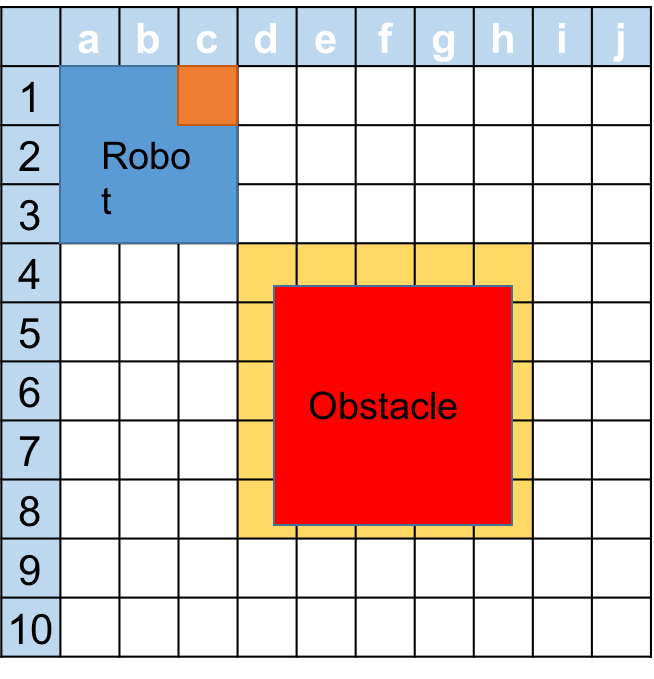
**Movement Part:**  
The basic idea of movement is to regard the site as a x\*y array, each grid is 5.5\*5.5cm, our robot will in a 3\*3grid every time (orange part of the robot is the sensor):



Then the robot will survey one grid for color and one grid for obstacle each time. For example, if our robot starts at ac13 (which means in the area from a-c and 1-3). Only **c1’s color** and **d1’s obstacle** can besurveyed**.**

**Obstacle/Faultline/Border detection:**

If there is an Obstacle like the figure shows. Our robot will mark all yellow part as Obstacle. Then our accuracy of the map depends on the size of every grid in the array. The the size of grid depends on the width of the ULtrasonicSensor. **The width I get from test is 5.5cm.**

**Function:**

**Regular moving functions:**  
Void forward() //move forward one grid

Void turnleft() //turn left 90, don’t move, stay in the 3\*3 grid

Void turnright() // turn right 90, don’t move, stay in the 3\*3 grid

When the robot turn right, sensor will keep decrease, -90, -180, -270…

When the robot turn left, sensor will keep increase, 90, 180, 270

Every time restart the sensor, the value will be reset to 0.

**Emergency moving functions:**

Void backward() // move backward one grid, face the same direction

Void stop() //emergency stop

**Automatic control:**

1. **Location**

The location of our robot will be marked as the colour sensor’s location, for example, as the figure shows, the location of our robot will be c1, if the robot turns left, the location of it will be a1.

1. **Survey**

Survey the site line by line. If there is an obstacle (as the figure shows), the robot reaches a5, and then turn to the side which has been mapped to avoid the obstacle. And the robot will turn to the obstacle every grid to detect the size of obstacle.

For example, if robot starts at a1, and go through like a1-b1-c1…f2-e2-d2...

When robot reaches a3, it will detect that there is an obstacle at b3, since line 2 has already been mapped, the robot will

1.mark there is an obstacle at b3,

2.turn right and go to a2, then turn left and move to b2,

3.move to c2 and detect if there is an obstacle at c3, if there is, move to d2, else move to c3, doing this loop until robot go back to line 2 or reach the border.

If there is no more move to let our robots get to a new line, System will navigate the robot to the reachable leftist side of a new line. (for example, if the robot has mapped the line1 and line2 and stuck at d2 because it is surrounded with obstacle, then it will go to a3 and start survey again.)

1. **Navigation**

In this part, our core function will be use BFS algorithm to find a path based on the map information. And here is a question that if we regard our robot as one grid on the map, we may get a path which we cannot pass, for example: if we want to move our robot from c1 to j10, we cannot find a path while our algorithm will give us a path like the grey path show:



so in Navigation, we first merge every 3\*3 grid together as one large grid, if there is an obstacle in one grid, we regard the large grid as an obstacle. Here will cause another problem that we may cannot generate a path we should do. **This problem will be fixed as a extension requirement.**

**Navigation** from known site to known site (use manhattan distance here)

Use BFS algorithm to find a path, and use regular moving functions to go the Destination.

1. Plan the path, show the result on GUI, if cannot found path to there, return “Can not find a path” on GUI.
2. Showing on GUI the position robot should be at that time as well as what the robot should do (like turn right, turn left, etc). So that operator can fix error caused by motor.
3. When finish the navigation, return “finish” on GUI.

**Navigation** form known site to unknown site (use manhattan distance here)

Use BFS algorithm to find a path to nearest grid to unknown site. And then use survey function to reach the unknown point.

**Navigation** form unknown site to unknown site

Can not support this function

1. **Return**

Operator will tell robot which corner to go back. And use navigation function to go back to that position. Since the robot can go into the container in every direction, so we don’t need to worry about the direction.