Control of Mobile Robotics

CDA4621

Fall 2017

Lab 3

SLAM: Simultaneous Localization and Mapping Task

Total: 100 points

Due Date: 12-6-17 by 8am

The assignment is organized according to the following sections: (A) Lab Requirements, (B) Task Description, and (C) Task Evaluation. Note that only the questions referred to in section (C) need to be answered. Nonetheless, you should be able to answer all questions in section (B) since some of them will come up in a quiz and possibly during a theory exam.

A. Lab Requirements

The lab requires use of the "Robobulls-2017" robot hardware provided at no charge for the duration of the semester. Required software can be downloaded free of charge from the web. All labs are to be done by teams of two students. Note that no diagrams or descriptions by hand will be accepted. Each group is required to submit its report through Canvas. Penalties will be applied for submitting late assignments (see syllabus). All documentation needs to be in PDF.

A.1 Hardware Requirements

The "Robobulls-2017" (Figure 1) is the main robot hardware used for the course.

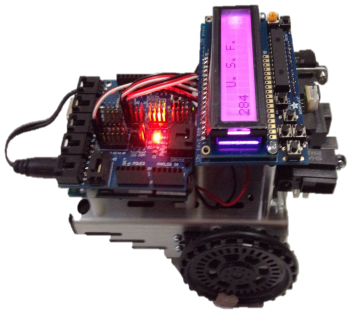

Figure 1: Robobulls-2017

A.2 Software Requirements

Arduino Software (Version 1.8.3 or later): https://www.arduino.cc/en/Main/Software

B. Task Description

The lab is divided into four parts: (1) "Robot Basics" where you will learn how to use the robot hardware, (2) line detection where you will implement an algorithm to detect colored lines in the floor, (3) maze navigation, where you will implement an algorithm to navigate within a maze, and (4) localization and path planning, where you will implement an algorithm to find a path from a start location to the goal.

B.1 Robot Basics

This section will cover basic usage of the hardware to gain an insight into what type of information can be found on a datasheet, how to properly use the robot hardware, and how to avoid typical errors made while working with the robot hardware.

B.1.1 Color Sensors

B.1.1.1 Color Sensor Basics

In this section, we will study some theory of how to use the color sensor in a correct manner. Go through the datasheet of the color sensor[1] and read the documentation of the Arduino's function "pulseIn"[2]. Think about the following:

1. To use the color sensor, you will have to use the Arduino's "pulseIn" function. What does the function do? What does it return?
2. In the color sensor, what are the pins S0 and S1 used for? What about S2 and S3?
3. How does the sensor encode the measured light intensity?
4. When you use the "pulseIn" function to read the color sensor, what does the return value represent? Hint, combine the answers to questions 1 and 3.
5. Go to the "Electrical Characteristics" section on the datasheet and look at the "Full-scale frequency" parameter. In what frequency range does the color sensor work at 100% frequency scaling? What about at 20% frequency scaling? What about at 2% frequency scaling?
6. What is the shortest amount of time that the "pulseIn" function will take before returning the result when using it to read the color sensor at 100% frequency scaling? What is the longest amount of time? (Hint, the shortest corresponds to the highest frequency and the longest to the shortest frequency).
7. When reading the color sensor, does the sensor return RGB values in the range 0-255?
8. Read the section "Measuring the frequency" on the color sensor data sheet. How fast can output data be collected?
9. The section "Measuring the frequency" specifies that at full scaling, one data point can be produced every 1 microsecond. Arduino's reference for the "micros" function specifies that it has a resolution of 4 microseconds. Taking this into account, would you use the sensor at 100% frequency scaling, 20% or 2%? Why?
10. Do you need to add delays between successive measures of the color sensor?

B.1.1.2 Calibrating the color sensor

NOTES: This section contains instructions on how to calibrate the color sensor. This section is optional, but it may be useful for using and understanding the color sensor.

As you should already know from the previous section, the value returned by the "pulseIn" function when reading the color sensor is not an RGB value. In fact, it is a time corresponding to half the period of the sensor's output frequency. So how do we get RGB values from the sensor? The answer is, we scale the values to the range 0-255. The problem is what value corresponds to (0,0,0) and what value corresponds to (255,255,255)?
To solve this issue, calibration is performed.
To find out what values correspond to black (0,0,0), put a black paper beneath the sensor. Use the sensor to measure the intensity of the light for red, green and blue. To reduce noise, take

[1] http://www.mouser.com/catalog/specsheets/TCS3200-E11.pdf
[2] https://www.arduino.cc/en/Reference/PulseIn

multiple measurements and then find the average or median. Store the results. Repeat the same process using a white paper.
If we let $B = (r_B, g_B, b_B)$ and $W = (r_w, g_w, b_w)$ be the stored values for black and white respectively, then when we read values (r, g, b) we scale them according to the formula $(r', g', b') = 255 * (\frac{r-r_k}{r_w-r_k}, \frac{g-g_k}{g_w-g_k}, \frac{b-b_k}{b_w-b_k})$. This way we get standard RGB values, although due to light intensity variations, the conversion might produce values below 0 or above 255.

B.2 Color line detection

Implement an algorithm that moves forward at full speed and detects lines in the floor. The algorithm should distinguish between red and blue lines. When the robot goes over a red line, the LCD screen should flash red. When the robot crosses a blue line, the LCD screen should flash blue.
It might be useful for section B.3 to implement the detection using time interrupts. A time interrupt example using Arduino was provided in the sample code. In the next section, the robot needs to detect the lines in real time, as it will have limited time to do so. If the code spends too much time on one part of the code, the lines might be skipped. This issue can be completely avoided if implementing line detection is done with time interrupts. That is, every fixed amount of time, the main program will be interrupted to execute the line detection code. This way, you can guarantee the timely execution of the line detection code.

B.3 SLAM

B.3.1 Task Description

The goal of this section is to develop a program that allows the robot to navigate and map a maze. The robot will have to navigate the full maze, and perform the task autonomously without any human intervention.
The robot may start from any cell and any orientation in the maze (see section B.3.2). This information will be provided to the robot at the start of the program by means of a selection menu as specifies in section B.3.3.

B.3.2 Maze

The maze will be composed of 4 by 4 grid cells, where each grid cell measures approximately 1.5 ft or 18 inches. The maze allows multiple configurations by switching the inner walls as shown in Figure 1. Red and blue tape will be placed on the carpet to mark the border between grid cells. The cells are numbered from left (west) to right (east), top (north) to bottom (south) as shown in Figure 2. This convention **MUST** be respected.

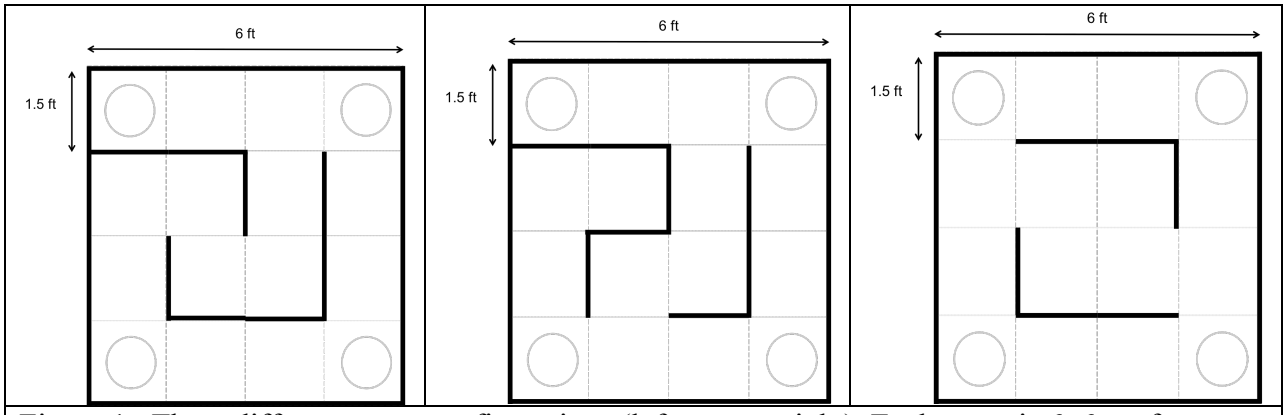

Figure 1. Three different maze configurations (left, center, right). Each maze is 6x6 sq. ft. consisting of a 4x4 marked grid cells of 1.5x1.5 sq. ft.

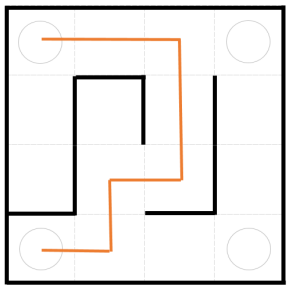

Figure 2. (a) Sample Navigation Robot Path

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Figure 2. (b) Sample Grid Cell Robot Path

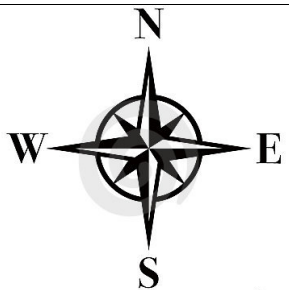

Figure 2. (c) Global Grid Cell Orientation

B.3.3 Subtasks

B.3.3.1 Options Menu

At the start of the program, the robot will have to implement an options menu to allow selecting the starting location and orientation. The different options should be navigated with the left and right LCD buttons, while parameters should be modified using the up and down buttons. Figure 3 shows sample screens (the ending location option will be used in section B.4 – path planning).

The option exit, should exit the options menu and start running the main program. You are free to add other options to the menu as you see fit, for example you could add options to calibrate the robot if necessary.

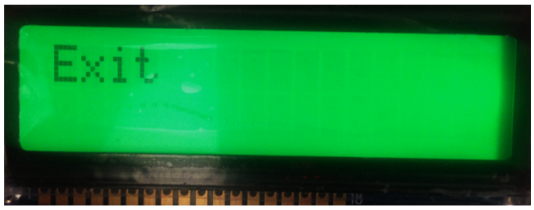

Figure 4. LCD Input buttons initial data.

B.3.3.2 Line Detection

While the main program is running, line detection should be implemented as in section B.2. When crossing a red line, the display should be flashed red, and when crossing a blue line, the display should be flashed blue.

B.3.3.3 Navigation and mapping

Once the algorithm starts running, the robot will have to keep track of its position and orientation through the whole program. At the same time, the robot will have to generate a map of the maze. By the end of the program, the robot will have to know whether a grid cell has walls or not on each of its sides.

B.3.3.4 Screen Format While Navigating

While the robot is running the main program, the LCD should display information of the program as seen on Figure 5. The first row indicates weather a grid cell has been visited or not. The map cell i should be displayed in column $i - 1$. For example, cell 1 is shown in column 0, while cell 16 is shown in column 15. Visited cells are marked with an X while non-visited cells are marked with an O.

The second row should use the first 5 columns to display the current grid being visited and the current orientation. Note that the second row ("1") is independent of information shown in row ("0") and the columns marked "0-4". The grid should be marked with a G followed by the grid number, and the orientation should be marked by an O, followed by the orientation (E for east, W for west, N for north, and S for south).

The remaining cells on the LCD display can be used as you see fit for debugging purposes.

The display must be refreshed whenever the robot enters a new cell or changes orientation.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	X	X	X	0	0	0	X	0	0	X	X	0	X	X	0	0
1	G	1		O	E											

Figure 5. Robot Path and Position Information as shown on the LCD

B.3.3.5 Algorithm completion

The SLAM algorithm finishes once the robot has visited all the grid cells and completed the mapping. When this happens, the robot should stop moving and it should display the message "DONE" on the LCD screen.

At this moment the robot will wait for the user to press "select", and then proceed to show the map information for cells 1 to 4 on row 0, and cells 5 to 8 on row 1.

Then, the robot will wait again for the "select" button to be pressed, and then proceed to show the remaining information. The map information for cells 9 to 12 should be displayed on row 0, and cells 13 to 16 on row 1.

Finally, the robot will wait one more time for the user to press "select" before ending the program.

When displaying the map information, for each cell a string of length 4 will be shown on the LCD. The first element will show whether there's a wall or not in the WEST, the second element will represent NORTH, the third EAST, and the fourth SOUTH. The presence of a wall will be marked with a capital 'W', while the absence will be marked with a lower case 'o'. For example, if a grid has walls to the east and west, and none to the north and south, the displayed string will look like this: "WoWo".

Figure 6 shows the series of screen displays that should be shown when completing the first maze (a) configuration as shown on Figure 1.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0						D	O	N	E							
1																

Done, wait for "select" button to be pressed

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	W	W	o	W	o	W	o	W	o	W	o	o	o	W	W	o
1	W	W	o	o	o	W	W	W	W	o	W	o	W	o	W	o

"Select" pressed, display info for first 8 cells and wait for select.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	W	o	W	o	W	W	o	o	o	o	W	W	W	o	W	o
1	W	o	o	W	o	o	o	W	o	W	o	W	o	o	W	W

"Select" pressed, display info for last 8 cells and wait for select.

Figure 5. Ending LCD display sequence

B.4 Path Planning

Add path planning to the program implemented in section B.3.
To do so, add new options to the options menu. Add one option for selecting the final goal, one option for executing the mapping and one option for executing path planning.
When the option mapping is selected, the robot will execute the program described in section B.3. When done, the program will store the map and go back to the options selection screen.
When the option path planning is selected, the robot will have to use the stored map, to move from the start location to the goal by following the shortest path. To find the shortest path, you have to implement any path planning algorithm of your choice.

The option path planning should be available from the start of the program. If the option path planning is selected before doing the mapping, the robot should use a preloaded map (a default map loaded at compile time).

C. Task Evaluation

The task will be evaluated at the final presentation scheduled during the final exam (Dec 6th) at the C4 lab. All students are required to assist to the presentation. This will be the only opportunity to present Lab 3.

During the presentation you will first be required to map an unknown maze configuration. The configuration will be chosen the day of the presentation. After performing the mapping, you will be required to do path planning. To perform path planning you may decide to use the map generated by the robot or to load the map manually into the robot.
Extra credit will be given to the top 5 teams that can perform path planning after mapping without having to manually load the map. The five teams will receive extra credit points according to their rank based on task completion time for the path planning part.

Grading will be done according to the following rubric:

- Line detection -10 points
  The robot must detect all lines flashing the screen accordingly as described.
  Possible outcomes: No lines detected 0 points, some lines detected 3 points, most lines detected 6 points, all lines detected 10 points.
- Options Menu – 10 points
  The options menu must display all the required options and work as described in the assignment. Possible outcomes: No implemented or not following specifications 0 points, partial implementation or not following some specifications 5 points, implementing all options following the specifications 10 points.
- Display while running – 10 points
  The display of the robot while running should match the specifications on section B.3.3.4. The possible outcomes are the same as in the options menu.
- Maze navigation – 10 points
  The robot should be able to navigate the whole maze autonomously without any help. The score will be given by the percentage of visited cells.
- Correct positioning – 20 points
  The robot should always display its correct position on the screen. The score will be given by the formula 20*X/16 where X is the number of grid cells until the robot displays the first incorrect value.
- Correct orientation – 20 points
  The robot should always display its correct orientation on the screen. The score will be given by the formula 20*X/16 where X is the number of grid cells until the robot displays the first incorrect value.
- Correct mapping – 20 points
  The robot should display the map correctly at the end of the algorithm as specified in the assignment. At the end of the algorithm, the two final screens will be compared to an expected output and the score will be given by percentage of cells that coincide with the expected output.
- Correct path planning – 10 points
  The robot can find and navigate the shortest path. Possible outcomes: Successful performance in the first attempt 10 points, successful performance in second attempt 6 points, successful performance in the third attempt 3 points, unsuccessful performance 0 points.