Default Program 4: Maze Solving - Arduino

Time required: 180 minutes

Requirements

- Button A: Remote Control
- Button B: Smart Obstacle Avoidance
- Button C: Smooth Line Following
- Button D: Maze Solving

The mBot is not an accurate robot. To maintain accuracy, some type of outside reference is needed. This maze solving program uses line following as an outside reference to keep the mBot driving straight. Smart Obstacle Avoidance is used to navigate and solve the maze.

We will combine line following and obstacle avoidance to successfully navigate a maze. The maze has walls and black tape. By switching back and forth between obstacle avoidance and line following, you can navigate the maze. This project shows how to go back and forth between two sensor inputs.

Build a Maze

Your maze does not have to be complex.

You can make a simple maze similar to what we used in class with boxes for the walls, and some sort of black tape for the line following. 3M Expressions Washi tape, Automotive cloth wiring harness tape, or electrical tape works.

You can use the paper line following track and your hand as a temporary obstacle.

The maze does not have to be big. You just need a line for the mBot to follow and an obstacle.

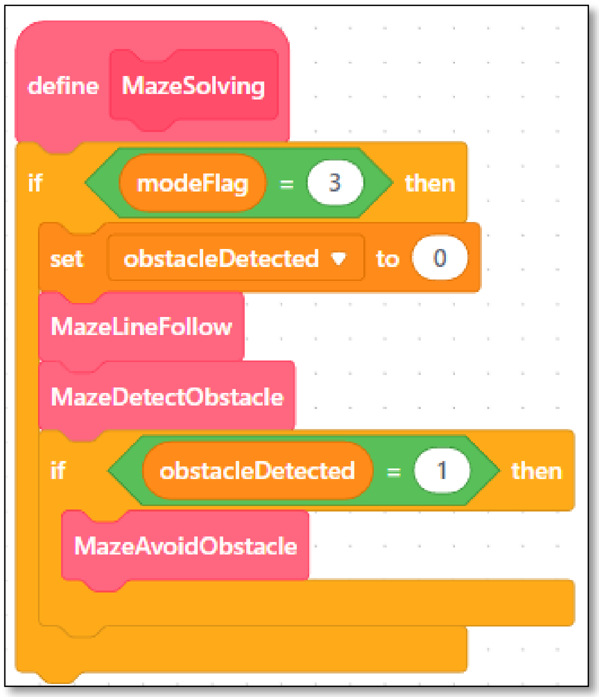
You want to demonstrate that you are following the line and avoiding an obstacle at the same time.

- The source of the idea and a video showing the maze that is used in class. https://www.thingiverse.com/thing:1169585
- https://youtu.be/rhM6JyZMujE (An in class video of 3 mBots solving the maze.)
- https://youtu.be/OvEOZGEdGLM (Another mBot solving the maze.)

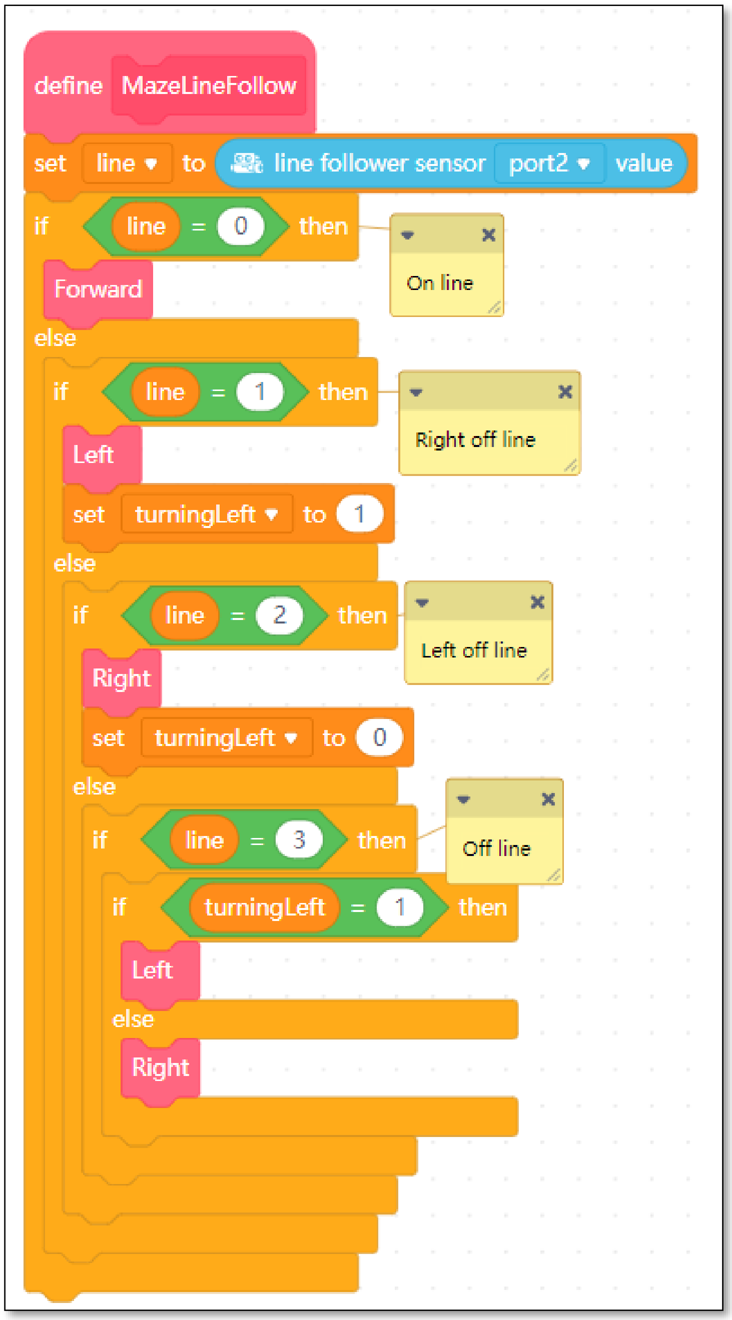
HINT: The mBot ultrasonic sensor distance should stop the robot on the line or just past it before it looks right and left for distance. When it starts looking for the line, it can easily find it.

Requirements

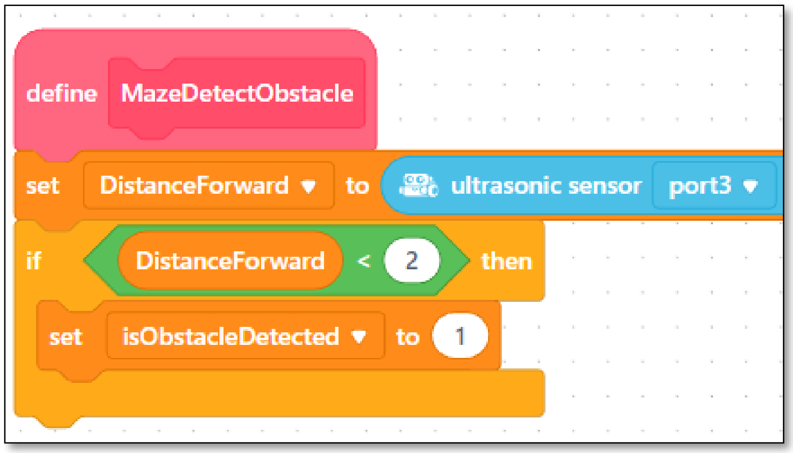
- Start the Arduino IDE. Open **DefaultProgram3 a**nd save it as **DefaultProgram4**.
- Add a **MazeSolving** function with the ir remote button D to activate it. Use the same techniques as previous default programs to add this block to the remote control.
- The **MazeSolving** function with have two functions in it
  - **MazeLineFollow()**
  - **MazeObstacle()**
- **MazeLineFollow** block is a modified duplicate of the **LineFollowing** block.
- **MazeObstacle** is a modified duplicate of **ObstacleAvoidance**.
- The mBot will switch from following the line and avoiding the walls of the maze.
- Find the line again after your robot senses an obstacle. There is new code that turns the robot until it detects the line.
- A mBlock version is shown to give you some ideas on how to solve this problem.



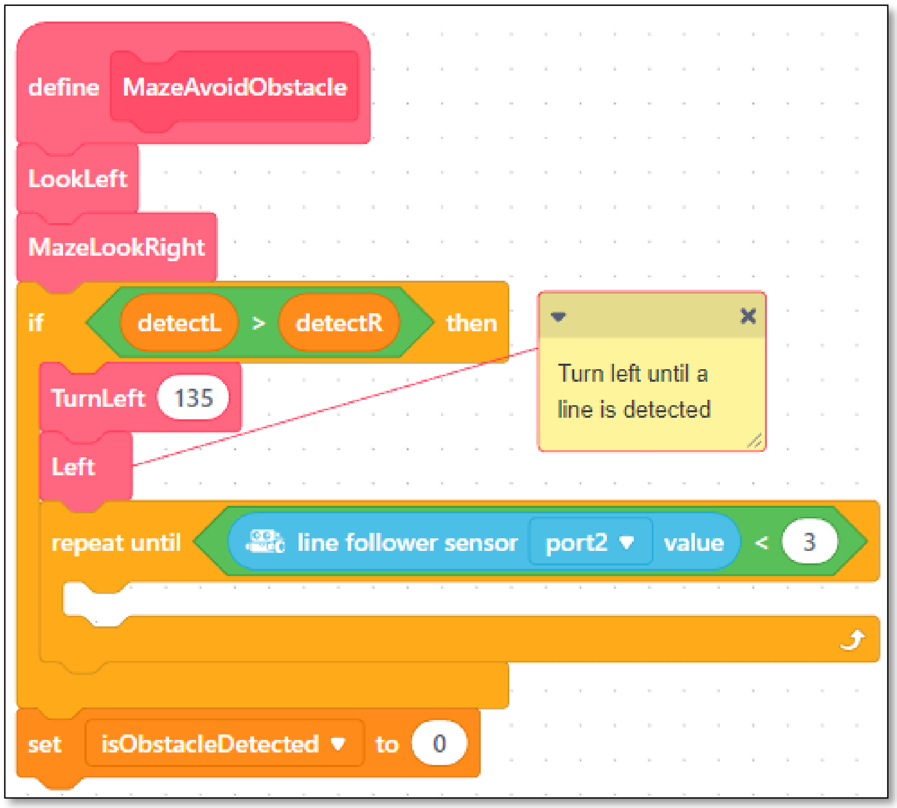
```
define MazeSolving
if modeFlag = 3 then
    set obstacleDetected to 0
    MazeLineFollow
    MazeDetectObstacle
    if obstacleDetected = 1 then
        MazeAvoidObstacle
```



```
define MazeLineFollow
set line to (line follower sensor port2 value)
if line = 0 then          // On line
    Forward
else
    if line = 1 then      // Right off line
        Left
        set turningLeft to 1
    else
        if line = 2 then  // Left off line
            Right
            set turningLeft to 0
        else
            if line = 3 then    // Off line
                if turningLeft = 1 then
                    Left
                else
                    Right
```


```
define MazeDetectObstacle
set DistanceForward to ultrasonic sensor port3
if DistanceForward < 2 then
    set isObstacleDetected to 1
```

The detection distance for the in class maze is 2 inches. You will want to adjust your detection distance for your maze.

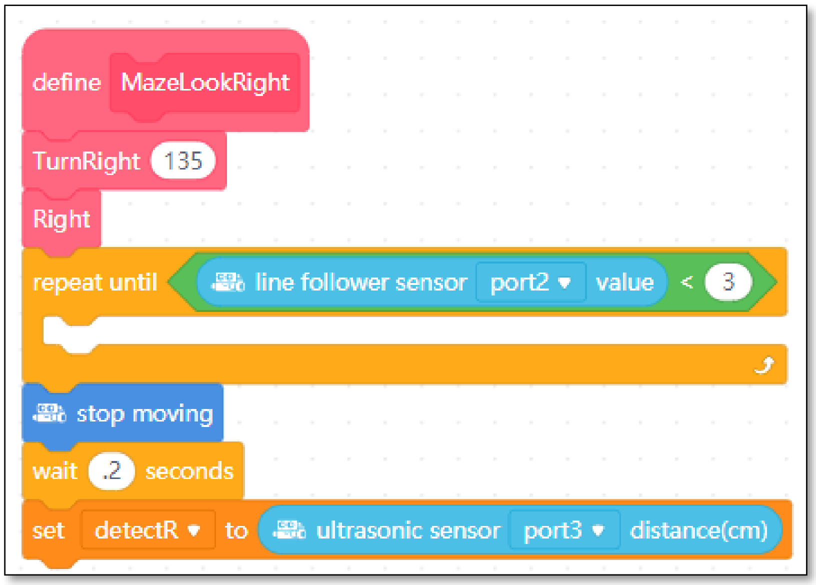

```
define MazeAvoidObstacle
LookLeft
MazeLookRight
if detectL > detectR then
    TurnLeft 135
    Left
    repeat until line follower sensor port2 value < 3

set isObstacleDetected to 0
```

Turn left until a line is detected

Using Lines to Make Accurate Turns

This is the part of the program that uses the lines to make more accurate turns.

MazeAvoidObstacle starts with **LookLeft**. This is the same LookLeft we have used previously. Turn left 90 degrees, take a reading with the Ultrasonic Sensor.


define MazeLookRight
TurnRight 135
Right
repeat until line follower sensor port2 value < 3
stop moving
wait .2 seconds
set detectR to ultrasonic sensor port3 distance(cm)

Instead of turning 180 degrees, **MazeLookRight** turns part way to 135 degrees. The robot then turns right until the line follower detects a line. When the line follower reads 3, that means that it is off the line, as anything less than 3 means that it detected a line. The robot keeps turning right until the line follower detects a line.

Repeat Until Linefollower Arduino Code

```
sensorState = lineFinder.readSensors();
while(sensorState > 2){
    sensorState = lineFinder.readSensors();
}
```

The same turning logic is used to turn left, turn part way, then keep turning until the line follower detects the line.

Assignment Submission

- **All students** → Attach finished programs to the assignment in Blackboard.
- **In class assignment submission** → Demonstrate in person.
- **Online submission** → A link to a YouTube video recording showing the assignment placed in the submission area in BlackBoard.