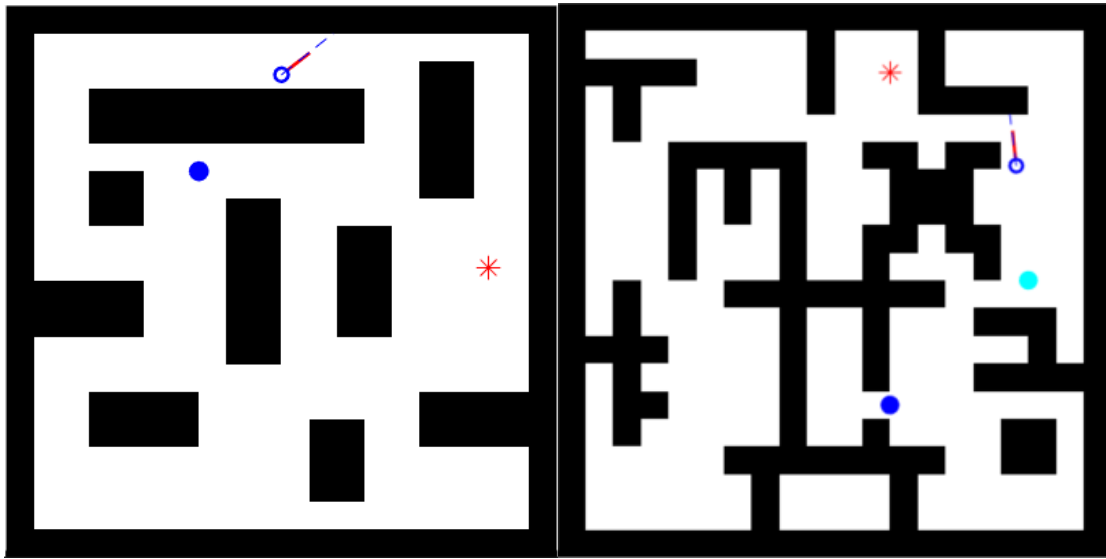


# mazeColliders

By Gia Nguyen (Tim)

Mechatronics 2 Robot Simulator



# 1. Introduction

This is a maze navigation simulation based on multiple MATLAB Toolboxes. The purpose of the simulation is to control a robot by a micro-controller with serial communication from a pre-set command bank. This project is designed and implemented on Windows for Arduino UNO(ATmega328).

## 2. Installation and Set up

### 2.1 Software Installation

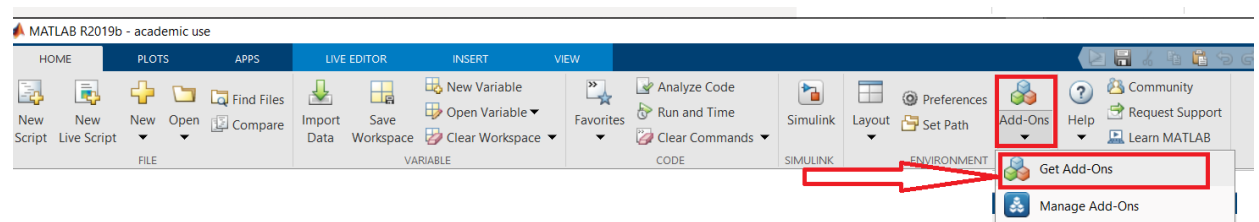
1. Arduino IDE (<https://www.arduino.cc/en/main/software>)
2. MATLAB : This simulation is designed to run on MATLAB environment. MATLAB is free for UTS student (<https://www.uts.edu.au/current-students/current-students-information-faculty-engineering-and-it/software-download>)

MATLAB Toolboxes required for the simulation

- Mobile Robotics Simulation toolbox: The script "startMobileRoboticsSimulationToolbox.m" will get called once to initialize the Mobile Robotics Simulation Toolbox, this toolbox is included in the Toolbox folder
- Robotic System toolbox: can be installed from MATLAB Add-Ons page (<https://au.mathworks.com/products/robotics.html>)
- Navigation toolbox: can be installed from MATLAB Add-Ons page (<https://au.mathworks.com/products/navigation.html>)

How to install:

Open MATLAB, on the toolstrip at "HOME" tab: Select Get Add-Ons, Search the toolbox name and select Install. Some toolbox required MATLAB to restart before the toolbox is usable.



### 2.2 Simulator Set up

To run this simulator:

- Arduino must be connected to the computer/laptop with the USB A-B cable
- MATLAB must be set to the MazeColliders folder

In MATLAB, After the first run, line number 5 (run startMobileRoboticsSimulationToolbox.m) can be commented out, the purpose of this line is to initialize the toolbox.

```

6  %% Change these values to suit your setup
7  run startMobileRoboticsSimulationToolbox.m %% Comment out after first run
8  port = serialportlist("available"); %% List the available Serial ports
9  SerialPort = "COM4"; %% Change to the port connected to the Arduino
10 BaudRate = 9600; %% Communication baud rate
11 mapLoad = imread('box.png'); % Load map from .png pixel drawing (200x200 pixel)

```

After plugged in the Arduino, the Serial port which is connected to the Arduino can be found from the Arduino IDE or by running command `//serialportlist("available")//` in MATLAB to find the connected port.

After the connected Serial port is identified, on line 7, change the value `"COM4"` to the port you connected the Arduino.

Communication baud rate can be adjusted on line 8.

Change the value of the task variable on line 9, `"A3"` or `"A4"`

Random goals can be generated when the variable `randomGoal` on line 10 is set to `true`, set this value to `false` if does not want random goal to be generated.

The value of line 11 is to set the map manually

Line 12 and 13 is the option to spawn random map and initial robot pose

Select "Run" from MATLAB editor to start the simulation.

Send the `"CMD_START"` command from Arduino to start controlling the robot, and `"CMD_CLOSE"` to clean up and stop the simulation, it is recommended to always stop the simulation with `"CMD_CLOSE"` instead of stopping from MATLAB.

## 2.3 Basic operation

In Arduino IDE :

- To send the command, use `//PrintMessage("command")//` in the DemoCode.ino file. The command/values returned is in String format.

Example :

```
PrintMessage("CMD_SEN_IR");
```

This will send the command `CMD_SEN_IR` (query the IR range) to the simulation.

- To receive the values, use `//Serial.readString()//` function  

```
String incomingByte = Serial.readString();
```

This will read the incoming data(String) from the simulation.

- If the Arduino IDE having trouble loading code to the Arduino similar to this, clear the Serial variable in MATLAB with `// arduinoObj = [] ;//` in the MATLAB command window or just

simply unplug and plug in the Arduino USB cable again.

```
An error occurred while uploading the sketch
Sketch uses 4516 bytes (14%) of program storage space. Maximum is 32256 bytes.
Global variables use 300 bytes (14%) of dynamic memory, leaving 1748 bytes for local variables. Maximum is 2048 bytes.
An error occurred while uploading the sketch
avrdude: ser_open(): can't open device "\\.COM4": Access is denied.
```

## 2.4 Simulator Settings

Line 9-19 of the Matlab simulator will allow the user to modify the simulator settings for the map, goal positions, and robot starting position.

```
9 - task = "A4" ;          %% Task A3(randomized map with two goal) or A4(manually set map with three goals)
10 - randomGoal = true;   %% Spawn random goals on map ?
11 - mapSet = "map\box.png"; %% set the map manually
12 - randomPose = true;   %% Random beginning pose of robot
13 - randomMap = true;    %% Randomize map for A4
14 - OneGoal=false;
15 - %%Assigning goal if randomGoal == false
16 - g1 = [13 6];         % GOAL [x y]
17 - g2 = [15 4];         % GOAL [x y]
18 - %%Assigning Pose if randomPose == false
19 - Pose = [9 7 15];     %Pose [x y theta]
```

### 2.4.1 Map

There are three sets of maps available. The complexity of the map varies between sets. The available maps are (Found in MazeColliders/map/):

- Box map (box.png)
  - An empty box used in A3 for functionality testing.
- A3 Maps (map3\_#.png)
  - Used in A3 for exploration tasks.
- A4 Maps (map4\_#.png)
  - Used in A4 for navigation tasks.

The map can be randomly (within A3 or A4 Maps) or manually set by the user. Setting the “randomMap” (**line 13**) variable will cause the system to:

- If true, it will randomly select a map within the mapset defined by the “task” (**line 9**) variable. A3=A3 Maps and A4=A4 Maps.
- If false, the system will select the map defined in the “mapSet” (**line 11**) variable. You will need to define the path and map name of the desired map to select. E.g. if the intended map is “map4\_1.png”, then the mapSet would be defined as “map\map4\_1.png”.

### 2.4.2 Robot Origin Pose

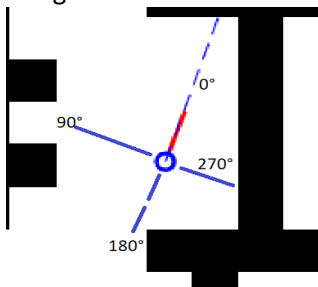
The robot’s origin pose (starting position and orientation) can be manually designated or randomly set. The “randomPose” (**line 12**) variable controls if the robot’s origin position is randomly or manually set. If true, then the robot will be randomly placed with a random orientation on an empty space within the map. If false, the “Pose” (**line 19**), will define the origin pose of the robot. The Pose variable needs to be an array with three elements, [xposition, yposition, orientation]. The values must be an integer within the range of the map.

### 2.4.3 Goal Position

Like the previous two settings, the goal positions can also be set to randomly positioned based on the “randomGoal” (**line 10**) variable. If true, the goals will be randomly placed on an empty space that is a set distance apart on the map. If false, then the system set the location based on the “g1” and “g2” (**line 16 and 17**) variable. The location is defined by the integer array [xposition yposition]. The simulator is set up in a way that g1 must always be collected before g2. Once the robot collides with a goal, it will disappear and count as collected. For A3, the exploration task will require the goal positions to be manually set as the same location.

### 3. Command Guide

List of commands, definitions, example calls.

Command	Definition	Usage
CMD_START	Start the simulation, this command needed to be called to "turn on" the robot	CMD_START
CMD_CLOSE	Clean up all the variable, disconnect the serial port for the next use and "turn off" the robot	CMD_CLOSE
CMD_ACT_LAT_*Direction*_*Distance*	Handles lateral movements of the robot. User must specify the direction (0=backwards, or 1 = forwards) and distance value (in metres)	CMD_ACT_LAT_0_10 This will move the robot backwards 10 metres CMD_ACT_LAT_1_0.1 This will move the robot forward 0.1 metres
CMD_ACT_ROT_*Direction*_*Angle*	Rotate the robot by a specific angle. User must specify the direction and the angle. (0 = Counter Clockwise, 1 = Clockwise)	CMD_ACT_ROT_0_25 This will turn the robot 25 degree counter clockwise CMD_ACT_ROT_1_1 This will turn the robot 1 degree clockwise
CMD_SEN_IR	Return the IR sensor measurement from the simulation in String type, when out of range, the return value will be NaN	CMD_SEN_IR
CMD_SEN_ROT_*Angle*	Adjust the angle of the sensor. 0° is where the robot is heading and increasing to 360 Counter-Clockwise 	CMD_SEN_ROT_180 This will move the sensor to the 180° position (Backward)
CMD_SEN_CHECK	Return the current angle of the sensor	CMD_SEN_CHECK

CMD_SEN_DIST	Return the distance the robot has travelled (robot odometer)	CMD_SEN_DIST
CMD_SEN_PING	Ping the nearest goal, this function returns the distance to the nearest goal if it stays within 5 metre of the goal, otherwise return 0	CMD_SEN_PING
CMD_SEN_ID	Identify which goal is nearby, only work if the robot is within 2 metre of the goal, return 1 or 2 depend on which goal is nearby, otherwise return 0	CMD_SEN_ID
CMD_SEN_GOAL	Check what goal the robot achieved, if the robot has reached goal 1, this will return 1, if reached goal 2 will return 2, if have not achieved any goal, will return 0	CMD_SEN_GOAL
CMD_SEN_COLL	Return the number of collisions between robot and the wall	CMD_SEN_COLL



## 4. Demo Code

This demo code will wait for 2 second and send the start command to turn on the robot, then do some basic navigation and sensor position, takes sensor reading, then the robot will go in a circle. Demo code must be run with map set manually to "map\box.png" and randomMap set to **false**

```
//This will be used on the Arduino UNO + LCD Shield

#include <LiquidCrystal.h>

double incomingByte;

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

void PrintMessage(String message)
{
    Serial.print(message);
    Serial.write(13); //carriage return character (ASCII 13, or '\r')
    Serial.write(10); //newline character (ASCII 10, or '\n')
}

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    lcd.begin(16, 2);
    lcd.setCursor(0, 1);
    delay(2000);          // Wait 2 second
    PrintMessage("CMD_START"); // Start the robot
}

void loop() {
    PrintMessage("CMD_ACT_LAT_1_2 "); // Move Foward 2m
    delay(500);
    PrintMessage("CMD_ACT_ROT_1_30 "); // Rotate Clockwise 30Deg
    delay(500);
    PrintMessage("CMD_ACT_LAT_0_1 "); // Move Backward 1m
    delay(500);
    PrintMessage("CMD_SEN_ROT_180"); // Rotate the sensor to the 45Deg position
```

```
delay(500);  
PrintMessage("CMD_SEN_IR");    // Query the sensor reading  
String incomingByte = Serial.readString(); // Read the incoming value  
lcd.print(incomingByte);      // Print the incoming byte to the lcd shield  
PrintMessage("CMD_SEN_ROT_0"); // Rotate the sensor to the 0Deg position  
for (int i = 0; i <=18; i++) {  
    PrintMessage("CMD_ACT_LAT_1_1"); // Move Foward 0.5m  
    PrintMessage("CMD_ACT_ROT_0_20"); // Rotate Counter Clockwise 30Deg  
}  
PrintMessage("CMD_CLOSE");      // Stop the program and clear the variable  
}
```