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

Course Title:	Microprocessor Systems
Course Number:	COE538
Semester/Year (e.g.F2016)	F2024

Instructor:	Dr. Sattar Hussain

Assignment/Lab Title:	Final Project

Submission Date:	Wednesday, November 27
Due Date:	Wednesday, November 27

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Mohamed	Malak	501218537	13	Malak Borham
Tandel	Dip	501171728	13	Dip
Kamboj	Divjot Singh	501165887	13	

*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: https://www.torontomu.ca/content/dam/senate/policies/pol60.pdf

Code Description:

The code developed for the eebot to navigate through the maze is structured into multiple modules, each responsible for a specific aspect of the robot's behaviour, including sensor management, motor control, pathfinding, and decision-making. The software is written in assembly language for the HCS12 microcontroller, allowing direct interaction with the robot's hardware components.

1. Initialization and Setup

   The code handles the initialization of the robot's components. This includes setting up the sensors, which are responsible for detecting the black guidance line on the maze floor and determining the robot's position relative to the path. The sensors are calibrated to detect the contrast between the black tape (the path) and the surrounding white surface. The robot's LCD is also initialized at this stage to provide real-time feedback, such as sensor data, battery voltage, and the state of the robot's actions (e.g., moving forward, turning). The code sets up the microcontroller's I/O pins for the sensor and motor interfaces. This allows the robot to read sensor values from its surroundings and control the motors that drive the robot's movement.

2. Sensor Management

   The sensor management routine continuously monitors the robot's sensors to detect changes in the maze's layout. The readings are continuously compared to predefined threshold values, which were determined through testing, to ensure that the robot remains on the black line. When the robot approaches an intersection, the sensors detect the shift from the path and trigger the intersection detection routine. This part of the code processes the sensor data to determine if the robot needs to make a decision about which direction to take.

3. Motor Control

   The motor control module is responsible for driving the robot's wheels and controlling its movements. The code allows the robot to move forward, make turns, and reverse as needed. The motor control is implemented using a simple on/off control mechanism, where the motors are turned on or off depending on the required action The speed and direction of the motors are adjusted by varying the signals sent to the motor driver circuits. In cases where the robot encounters a dead end, the bumper switch is activated, triggering the front bumper detection routine. This routine commands the robot to stop, reverse, and perform a 180-degree turn to retrace its steps.

4. Pathfinding and Decision-Making

   The pathfinding logic of the eebot revolves around its ability to explore the maze and learn from its environment. At each intersection, the robot chooses a direction based on its sensor readings.

Algorithm Explanation:

1. Initialization and Sensor Setup:

   The eebot starts its journey at the beginning of the maze, following a black guidance line with its line-following sensors. These sensors distinguish between the black tape (which represents the path) and the surrounding white region, allowing the robot to follow the track precisely. If the robot strays from the line, it will correct its direction based on the differential sensor readings (E-F).

2. Path Following and Intersection Detection:

   The robot is programmed to move forward and look for intersections where the path splits into multiple directions. The robot stays within the black line using the differential sensor readings (E-F). If the reading drops below $84, the robot shifts right since the left sensor is reading more of a white surface, and vice versa if the reading is above $A7. At each intersection, the robot must make a decision about which way to take. The decision-making process follows a set of simple criteria that prioritize right turns over other routes. The robot continues to go straight unless the sensors determine that a right turn is accessible (sensor D); if so, it turns right and proceeds, making a left only if it is not possible to go straight or right, but possible to go left. The left turn is also small, making it merely useful for corrections.

3. Handling Dead Ends:

   When the robot reaches a dead end, its front bumper is triggered, indicating that it has hit an obstacle. The eebot does a 180-degree spin and returns to the previous intersection. Once back at the previous intersection, the eebot has no choice but to turn right, going the alternate way, assuming it was a left turn before (turning around makes it a right turn).

   In summary, the algorithm guiding the eebot through the maze is a combination of basic path-following behaviour, decision-making at intersections, and error recovery from dead ends.

Challenges Faced:

One of the key challenges we faced during this project was determining the threshold values for the sensors to reliably detect the black tape on the maze. Initial trials showed inconsistent readings, with the sensors occasionally failing to detect the line or falsely identifying the surroundings as part of the track. To address this, we conducted numerous experiments, adjusting the threshold values. Each trial involved observing the sensor outputs in real time and comparing them against expected results, allowing us to identify patterns and refine the threshold settings. By the end of these trials, we achieved reliable sensor performance, enabling the robot to consistently track the maze path and respond correctly to intersections and obstacles.

Additionally, precise motor control was another challenge we had to tackle. The robot's ability to execute smooth turns and accurate U-turns was critical for navigating the maze. However, during the initial phases of testing, the robot's movements were often imprecise, with turns that were either too wide or too sharp, causing it to deviate from the intended path. To solve this, we calibrated the motor control logic for turning and movement (delay time). By carefully adjusting the motor control parameters and testing the robot repeatedly in different sections of the maze, we were able to achieve the level of precision necessary for the eebot to navigate effectively and turn for the correct amount of time.

We also later realized that once the voltage level of the eebot changed, it also changed the speed of the motors, adding additional factors to consider. At the end of one of our trials, the eebot had successfully travelled the maze. However, when coming back to the same eebot with the same code a day later when the eebot was fully charged, it resulted in erratic behaviour that we determined was due to the motor speed being faster than usual. We therefore had to adjust the delays once again.

Finally, the testing time available to validate the robot's performance in the maze was limited due to constraints such as shared lab equipment, and waiting in long lines to test the robot on the one track available to all groups. Despite these challenges, we made the most of the time available by carefully planning and focusing on the most critical aspects of the robot's functionality.