# Line-Maze Solver Robot

Hardware Graduation Project

Prepared By
Wesam Abu Shunnar
Mohammad Asaad

Supervised By
Dr. Aladdin Masri

# Outline

- Introduction
- Project tools
- Progress of project
- Limitation of Project
- Future work
- View a demo for the project

# Introduction

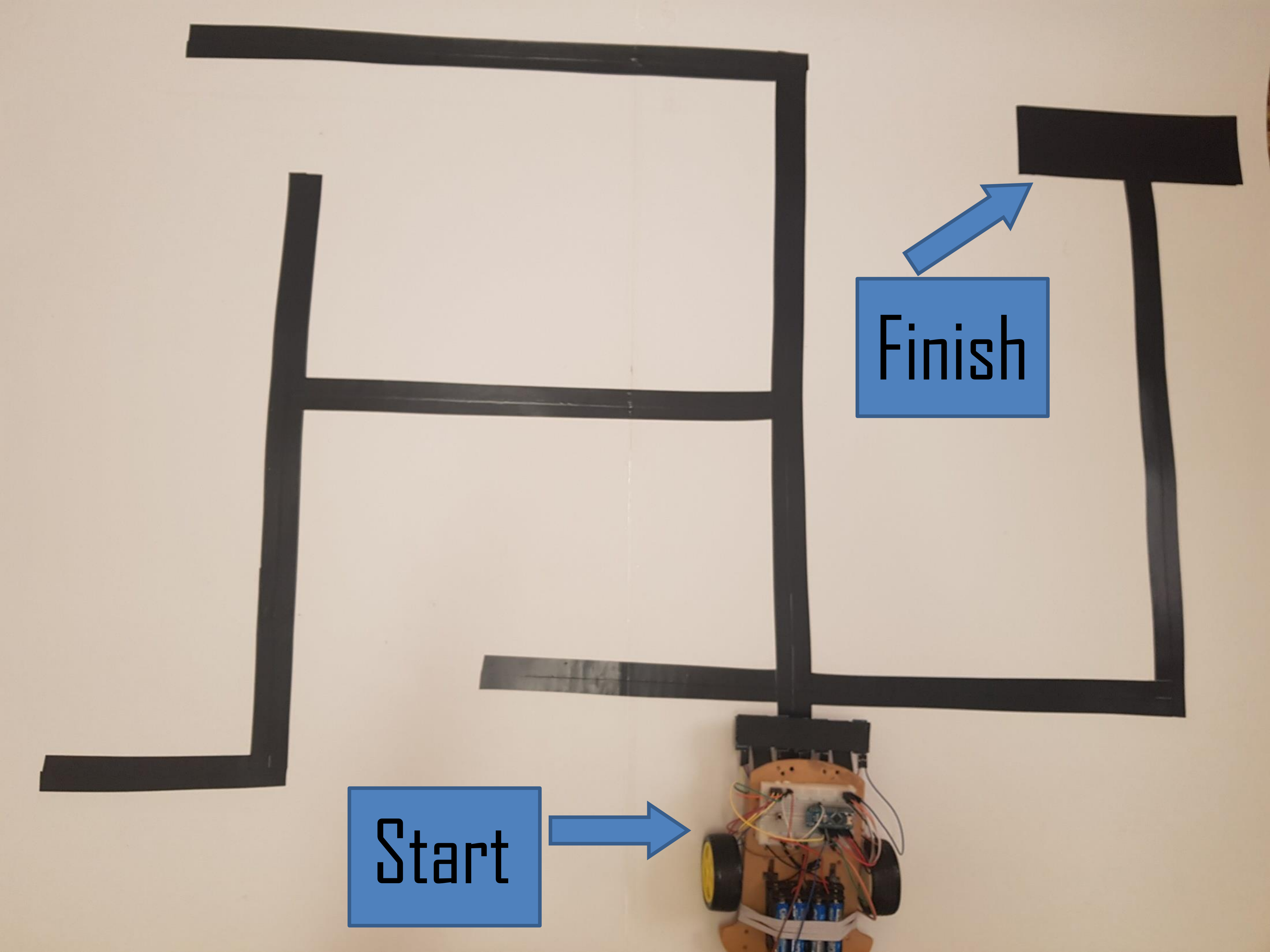
- In this project we developed a robot that will solve a Line-Maze

- The line maze will be black lines on a white background

- The robot will use Left Hand Rule to solve the Maze

# What is a Line Maze?

- A line maze is usually a black line on a white background.
- Each line maze has a Start point and a Finish point.
- The robot is expected to follow the lines and find it’s way from Start to Finish.



Finish

Start

# Project Tools

Materials to build the Robot

- Plastic base
- 2 plastic wheels
- 1 Ball caster
- Bread Board
- Wires
- 8 Batteries(each 1.5 volt)

# Electronic Tools

- ➢Arduino Nano

- ➢7 Infrared sensors

- ➢Continuous Rotation Servo Motors

# Left Hand Rule

The robot will always use the left hand rule, which means:

- Robot prefer a left turn over going straight ahead or taking a right turn.
- Robot prefer going straight over going right.
- If the maze has no loops, this will always get you to the end of the maze.

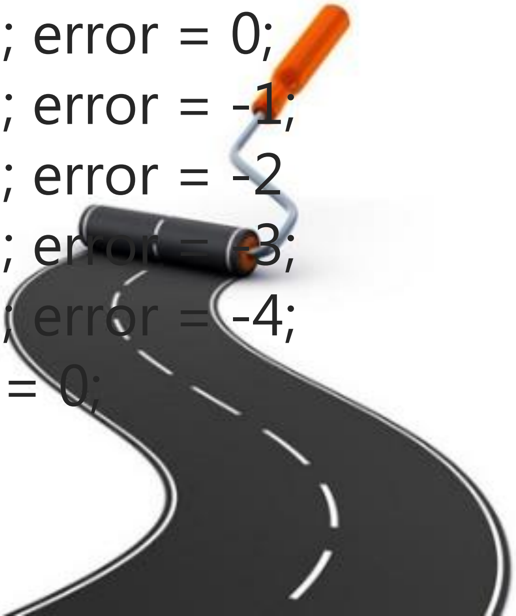
# How the sensors work?

- Five middle sensors to detect lines.

- Two side sensors to detect intersections.

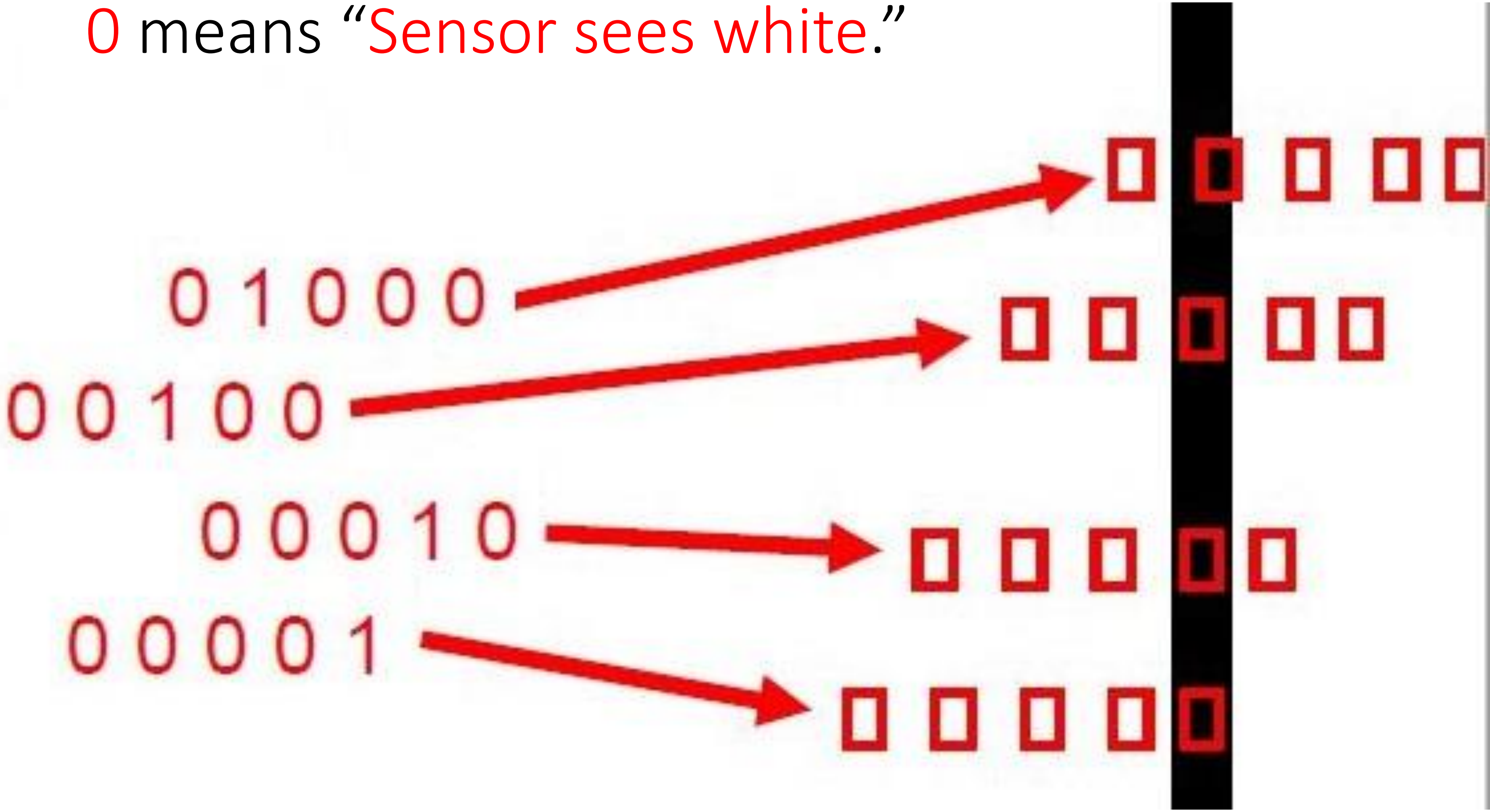
# Line Sensors

- 0 X X X X X 1 ==> mode = RIGHT_TURN; error = 0
- 1 X X X X X 0 ==> mode = LEFT_TURN; error = 0;
- X 0 0 0 0 0 X ==> mode = NO_LINE; error = 0;
- X 0 0 0 0 1 X ==> mode = FOLLOWING_LINE; error = 4;
- X 0 0 0 1 1 X ==> mode = FOLLOWING_LINE; error = 3;
- X 0 0 0 1 0 X ==> mode = FOLLOWING_LINE; error = 2;
- X 0 0 1 1 0 X ==> mode = FOLLOWING_LINE; error = 1;
- X 0 0 1 0 0 X ==> mode = FOLLOWING_LINE; error = 0;
- X 0 1 1 0 0 X ==> mode = FOLLOWING_LINE; error = -1;
- X 0 1 0 0 0 X ==> mode = FOLLOWING_LINE; error = -2
- X 1 1 0 0 0 X ==> mode = FOLLOWING_LINE; error = -3;
- X 1 0 0 0 0 X ==> mode = FOLLOWING_LINE; error = -4;
- X 1 1 1 1 1 X ==> mode = CONT_LINE; error = 0;



1 means “Sensor sees black.”

0 means “Sensor sees white.”



# How the robot behaves?

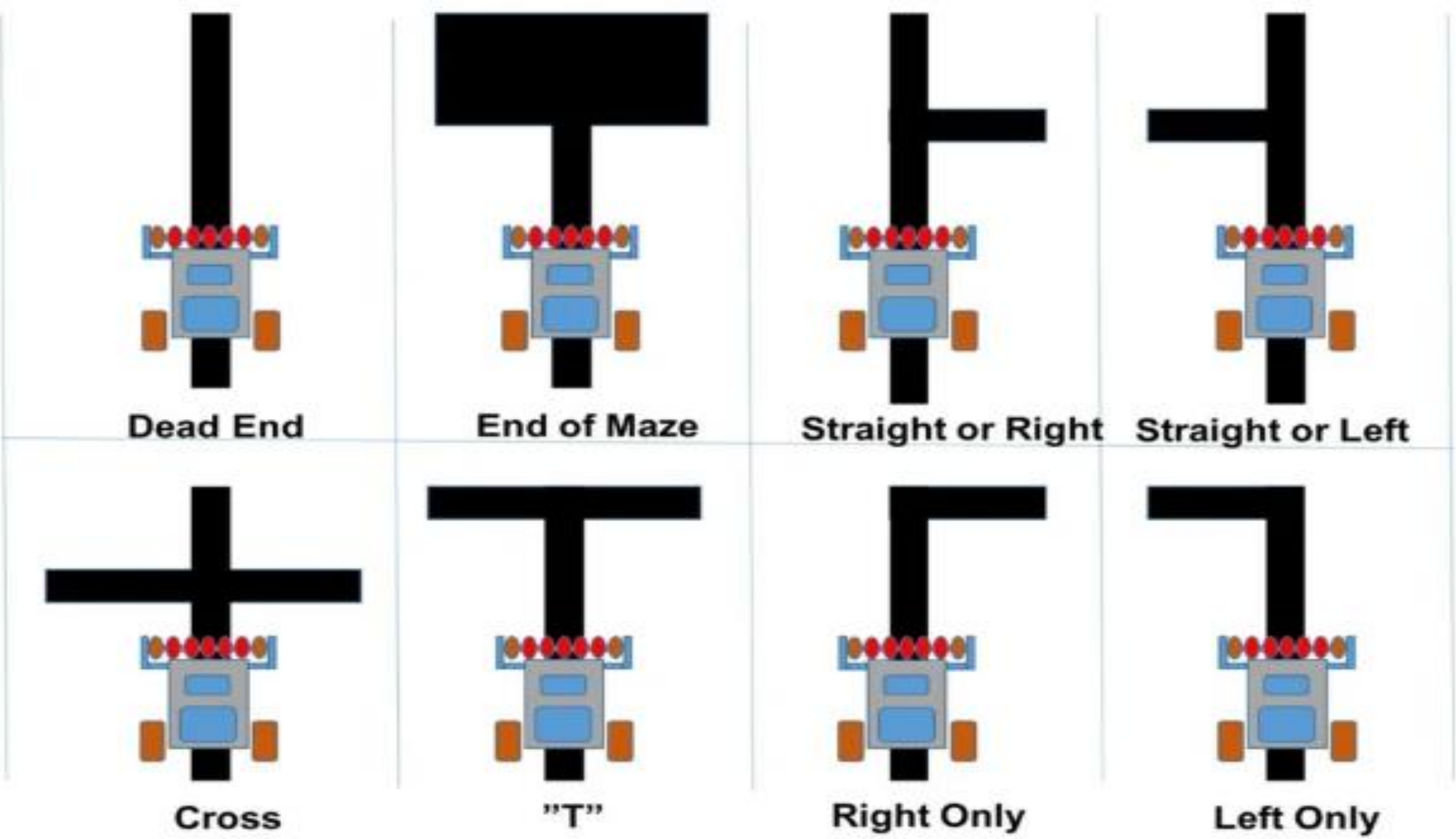
- Following the line, looking for the next intersection
- At an intersection, deciding what type of intersection it is.
- At an intersection, making a turn.

# Follow the Line

- Following the line is relatively easy.

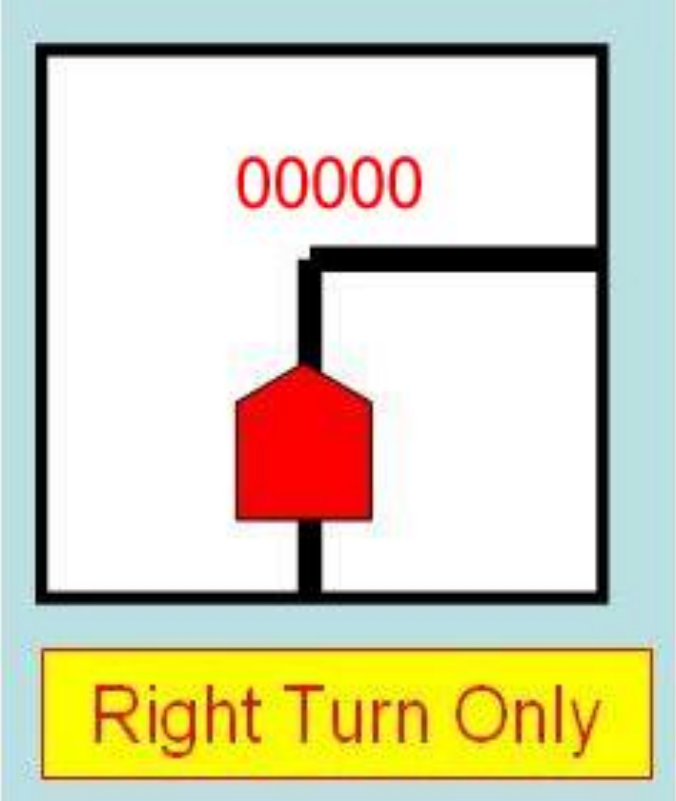
- Robot will adjust its movement to follow the line in the correct direction when straying from the line

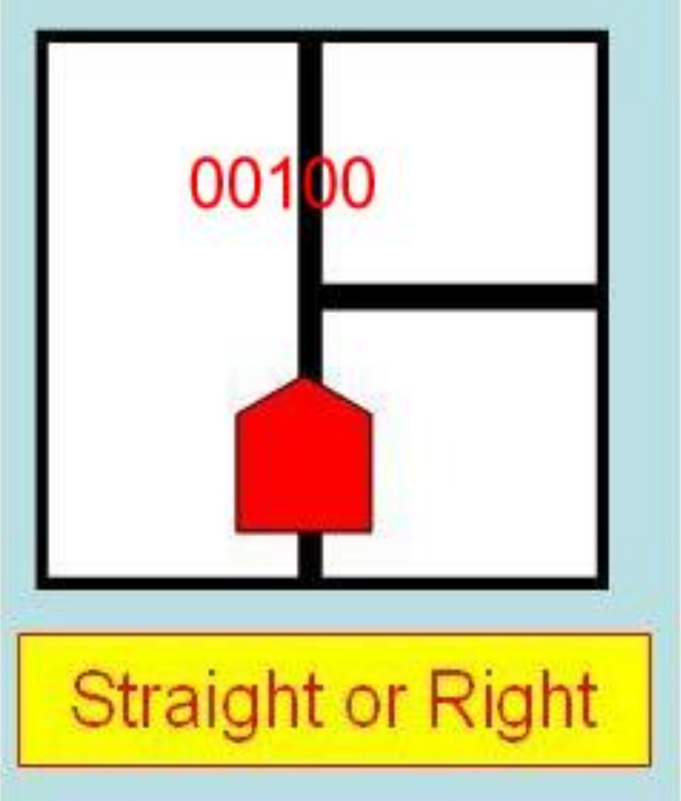
# Possibilities in the Maze



# Intersections

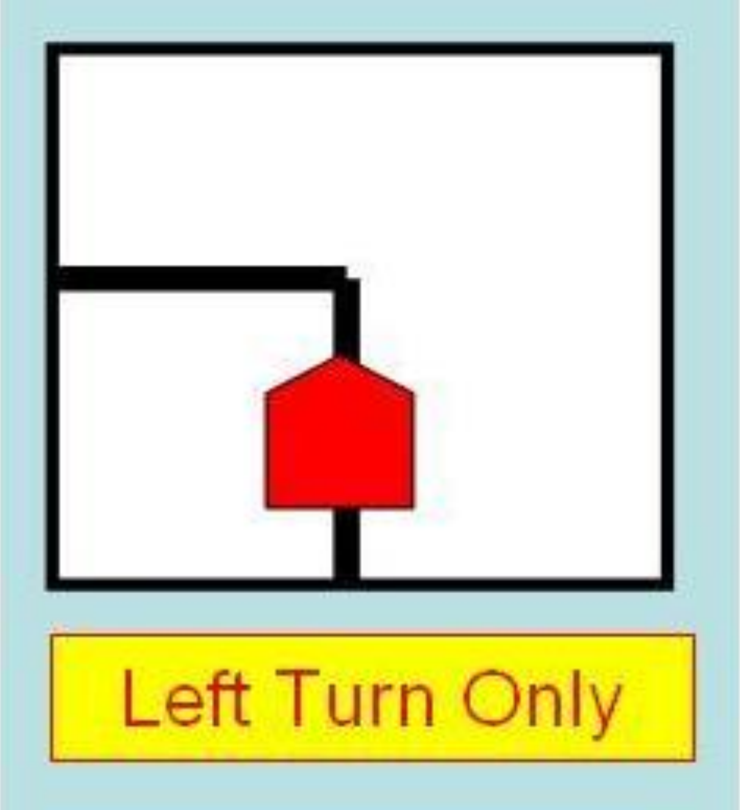
- “Right Only” or “Straight or Right”
- Moves the robot forward one inch.
- Read the sensors again

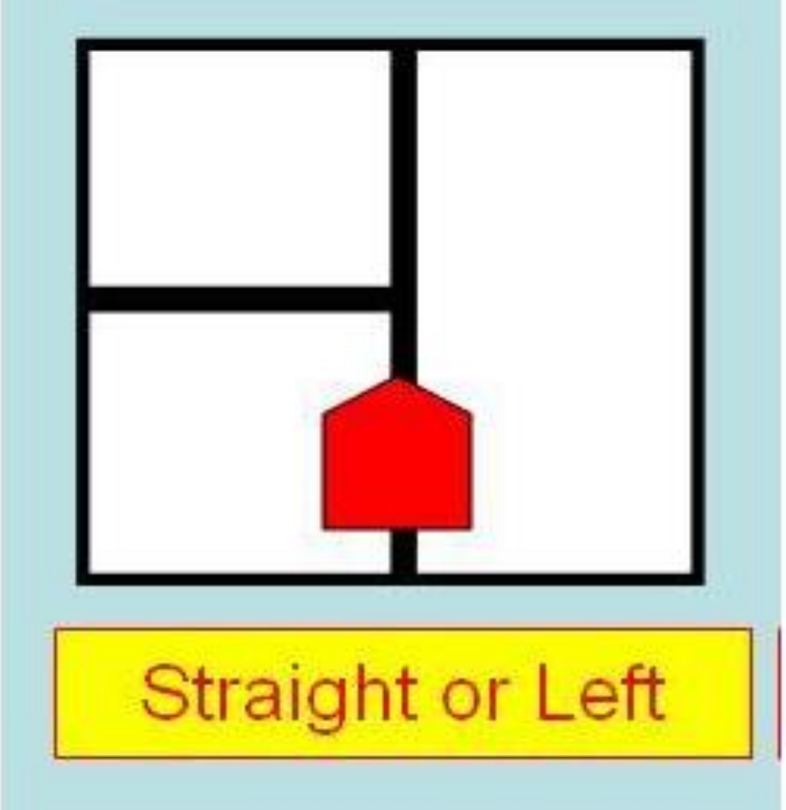




# Intersections

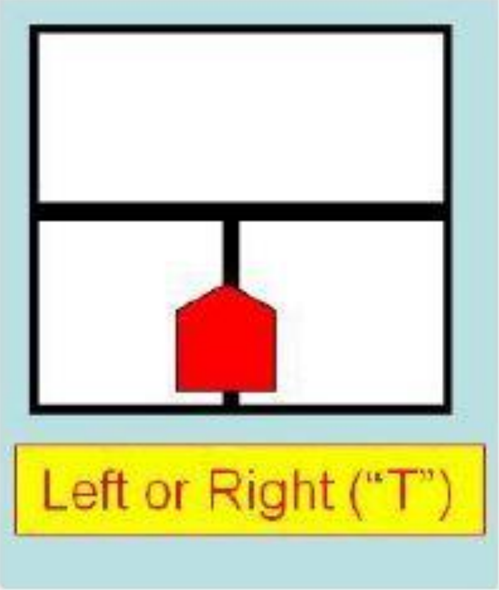
“Left Only” or “Straight or Left”

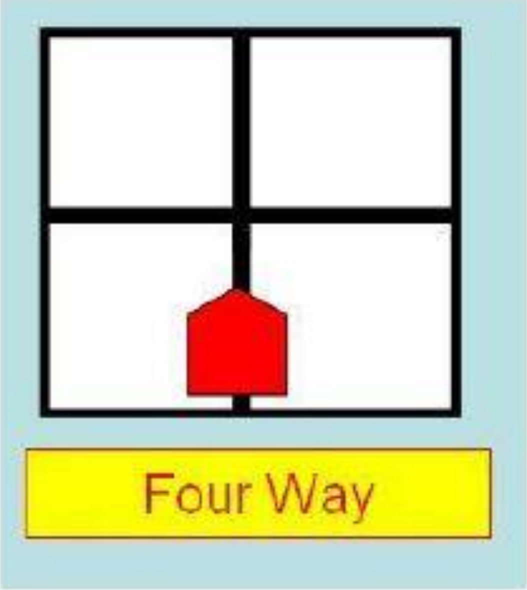


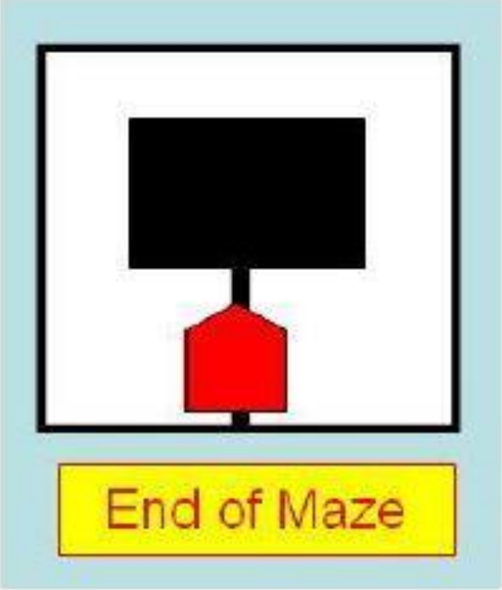


# Intersections

- ALL of these three intersection types will read "11111"
- Moves the robot forward one inch.
- Read the sensors again.

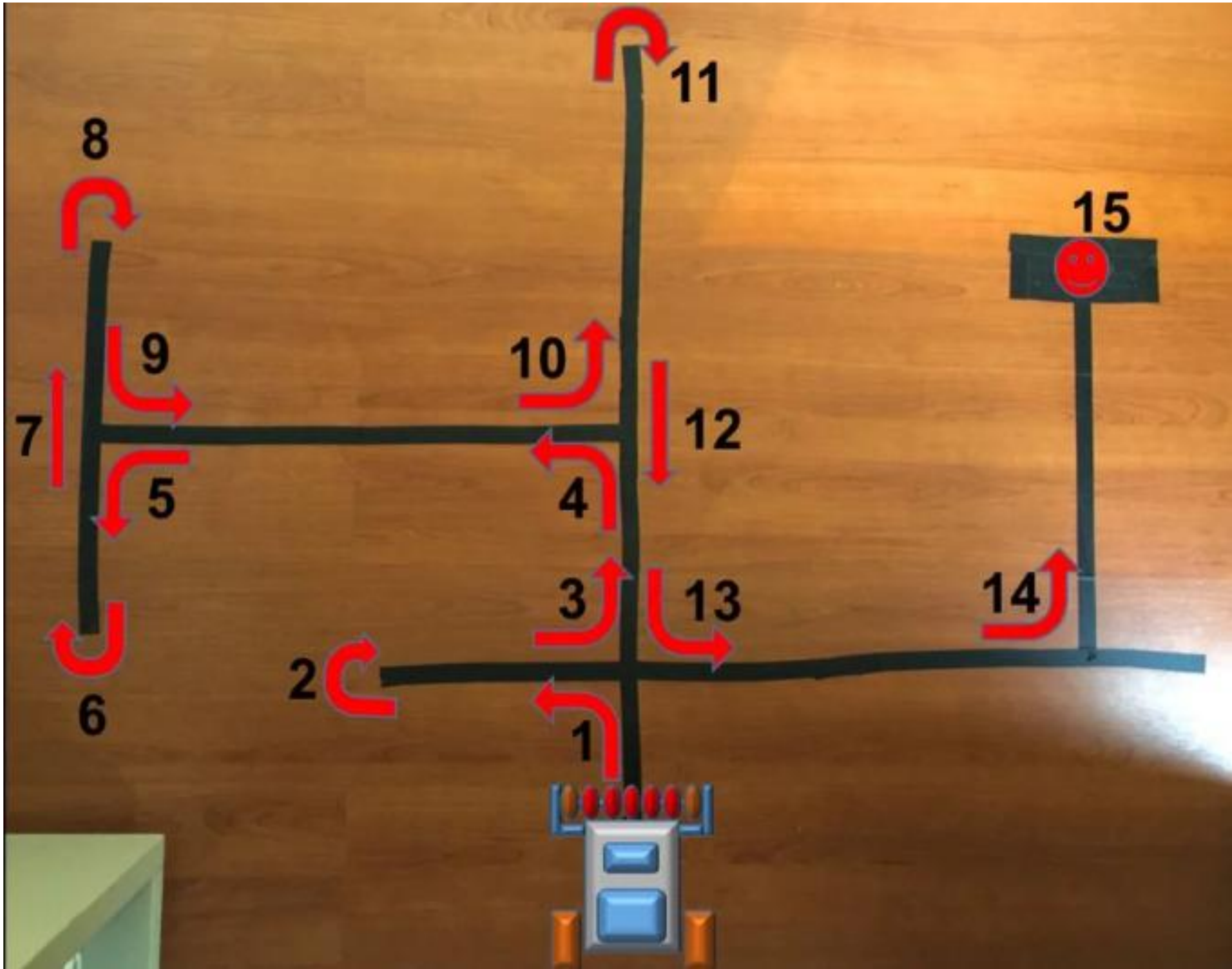






# Storing the Path

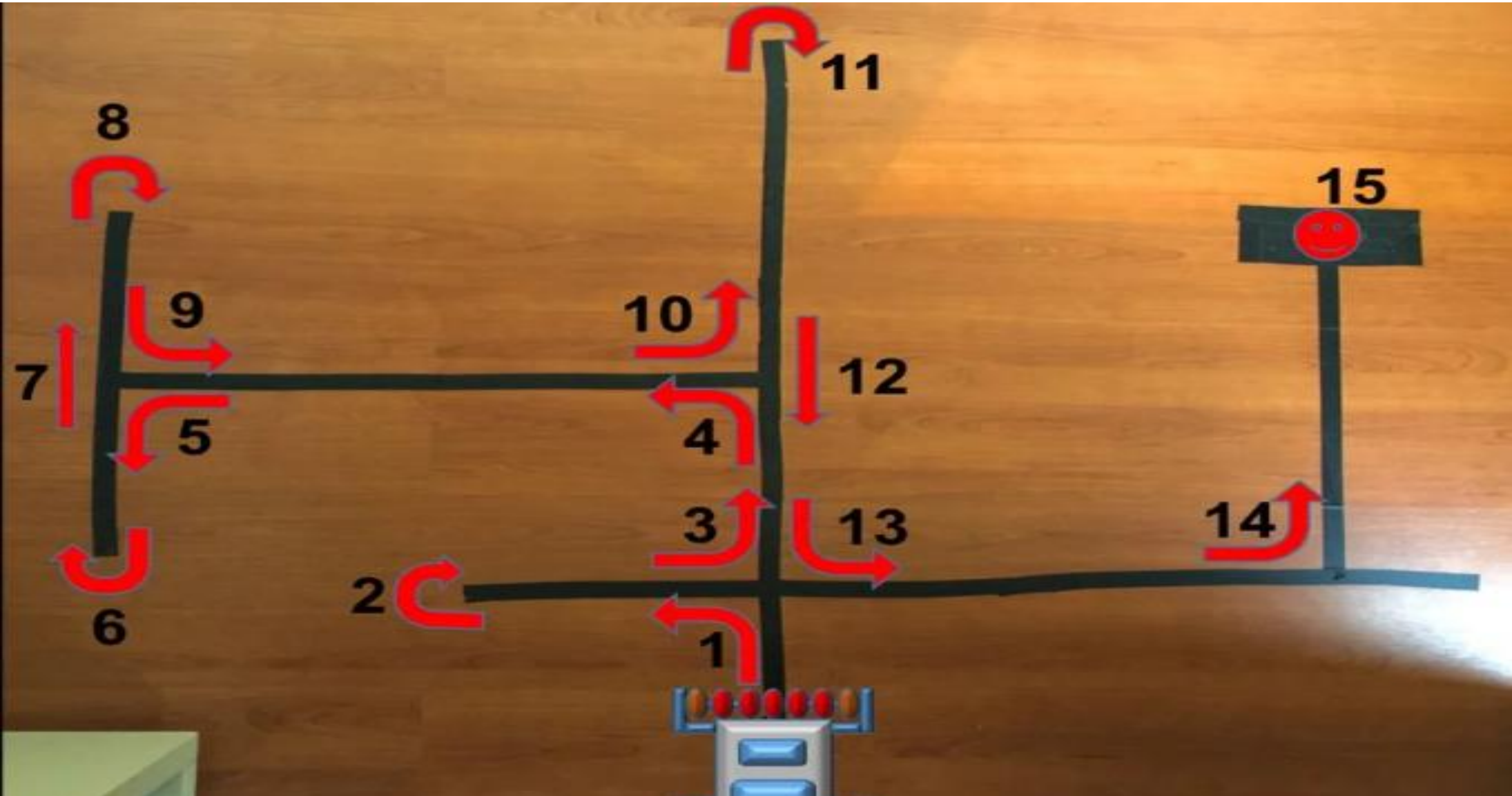
- Left (L)
- Back (B)
- Left (L)
- Left (L)
- Left (L)
- Back (B)
- Straight (S)
- Back (B)
- Left (L)
- Left (L)
- Back (B)
- Straight (S)
- Left (L)
- Left (L)
- End



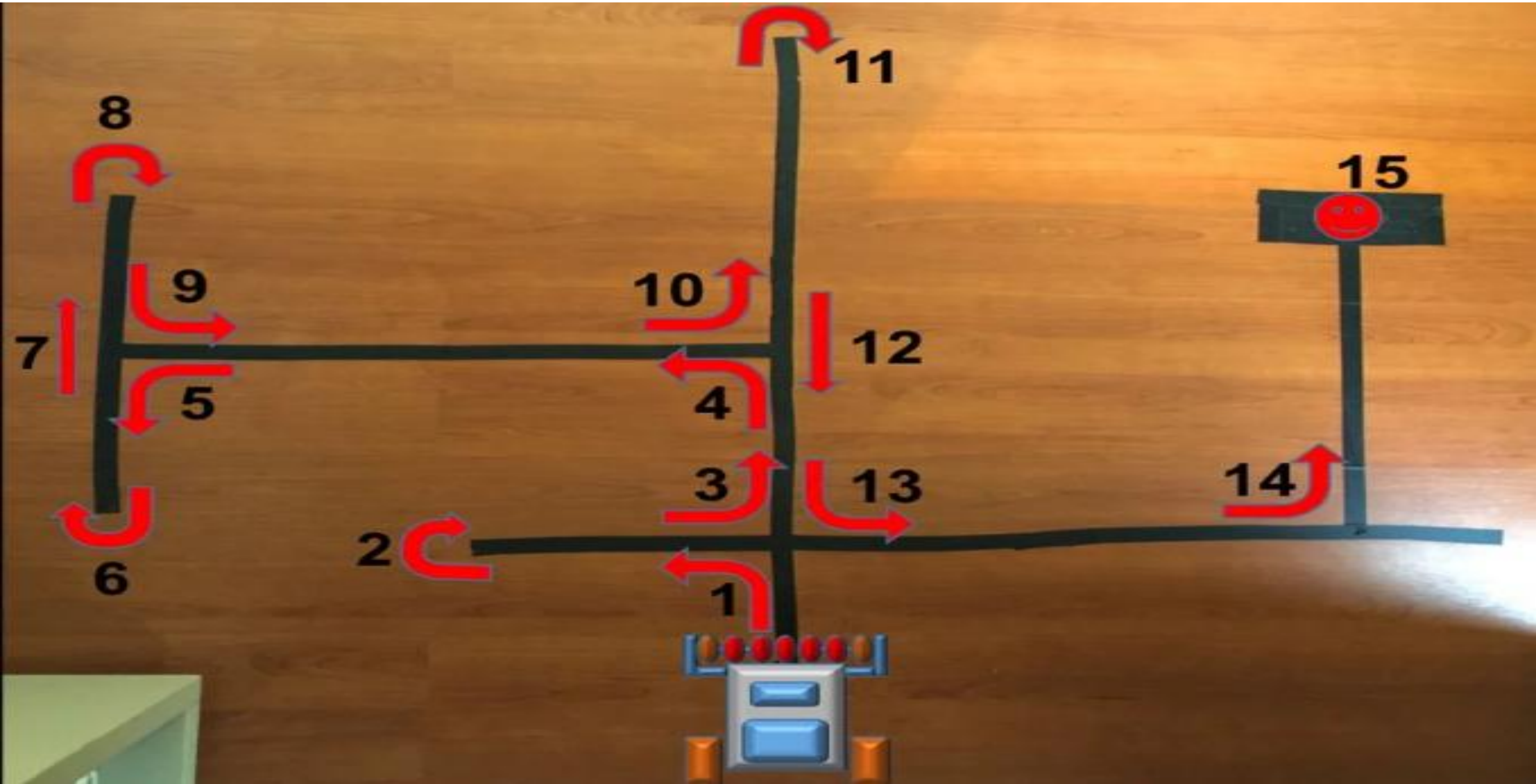
# Simplifying the path

- LBR = B
- LBS = R
- RBL = B
- SBL = R
- SBS = B
- LBL = S

- path = [**LBL**LLBSBLLBSLL] ==> LBL = S
- path = [SL**LBS**BLLBSLL] ==> LBS = R
- path = [SL**RBL**LBSLL] ==> RBL = B

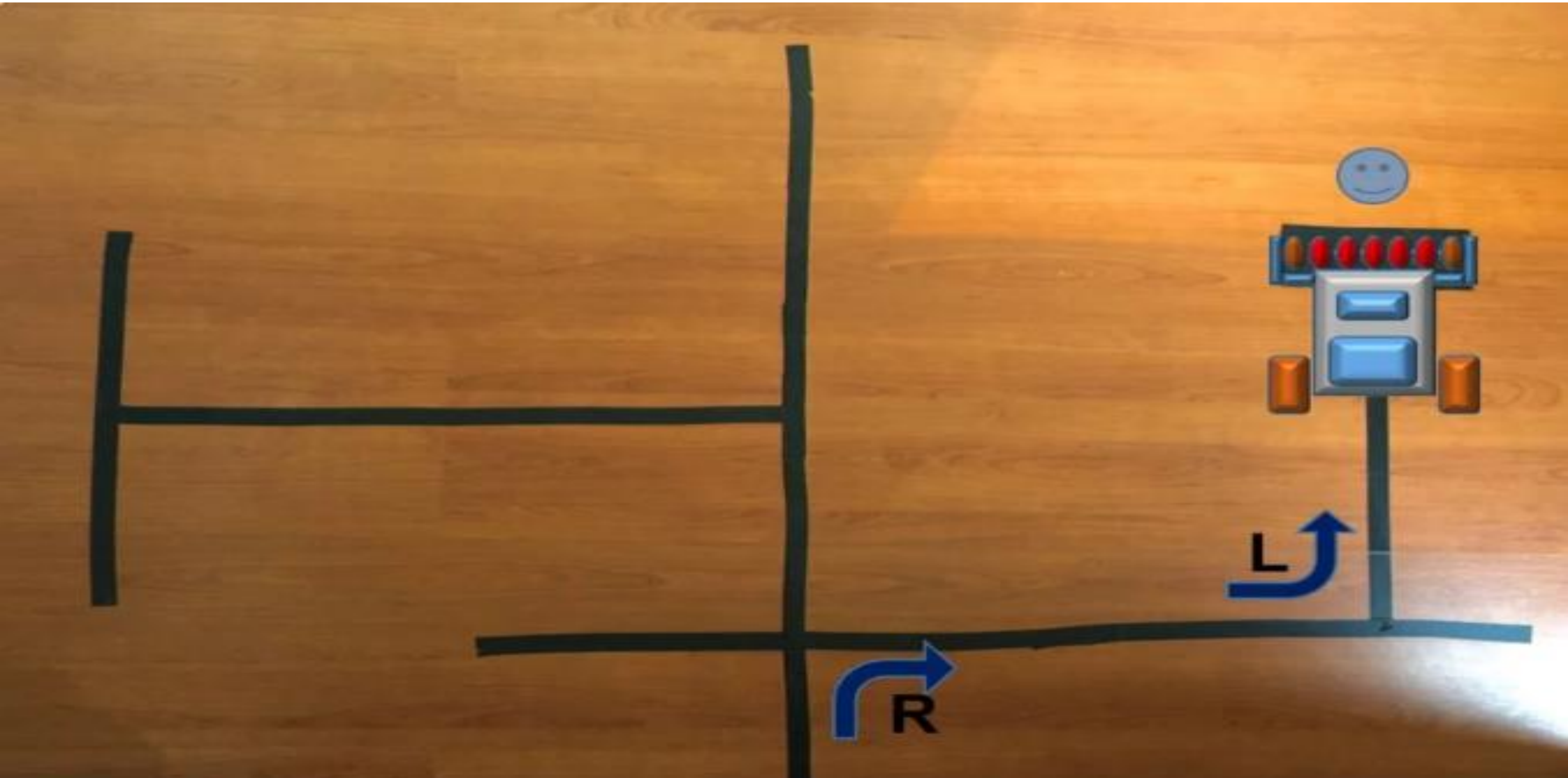


- path = [S**LBL**BSLL] ==> LBL = S
- path = [S**SBS**LL] ==> SBS = B
- path = [**SBL**L] ==> SBL = R
- path = [RL]



# Second Pass

## Solving the Maze As Fast As Possible.



# Limitations of Project

- Sensitivity of sensors, which affect the logic of the algorithm.
- Lighting, which may affect the sensors readings.
- The ability to adjust motors movement.

# Future Work

- Change and develop the design of the robot.
- Add an algorithm to solve loops problem.

Start
DEMO

# Any Questions ?

Thank you!!