## Lab 05: A* Path Planning – Tutorial

**Note:** The idea behind the implementation of this code is to use the sample code obtained from NI.com and add additional code that will use the output from that sample VI to navigate the map. This involved obtaining the array and setting the starting position of the robot. The array elements around the robot are analyzed and checked to see if a value of -2 is assigned to one of those blocks. (A -2 signifies that the block is a part of the calculated path. According to which block holds the -2, the robot will turn a certain degree, move forward, and turn back to a certain degree to come back to its original stance.

**Step 1**: Create a new project and title it as Lab5AStar. Download the file "A-Star simple example" from Ni.com

**Step 2**: Add the file "A-Star simple example" to the project by simply right clicking on sbRIO and clicking on add file.

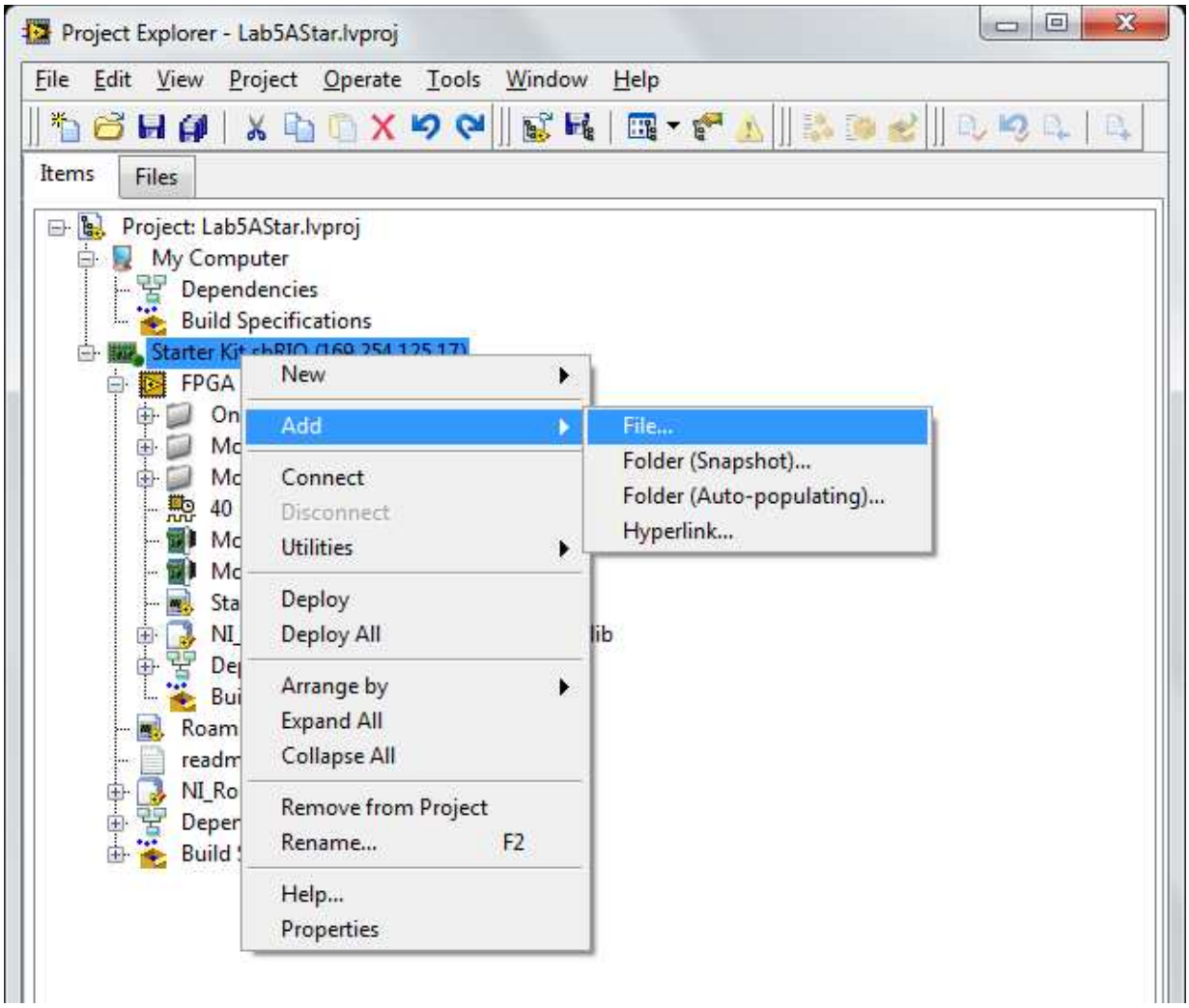


Figure 5.1

**Step 3**: We will now start to build the AStar subVI. Just in case, open the A-Star simple example VI, go to the block diagram, select all, and copy. Open a new VI and paste the copied code to the block diagram.

**Note:** To create a subVI, the connector pane must be the same for all other subVIs used in the main code. Also, connections must be assigned to either control elements or indicators in the code.

**Step 4**: Right click on the image shown in the top right and click on show connector to see the connector pane for the VI. This will already be visible in LV 2011.)

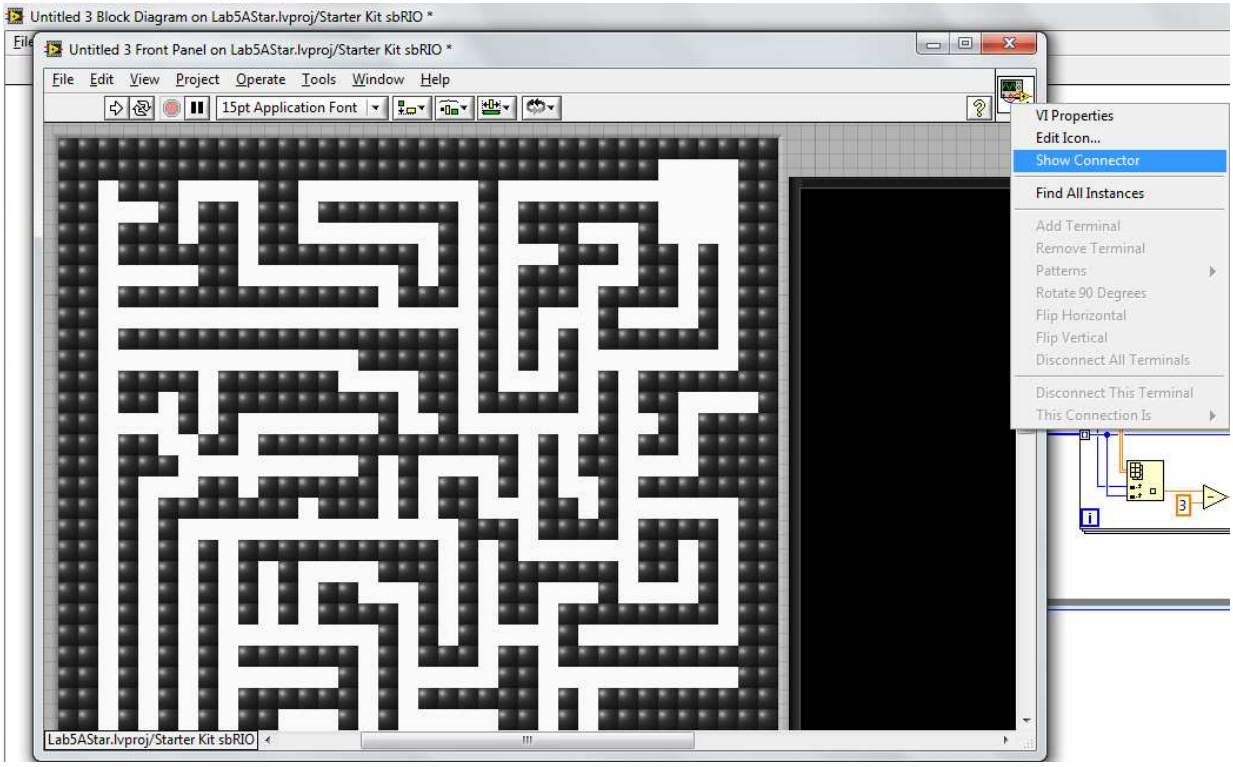


Fig. 5.2

**Step 5:** The connections then need to be made. All inputs for the VI are shown on the left and all output connections are shown on the right side. Connect the map as an input and the array on the front panel as the output.

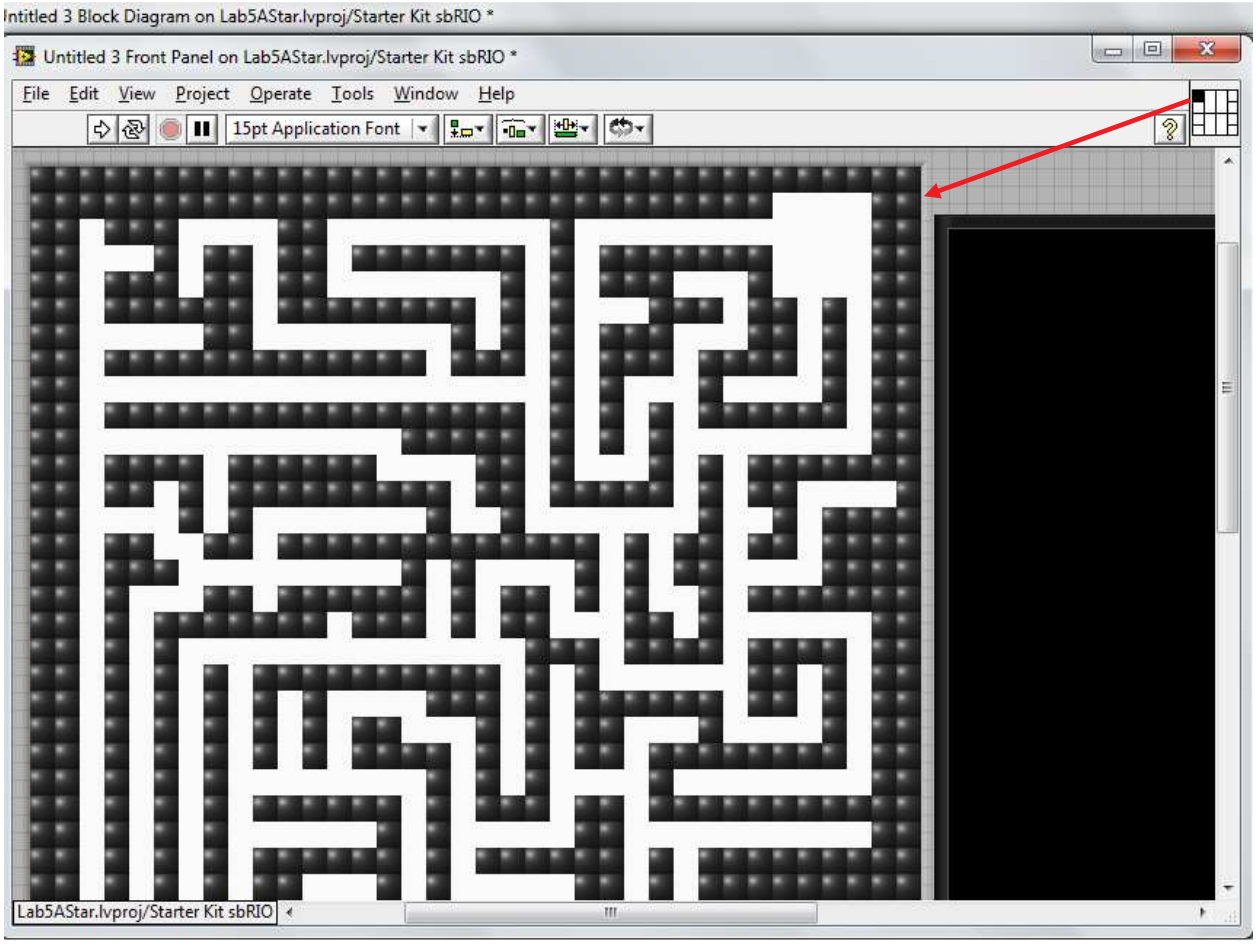


Fig 5.3

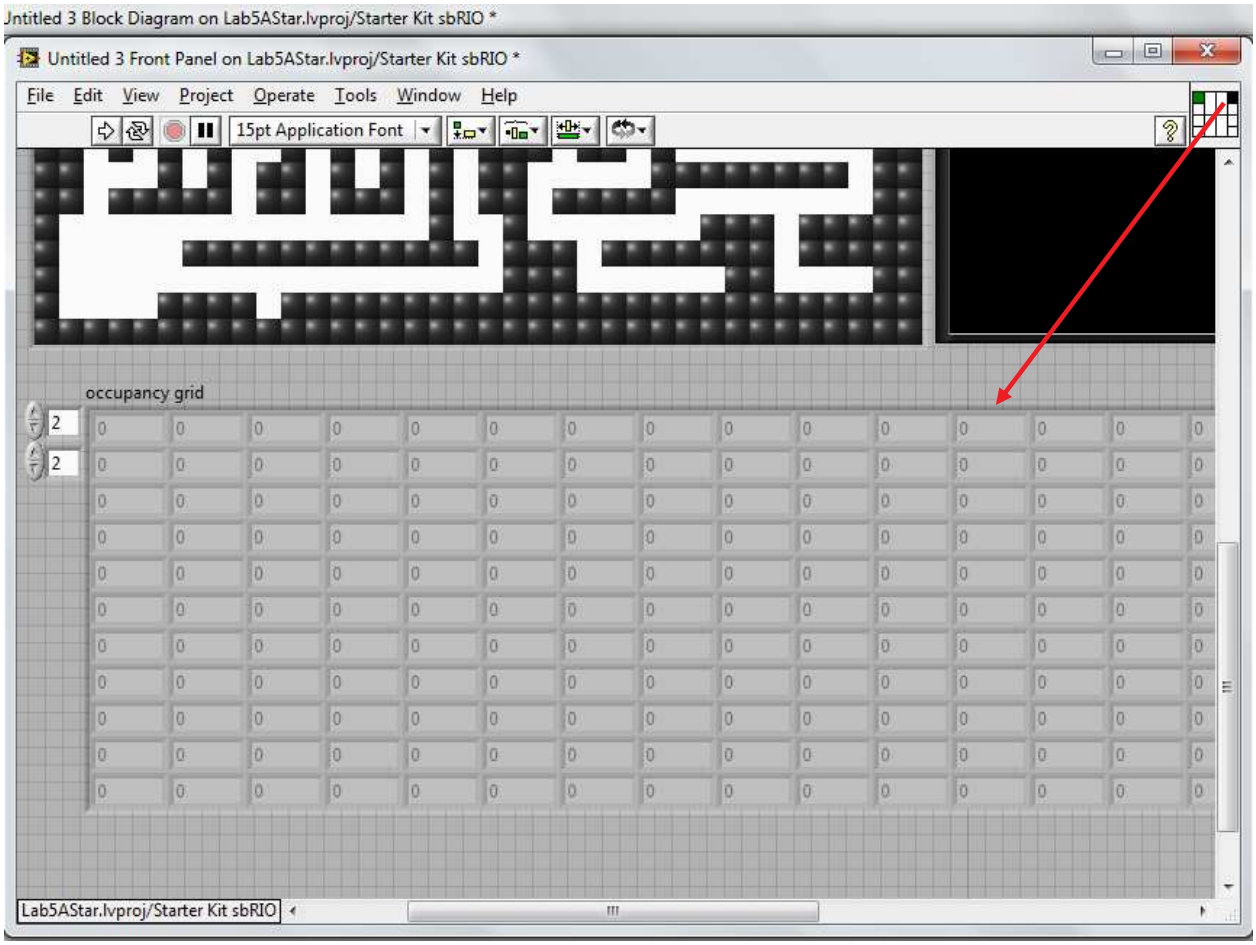
Untitled 3 Block Diagram on Lab5AStar.lvproj/Starter Kit sbRIO *
Untitled 3 Front Panel on Lab5AStar.lvproj/Starter Kit sbRIO *
File Edit View Project Operate Tools Window Help
15pt Application Font
occupancy grid
2
2
Lab5AStar.lvproj/Starter Kit sbRIO

Fig 5.4

**Step 6**: To finish up with this subVI, select the timed loop and right click on it. Then select remove timed loop.

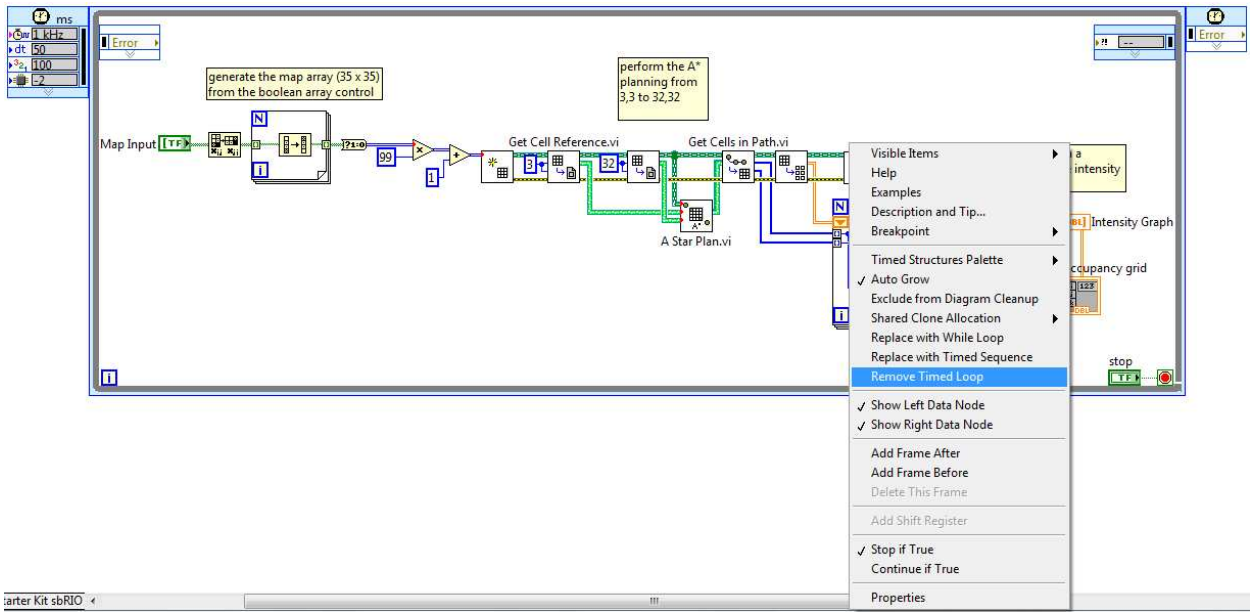


Fig 5.5

**Step 7**: Before closing the AStar subVI, create a new VI and title it “Determining Direction.” This VI will implement the idea behind implementing the A* algorithm. This VI will look at all array elements around the robot’s position. Thus, copy the occupancy grid indicator that shows the resulting array on the front panel of AStar subVI.

**Step 8**: To start building the new VI, create two numeric constants and paste the occupancy grid indicator that was copied earlier. Make these constants and indicator controls for the VI by right clicking each and selecting “make control.”

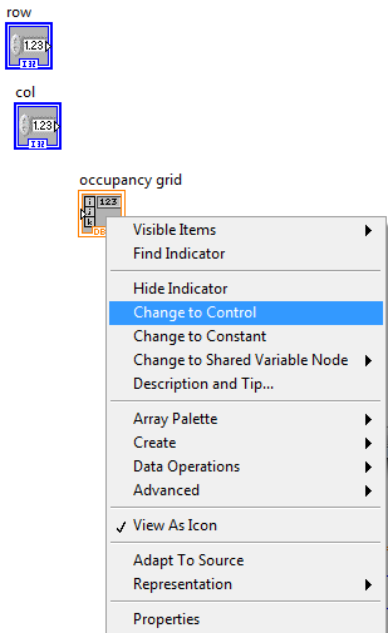


Fig 5.6

**Step 9**: The arrays in LabVIEW are a bit different from other languages in the fact that the rows and cols appears to be reversed. A 2-D array will have what appears as rows to be cols and what appears as cols to be rows. By analyzing these rows and cols around the robot, you can see how each position will correspond to the current row and col as shown below.

| | | |
|---|---|---|
| Front Diagonal Left<br>Row -1<br>Col +1 | Forward position<br>Same Row<br>Col +1 | Front Diagonal Right<br>Row +1<br>Col +1 |
| Left position<br>Row -1<br>Same Col | | Right position<br>Row +1<br>Same Col |
| Back Diagonal Left<br>Row -1<br>Col -1 | Back position<br>Same row<br>Col -1 | Back Diagonal Right<br>Row +1<br>Col -1 |

Fig. 5.7

From this knowledge we will set up a check for each of these positions. Place an index array subVI, which can be found under the programming template and under the array template. When first placed, it will only allow a row input but expanding it vertically will show a col input as well. Based on what position we check we will either increment or decrement the row or col input value. The outputs of these index array subVIs will be used as the inputs to the case statements made later on.

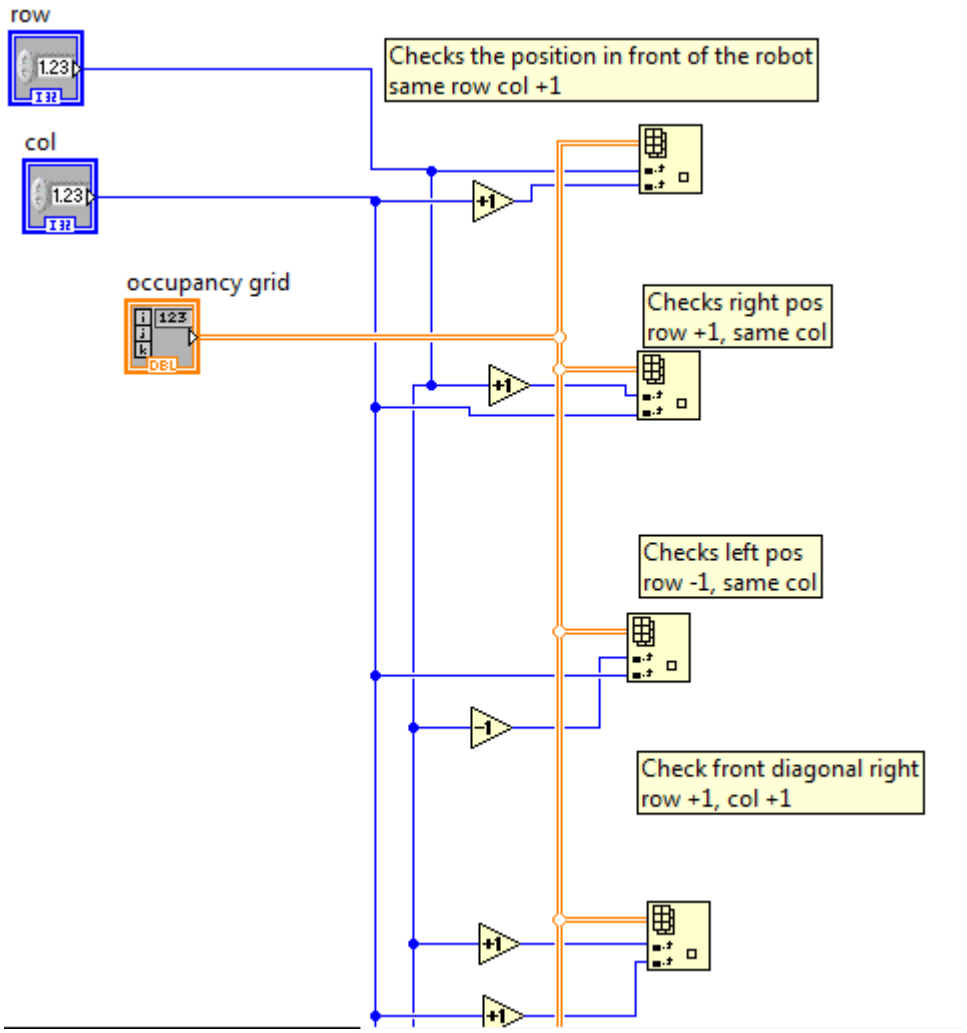


Fig. 5.8

**Step 10**: The case statements will then be made. If a -2 is found from its corresponding index array subVI output, then it will execute the code to make the robot move accordingly. If not found, then the case statement will then go to another case statement that is inside and check again. This is done again and again until all checks have been made. Since we will be using the write DC motor VI to write values to the FPGA to set the velocity setpoints, we will create a control for both the initialize reference input and a control to the error input for this write dc motors.

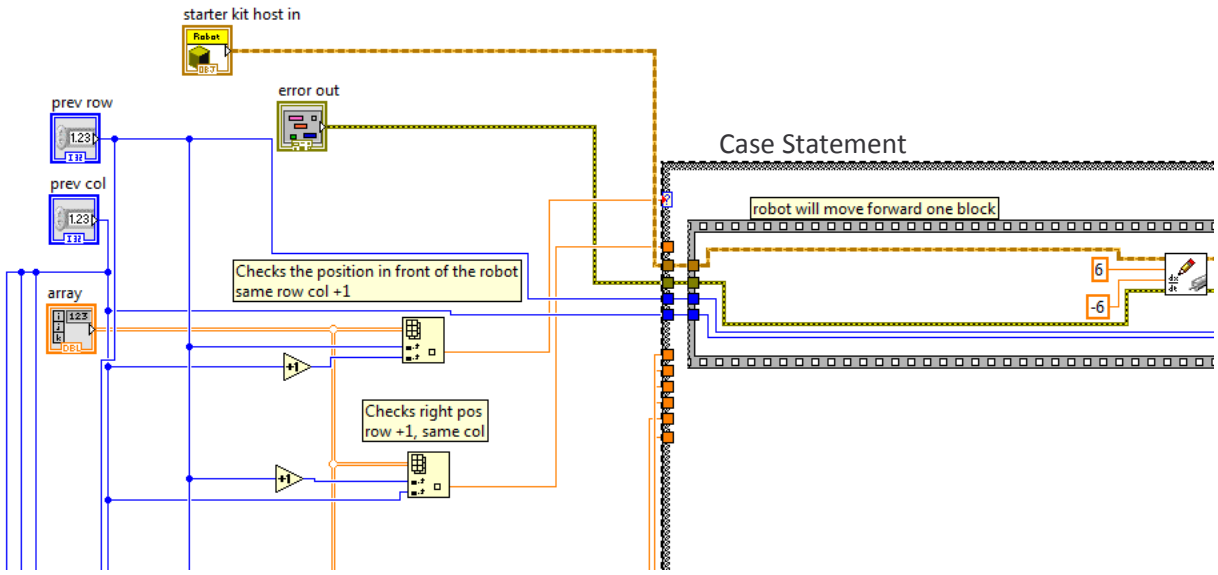


Fig. 5.9

**Step 11**: Like in the figure above, inside the -2 case for each case statement values will be written to the FPGA depending on which direction the robot will move. The time depends on how big the squares in the map are set up to be. For this map, the squares were set up to be 2 feet by 2 feet and the values were calculated using the equations from the meter square project. At the end of the flat sequence that handles how the robot will move, the row and col values are changed by either incrementing or decrementing them depending on which position the robot is moving to. Create two indicators outside of the case statement that will hold the values for the next row and next col. Two indicators must also be created for the FPGA VI reference and error out.

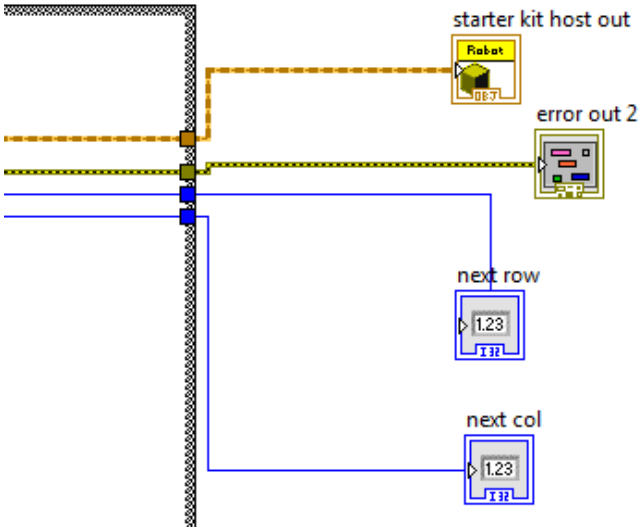


Fig. 5.10

**Step 12**: To finish up this subVI, the connections to the connector pane must be done. The left side will connect to the row, col, and array input. The two middle connections can be used to connect the FPGA VI Reference and error out inputs. Only the right side connections are considered to be outputs and will be used to connect the next row, next col, FPGA VI Refernce, and error out outputs.

**Step 13**: Now we are ready to make the main VI. Make a new VI and title it "Implementation." A flat sequence will be used to step through the process involved in obtaining the number of blocks in the path and then moving the robot in the appropriate directions. Also like in the AStar sample code, a map will be needed to act as an input. Thus, copy the map with black and white blocks from the sample code and paste it on the front panel of the Implementation VI. As a precautionary measure, we will set the values of the velocity to zero and make it wait for a a small amount of time (2ms) since a delay was sometimes seen in the first turn the robot made. Making the appropriate connections it should look like this:

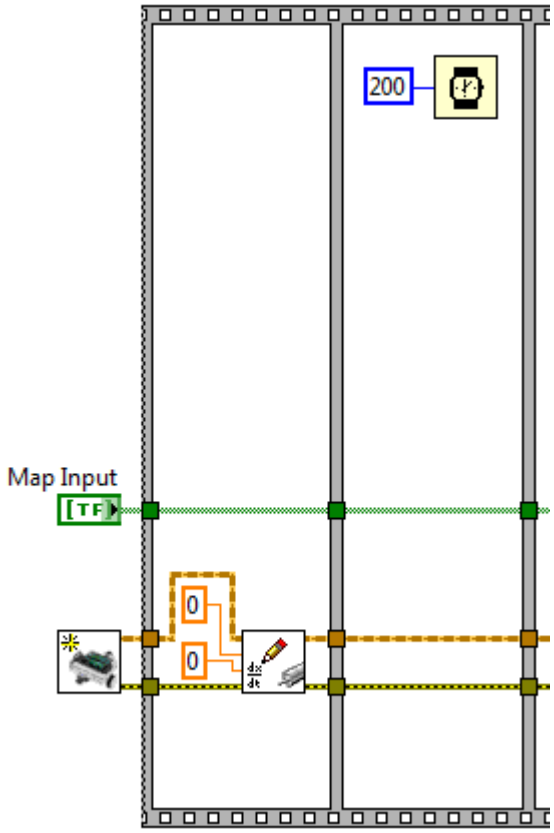


Fig. 5.11

**Step 14**: The next step is to obtain the path that the robot will take. Depending on how your map looks and the settings that are in your AStar subVI like the starting and end points, the path may be different. To select a VI to use as a subVI, right click anywhere on the block diagram and select "select a VI…" a dialog box will then appear allowing you to browse your directories and select a VI. Select AStar subVI and since we have made the connections to the connector pane, the VI will have the inputs and outputs

we have selected for it once we place it on the block diagram. The map input is then passed to this subVI and an output array of the map with the path in it is obtained.

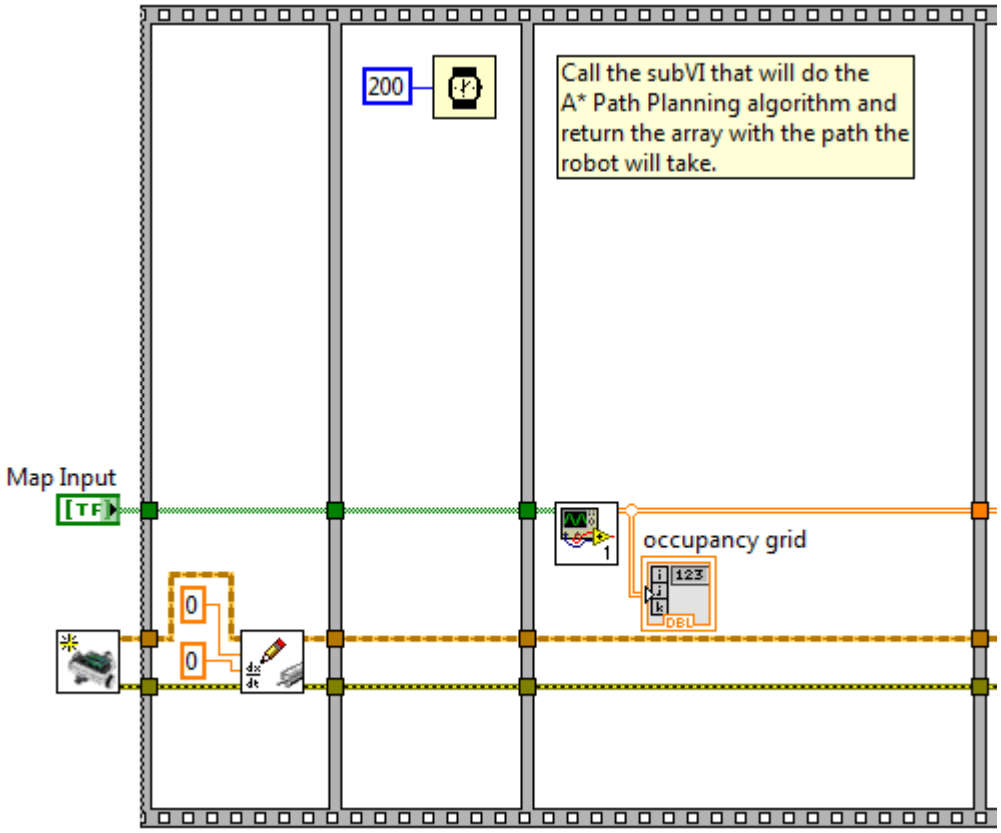


Fig. 5.12

**Step 15**: Now that we have the array with the path in it, we will search the array to see how many blocks are in the path. To do this, two for loops are needed. In the second for loop, A case statement is used to check if the block contains a -2 like how we checked in the Determining Directions subVI. The for loops can be auto-indexed but for these for loops we tied the value 36 to N because the 2-D array is a size of 36 x 36. A numeric constant is also created that is first set to zero. This will be used in the case statement when a -2 is found in one of the blocks in the array. If a -2 is found, then the constant is incremented. The end result should appear like this. The shift registers are used to pass values from previous loop iterations to the next iteration.

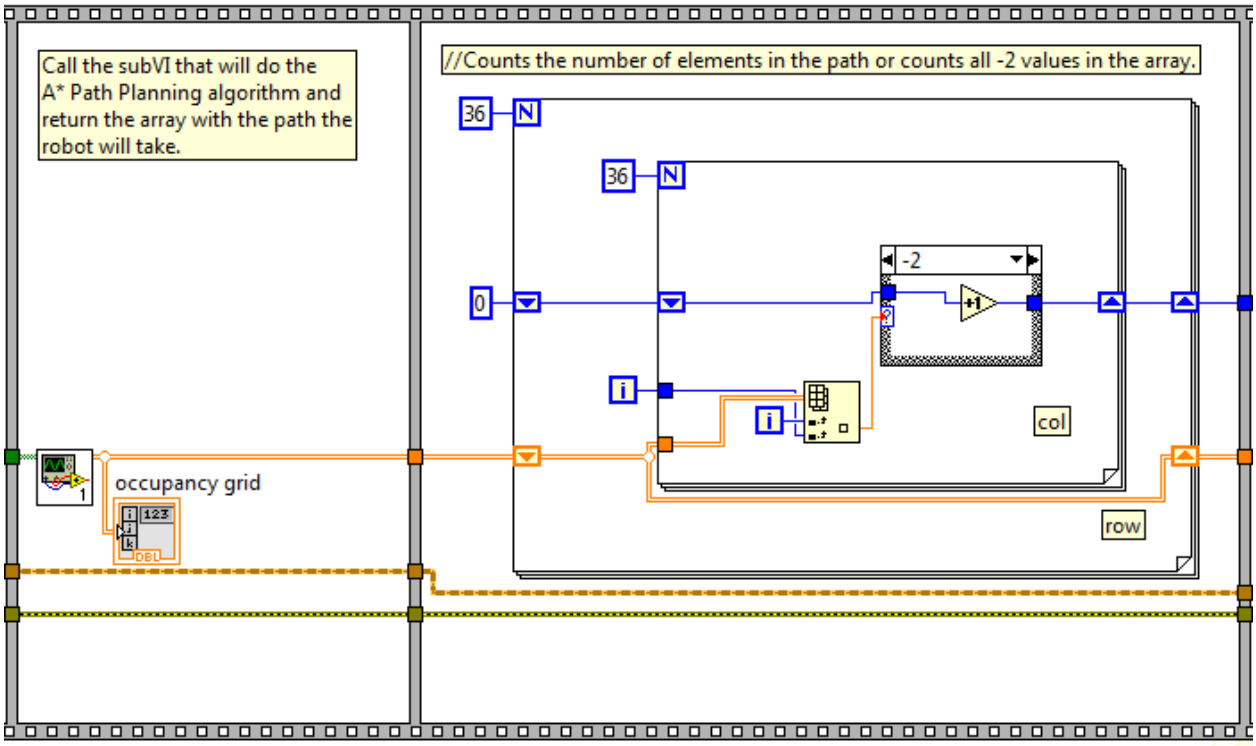


Fig. 5.13

**Step 16**: The last segment of code involve taking the number of blocks found in the path and connecting it to the N value in another for loop. This for loop will have two numeric constants on the outside of the loop that will be passed to the for loop. These constants will act as the initial row and col where the robot is first located. Shift registers will be used to pass these values to the next loop iteration.

**Step 17**: A flat sequence will be placed in the for loop that will call the Determining Direction subVI. The array with the path must then be updated to avoid seeing two blocks when checking the array elements around the robot by setting the current position's value to 1 using the replace array subset subVI which can be found in the same place the index array subVI was found. To see the robot move through the path, a wait statement is used and set to 200ms. The resulting steps look like this:

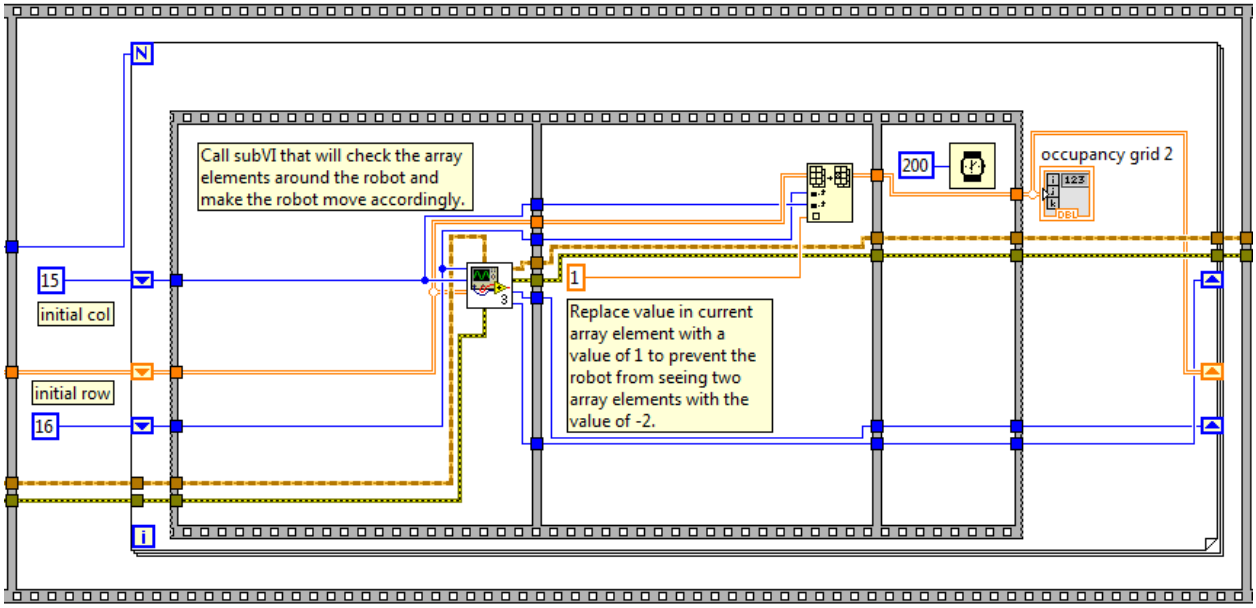


Fig. 5.14

**Step 18**: To prevent the robot from constantly moving after closing reference, the velocities are set to zero and then a wait statement is used. After these last blocks in the flat sequence, there is a close statement and we are done with lab 5.

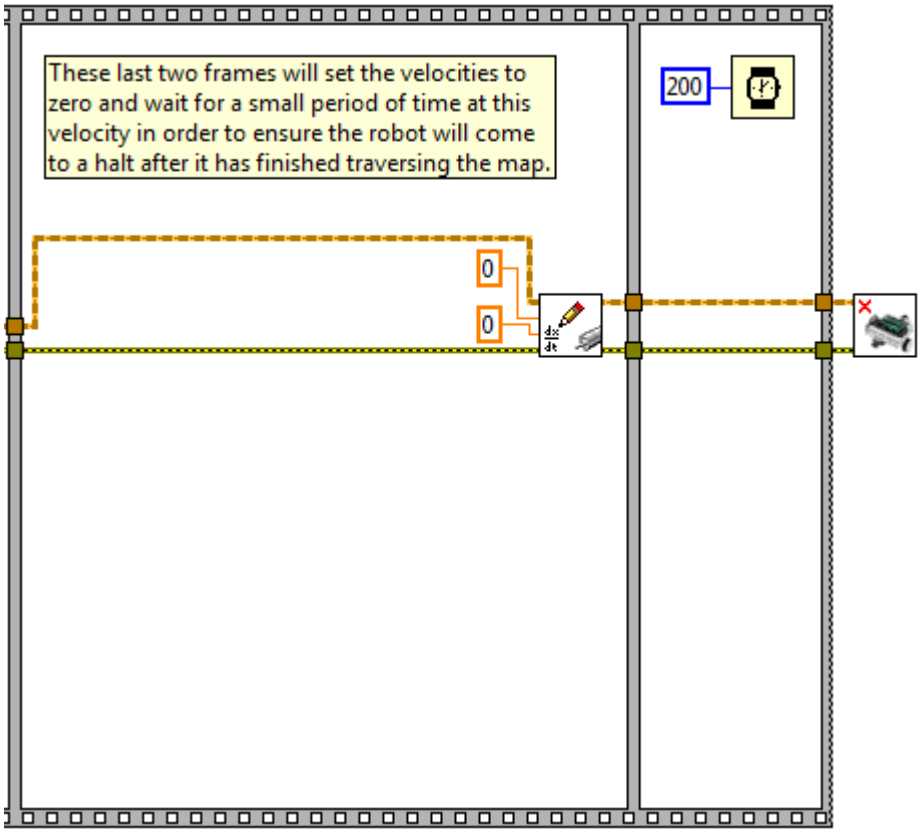


Fig. 5.15