Final Project Report

Bernardo de Barros Barreto Costa Machado, Jaroslaw Socha

07/05/2024

Table of Contents

1. Introduction

The purpose of this report is to describe and evaluate the performance of a controller designed to navigate an unknown map and rescue two people autonomously in an environment that emulates a disaster. The motivation of this project is to combine knowledge learned throughout the laboratory course to create an autonomous robot to complete specific tasks.

The initial requirements are as follows: There are 10 base scenarios which differ in the distribution of obstacles and lighting conditions, there are two people at the other side of the scenario (indicated by the yellow lines on the floor) represented as two green cylinders. An example of one scenario may be seen in Figure 1 below.

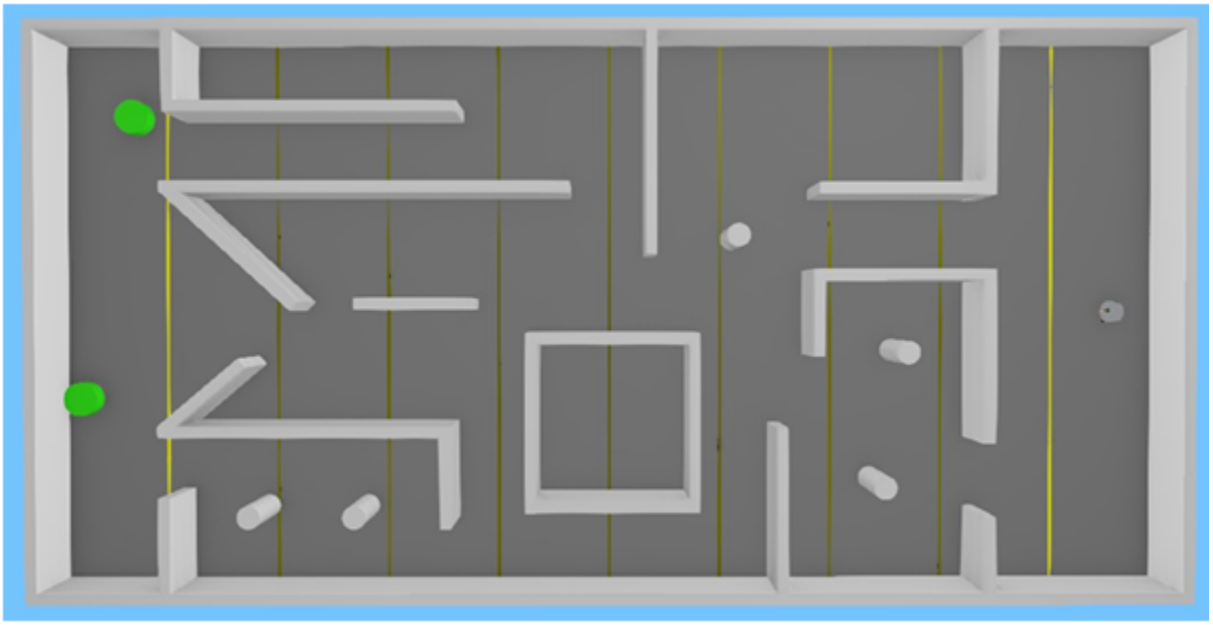


Figure 1: Scenario 1 provided for testing

The robot to be used is the Pioneer II - which may be seen in Figure 2 below - and the sensors allowed are the compass, GPS, distance sensors, encoders, and forward/spherical cameras. It is important to note that the GPS has a resolution of 3 meters. The robot will navigate to the other side of the map, avoiding obstacles and walls, identify the two people, then navigate back to the departure point. A maximum of 4 minutes is allowed for the tasks. To properly rescue the people the robot must stop within 1 meter of the person, spin around once and a message should be outputted to the terminal that a person was found.

Figure 2: Pioneer II Robot

2. Problem Background

As described in the introduction the problem is a simulation of a disaster zone. Given two yellow lines, one being the departure zone and the other the arrival zone where the people to be rescued are located, the robot must navigate autonomously with the designed controller. Upon arrival to the area where people will be located, it must find them and stop within a meter of them. Then it must spin and output a message to the terminal. Finally, it will navigate back to the departure zone. In Figure 3 the maps that the robot may be tested on from a top down view of them.

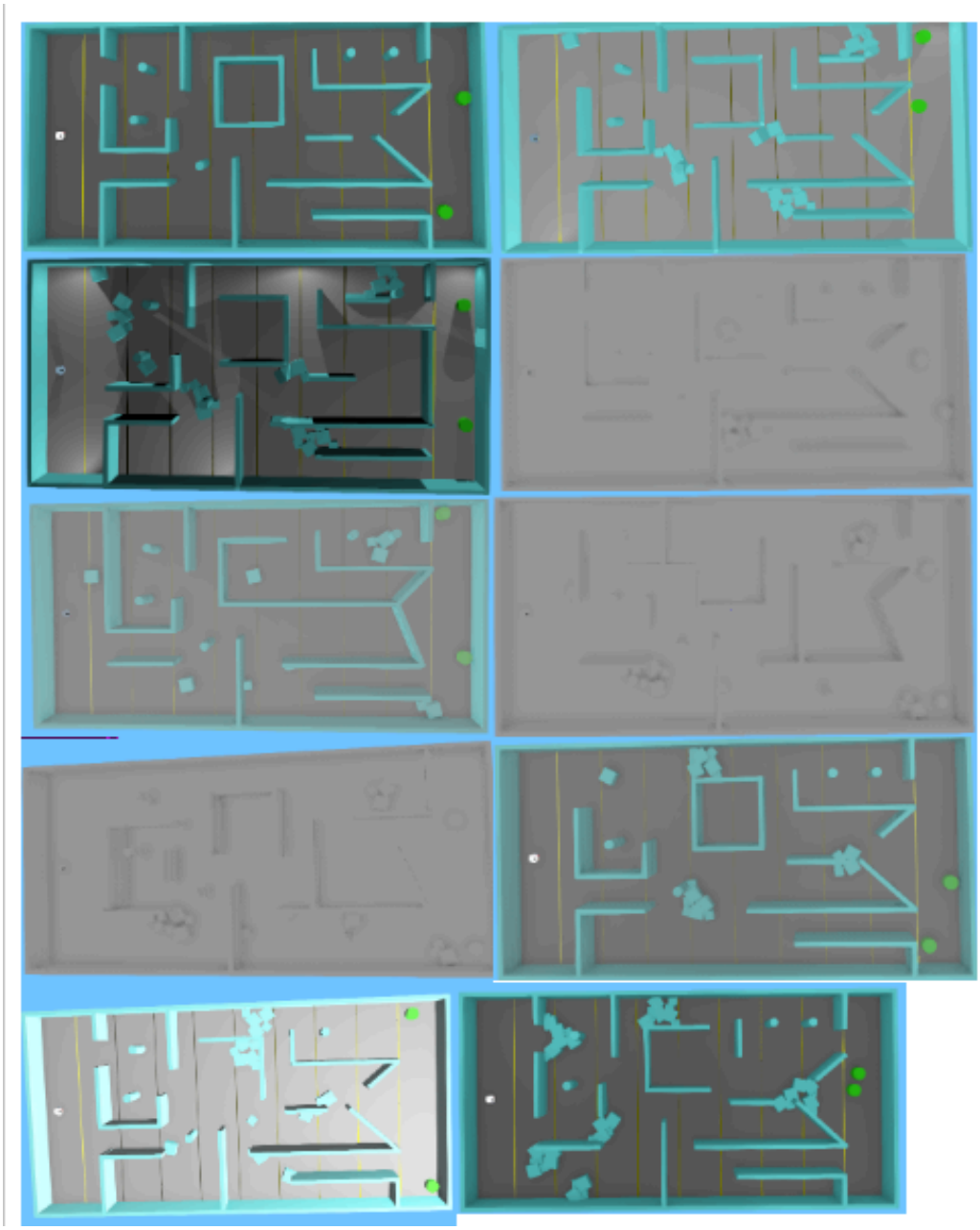


Figure 3: 10 Scenarios Provided

There are a total of 4 minutes to complete all the tasks, if not done in 4 minutes the robot will be scored by how far it traveled as well as how many people it found. Collisions are not allowed.

The main problem to get around is navigating in an efficient manner, this requires the use of Simultaneous localization and Mapping (SLAM). Furthermore, the robot's sensors may be affected by conditions on each map. For example, differing lighting conditions, fog, or non-conforming obstacles that may be in its path.

3. Solution Achieved

The solution achieved is better described in three parts. The first part is the navigation of the robot to the arrival area. Second, the controller instructs the robot to look for the people required. Finally, the robot will need to navigate back to its original position. The sensors used in the solution include: front camera, infrared distance sensors, odometers, and GPS.

For the first part, the robot will utilize the Bug 2 algorithm previously introduced in the laboratory classes. In addition to this, the controller will also construct a map of every position the robot has been in while navigating to the final position. The way that bug 2 works is by knowing its initial position, acquired by the GPS and drawing a line to its goal position determined in the code. The robot will then follow this line and if it encounters an obstacle (determined by distance using the infrared distance sensors) it will begin wall following with the obstacles wall on its right. The infrared distance sensors are the best option to use in this scenario as they are independent of lighting conditions and will work in all maps. Once it re-encounters the line (on the other side of the obstacle) it will turn back towards the goal and follow the line once again. An example path for the robot may be seen in Figure 4 below.

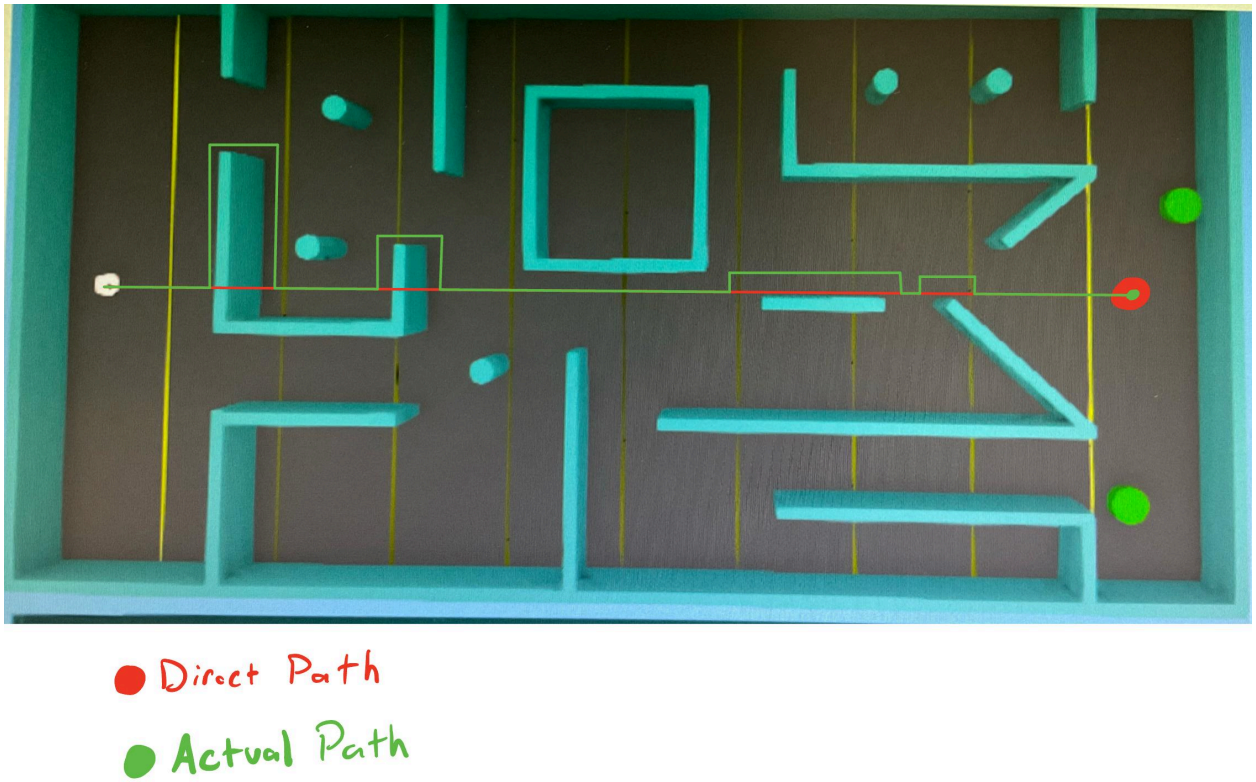


Figure 4: Map 1 Bug 2 Paths

For this algorithm there was a fine tuning of the speed that had to be done in order to ensure that no collisions would occur while also moving forward towards the goal position in a timely manner. Repeating this process until it reaches the goal position. A flowchart of the algorithm can be seen in Figure 5 below.

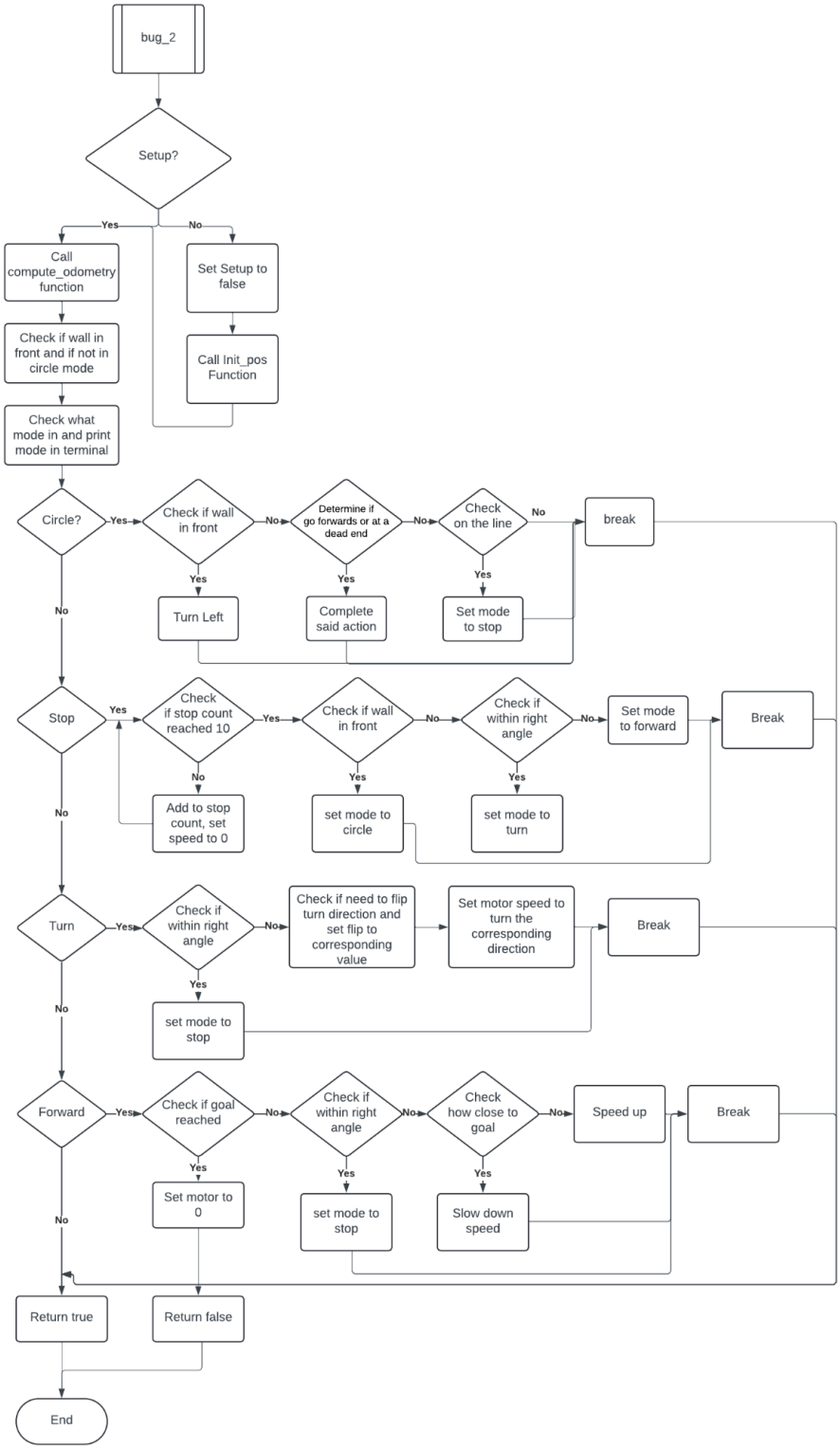


Figure 5: Flowchart of Bug 2 algorithm

It is important to note that in parallel to the bug 2 algorithm the robot will be storing its position in an array for later use when traveling back to its original position after identifying people. This

array is put into a text file which can allow the user to easily see the map created once the program runs to completion. This can be seen indicated in Figure 6 below.

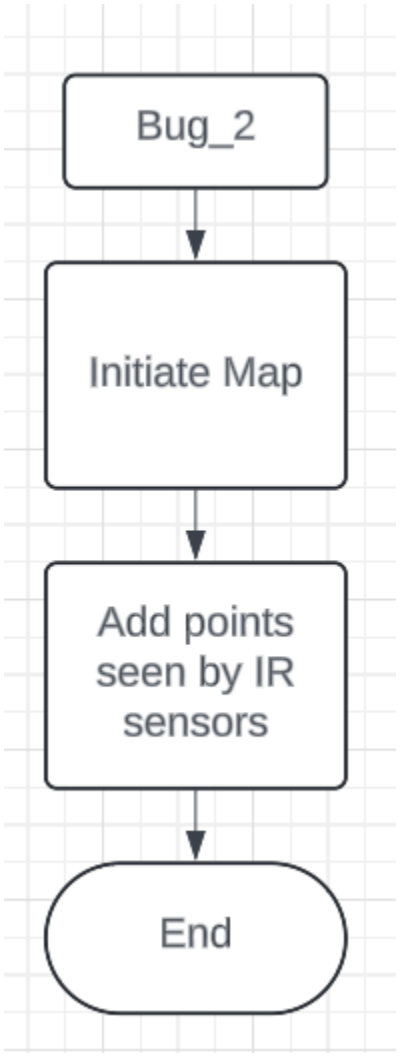


Figure 6: Flowchart of mapping algorithm

Once at its goal position the robot will determine where the people are and go to them, stopping within 1 meter then spinning around and outputting to the console that it has found a person. The way it does this is by spinning around while looking for green in the forward camera. When green was found it went towards it until a specific threshold was met. The way that the robot goes towards the posts is also determined by the front camera. The image is segmented into four different parts, the left third, the middle third, the right third, and the center box (from a 3x3 grid). If green is identified on the left or the right the robot will turn this way until the middle third identifies green. Once that occurs, the robot will go forwards towards the post and identify it using the total percentage of green in the forward camera image. After this, the robot spins around and outputs to the terminal and then proceeds to turn the opposite way it originally did (Clockwise) to find the next person. The center box image is used to identify the green post if it is far away as the accuracy of the larger image may not be clear enough. Once

green is found in the center box the robot moves forward towards it until the other three segments of the image may be used to identify its precise location. The flowchart for the function that identifies people may be seen in Figure 7 below while the extra functions used may be seen in Appendix 1 at the end of the document.

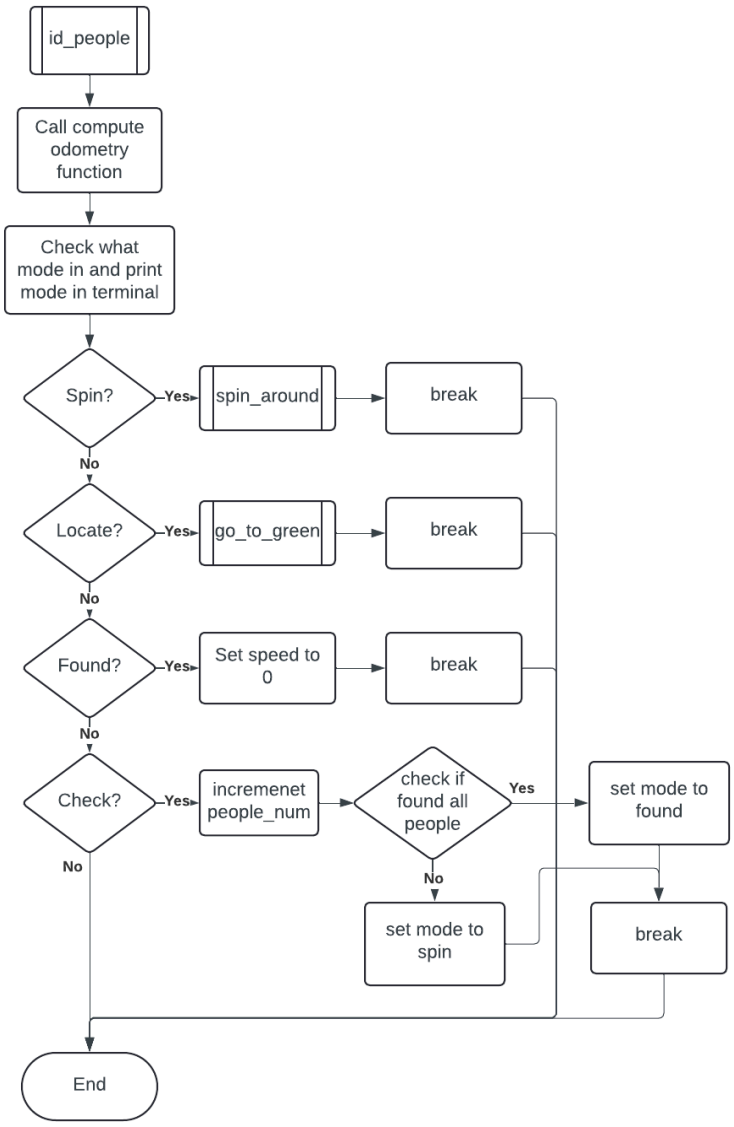


Figure 7: Flowchart of people finding algorithm

For finding the people a standard threshold had to be set for the RGB of the green posts. Through trial and error this was found to be 35 % to be within the 1 meter distance and a threshold of 80. The front camera image for this point may be seen below in Figure 8.

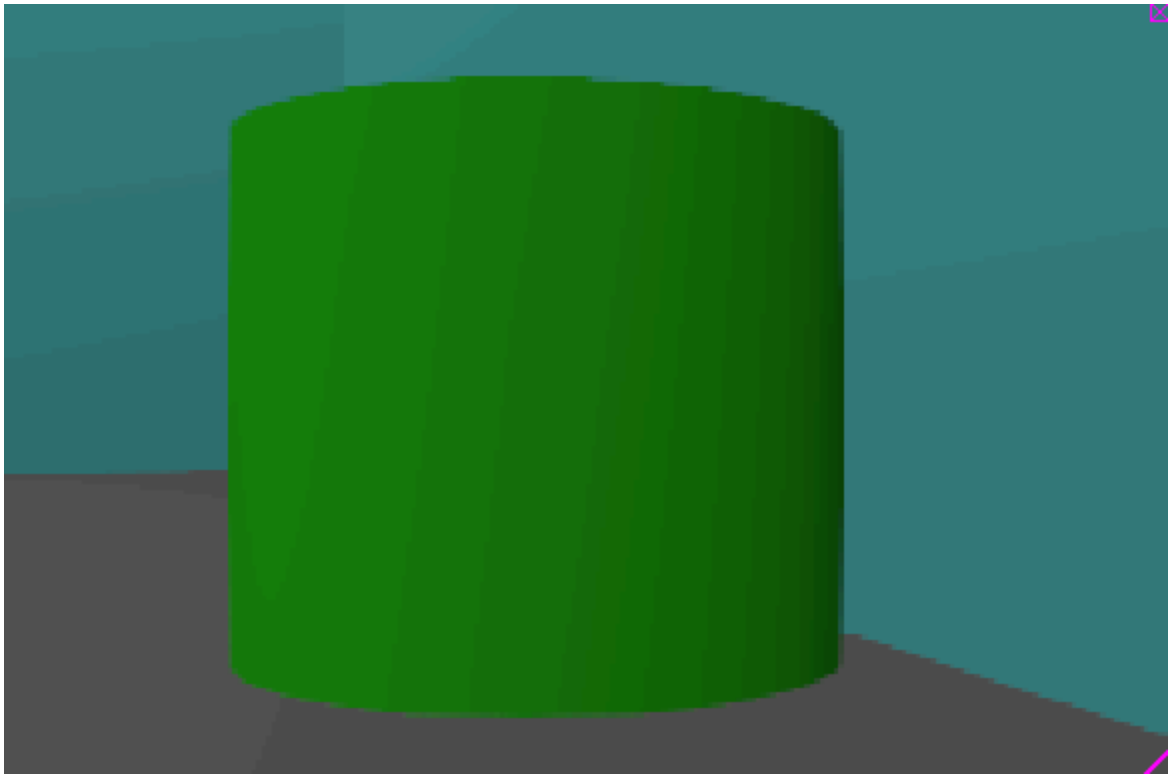


Figure 8: Front image camera for 1 meter distance to post

After the people are identified the robot will navigate back to its original position. As mentioned earlier, the robot makes a map of its surroundings by utilizing sensors. At first, the map is unexplored, and with every step the robot uncovers unexplored cells according to what side and front sensors can see. Using the map, the controller determines the shortest path back to its original position. This allows the robot to go back faster than it took to get to the arrival position, saving time in the process. For this purpose Djikstra's algorithm has been used. It works by calculating the number of steps to the initial position, then determining the best path. The heuristic function used rewarded the robot for getting closer to the goal, for getting farther than the starting position and punished it for sticking too close to the walls (to prevent collisions). After the path had been calculated in a point by point manner, the final path was calculated as a truncated set of points to follow, with an emphasis on the longest stretches of straight lines there could be.

From this the robot will follow the shortest and safest path back to the initial position. The path it follows is also highlighted in the text document on the map, along with the truncated points list.

4. Results

After the controller was properly tested and debugged, a final test on each map was done and the number of collisions, time it took for the robot to complete the tasks, and number of people found was tallied. If the map was not able to be completed, the number of lines traveled was included - where each line is 2 meters and there and back has 18 lines normally, otherwise this is not applicable. These results may be seen in Table 1 below. These tests were run on a windows system, some slight discrepancies in time completed may be present due to the presence of a graphics card on the testing device.

Table 1: Results from simulation of controller on each map

	# Collisions	Time Completed (min:sec)	# People Found	# Lines Traveled
Map 1	0	3:15	2	N/A
Map 2	0	6:47	2	N/A
Map 3	1	6:21	2	N/A
Map 4	1	3:07	2	N/A
Map 5	0	4:27	2	N/A
Map 6	0	4:33	2	N/A
Map 7	1	6:02	2	N/A
Map 8	0	3:15	2	N/A
Map 9	0	3:39	1	N/A
Map 10	1	5:35	1	N/A

The maps created by the robot may also be seen for Map 1 below in Figure 9 as it completes its tasks. Maps follow the general pattern shown. After the full map is created and the people are found the controller plots the shortest path and that can also be seen as the thin line going through the explored regions, this may be seen in Figure 10.

Figure 9: Mapping progression for Map 1

Figure 10: Map with shortest path taken by robot (World 3). Orange points are the truncated points for the robot to follow

5. Discussion

The results are expected for the controller designed as the broad application covered most maps and obstacles presented during testing. The main factor that was affected by changes in maps was speed, this is varying between 3 minutes and 6 minutes for the longest map. The average speed across all ten maps was 4:47. Although not below the 4 minute requirement, the controller correctly identifies two people with minimal collisions in most maps. The mapping works as intended and the final maps are seen above in Figure 9 and 10.

The main limitation of the controller design is the implementation of the people finding algorithm. Due to the nature of the first map (considered the testing map) the green posts were positioned in a manner where they can be easily identified. The main limitation is when the posts line up with the path of the cameras of the robot. This leads to the robot incorrectly identifying the same post twice and then proceeding with its return journey to the starting position. The other reason why the people finding it is flawed is due to the use of cameras and no other sensors. When lighting conditions are different the thresholds set for the color green change slightly leading to the cameras misidentifying other parts of the map as green, or not finding green at all. The same happens when fog is present in maps. This is what happens in map 9 and 10 where the robot is only able to identify 1 person. In map 9 the reason is that it is very bright and the threshold set for identifying green would have to be modified for the algorithm to work properly. Furthermore, the collisions present in map 3, 4, 7, 10 are due to the camera not identifying that the person is in front properly due to fog, low light levels, and shadows. Eventually in all maps but 10 it does identify that the person is there, however, it is too late as it has run into the person.

The other limitation was the bug 2 algorithm. On the way to the final destination where the people are, the bug 2 algorithm was implemented. When the robot encounters a wall in front of it and nothing to its right, it automatically turns left (right wall following). This can lead the bug 2 to be in a loop and not be able to traverse the map initially. To resolve this issue the map was used to determine if the robot has been at that location previously and then it will turn right (left wall follow) from then on, unless another loop is encountered and it will flip back then. This leads to significant time consumption. The main reason maps 2, 3, 5, 6, and 7 take more than four minutes is due to this. There are certain obstacles and patterns in the maps that make the robot loop, leading to more time to traverse the map on the way to the people.

Another aspect was connected to mapping, and more precisely the sensor range. Due to low distance sensor range, the robot couldn’t map far out (it could do barely more than a meter on each side), so sometimes when going back it was traversing around free space, just because it wasn’t mapped out by its sensors.

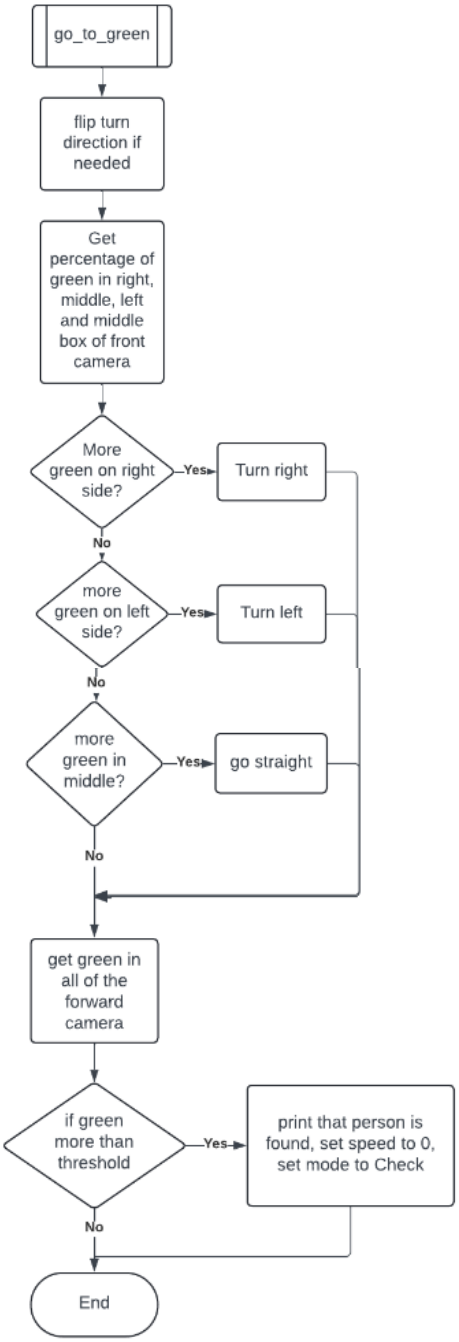
The final factor was the speed. Due to the sensor's accuracy and the information that is required for the mapping, the speed had to be reduced in order to traverse the map and localize properly. This leaves the controller with little time in the process of going to the final destination, identifying people, and returning. The main reason this happens is because the speed had to be reduced to ensure proper functioning of the sensors during the tests.

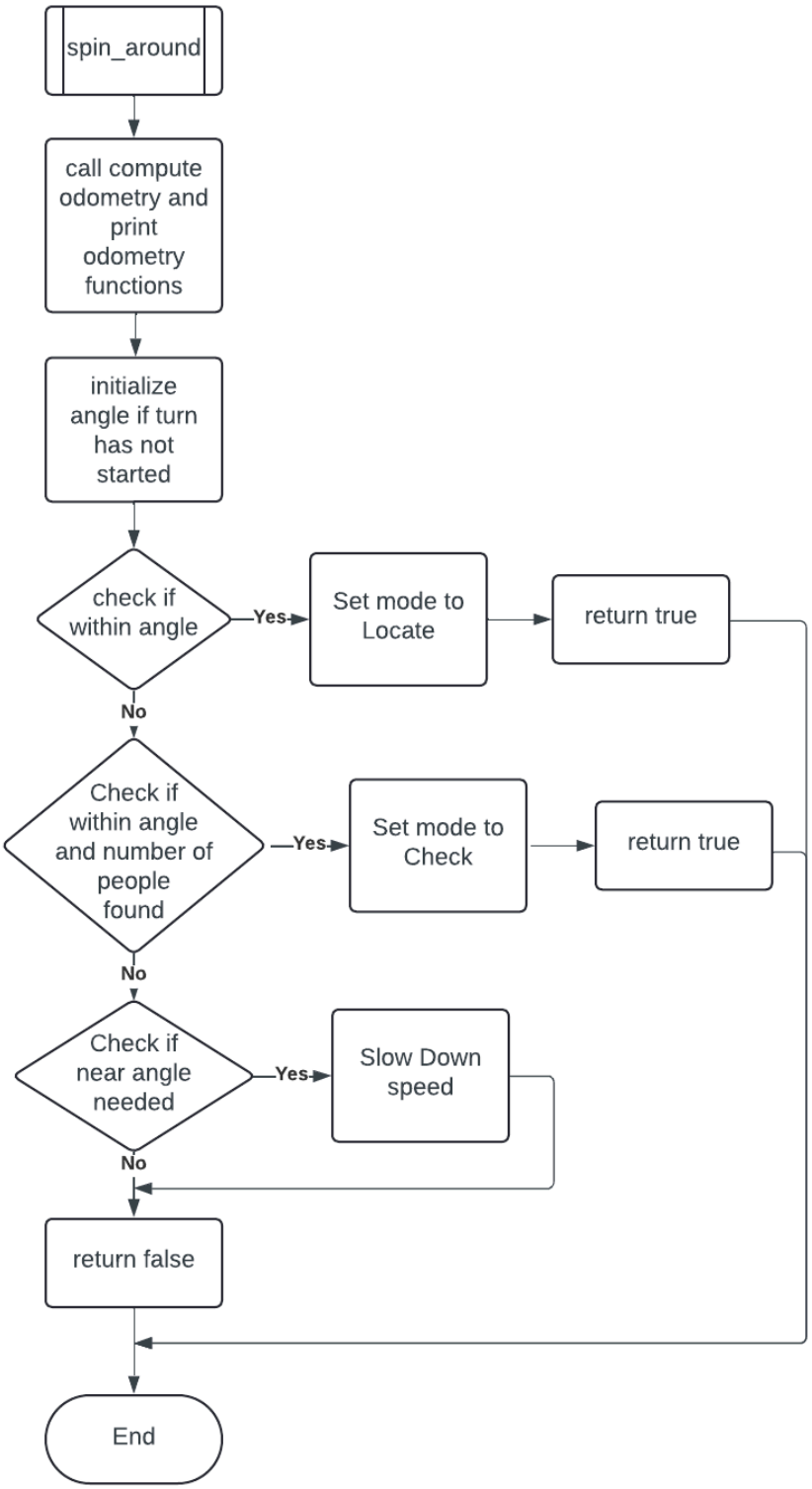
6. Conclusion

In summary, a controller was designed to fulfill the three main tasks of navigating to a point, identifying people (green cylinders), and then navigating back to its starting position, in under 4 minutes, with the least amount of collisions possible. This was done by using the bug 2 algorithm while creating a map of the scenario provided, using an algorithm to find the people, and then using Djikstra's to navigate faster on the way back to its starting position. The results of this approach led the robot to succeed in 5 of the maps completely, 3 others with one collision, and the last 2 with 1 person identified. The ways to improve the current design are to improve speed on maps that require looping due to the nature of the bug 2 algorithm, improvement of identification of people with adaptable thresholds and being able to handle changing light and environment conditions. Additionally, speeding up the robot to be able to fall within the 4 minute requirement.

Appendix 1: Flowcharts for Functions

Flowchart of go_to_green function seen in Figure 6 flowchart.





Flowchart of spin_around function seen in Figure 6 flowchart.