9

ROBOT IN THE MAZE

STEMJAM Teaching Guide

Developing make spaces to promote creativity around STEM in schools
Acronym: STEMJAM
Project no. 2016-1-ES01-KA201-025470

www.stemjam.eu

Co-funded by the
Erasmus+ Programme
of the European Union

MAZE

ABSTRACT

The aim of activity is solve the maze.

In first version the mazes is made from black line. Robot use two sensor of line

The second option is to build the maze with walls. For its resolution the robot uses an ultrasonic sensor to follow the wall and a line-follower sensor to detect the front walls.

DIDACTIC OBJECTIVES

- To learn how to develop an algorithm for solving the maze.
- To learn the operation of different sensors and components.
- To develop computational thinking.

STEM Subject: Science☒ Technology □ Engineering□ Mathematics☒

Education Level: 12-14 years□ 14-16 years☒

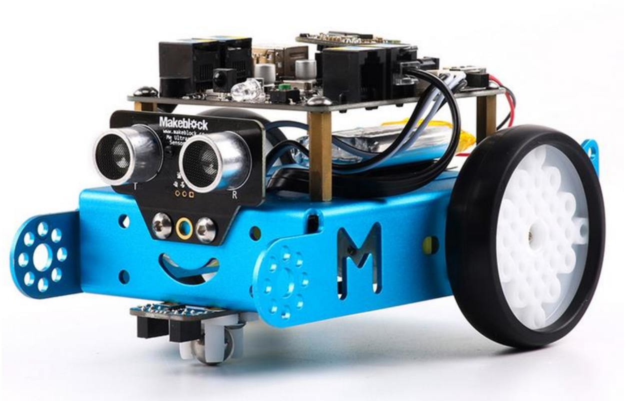
PROBLEM STATEMENT

The mBot has to find itself the way out of a maze, by:

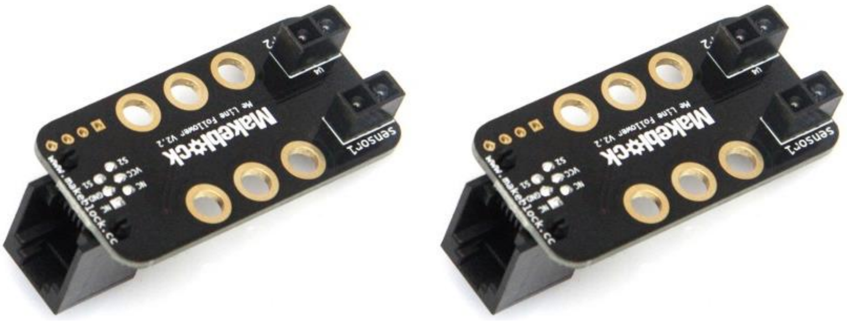
1. Using two line sensors in the maze made from the black line
2. Using both the ultrasonic sensor and the line-follower sensor to compare distances to the wall.

BOM (Bill of Materials Needed)

- mBot => Ref. 90054



- (x2) **Line Follower** sensors:



- Me **Ultrasonic Sensor**:

- Different beams and structures:

- Maze with black line.

First version

ELEMENT	ID	CABLE	AMOUNT	PORT 1			PORT 2			PORT 3				PORT 4				P.MOT 1	P.MOT 2
				Y	B	W	Y	B	W	Y	B	W	Bl	Y	B	W	Bl	W*	W*
Mbot Robot 2´4G			1																
Motor 1	W*		1															W*	
Motor 2	W*		1																W*
Me Line Follower	B		2					B			B								
RJ25 cables			2																
Structures and beams			1																
Laptops			1																
Attrezzo (not essential)																			

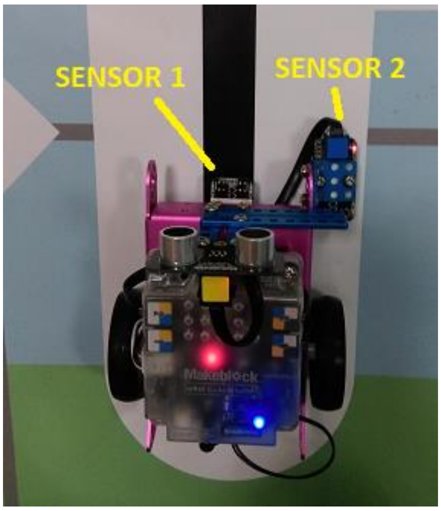
Second version

ELEMENT	ID	CABLE	AMOUNT	PORT 1			PORT 2			PORT 3				PORT 4				P.MOT1	P.MOT2
				Am	Az	Bl	Am	Az	Bl	Am	Az	Bl	Ng	Am	Az	Bl	Ng	Bl*	Bl*
Mbot Robot 2´4G			1																
Motor 1	Bl*																	Bl*	
Motor 2	Bl*																		Bl*
Ultrasonic sensor	Az	1	1							Az									
Me Line Follower	Az	1	1					Az											
RJ25 cables			2																
Structures																			
Brass Stud M4x20			2																
Cut-out beam			1																
Nuts and bolts (Pairs)			6																
Atrezzo - Maze (with white walls)			1																

ACTIVITY DESCRIPTION

First version

Robot is eqipped in two line sensors. Sensor nr 1 should be following the black line and realising the classical algorithm „go along the line”. Sensor nr 2 checks what colour is located on the robots right.



The basic algorithm of moving around the maze is an algorithm of „keeping to the right”.

It can be defined as follows:

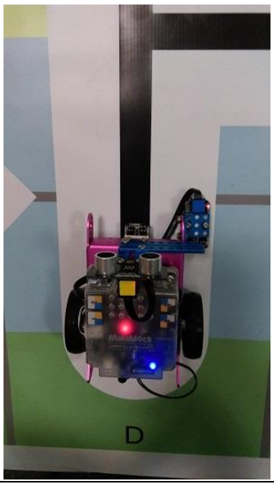
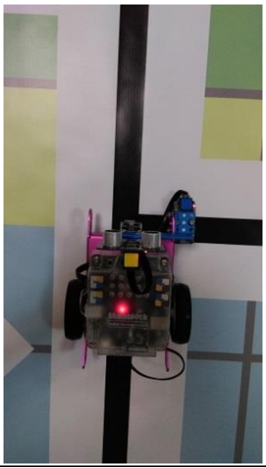
	Robot goes straight when the sensor nr 1 is black, and sensor nr 2 is white.		When the side sensor nr 2 goes onto the black line robot should turn right.
		When both sensors see the white colour it means turning left until it finds the black colours by the sensor nr 1. In case when the line is finished, robot will turn around, and in case when there is a left turn, robot will turn left.	

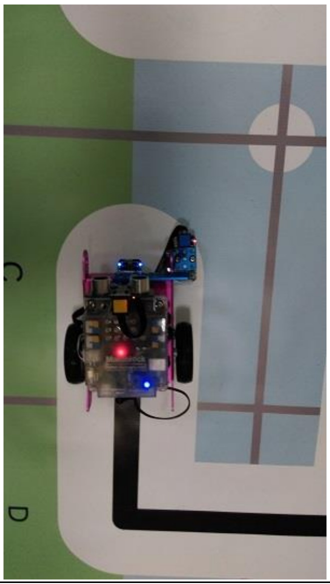
The readings from line sensor are:

two white colors	3
two black colors	0
left black, right white	1
left white, right black	2

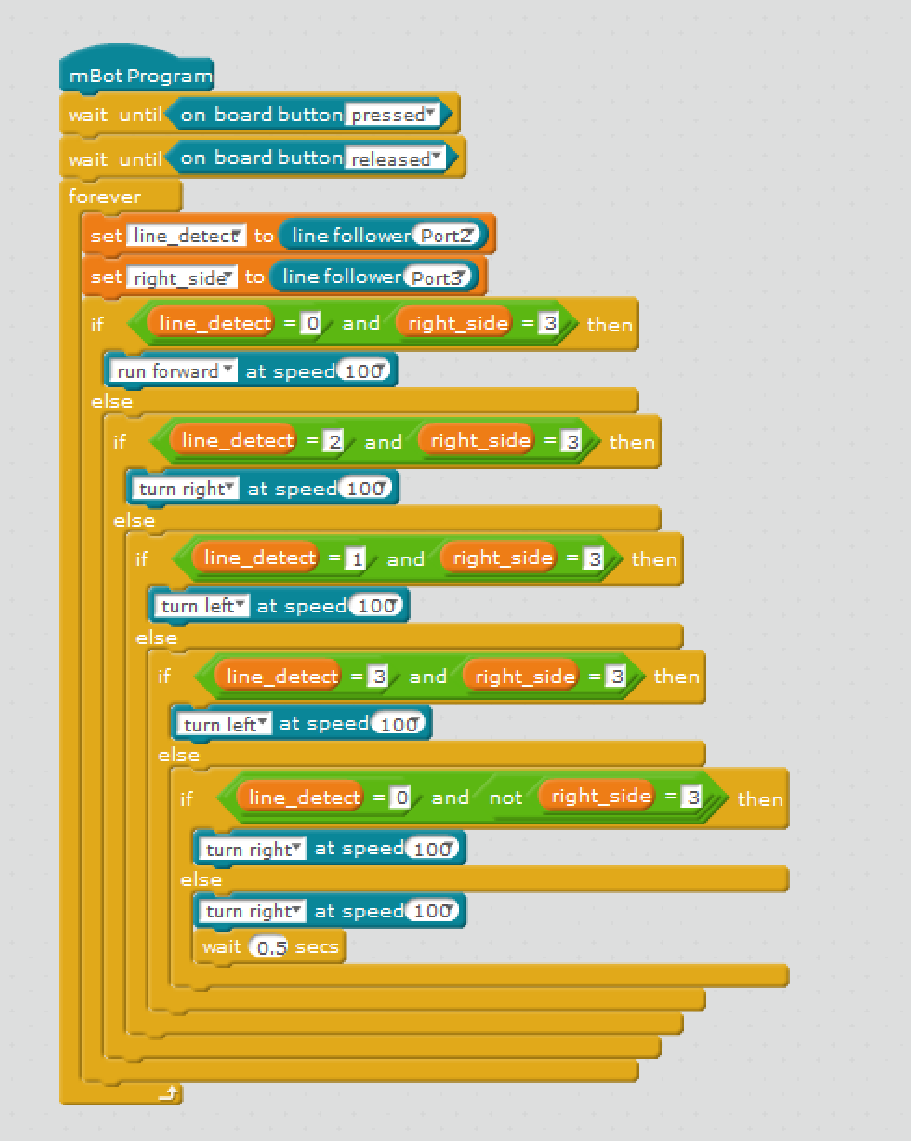
Now we can create the conditions to control the movement of robot. Lets see the picture one again:

We will create two varaibles: *line_detect* and *right_side*

	Robot goes straight when the sensor nr 1 is black, and sensor nr 2 is white. But sensor 1 can detect that robot loses the line. That time we have to correct the path like in line follower program.	If *line_detect*=0 and *right_side*=3 Run forward If *line_detect*=1 and *right_side*=3 Turn left If *line_detect*=2 and *right_side*=3 Turn right
	When the side sensor nr 2 goes onto the black line robot should turn right.	If *line_detect*=0 and *right_side*=0 Turn right This is perfect situation, but when you try to run this program robot will not turn right fluently. During the turning the right sensor changes readings to 1 or 2. So we change: If *line_detect*=0 and ***right_side*≠3** Turn right

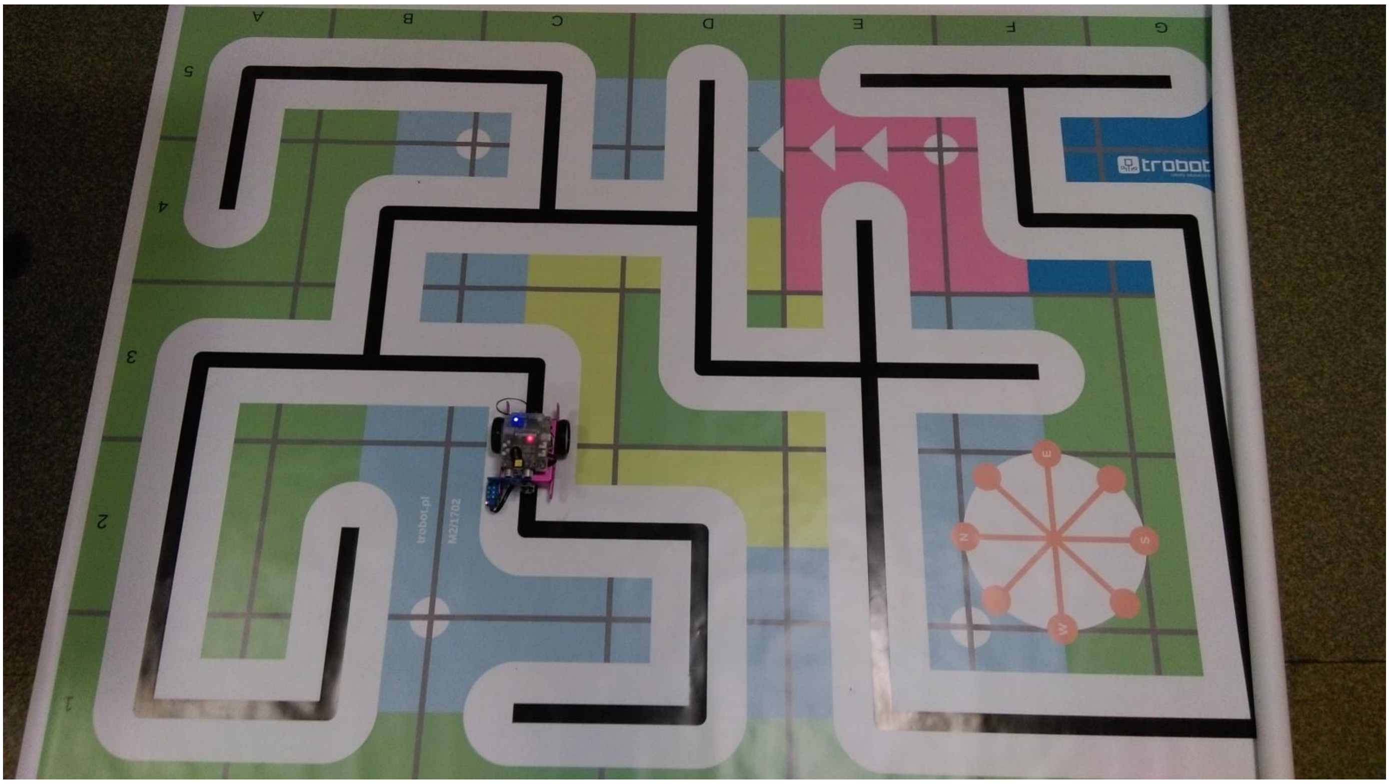
	When both sensors see the white colour it means turning left until it finds the black colours by the sensor nr 1. In case when the line is finished, robot will turn around, and in case when there is a left turn, robot will turn left.	If *line_detect*=3 and *right_side*=3 Turn left

All this cases we put to forever loop. Remember of nesting the if-else instruction.



```
mBot Program
wait until on board button pressed
wait until on board button released
forever
    set line_detect to line follower Port2
    set right_side to line follower Port3
    if line_detect = 0 and right_side = 3 then
        run forward at speed 100
    else
        if line_detect = 2 and right_side = 3 then
            turn right at speed 100
        else
            if line_detect = 1 and right_side = 3 then
                turn left at speed 100
            else
                if line_detect = 3 and right_side = 3 then
                    turn left at speed 100
                else
                    if line_detect = 0 and not right_side = 3 then
                        turn right at speed 100
                    else
                        turn right at speed 100
                        wait 0.5 secs
```

Here you can see the maze made from the lines:



Second version

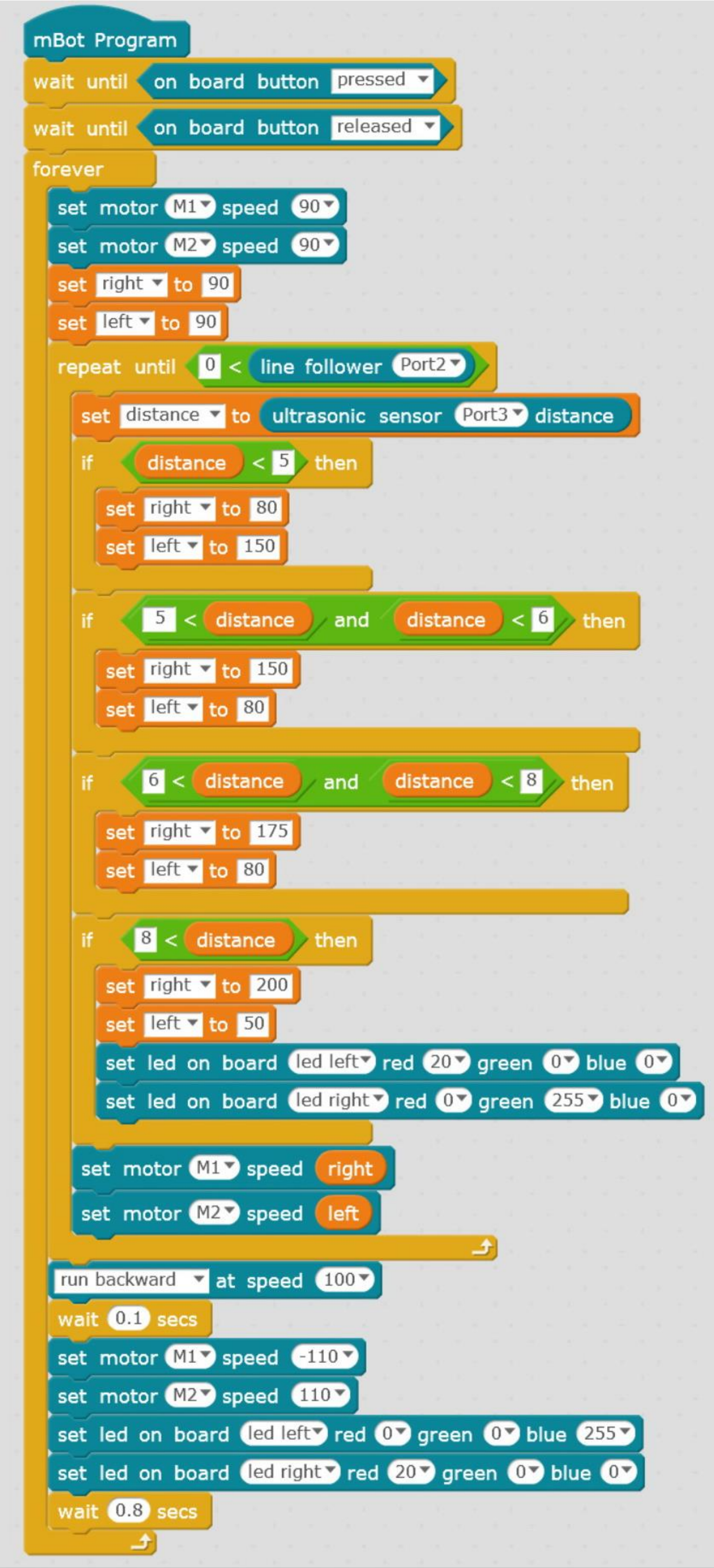
The robot will stay still until its board button is pressed and released. From this moment on the main program will start running, combining the line-follower sensor and the ultrasonic sensor to detect the maze walls.

The first condition is stablished with the line-follower sensor, which is placed on the front of the mBot:

- If it detects white is because it is walking forward to the wall, so it will turn right 90 degrees.
- If it does not detect white, the second condition stablished with the ultrasonic sensor is started. This sensor is placed in the right side of the mBot and will perform four different routines, depending on the distance to the wall:
    - Distance is smaller than 5 cm: the mBot is too close to the maze wall, so it will turn left slowly.
    - Distance is between 5 and 6 cm: the mBot turns right slowly, correcting the slight path diversion.
    - Distance is between 6 and 8 cm: the mBot turns right a little faster, due to the diversion is more noticeable.
    - Distance is larger than 8 cm: it means the mBot has arrived to a wall corner, so it will tur right very fast to find again the wall to follow.

The mBot will work autonomously, being the main code recorded on its Arduino board.

MAIN CODE (PC PROGRAM)



```
mBot Program
wait until on board button pressed
wait until on board button released
forever
    set motor M1 speed 90
    set motor M2 speed 90
    set right to 90
    set left to 90
    repeat until 0 < line follower Port2
        set distance to ultrasonic sensor Port3 distance
        if distance < 5 then
            set right to 80
            set left to 150
        if 5 < distance and distance < 6 then
            set right to 150
            set left to 80
        if 6 < distance and distance < 8 then
            set right to 175
            set left to 80
        if 8 < distance then
            set right to 200
            set left to 50
            set led on board led left red 20 green 0 blue 0
            set led on board led right red 0 green 255 blue 0
        set motor M1 speed right
        set motor M2 speed left
    run backward at speed 100
    wait 0.1 secs
    set motor M1 speed -110
    set motor M2 speed 110
    set led on board led left red 0 green 0 blue 255
    set led on board led right red 20 green 0 blue 0
    wait 0.8 secs
```

We will record this code into the Arduino board of the mBot. This way it will work independently from the computer and there will not be needed a laptop close to the maze to complete the activity.

How to load a program into the Arduino mBot board using mBlock:

In order to load a program on the board using mBlock:

1. Choose mBot in the Board tab of the mBlock menu.

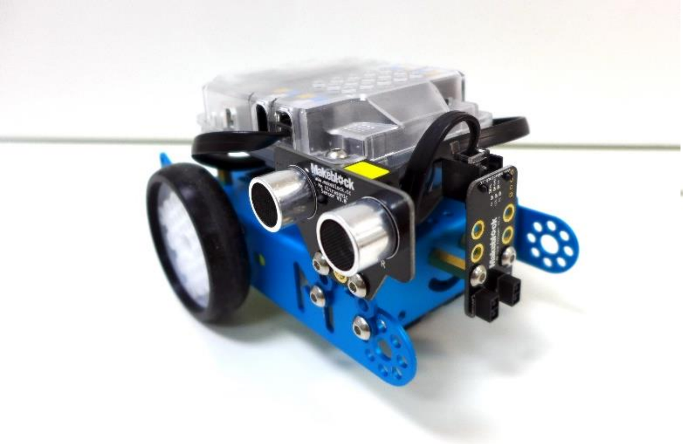
2. Connect the USB and choose "Serial Port" in the connect tab.

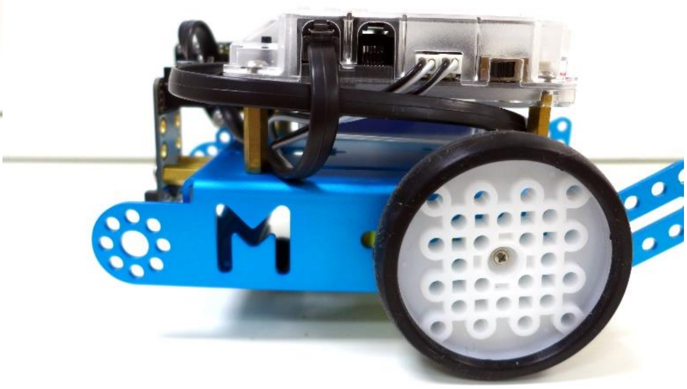
3. In the tab edit, choose "Arduino Mode" (In the program that we are going to load, instead of the green flag, we will put the blue command "mBot program")

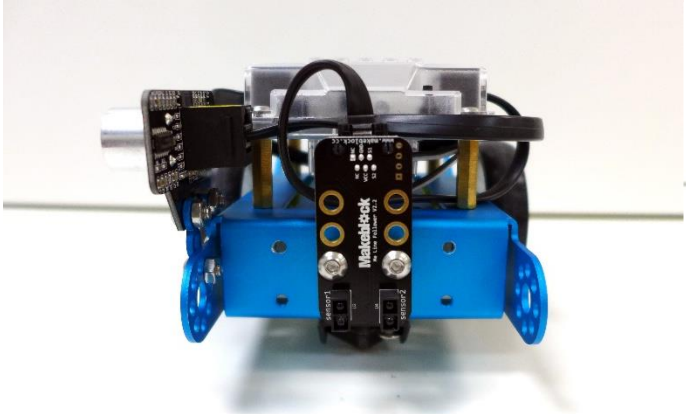
4. A window with the code will open to record it on the Arduino board of mBot. You can, if you want, modify your program. Finally, click on Upload to Arduino.

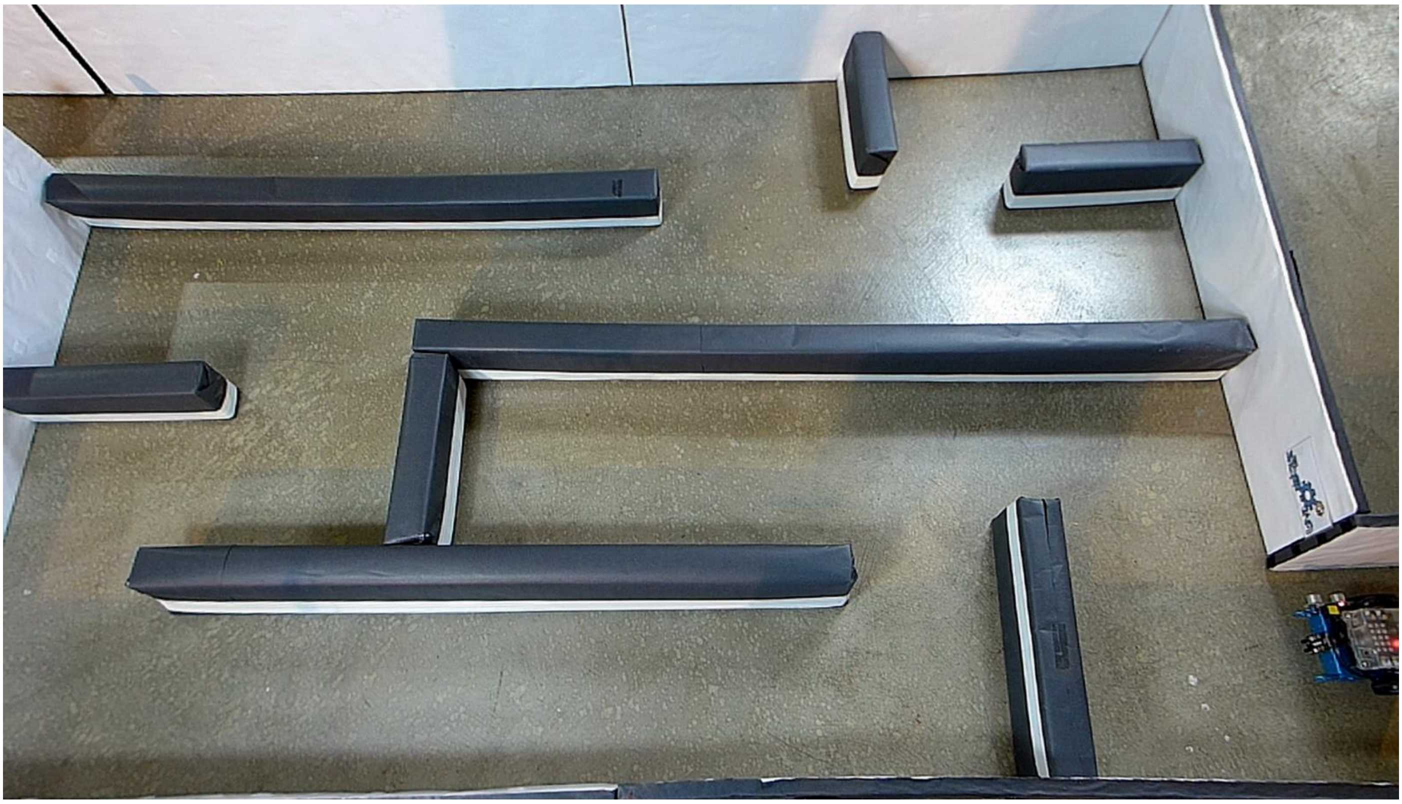
5. If there have been no errors, a message will be sent informing that the program has been recorded correctly. At this moment you will be able to start enjoying the program introduced in the robot, without the computer turned on. For doing this, you must disconnect the USB cable and connect the batteries (or lithium battery) of the robot. You will see that your mBot works independently.

Structural composition: once the programming is finished, we start BUILDING UP THE STRUCTURE where all the mechanical elements will be set, just as the electronic elements.



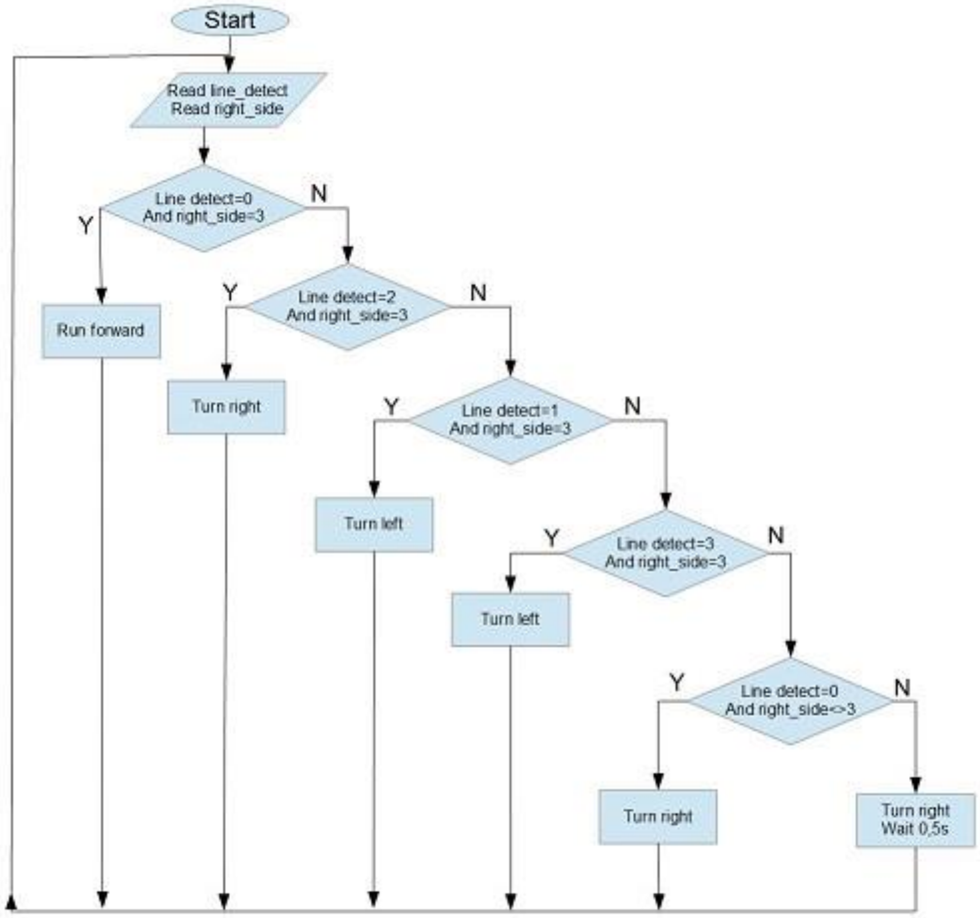




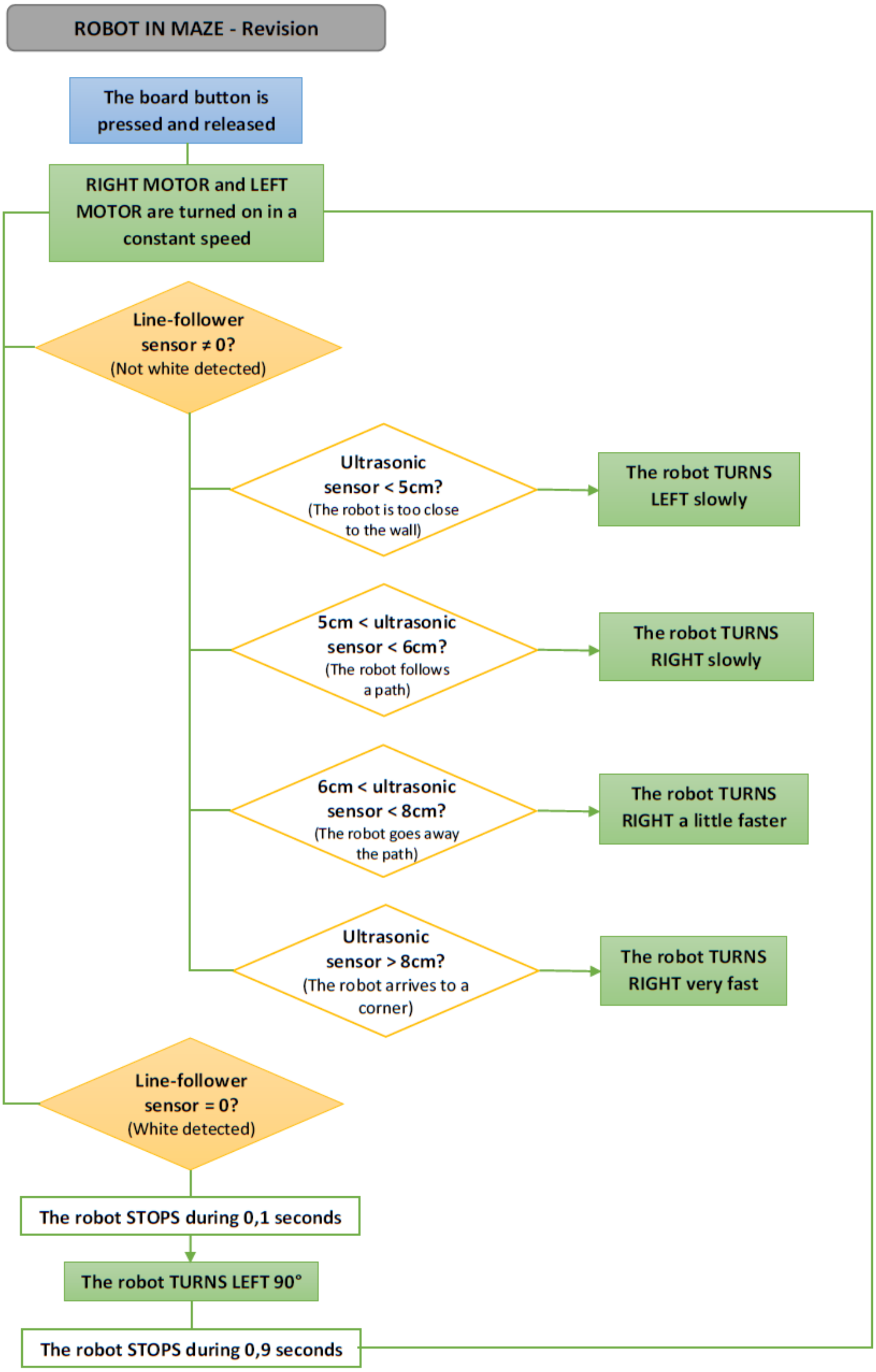


FLOW CHART

First version



Second version


ROBOT IN MAZE - Revision
The board button is pressed and released
RIGHT MOTOR and LEFT MOTOR are turned on in a constant speed
Line-follower sensor ≠ 0? (Not white detected)
Ultrasonic sensor < 5cm? (The robot is too close to the wall)
The robot TURNS LEFT slowly
5cm < ultrasonic sensor < 6cm? (The robot follows a path)
The robot TURNS RIGHT slowly
6cm < ultrasonic sensor < 8cm? (The robot goes away the path)
The robot TURNS RIGHT a little faster
Ultrasonic sensor > 8cm? (The robot arrives to a corner)
The robot TURNS RIGHT very fast
Line-follower sensor = 0? (White detected)
The robot STOPS during 0,1 seconds
The robot TURNS LEFT 90°
The robot STOPS during 0,9 seconds

STUDENT'S EVALUATION

This exercise teaches algorithmic thinking. The right-hand algorithm can also be used for a robot with two distance sensors - this case is difficult and is a good exercise for students to overcome obstacles. The algorithm is clear, but it is technically necessary to choose the right time and distance.

SCALABILITY

The first ride is used to get to know the labyrinth (its construction). Next, the robot calculates the shortest route to the destination and the next passage is performed without dead ends.

This program requires a program written in arduino.