. For example, using the same bot each time we test our code. Using the same bot each time would ensure that the guider code

would work each time since the threshold voltages are specific to each bot based on their photo resistors. Secondly, it would be much more efficient if tasks were divided between group members. In the beginning, all three group members were working on the code which is very time-consuming since we can only have the most up-to-date code on a single workstation. Instead, it would be better if we had just two group members working on the code at a time or even each group member working on something different entirely. The last difference in our approach to this project would be to read the project report/requirements thoroughly before starting the project as we forgot to implement the rear bumper stop. Although it is not a significant issue, it is important for the bot to function to the outlined specifications.

Our experience included several challenges during the project. One significant issue was encountering an unknown syntax error despite making no changes to the code. Initially, our program was working perfectly, but when we attempted to test the bot again after five minutes, syntax errors appeared unexpectedly. This problem stumped us for nearly 30 minutes. Restarting the software did not resolve the issue; however, restarting the virtual machine ultimately fixed the problem. Additionally, we observed inconsistent results when using a different bot. If the selected bot differed from the one used in a previous lab, its behavior could vary due to differences in calibration. Another difficulty involved implementing the rear bumper stop. For a time, it appeared as though the code for this feature had not been implemented at all, despite our efforts. These challenges required significant troubleshooting and problem-solving to overcome.