Erick Chacon
echacon@uncc.edu

Demonstration points:

Requirements:

Req. 1 No sensors will be used. _____

Req. 2 The algorithm will be implemented to have the robot navigate and make decisions _____ on its own.

Req. 3 Robot Will Operate Autonomously. _____

Req. 4 Students will have access to the testing area prior to final demonstration. _____

Pre Lab:

1) How do you go about building the map to use for the A* Path Planning algorithm?

The map for the A* was represented as a 2D array of non-zero values in LabVIEW. Each value of the array (node) was denoted with a weight which would determines whether the node was to be traversed or not. A traversable node was associated a weight of 1 whereas a non-traversable node was given value of 100. In the given LabVIEW example, we have built a simple map of two values (white & black squares) using a 2D array of Boolean controls to serve as the map input. The white squares denoted the traversable region whereas the black region was the obstacle region. Herein a 36x 36 2D array was defined and the start and destination nodes were provided to it as (0, 1) and (2, 5) as shown by the Figure 1.

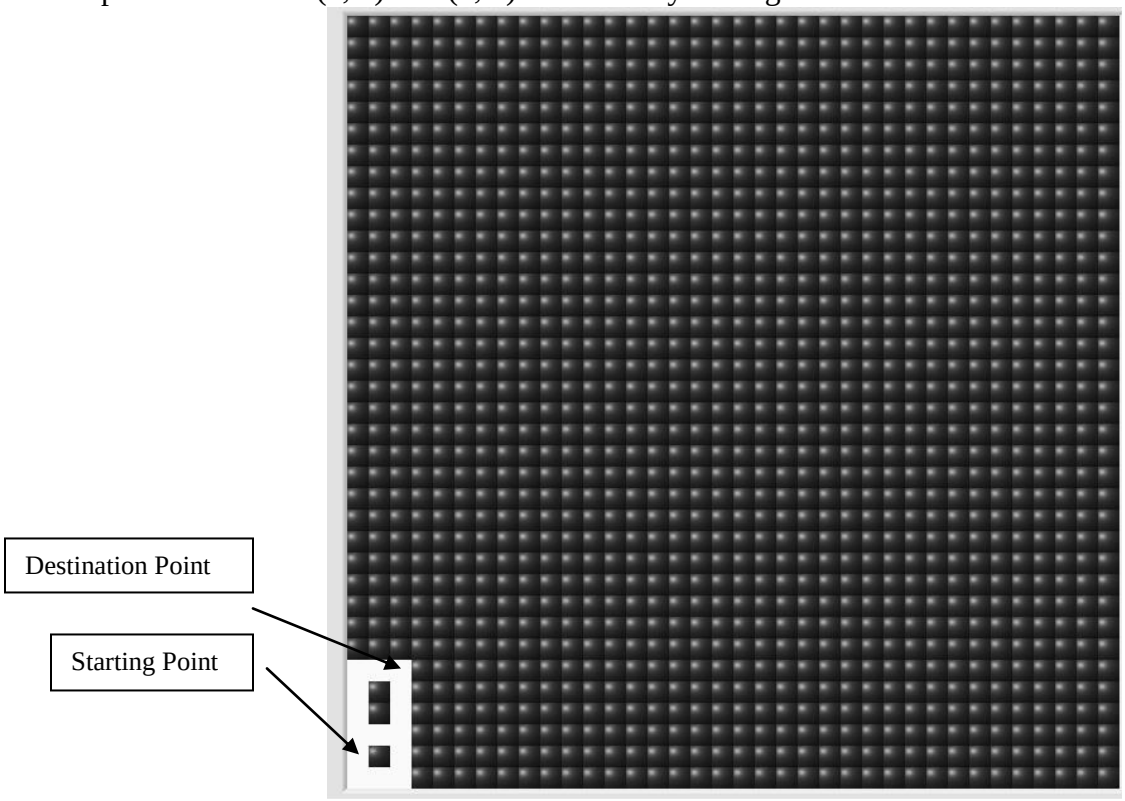

Figure 1. Map input for given problem

2) Explain the procedure involved after building the map.

After the building the map, the start and end points were provided as inputs to the AStar_Sub VI. Depending on the start point, end points, the traversable locations & obstacles the A Star algorithm would provide the shortest path from the start and end point. Depending upon the path values, the direction maneuvering decisions were made by the Determining Direction sub Vi. The maneuvering decisions were further used to govern the motion of each motor.The A Star Algorithm uses F=G+H formula to compute its next plausible node. Herein G is the movement

cost to move from start point to the node on the grid. H is the estimated movement cost to move from that node on the grid to the destination node. The A Star algorithm provided the solution map with blue squares indicating the shortest path from the start point to the end point. The blue blocks hold value of -2 in the array and the motion of the robot were planned along these blue blocks as displayed by Figure 2. Moreover the black square (obstacle) regions were allotted an value of 100 whereas the traversable blocks which were not part of the planned path were given a value of 1.

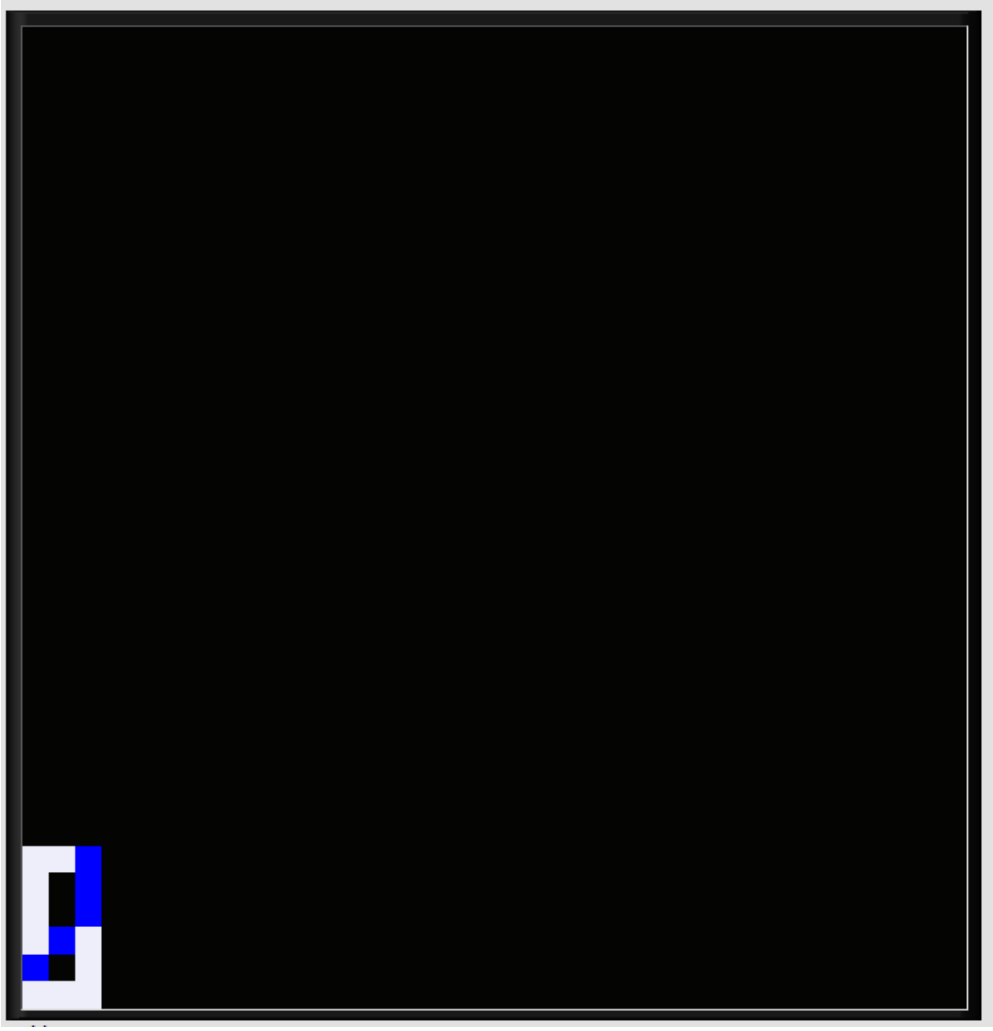

Figure 2. Shortest path provided by the A* algorithm for the input map array.

General Learning Objectives:

The general learning objectives of this lab was to introduce the A* planning algorithm to the users & implement it to navigate the robot through a pre-defined grid.

General Steps Needed to Complete the Lab:

The general steps needed to complete this lab were to follow the tutorial provided to us. The A* algorithm was obtained from NI and additional changes were made to this VI to form the AStar subVI. The output solution array from this sub VI was used to determine the shortest path for the robot path planning. Another sub VI called the Determining direction VI was implemented to determine direction of motion from this solution array. The switch case logic to control the input to the robot motors depending on the determined direction was included in the Determining direction sub VI.

Procedure / Detailed Steps to Complete the Lab:

The lab experiment was broken down into 9 steps:

1. A new project entitled Lab5AStar was created in LabVIEW and a file called 'A-Star Simple Example VI' was obtained from the Moodle2 lab resource. This downloaded program was added to the project.

2. The 'A-Star Simple Example VI' was copied into a new VI called the AStar subVI. While making this sub VI it was ensured that the connector plane was same for all other subVI to be used in the main VI. Thus the input map was connected as an input while the occupancy grid was connected to the output plane. The completed front panel of AStar subVI is displayed by Figure 3. The timed loop was removed from the back panel by selecting the Right click based option 'Remove Timed Loop'. The occupancy grid was added as an indicator to the completed back panel of AStar subVI is displayed by Figure 4. Before closing & saving this subVI the occupancy grid indicator was copied from it.

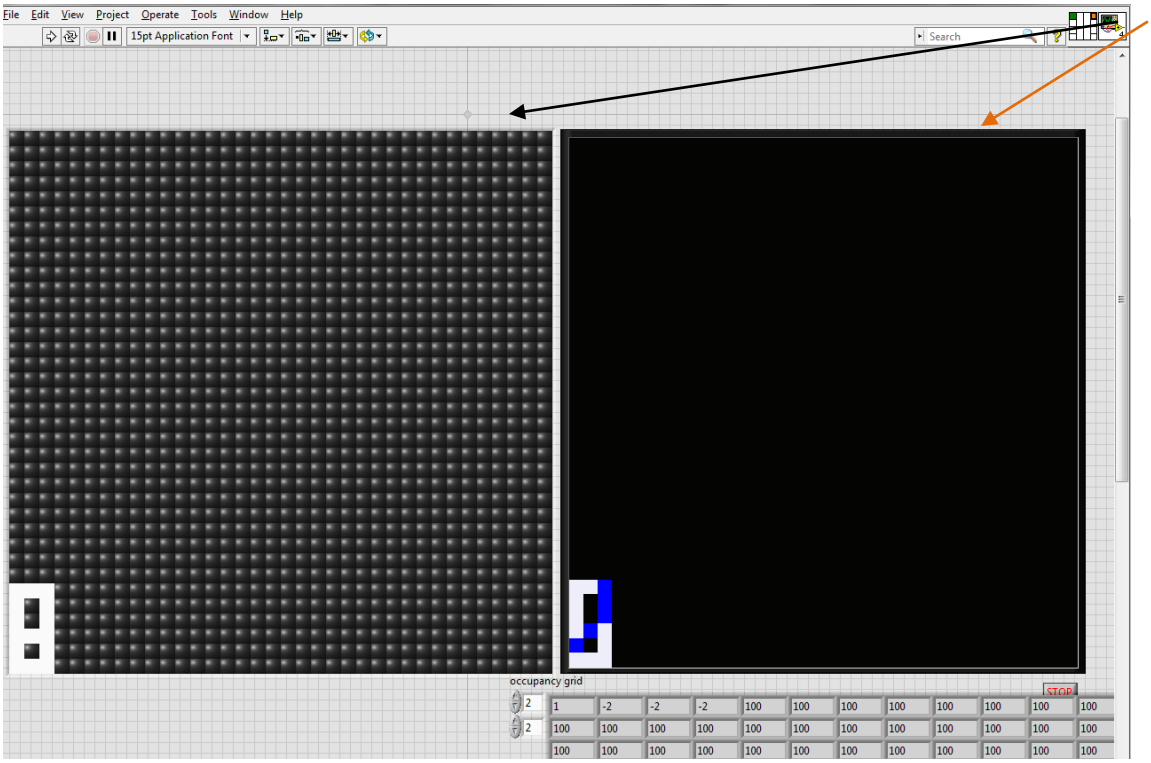

Figure 3. Completed Front Panel for A* sub VI

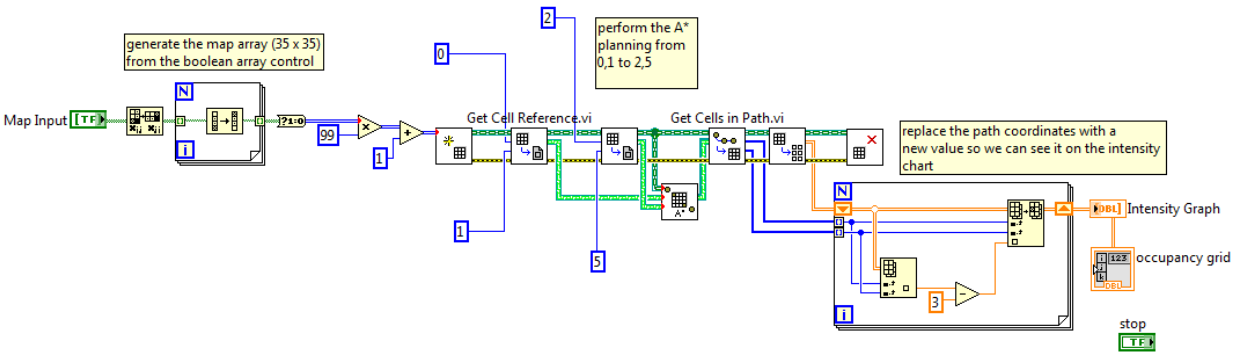

Figure 4. Completed back panel for A* sub VI

3. A new VI was created and was named "Determining direction". The previously copied occupancy grid indicator was pasted onto this new blank VI. Two numeric constants were added to the VI along with the occupancy grid. The constants were converted to indicators. An index array subVI was placed onto the program in order to set up the array in which the program would determine which direction to go. The inputs to the index array are the row and column values. Depending on what this value is, the output must represent what direction it will move the robot towards. By adding increment and decrement, it was possible to change the value of the row and col in order to match the direction according to its corresponding command. This was done for all 8 possible directions in which the robot could move. This is displayed by Figure 5.

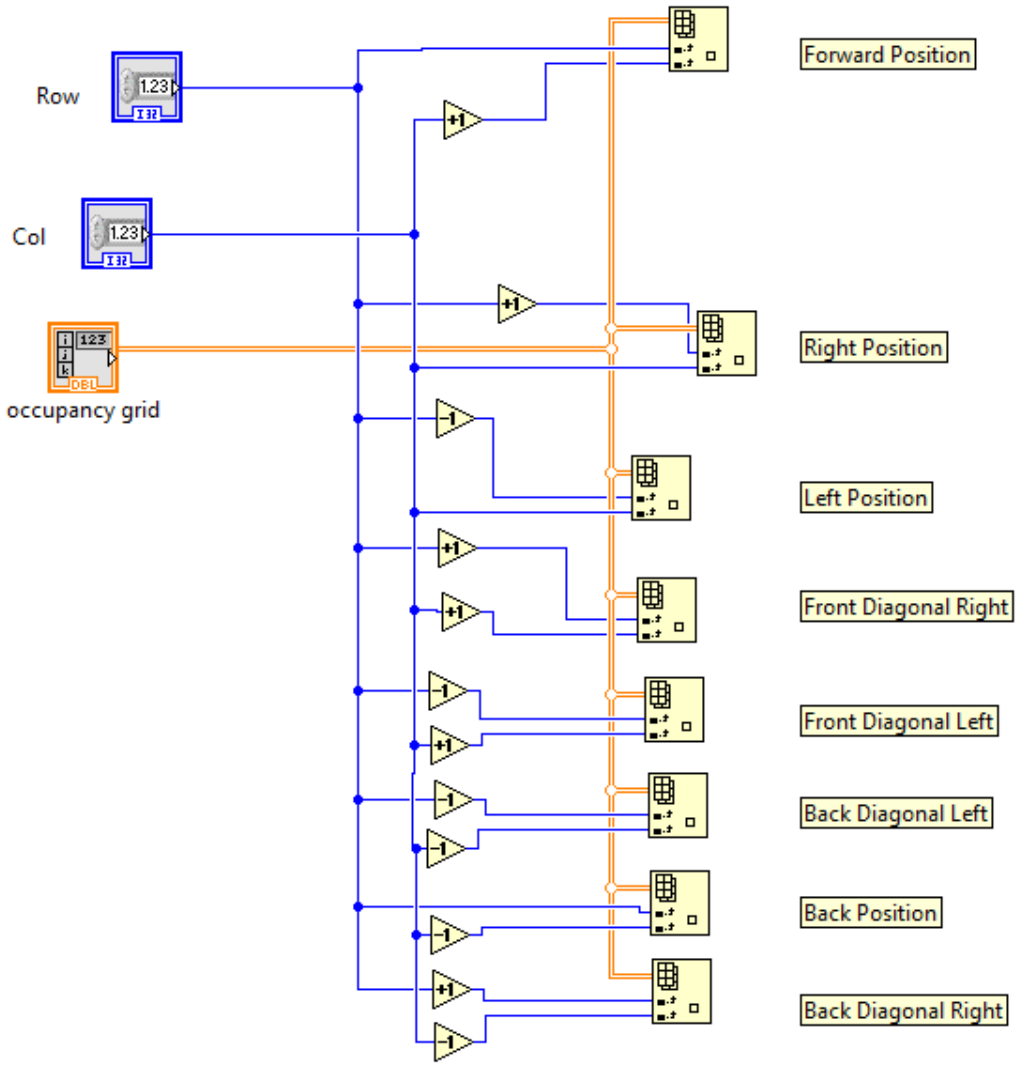

Figure 5. Occupancy grid based direction determination.

4. The following step was to create a control for the initialize reference input and error input for the write DC motors. Two indicators were created outside the case structure to hold the value for the next col and row. A flat sequence structure was placed inside a case structure. A DC motor write icon along with a timer icon was placed inside the flat sequence. This was done for the motor to move the robot forward. The array output for the forward command was input into this case structure. A new case structure was placed within the forward motion case structure. A flat sequence was also placed inside this new case structure along with a DC motor write icon. In this case, the motor values were set for the next robot motion which was to move left. The timer value was set from a previous value which was used in the meter square lab. The difference was that instead of 2 meters, the robot had to be calculated for 1 foot per block. The 90 degree timer was set to the same value and the 45 degree was half that of the 90 degree value. This was done for the remaining motions which totaled to 8 case structures within each other. This was done in follow the shortest path displayed in Figure 2.But due to existing obstacle constrains modifications to the path were made to follow a Forward-90 degree Right Turn-Forward -90 degree Left Turn –Forward-Forward motion path instead of the A star sub Vi provided Forward Diagonal Right- Forward Diagonal Right-Forward-Forward motion path. The actual path taken is shown by the white line whereas the proposed is shown by a red line in Figure 6.

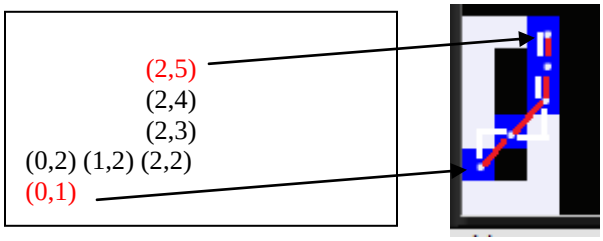

Figure 6.Path proposed & actual.

To accommodate for this change, cases to handles various inputs of -2,1 & 100 were devised and implemented as shown by figure 7. Thus the occurrence of 1 was used to replace the existing forward diagonal motion routine with a forward-90 degree right turn & forward motion .Whereas the detection of 100 was used to replace the existing forward diagonal motion routine with a forward-90 degree left turn & forward motion .These were as illustrated by Figure 8 & 9 respectively. The default case of 0 for the forward diagonal right is as shown in the section of the entire program as shown in Figure 10.This embedded casing can be seen in this figure. The execution case for the forward motion & its default case are connected as shown in Figure 11.

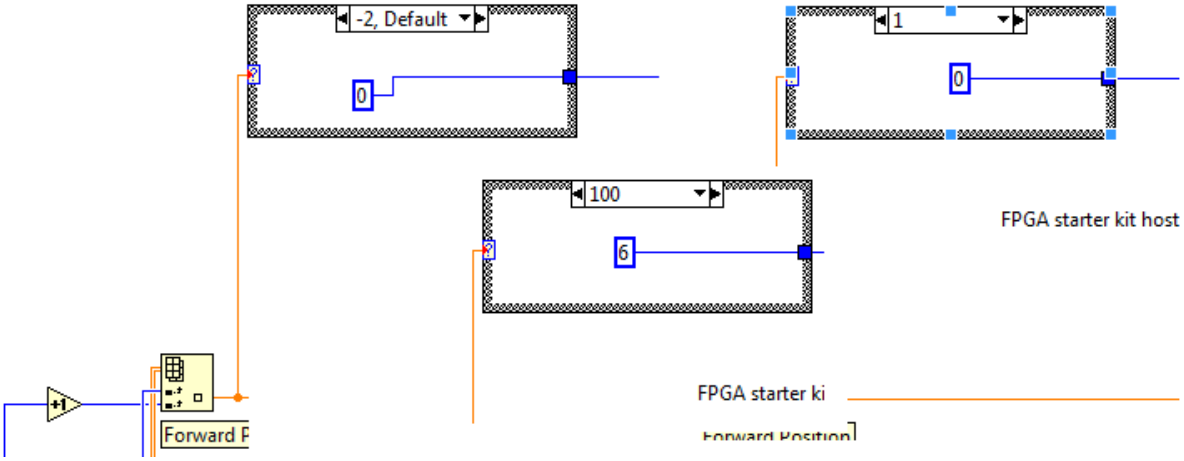

Figure 7.Cases to for various array values

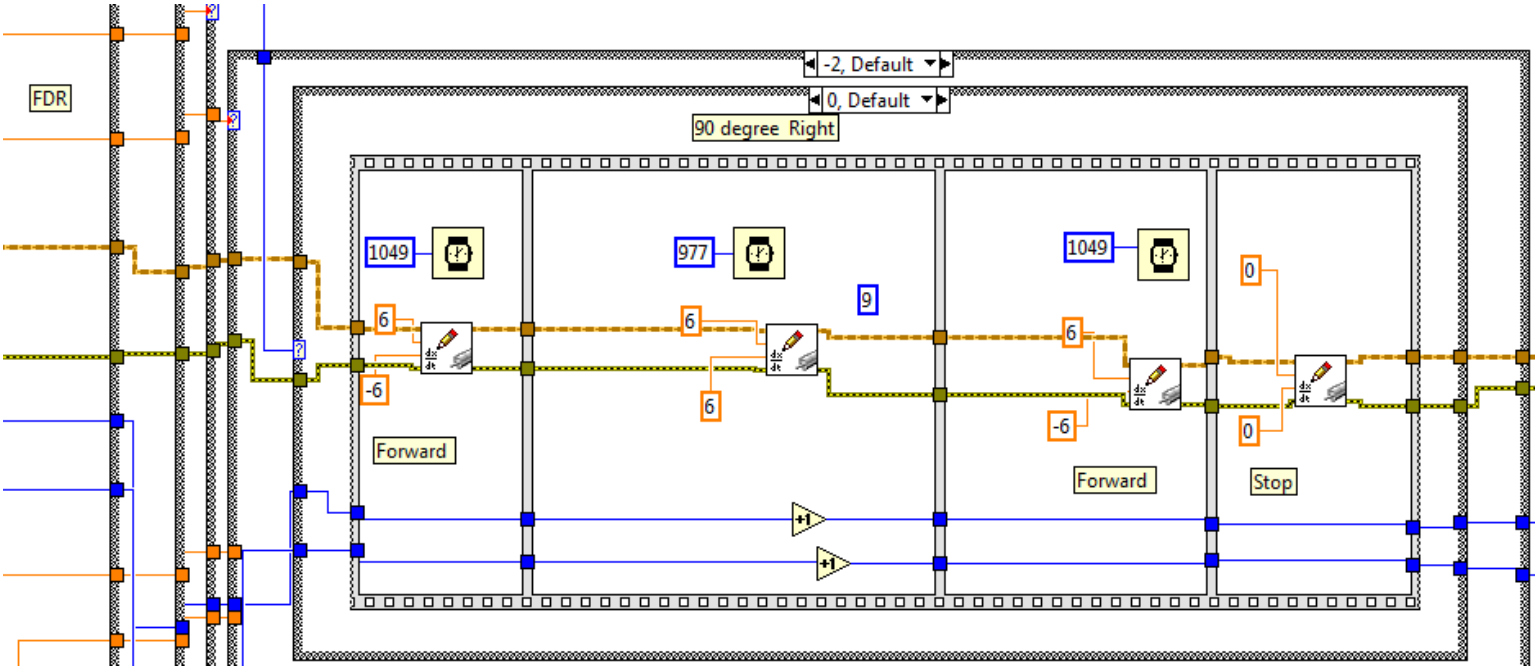

Figure 8.Case for occurrence of 1& -2.

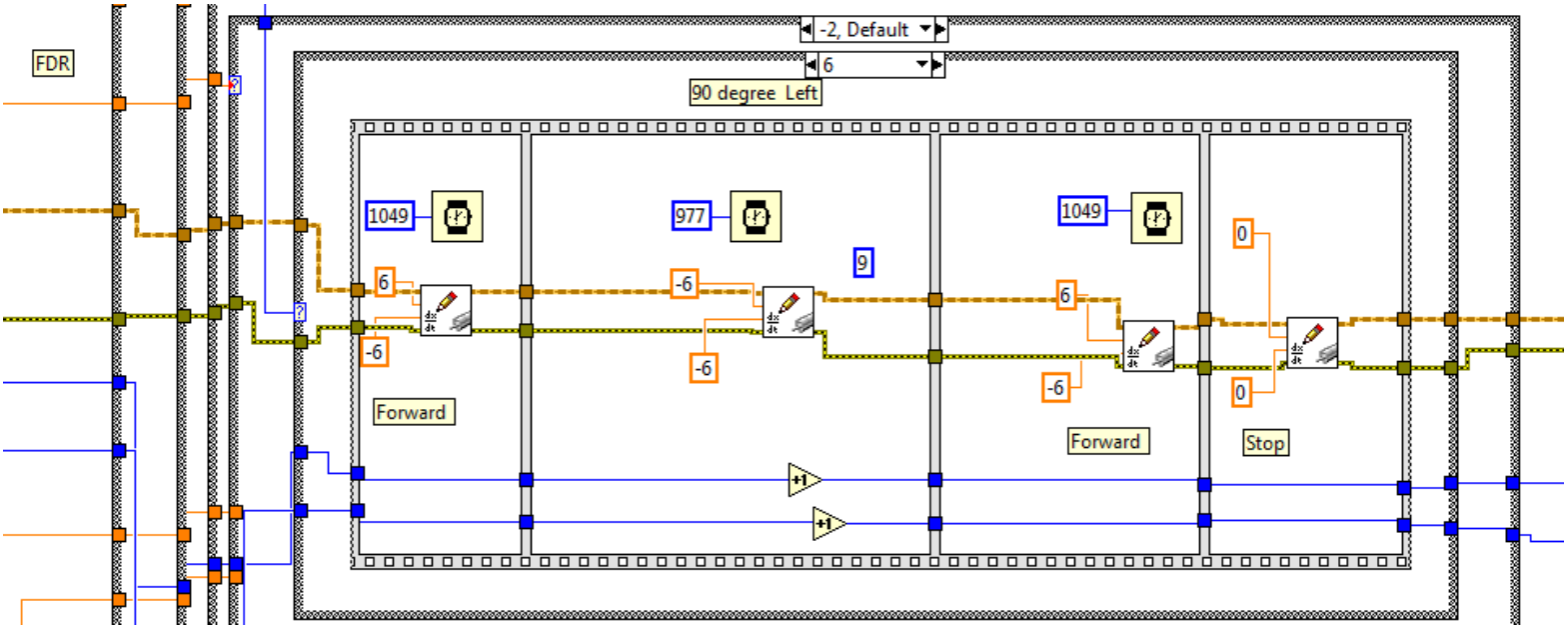

Figure 9.Case for occurrence of 100.

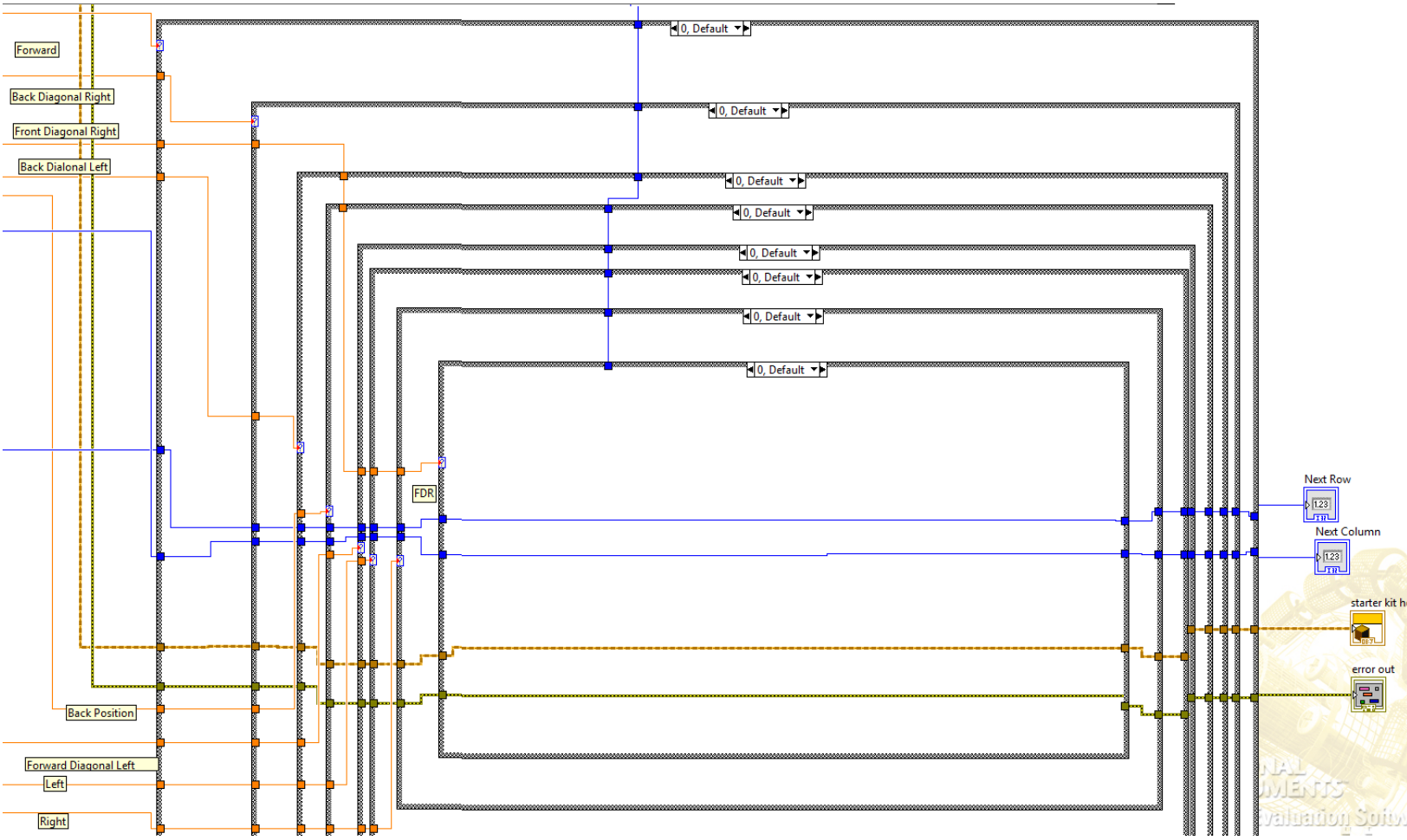

Figure 10.Section of the completed Determining direction VI .

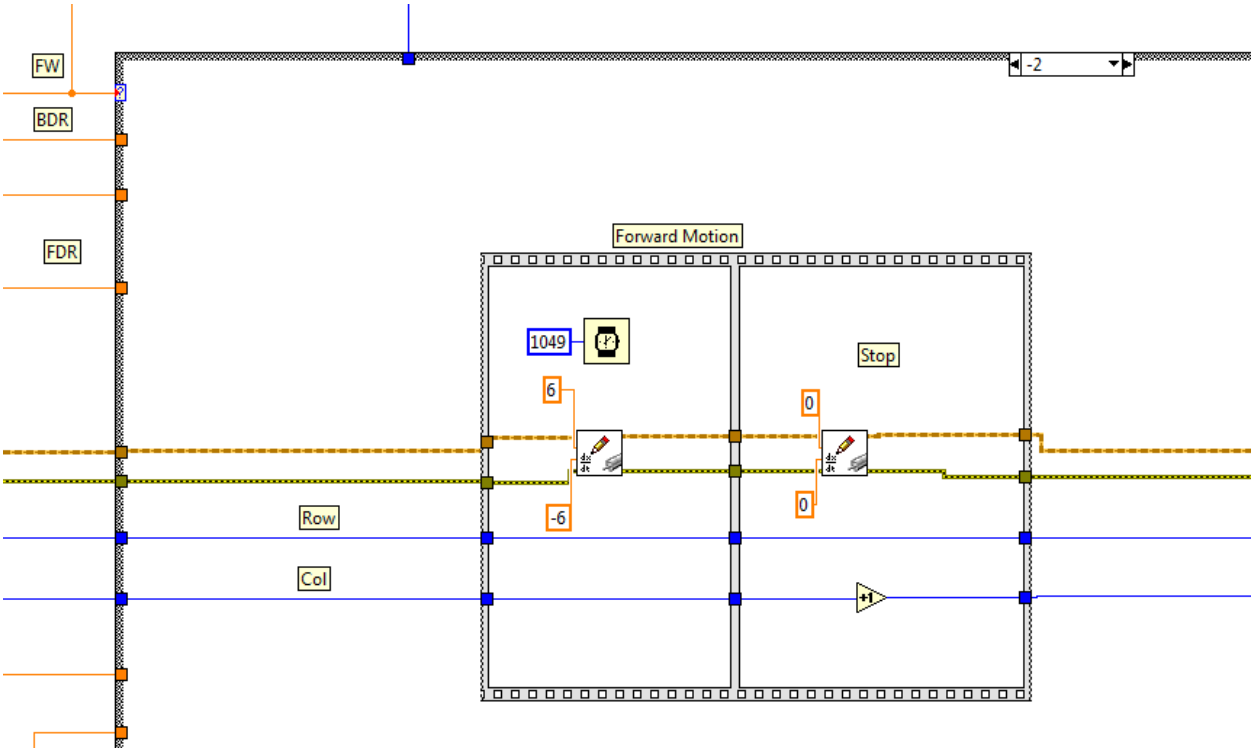

Figure 11.Execution stage for forward motion.

5. In the connector plane of the determining direction subVI components were connected. On the left side of the connector pane, the row, col, and array input were connected. The two middle connections were connected to the FPGA VI reference and the error out inputs. The next row, next col, FPGA VI reference and error out outputs were connected to the right side of the connector pane. Once this was done, the determining direction subVI was completed.
6. A new VI was made which was called “Implementation”. A flat sequence was used in order to step through of each of the steps needed in order to gather the number of blocks in the path and to move the robot to the desired position from a specified starting point. By copying the map from the AStar sample code and pasting it to the main VI, the program was made to gather information about the map. An initialize and map icon outside the flat sequence were connected to the DC motor write in one frame and a timer was connected on the following frame. In order to allow the robot to obtain the path it must take in order to complete its task AStar subVI was placed in the main VI in a following frame. The Map output wire was then inputted into this VI and an occupancy grid indicator was generated as a branch of the output wire from the VI.
7. Further to check the occurrence of -2, a case statement was embedded within two for loops to check the whole 36x36 array. If -2 were found then the constant pertaining to the loop was incremented. This is shown by Figure 12.

8. The completed Implementation Vi can be illustrated as per Figures 13 & 14.

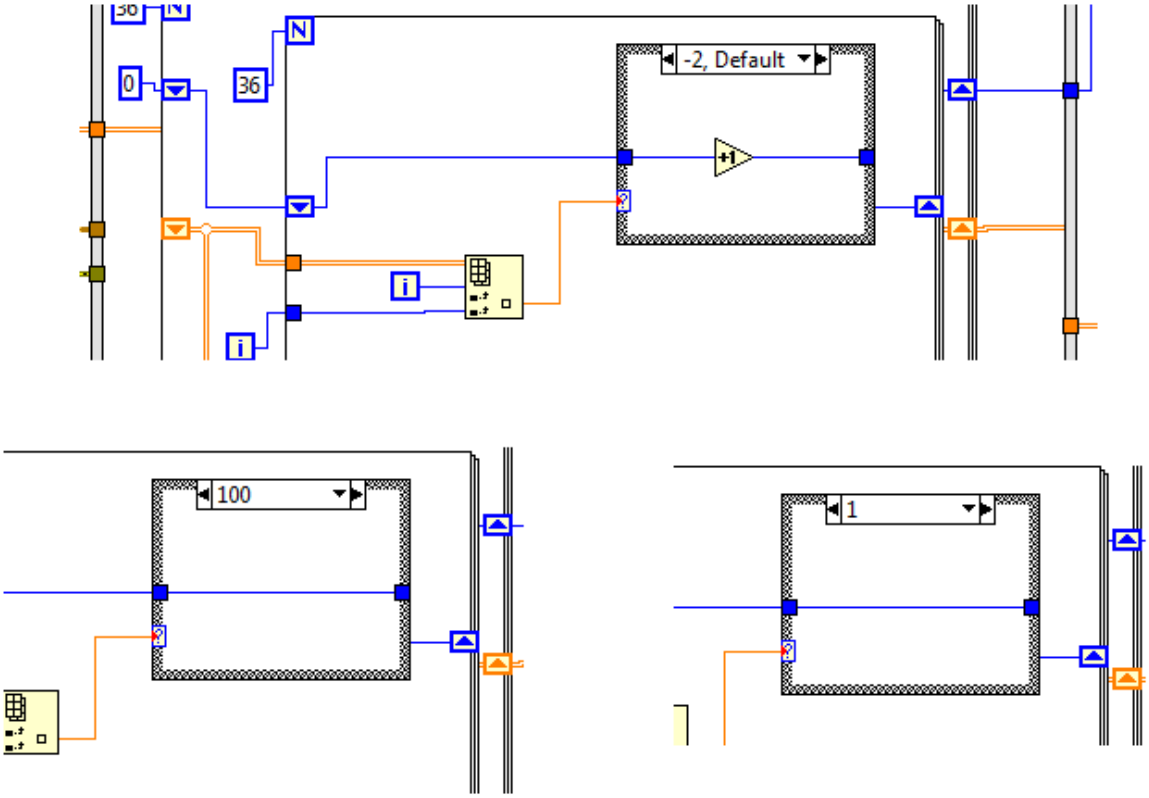

Figure 12.Case Structure which increments on finding -2.

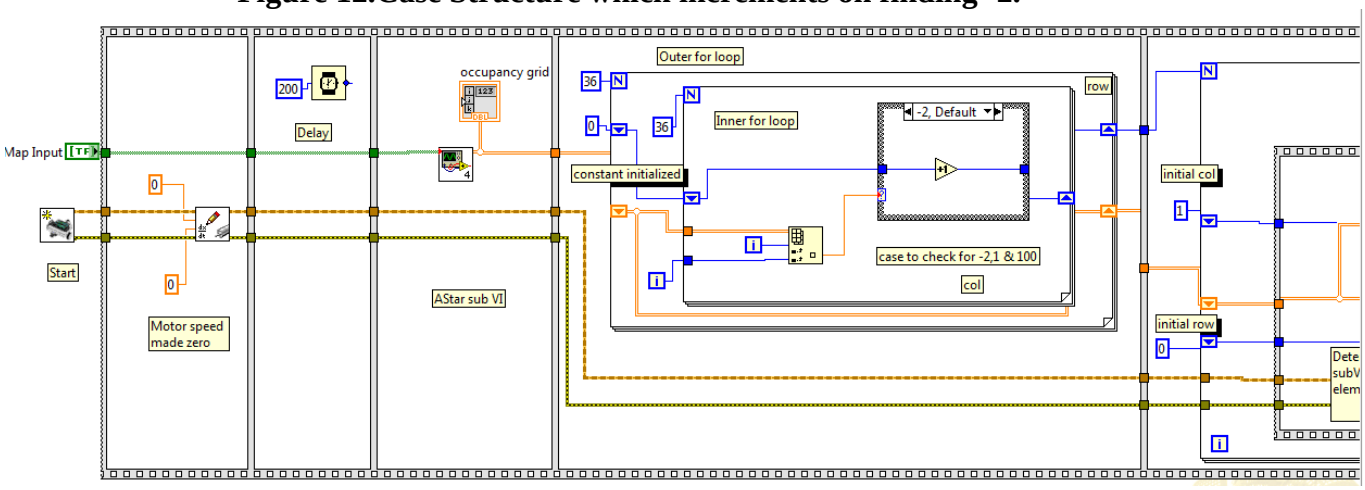

Figure 13.Completed Implementation VI part I

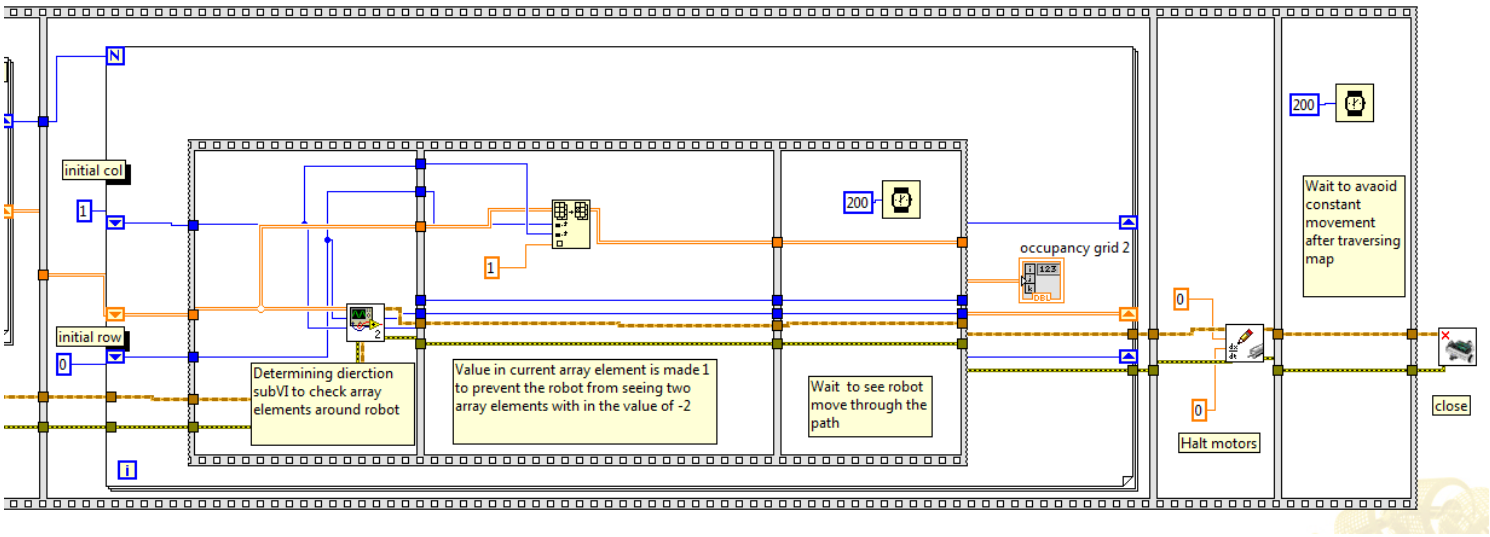

Figure 14. Completed Implementation VI part II

9. The program was later downloaded onto the sbRIO starter kit robot and it functioned by taking the forward motion followed by 90 degree right turn followed by a forward motion & then stopped. when it encountered 100 the algorithm proposed a forward diagonal right but the case statement logic chose the condition in which the robot moved forward followed by 90 degree left turn followed by a forward motion & then stopped .The remaining course of motion was traversed by calling the forward motion case (shown in Figure.11) twice.

Observations while completing/testing the Lab:

It was observed in the path finding algorithm did provide the shortest path for the given start and end points. Although, a path planning algorithm might provide the user with the best optimum path, such a path may not always be possible due to physical constraints. Modifications might have to be made to the proposed path as experienced in this lab exercise.
The path taken by the robot was starting from (0,1) to (0,2),(1,2),(2,2),(2,3),(2,4) & finally to (2,5) as shown in Figure 6.
Copying an icon incorrectly may cause it to be pasted as an image onto the VI. This error was observed when a wait icon was incorrectly pasted in the determining direction VI. The resolving of this issue was crucial to achieve proper functioning of the lab.

Aspects such as difference in drive to both motors, slipping, friction etc. can affect the proposed path and hence a provision to monitor the system behavior & having a closed loop feedback might prove useful.

Lessons Learned:

In this lab we learned the following concepts & implementations.

a. The use & implementation of A* algorithm.

b. The usage of Map as input & occupancy grid.

c. A proposed path may not always be the best optimal path and the user might have to make some modifications in the proposed path for path planning in real time.

References:

Lab 5 Assignment & Tutorial