Mobile Robotics Final Project Report

Winter 2015-2016

Jordan Patterson
Chris Collinsworth
and
Cog the Arduino robot

February 21, 2016

ECE 425
Mobile Robotics
Professor C. A. Berry

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Abstract

The purpose of the final project was to successfully implement path planning, localization, and SLAM in an Arduino robot by utilizing a multitude of topics and concepts covered throughout the quarter. Topics such as path planning, wall following, and object detection were utilized to accomplish these tasks. Map making and localization were successfully implemented in the robot, but SLAM was not due to memory constraints with the Arduino's microcontroller. All of the tasks for map making and localization were successfully completed during the demos, with a multitude of different world's being used to do so.

Table of Contents

I. Objective

The purpose of this project was to learn about map making, localization, and Simultaneous Localization and Mapping (SLAM) with regards to an Arduino robot. The first part of the project involved placing a robot in a 4x4 grid and letting it move about until all cells of the 4x4 grid were mapped as either open or closed. Once the grid was completed, the robot would plan a path to another location; in this case, the robot had to plan and navigate a path back to its starting location.

The second part of the project entailed giving the robot a completed 4x4 grid to begin with. However, the robot initially did not know its location in the grid and had to localize itself. To do so the robot had to sense landmarks in its surroundings, compare the sensed landmarks to landmarks in the grid, and determine all possible locations in the grid that matched. The robot would then move to another cell in the grid, repeat the aforementioned process, and narrow down its viable locations until successfully localizing itself in the grid. The robot then had to plan a path and navigate to a location within the grid.

The third and final part of the project required implementing SLAM in the Arduino robot. Unfortunately due to memory constraints, it was not possible to successfully implement SLAM.

II. Theory

The theory behind map making in part one entailed the use of an occupancy grid in combination with algorithms such as Histogrammic in Motion Mapping (HIMM) to map the world. With HIMM, the robot reads its surroundings via sensors and increments values in a cell between 0 and 15. Each time a cell is registered as occupied, the value increases by 3. When a cell is considered open, it decreases by 1. If it is at 0, it caps at 0. The final result is a matrix of values between 0 and 15.

Map making also involved using a wavefront algorithm to plan out a path to another location. This process is accomplished by beginning at the goal location and decrementing the values of the surrounding cells. The process is then repeated for each of these surrounding cells, by decrementing the values of all cells surrounding these cells and continuing on in a wave-like manner. The robot then follows the path incrementally until it reaches the goal.

Localization for part two of the project involved using a form of the Partially Observable Markov Decision Process (PMDP) method. This uses probability to determine the location of the robot, with each cell containing a certain probability. As the robot moves, a cells probability of containing the robot changes as the options are narrowed down. Once the robot narrows its location down to a single cell, it localizes itself within the world and uses a similar wavefront algorithm to return to a predetermined location.

Part three of the project, which involves implementing an algorithm for SLAM, was not able to be accomplished due to memory constraints. However, the theory for implementing SLAM will still be discussed as though it was. When implementing SLAM, the robot initially scans the environment using sensors and performs some type of landmark extraction. The robot updates its map using the data from the sensors and localizes itself with respect to the sensed landmarks. The robot then attempts to navigate the environment. As it does so, the robot reads its sensors and performs landmark extraction again. An Extended Kalman Filter (EKF) can then be used to correct the robot's location based on the current and previous surrounding landmarks. The EKF accomplishes this by keeping track of the uncertainty in the landmarks as well as the uncertainty in the robot's position. When the robot comes across a new landmark, it gets added to the EKF so it can be re-observed in later iterations.

III. Methods

As previously stated, a method similar to HIMM was used for map making. Instead of using values between 0 and 15, only values of 0 (free space) or 99 (occupied space) were used. The robot was initially placed at its starting location (20) in order to identify itself. The robot then proceeded to read data from its sensors, update the grid accordingly, and navigate in a direction free of obstacles. This process was repeated until the grid was completely filled. The robot occupied cell was given a value of 2 which allowed the user to track the robot's movement on the grid. Once the grid was complete, there was not enough memory to utilize the wavefront algorithm from lab 7. Instead, a similar algorithm from the first part of map making was used. Because the current location of the robot and the goal location (starting location) were both known, if statements were used to move the robot depending on the landmarks identified through the sensors. The path on the grid was then replaced by 5's in order to show the final path to the user.

Localization used a similar idea to the PMDP method. Initially, the sensors were read and all of the possible locations of the robot on the grid were printed to the LCD. The robot then proceeded to move forward and perform more sensor readings. It then determined if that new location narrowed down the possible locations of itself in the grid. This process continued until only one possible location remained, at which point the robot would localize. After localizing, the robot switched states to plan a path to a goal via the wavefront algorithm. Also, picking up the robot and placing it in a surrounding cell results in the robot re-localizing and then continuing on its path to the goal. This additional feature allowed for the robot to re-localize if it strayed from the intended path.

SLAM, as previously stated, was not implemented due to memory constraints with the Arduino robots. Nevertheless, the method for implementing SLAM will be discussed. After landmark extraction, the SLAM process can be implemented in three steps:

1. Use the EKF to update the estimated current position of the robot via the Odometry data
2. Use the EKF to update the estimated current position of the robot from re-observed landmarks
3. Add new landmarks to the EKF that are associated with the current position of the robot

The first step just requires estimating the x- and y-coordinates of the robot with respect to Odometry data. The second step entails re-observing landmarks, updating the uncertainty associated with each landmark in the EKF, and using this information to compensate for Odometry errors by updating the x- and y-coordinates of the robot. In the third and final state, new landmarks are added to the world and their corresponding uncertainties are added to the EKF. The robot then continues to repeat this process.

IV. Results

MAP MAKING

In the map making portion, we were able to successful map out this world:

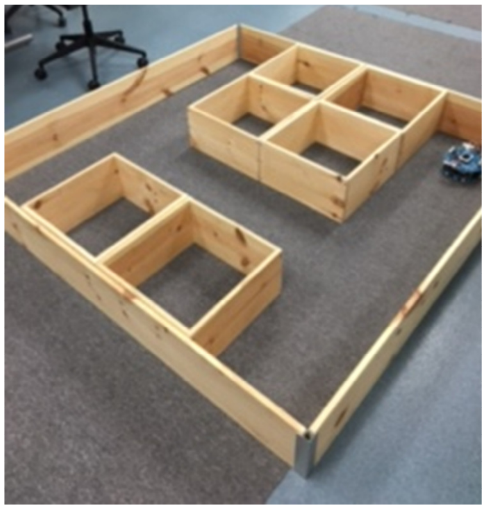
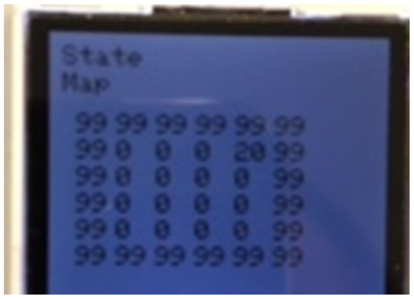


Figure 1. World setup for part 1

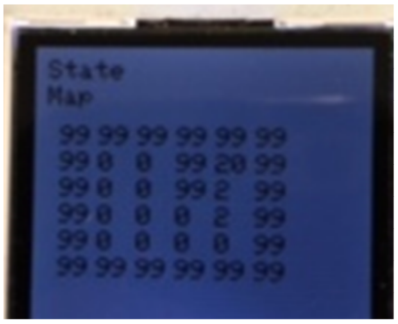
Before the robot moves or registers anything, the LCD only displays 0’s (free spaces), 99’s (obstacles), and a single 20 (starting location) as shown in Figure 2.



```
State
Map
 99 99 99 99 99 99
 99 0  0  0  20 99
 99 0  0  0  0  99
 99 0  0  0  0  99
 99 0  0  0  0  99
 99 99 99 99 99 99
```

Figure 2. LCD prior to map making

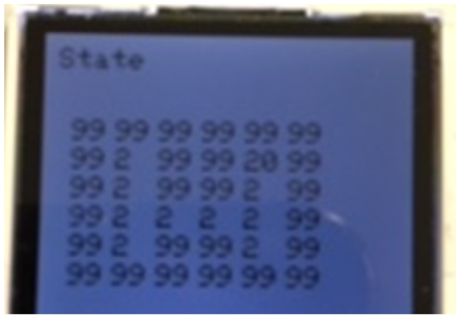
Figure 3 shows the LCD screen after two iterations of the map making process. It adds the two 99’s where the walls are and leaves the center isle open as a 0.



```
State
Map
 99 99 99 99 99 99
 99 0  0  99 20 99
 99 0  0  99 2  99
 99 0  0  0  2  99
 99 0  0  0  0  99
 99 99 99 99 99 99
```

Figure 3. LCD after two map making iterations

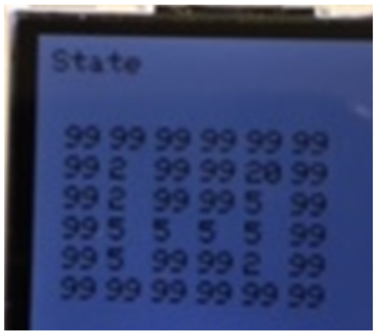
Figure 4 is the complete map. The robot ends up in the bottom left corner of the map after planning and will then turn around and return to the starting location.



```
State

 99 99 99 99 99 99
 99 2  99 99 20 99
 99 2  99 99 2  99
 99 2  2  2  2  99
 99 2  99 99 2  99
 99 99 99 99 99 99
```

Figure 4. Completed planned map

Finally, figure 5 shows the map after the robot returns to the starting location.



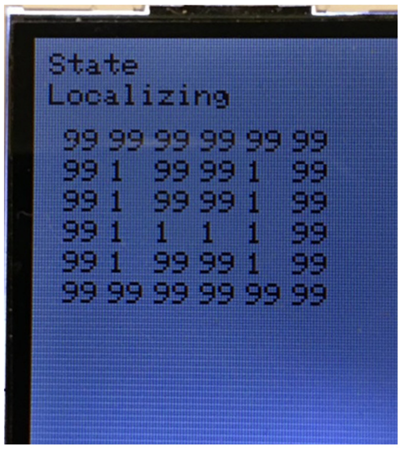
```
State

 99 99 99 99 99 99
 99 2  99 99 20 99
 99 2  99 99 5  99
 99 5  5  5  5  99
 99 5  99 99 2  99
 99 99 99 99 99 99
```

Figure 5. Final map

The robot was able to successfully map out the world and then return to the starting location.

Localization

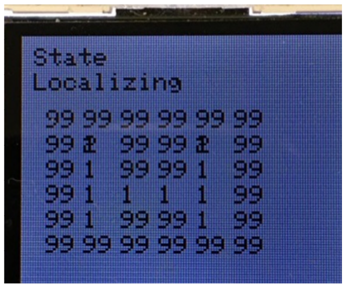
For our demonstration, the robot started in a world similar to Figure 1 but instead started by facing the wall. Figure 6 shows the map before the robot received its data.



```
State
Localizing
 99 99 99 99 99 99
 99 1  99 99 1  99
 99 1  99 99 1  99
 99 1  1  1  1  99
 99 1  99 99 1  99
 99 99 99 99 99 99
```

Figure 6. Map before sensing

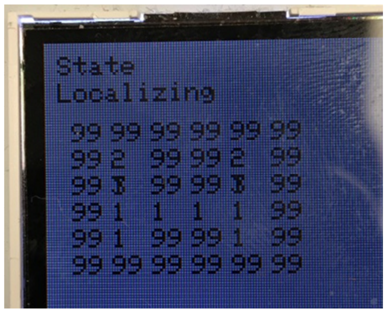
Figure 7 shows the map after the first iteration. The 2’s on the map show the possible locations for the robot’s location.



```
State
Localizing
 99 99 99 99 99 99
 99 2  99 99 2  99
 99 1  99 99 1  99
 99 1  1  1  1  99
 99 1  99 99 1  99
 99 99 99 99 99 99
```

Figure 7. LCD map after one iteration

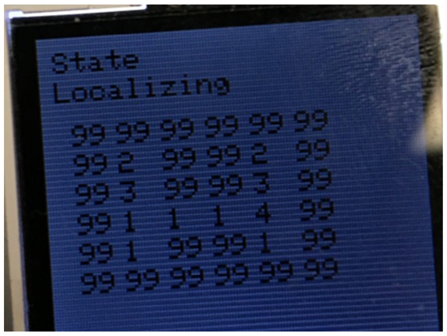
Figure 8 shows the map after the robot moves forward on the second iteration. The 3’s on the map show the next possible locations of the robot.



```
State
Localizing
 99 99 99 99 99 99
 99 2  99 99 2  99
 99 3  99 99 3  99
 99 1  1  1  1  99
 99 1  99 99 1  99
 99 99 99 99 99 99
```

Figure 8. Map after two iterations.

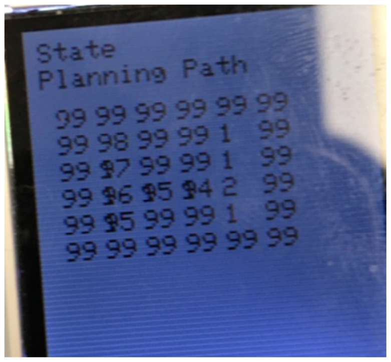
Figure 9 shows the map after the robot moves forward again on the third iteration. The single 4 shows where the robot is and since there is only one on the screen, it has successfully been localized!



```
State
Localizing
 99 99 99 99 99 99
 99 2  99 99 2  99
 99 3  99 99 3  99
 99 1  1  1  4  99
 99 1  99 99 1  99
 99 99 99 99 99 99
```

Figure 9. Final localization.

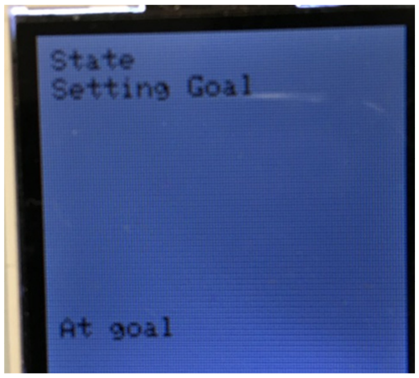
Figure 10 shows the map after localization, where the robot has used the wavefront algorithm to plan a path to the goal (98) on the map.



```
State
Planning Path
 99 99 99 99 99 99
 99 98 99 99 1  99
 99 97 99 99 1  99
 99 96 95 94 2  99
 99 95 99 99 1  99
 99 99 99 99 99 99
```

Figure 10. Wavefront algorithm

Once the robot has reached the goal, figure 11 shows the final printing statement on the LCD.



```
State
Setting Goal






At goal
```

Figure 11. Final state when robot is at goal

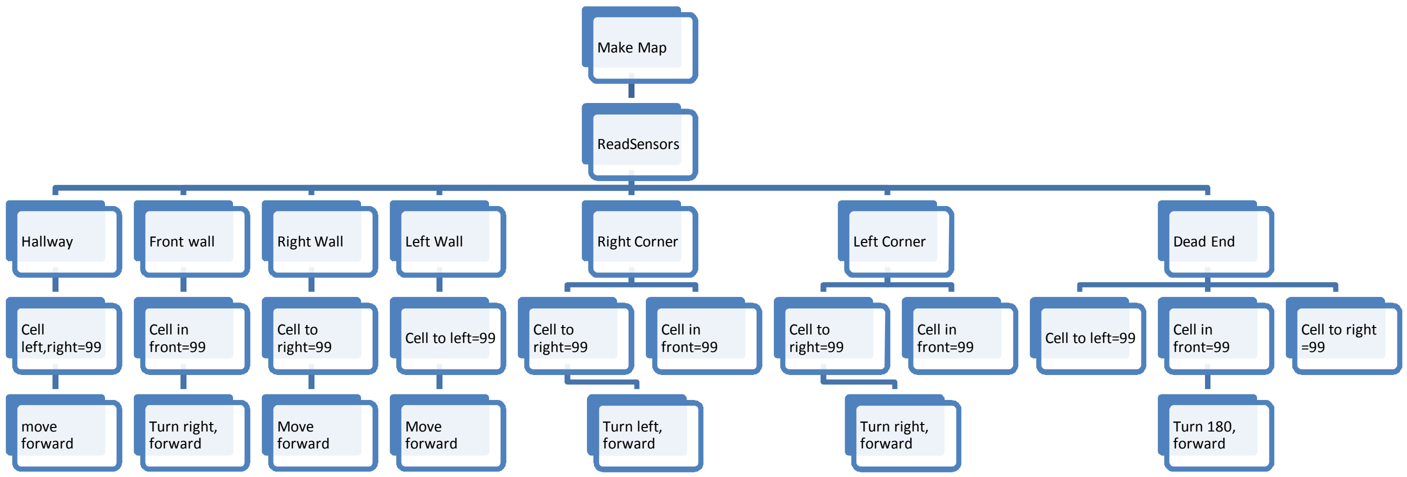
V. Conclusions and Recommendations

All tasks were achieved, with the exception of SLAM. The robot was successfully able to implement map making in an unknown world and return to its starting location. It used its sensors to identify surrounding landmarks and create an occupancy grid of the world. However, due to memory constraints the full wavefront algorithm was not implemented. Instead, a similar algorithm with map making was used for the robot to return to its starting location.

For part two, the robot was able to localize after a few iterations in the world. It would then plan out a path to a specified goal location. This location was hard coded into the robot because of memory constraints; if there were no memory constraints, the goal location would have been user specified via push buttons. Picking the robot up and moving it to a surrounding cell resulted in the robot re-localizing and then continuing on its path to the goal; this supplementary feature was added to allow the robot to re-localize if it wondered away from the planned path. SLAM was too memory intensive for the Arduino robots to handle and, instead, was discussed throughout this report as how it would have been implemented.

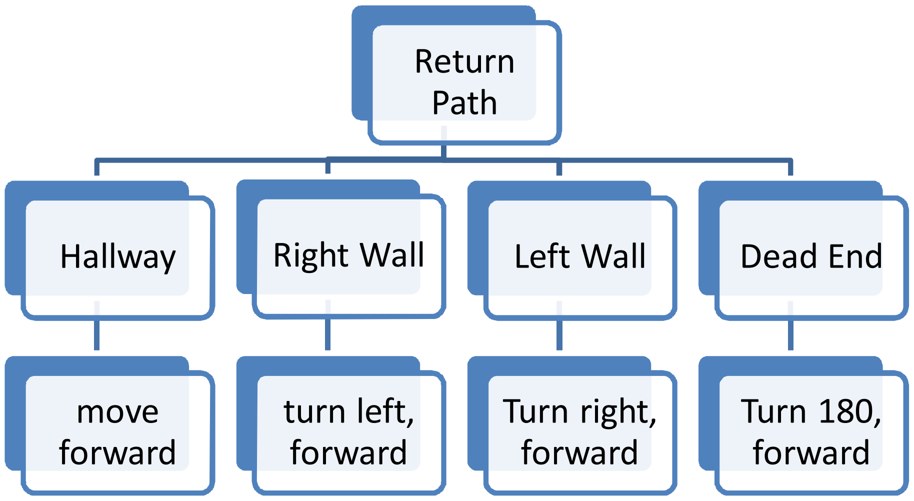
Appendix/Supplementary Materials

Map making diagram:

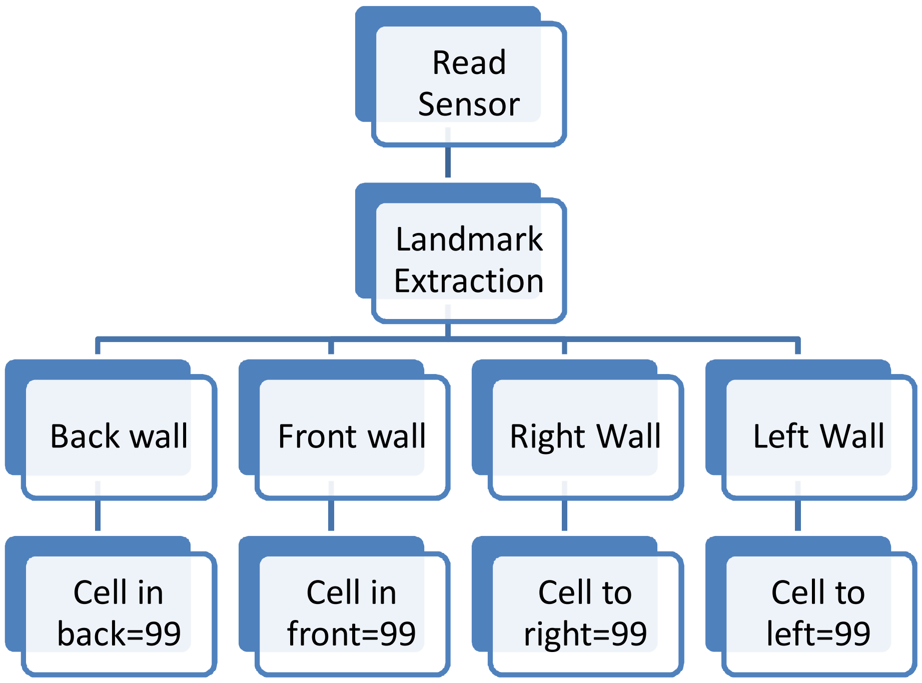


Repeated until the map is full.

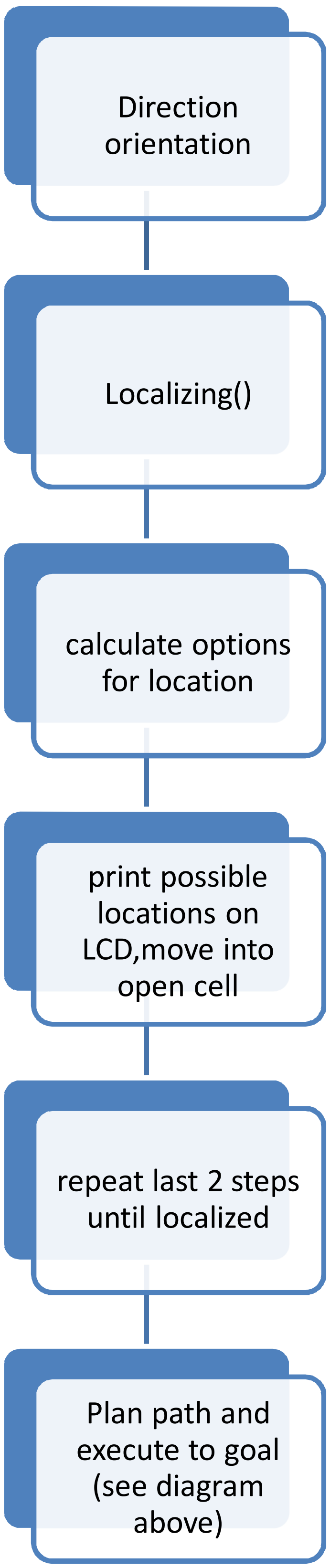
For the map, this is the process for returning to the starting location:



Localization diagram:



After sensors are read, calculate the next direction.


Direction orientation
Localizing()
calculate options for location
print possible locations on LCD,move into open cell
repeat last 2 steps until localized
Plan path and execute to goal (see diagram above)

References

Berry, C. A. (2012). *Mobile Robotics for Multidisciplinary Study*. Retrieved February 15, 2016, from http://www.morganclaypool.com/doi/pdfplus/10.2200/S00407ED1V01Y201203CRM004

Berry, C. A. (n.d.). *Lecture 7-2 Localization*. Retrieved February 15, 2016, from http://moodle.rose-hulman.edu/pluginfile.php/535303/mod_resource/content/1/ECE425 Lecture 7-2 (W1516).pdf

Berry, C. A. (n.d.). *Lecture 7-1 Map Making*. Retrieved February 15, 2016, from http://moodle.rose-hulman.edu/pluginfile.php/535300/mod_resource/content/1/ECE425 Lecture 7-1 (W1516).pdf

Blas, Morten Rufus (2005). *SLAM for Dummies*. Retrieved February 21, 2016, from http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam_blas_repo.pdf