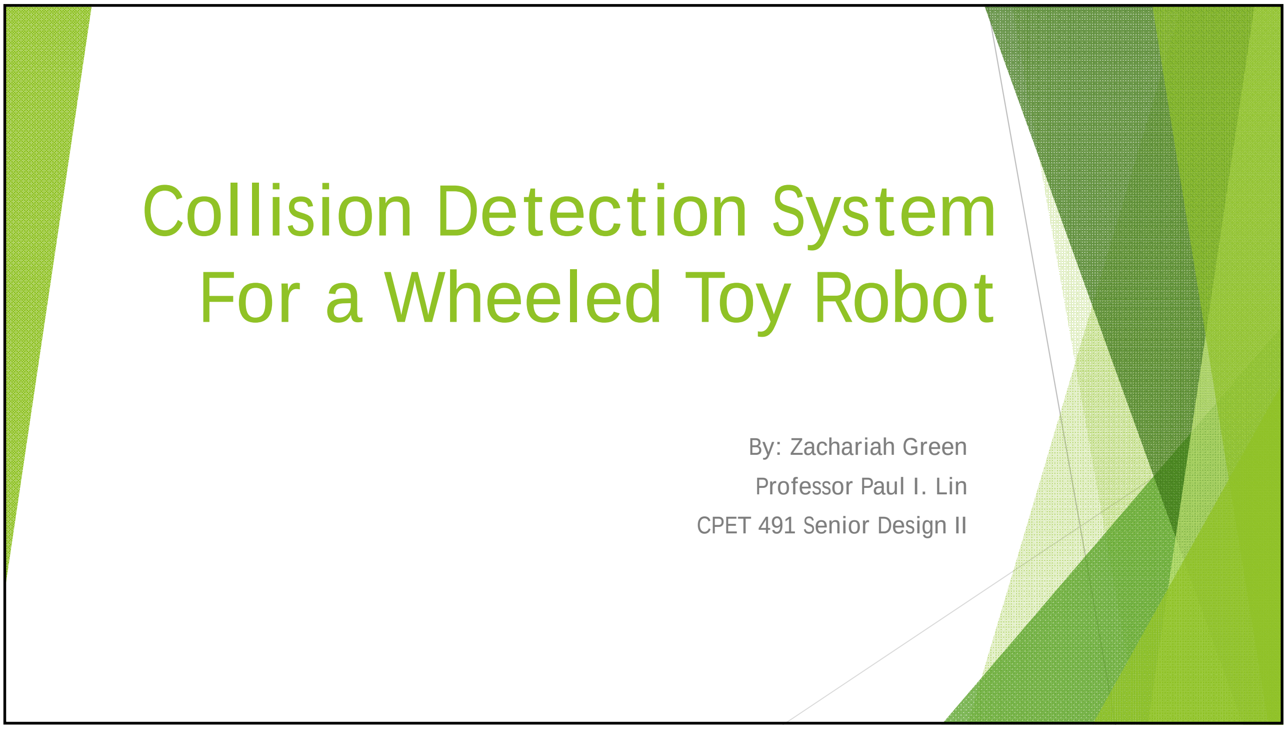


Collision Detection System For a Wheeled Toy Robot

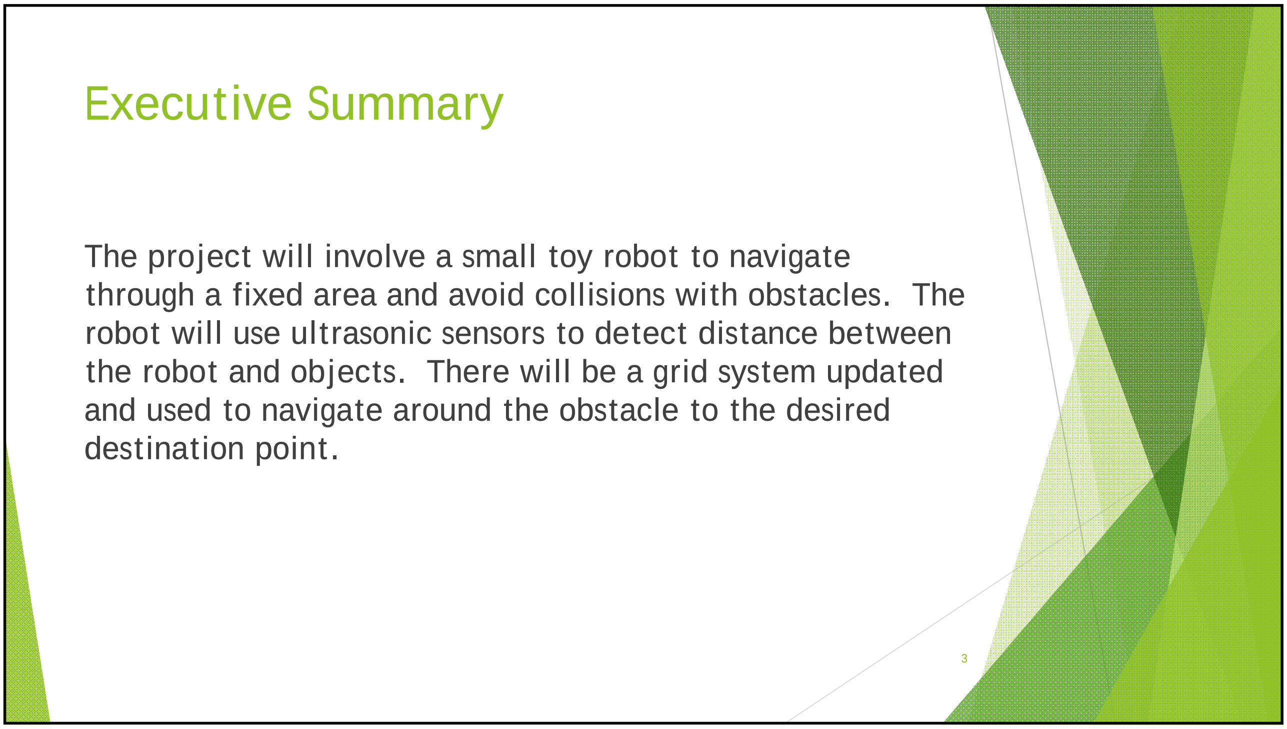
By: Zachariah Green

Professor Paul I. Lin

CPET 491 Senior Design II

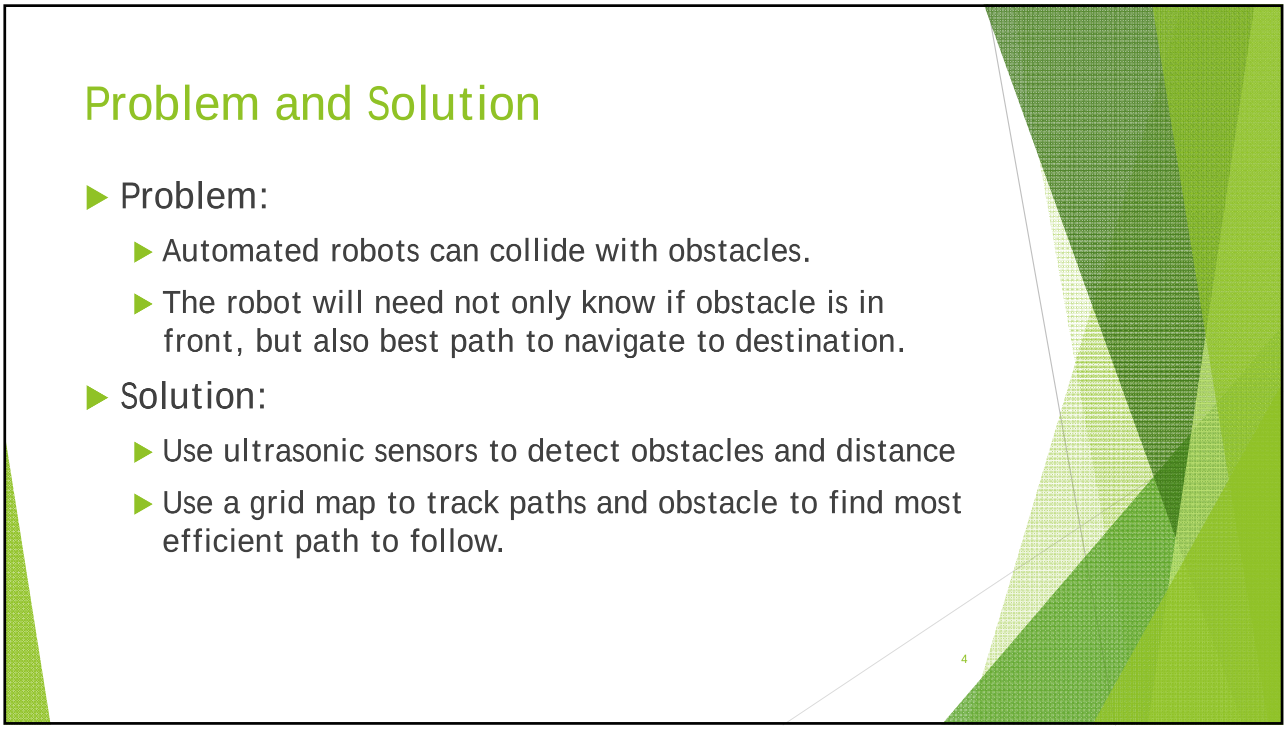
Outline

- Executive Summary
- Problem Statement and Solution
- System Requirements
- Robot Design
- Navigation Design
- System Validation
- Conclusion and Lessons Learned



Executive Summary

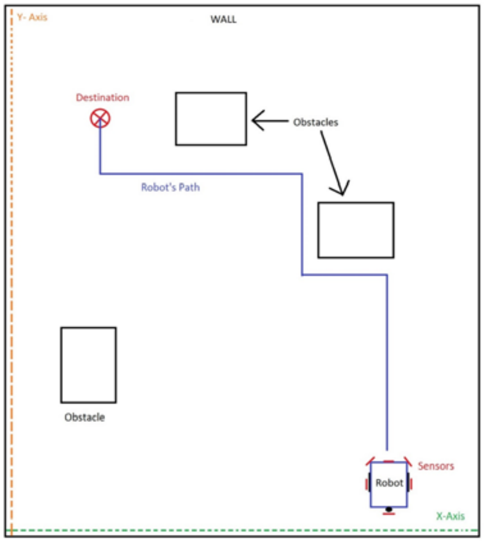
The project will involve a small toy robot to navigate through a fixed area and avoid collisions with obstacles. The robot will use ultrasonic sensors to detect distance between the robot and objects. There will be a grid system updated and used to navigate around the obstacle to the desired destination point.



Problem and Solution

- Problem:
  - Automated robots can collide with obstacles.
  - The robot will need not only know if obstacle is in front, but also best path to navigate to destination.
- Solution:
  - Use ultrasonic sensors to detect obstacles and distance
  - Use a grid map to track paths and obstacle to find most efficient path to follow.

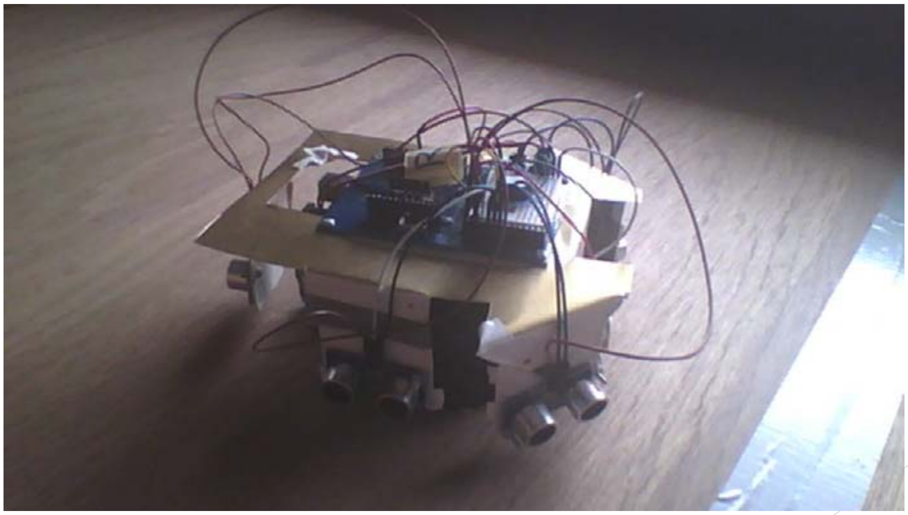
Overview Diagram



System Requirements

- The system shall not collide with stationary objects, while in motion.
- The systems shall store information about position of objects in relation to the system.
- The system should find the path to take least amount of travel time.
- The system shall operate in a fixed test area, indoor, flat wood floors and at room temperature.

Robot

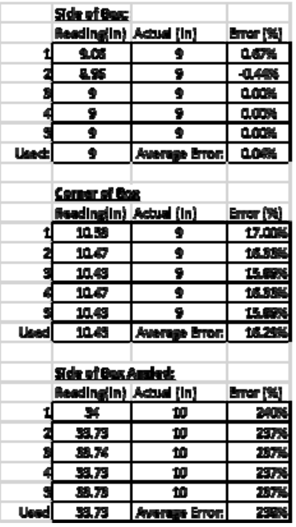


Program’s Main Components

- Ultrasonic sensor detect where the obstacles are located.
- Obstacle Grid stores where the obstacles are located in the test area
- Heuristic Grid shows the path to the destination and uses the Obstacle Grid to know how to avoid obstacles.
- The robot has different movement functions used to move the robot to specific points on the grid map.
- The robot slowly steps toward the destination and has to stop each time it takes a reading from the sensors, and updates the Obstacle and Heuristic Grids.

Ultrasonic Sensors

- Uses high frequency sonic pulse to detect distance
- Soft and furry substances can absorb sound and give bad readings.
- The object must be at a right angle to get the best reading.
- Robot only moves four directions to keep the objects at right angles.
- Readings aren't constant and created a function to filter out the error.



Side of Box:

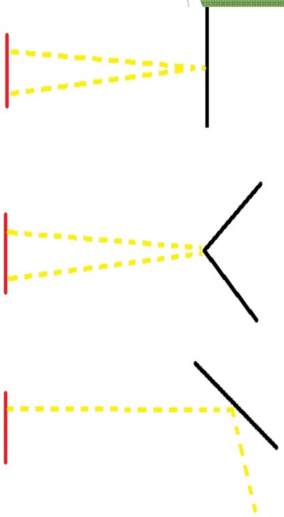
	Reading(in)	Actual (in)	Error (%)
1	9.06	9	0.67%
2	8.96	9	-0.44%
3	9	9	0.00%
4	9	9	0.00%
5	9	9	0.00%
Used:	9	Average Error:	0.04%

Corner of Box

	Reading(in)	Actual (in)	Error (%)
1	10.53	9	17.00%
2	10.47	9	16.33%
3	10.43	9	15.89%
4	10.47	9	16.33%
5	10.43	9	15.89%
Used	10.43	Average Error:	16.29%

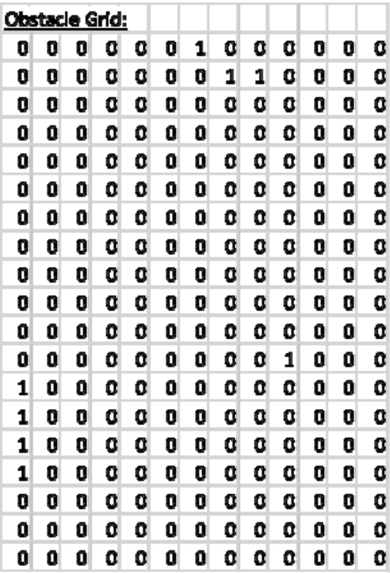
Side of Box Angled:

	Reading(in)	Actual (in)	Error (%)
1	34	10	240%
2	33.73	10	237%
3	33.74	10	237%
4	33.73	10	237%
5	33.73	10	237%
Used	33.73	Average Error:	238%



Obstacle Grid

- Two dimensional array to store the different grid points
- Zero shows a free space without obstacle, and one marks an obstacle
- The grid is a quarter the size of the actual test area.
- The grid is populated while the robot is in motion
- Use a GridPoint class for points

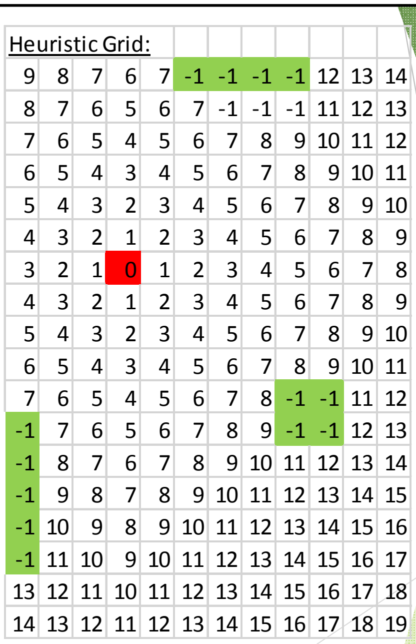


Obstacle Grid:

0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

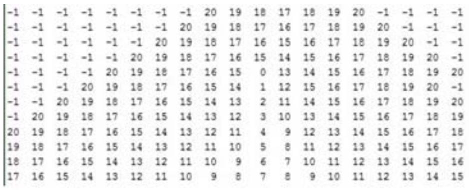
Heuristic Grid

- The grid represents the intersection points within the Obstacle Grid.
- The values stored at each point represent the number of steps needed to reach the destination.
- The values are populated using a recursive function that steps away from the destination and slowly works throughout the grid
- The robot uses these points for where to move.

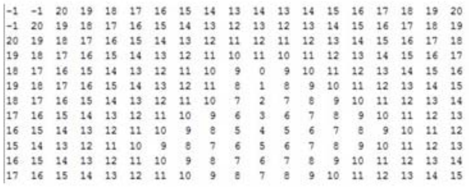


Heuristic Grid:											
9	8	7	6	7	-1	-1	-1	-1	12	13	14
8	7	6	5	6	7	-1	-1	-1	11	12	13
7	6	5	4	5	6	7	8	9	10	11	12
6	5	4	3	4	5	6	7	8	9	10	11
5	4	3	2	3	4	5	6	7	8	9	10
4	3	2	1	2	3	4	5	6	7	8	9
3	2	1	0	1	2	3	4	5	6	7	8
4	3	2	1	2	3	4	5	6	7	8	9
5	4	3	2	3	4	5	6	7	8	9	10
6	5	4	3	4	5	6	7	8	9	10	11
7	6	5	4	5	6	7	8	-1	-1	11	12
-1	7	6	5	6	7	8	9	-1	-1	12	13
-1	8	7	6	7	8	9	10	11	12	13	14
-1	9	8	7	8	9	10	11	12	13	14	15
-1	10	9	8	9	10	11	12	13	14	15	16
-1	11	10	9	10	11	12	13	14	15	16	17
13	12	11	10	11	12	13	14	15	16	17	18
14	13	12	11	12	13	14	15	16	17	18	19

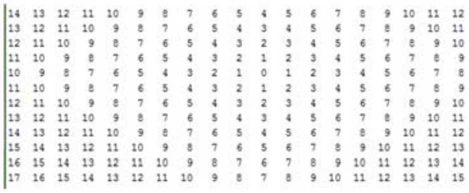
Heuristic Grid cont.



```
-1  -1  -1  -1  -1  -1  -1  -1  20  19  18  17  18  19  20  -1  -1  -1  -1
-1  -1  -1  -1  -1  -1  -1  20  19  18  17  16  17  18  19  20  -1  -1  -1
-1  -1  -1  -1  -1  -1  20  19  18  17  16  15  16  17  18  19  20  -1  -1
-1  -1  -1  -1  -1  20  19  18  17  16  15  14  15  16  17  18  19  20  -1
-1  -1  -1  -1  20  19  18  17  16  15   0  13  14  15  16  17  18  19  20
-1  -1  -1  20  19  18  17  16  15  14   1  12  15  16  17  18  19  20  -1
-1  -1  20  19  18  17  16  15  14  13   2  11  14  15  16  17  18  19  20
-1  20  19  18  17  16  15  14  13  12   3  10  13  14  15  16  17  18  19
20  19  18  17  16  15  14  13  12  11   4   9  12  13  14  15  16  17  18
19  18  17  16  15  14  13  12  11  10   5   8  11  12  13  14  15  16  17
18  17  16  15  14  13  12  11  10   9   6   7  10  11  12  13  14  15  16
17  16  15  14  13  12  11  10   9   8   7   8   9  10  11  12  13  14  15
```



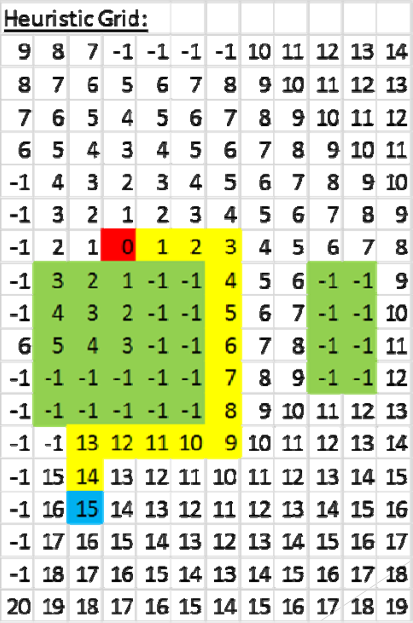
```
-1  -1  20  19  18  17  16  15  14  13  14  13  14  15  16  17  18  19  20
-1  20  19  18  17  16  15  14  13  12  13  12  13  14  15  16  17  18  19
20  19  18  17  16  15  14  13  12  11  12  11  12  13  14  15  16  17  18
19  18  17  16  15  14  13  12  11  10  11  10  11  12  13  14  15  16  17
18  17  16  15  14  13  12  11  10   9   0   9  10  11  12  13  14  15  16
19  18  17  16  15  14  13  12  11   8   1   8   9  10  11  12  13  14  15
18  17  16  15  14  13  12  11  10   7   2   7   8   9  10  11  12  13  14
17  16  15  14  13  12  11  10   9   6   3   6   7   8   9  10  11  12  13
16  15  14  13  12  11  10   9   8   5   4   5   6   7   8   9  10  11  12
15  14  13  12  11  10   9   8   7   6   5   6   7   8   9  10  11  12  13
16  15  14  13  12  11  10   9   8   7   6   7   8   9  10  11  12  13  14
17  16  15  14  13  12  11  10   9   8   7   8   9  10  11  12  13  14  15
```



```
14  13  12  11  10   9   8   7   6   5   4   5   6   7   8   9  10  11  12
13  12  11  10   9   8   7   6   5   4   3   4   5   6   7   8   9  10  11
12  11  10   9   8   7   6   5   4   3   2   3   4   5   6   7   8   9  10
11  10   9   8   7   6   5   4   3   2   1   2   3   4   5   6   7   8   9
10   9   8   7   6   5   4   3   2   1   0   1   2   3   4   5   6   7   8
11  10   9   8   7   6   5   4   3   2   1   2   3   4   5   6   7   8   9
12  11  10   9   8   7   6   5   4   3   2   3   4   5   6   7   8   9  10
13  12  11  10   9   8   7   6   5   4   3   4   5   6   7   8   9  10  11
14  13  12  11  10   9   8   7   6   5   4   5   6   7   8   9  10  11  12
15  14  13  12  11  10   9   8   7   6   5   6   7   8   9  10  11  12  13
16  15  14  13  12  11  10   9   8   7   6   7   8   9  10  11  12  13  14
17  16  15  14  13  12  11  10   9   8   7   8   9  10  11  12  13  14  15
```

System Validation

- First initializes the starting position of the robot
- Moves to the nearest Heuristic Grid point
- Steps through the different Heuristic Grid points towards the destination using the values in the Heuristic Grid.
- Each time it stops, scans for obstacles and updates the Obstacle and Heuristic Grids.
- Sounds a buzzer once it reaches the destination.

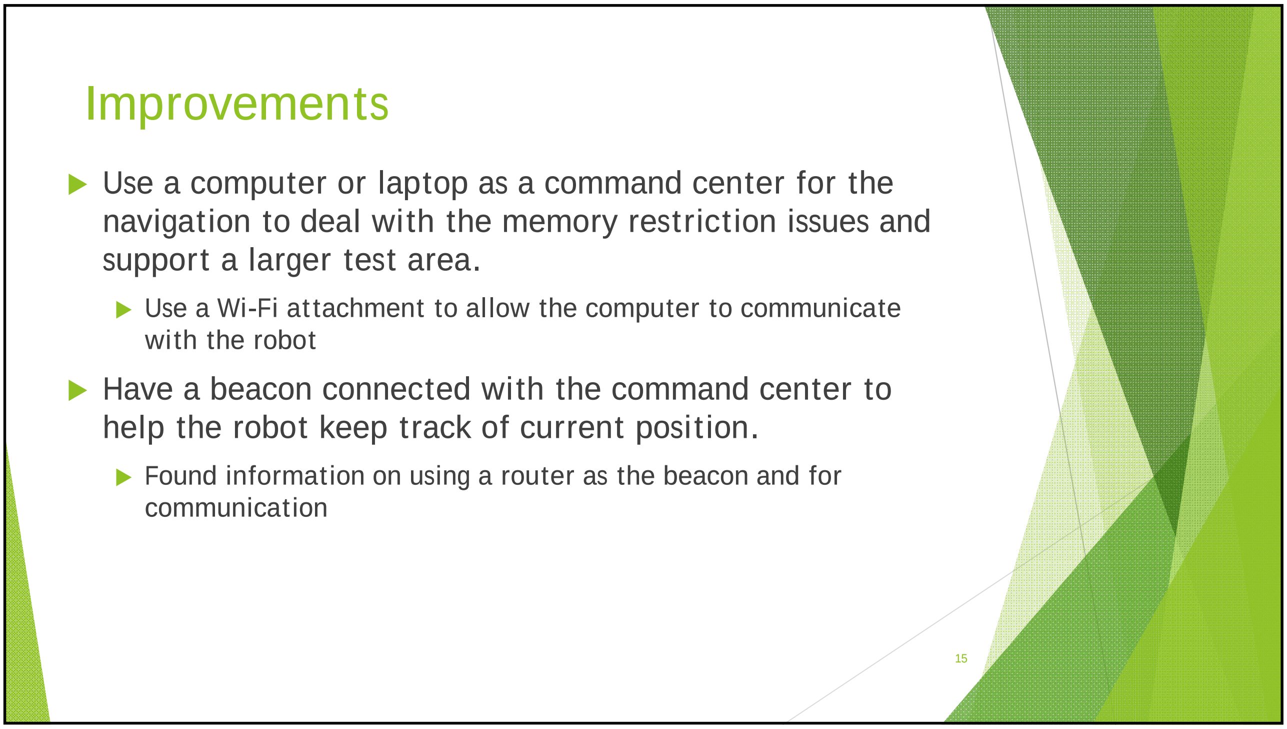


Heuristic Grid:

9	8	7	-1	-1	-1	-1	10	11	12	13	14
8	7	6	5	6	7	8	9	10	11	12	13
7	6	5	4	5	6	7	8	9	10	11	12
6	5	4	3	4	5	6	7	8	9	10	11
-1	4	3	2	3	4	5	6	7	8	9	10
-1	3	2	1	2	3	4	5	6	7	8	9
-1	2	1	0	1	2	3	4	5	6	7	8
-1	3	2	1	-1	-1	4	5	6	-1	-1	9
-1	4	3	2	-1	-1	5	6	7	-1	-1	10
6	5	4	3	-1	-1	6	7	8	-1	-1	11
-1	-1	-1	-1	-1	-1	7	8	9	-1	-1	12
-1	-1	-1	-1	-1	-1	8	9	10	11	12	13
-1	-1	13	12	11	10	9	10	11	12	13	14
-1	15	14	13	12	11	10	11	12	13	14	15
-1	16	15	14	13	12	11	12	13	14	15	16
-1	17	16	15	14	13	12	13	14	15	16	17
-1	18	17	16	15	14	13	14	15	16	17	18
20	19	18	17	16	15	14	15	16	17	18	19

Conclusions and Lessons Learned

- Conclusions
  - The robot does navigate, but has to stop each step to make sure to get good sensor readings.
  - The angled sensors cause more problems than help due to the angled reading giving bad information and “shadow” obstacles
- Learned
  - Work within limited memory of the microcontroller.
  - Overconfidence when planning and have more problems than intended
  - Program how to calculate scenarios instead of how to react to scenarios.



Improvements

- Use a computer or laptop as a command center for the navigation to deal with the memory restriction issues and support a larger test area.
  - Use a Wi-Fi attachment to allow the computer to communicate with the robot
- Have a beacon connected with the command center to help the robot keep track of current position.
  - Found information on using a router as the beacon and for communication

15

Questions

?

16

Demo

17