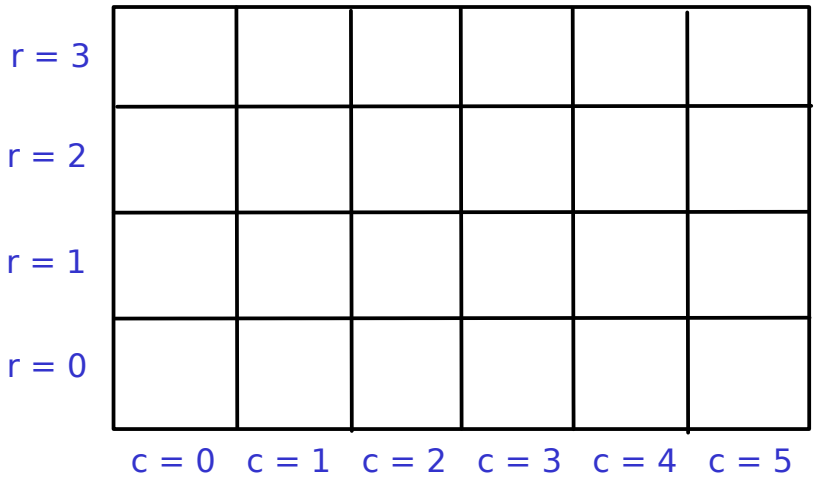
Final Project

Maze Robot (Might be able to finish in 3~4days)

We must number the maze as 2 by 2 matrix
Measures how much time it took per cell



Constraints

- Size: 4 x 6 Grid
- Walls on the outside
- No islands
  (each walls are connected to the outer wall)
- Only one solution to the maze

Required

- Start from a given cell (at the demo start entered using brick buttons), facing North (We can define)
- Target cell is given at the demo start/beginning
  Entered using brick buttons
- Robot should autonomously navigate from start to end cells, saving, wall data in a 2D array of structures
- Robot return to where is started

Mechanical Constraints

- Small size, reasonable sound (for the placement height of the ultrasonic sensor)
- Gear ratio for accurate turning
- Axis of rotation at robot center
- Wheel size (smaller, wider is better)
  smaller (accuracy) wider (more friction to prevent skidding)
- To make the robot do exact (accurate) turning, bump into the wall (flat beam at the front)
- Symmetric (Exactly same on both sides)

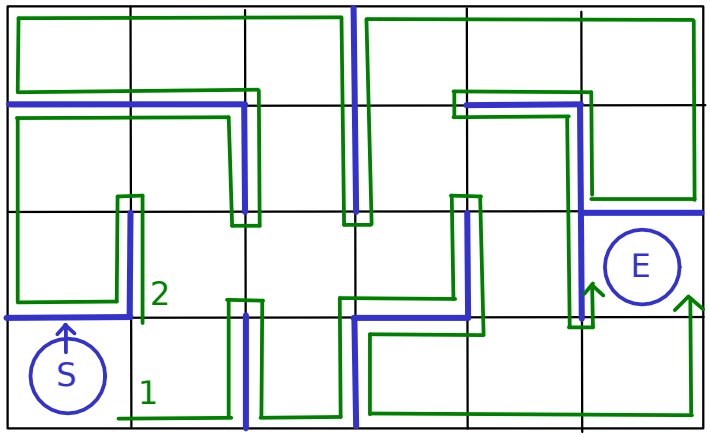
Seonsors

- Encoders
- Ultrasound to detect walls
- Touch sensors
- Never use gyroscope (Will have more skidding)

Example)

1. Stick to the right side

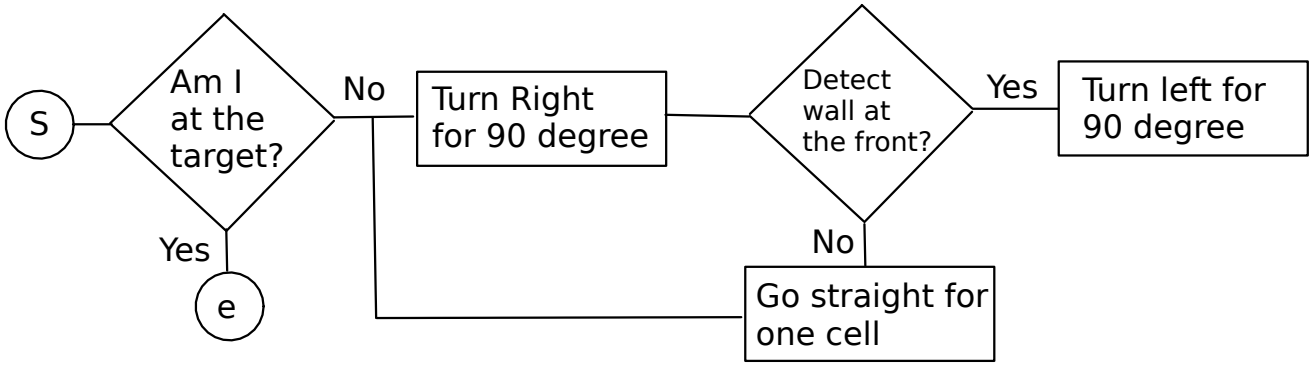
2. Stick to the left side



Softwares

- Flowchart

  Right wall follower (Priority= RFLB)



- How to go back

  1. Using coordinate system (row, col)

  2. Using directions {0 = North, 1 = East, 2 = South, 3 = West}

     We have saved how we moved, using 0 1 2 3

     We want to change 0 > 2 1 > 3 2 > 0 3 > 1

     We cancel if it is has 0 and 2 together or 1 and 3 together

     And go back (if it was 0 1 2 3, change to 2 3 0 1 then move 1 0 3 2)

  Ex) Left wall follower

  - We went

  E N W S N E S E N N W W E E S S E N N E E S N W W S E S S E N

  1 0 3 2 0 1 2 1 0 0 3 3 1 1 2 2 1 0 0 1 1 2 0 3 3 2 1 2 2 1 0

  - We cancel one and scan from the beginning and re-cancel

  1 1 1 0 1 2 2 1 0

  - We change 1s to 3s, 3s to 1s, 0s to 2s, 2s to 0s

  3 3 3 2 3 0 0 3 2

  - We go back in the opposite order

  2 3 0 0 3 2 3 3 3 (this way)

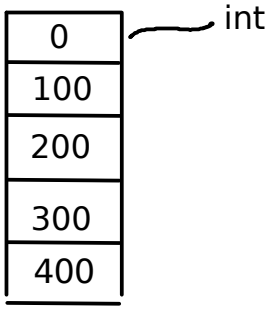
1D array in C

```
// Array with size 5
int MyArray[5];

// Index starts at i = 0
for (int =0;i<5;i++) {
        MyArray[i]=i * 100;
}

// Reading the rray
int x = MyArray[3];
// This will read 300
```

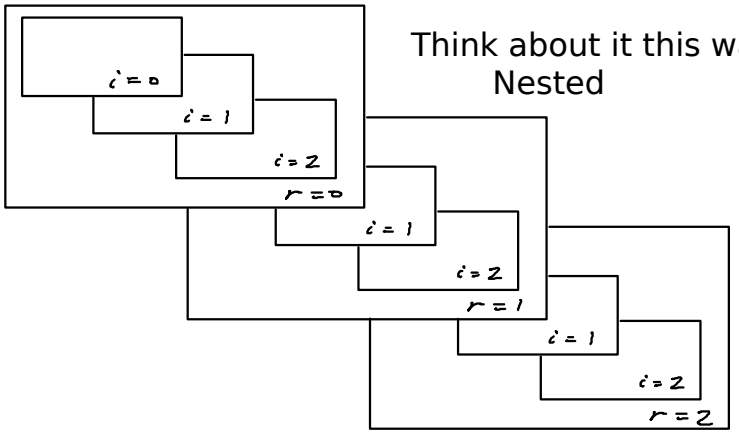
We want array called MyArray to save up these values



2D Array in C

We want this matrix

100	200	300
400	500	600
700	800	900



Think about it this way
Nested

Nested Loops

```
int MyArray[3][3];
int k = 100;

for(int r=0;r<3;r++){
        for(int c=0;c<3;c++){

                MyArray[r][c] = k;
                k = k + 100;
        }
}

// To display
int x = MyArray[1][2];
// This will print 600
```

k	r	c	MyArray[r][c]
100	0	0	[0][0] = 100
200	0	1	[0][1] = 200
300	0	2	[0][2] = 300
400	1	0	[1][0] = 400
500	1	1	[1][1] = 500
600		1	2 [1][2] = 600
		.	
		.	
		.	