

Workshop Week 10:

Intro to Maze

Objective:

Learn how to represent a maze in C, and how to set the walls and values of the maze cells. Learn about data structures and how they can be used to represent a maze cell. Print the maze into a terminal to visualize the maze and mouse position.

Background information:

We are now moving into what we consider the second part of micromouse project, the maze/path planning software. In this second part, we will write the code to represent the maze inside the Arduino's memory, scan for walls, find the best neighbor and orient the mouse towards the best neighbor. We will also start thinking about the end goal of the micromouse and working towards finishing the final algorithm of the micromouse project.

As of right now, our mouse can move a maze cell distance and turn accurately to a specified angle. This is great if we only want to move on a straight and turn mindlessly, but we know that the mouse should accomplish a specific task, which is reaching the center of the maze. To do this we need to have a path planning algorithm, and to do path planning we need a map of the mouse environment. The map of the micromouse is simple, 8x8 cells each with 4 walls on it's NORTH, EAST, SOUTH and WEST borders. The cells on the borders can be assumed to always have a wall on its perimeter, so that leave us with mapping the rest of the cells on the maze.

In the interest of time, we will provide you with the code for the maze. You will still be responsible for writing some of the necessary functions to finish the path planning algorithm, but the maze representation and printing functions will be provided to you.

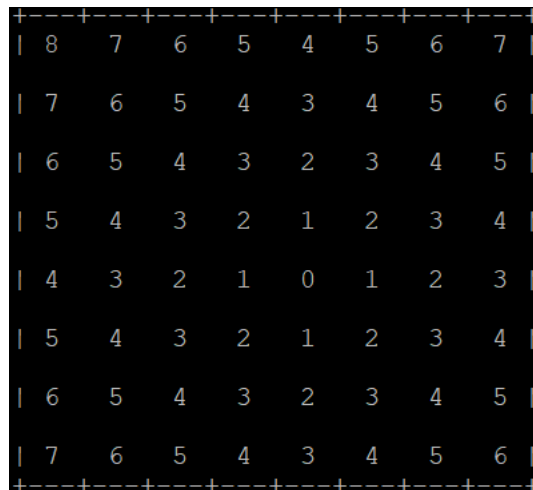
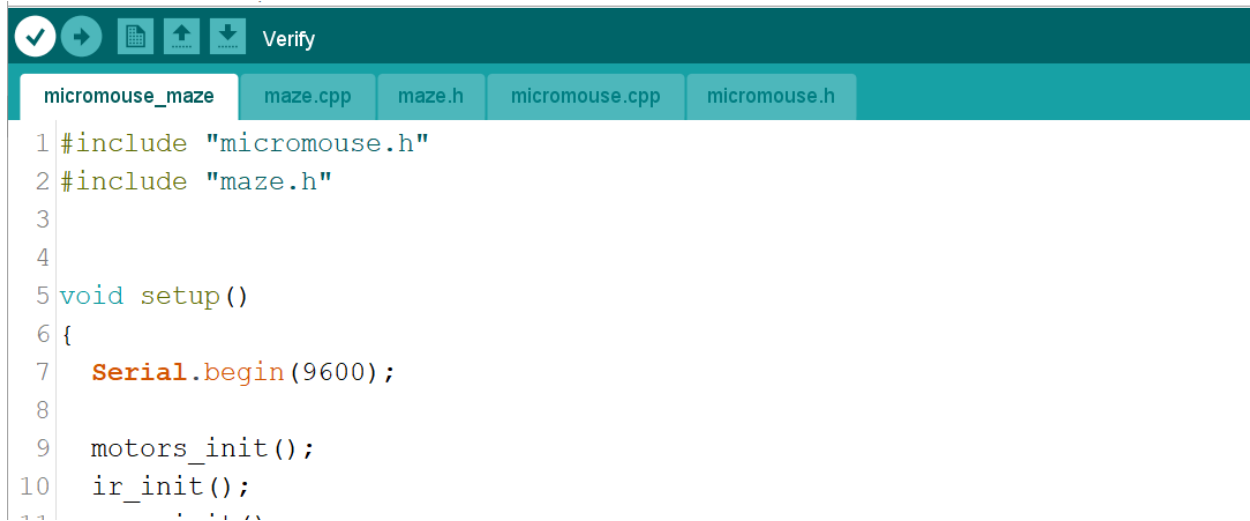


Figure 1 Printed maze

Setup Maze Sketch

To start, [download](#) and copy the maze.cpp and maze.h files into your sketch folder. You might have to close and open the Arduino sketch to see the files. You should see a project structure with the files shown below.



The screenshot shows the Arduino IDE interface. At the top, there's a toolbar with icons for checking, running, and uploading, followed by a 'Verify' button. Below the toolbar, the project structure is displayed with tabs for 'micromouse_maze', 'maze.cpp', 'maze.h', 'micromouse.cpp', and 'micromouse.h'. The 'micromouse_maze' tab is active, showing the following code:

```
1 #include "micromouse.h"
2 #include "maze.h"
3
4
5 void setup()
6 {
7     Serial.begin(9600);
8
9     motors_init();
10    ir_init();
```

Verify and upload the sketch to your Arduino.

Initializing the Maze and Mouse position

Now that you have the maze files into your sketch, we can start using them to initialize the maze, the mouse position within the maze and visualize the maze and mouse position on a serial console.

Write the following code into your Arduino Sketch:

```
1. #include "micromouse.h"
2. #include "maze.h"
3.
4. void setup()
5. {
6.     Serial.begin(9600);
7.
8.     motors_init();
9.     ir_init();
10.    gyro_init();
11.
12.    /*Initialize maze with goal position at the center*/
13.    maze_init(MAZE_WIDTH/2, MAZE_HEIGHT/2);
14.
15.    /*Initialize mouse position at the bottom
16.    left corner of the maze. Initialize heading to NORTH*/
17.    mouse_set_heading(NORTH);
```

```

18. mouse_set_x(0);
19. mouse_set_y(MAZE_HEIGHT-1);
20.
21. /*Print maze. This is where you could use the
22.  * HC-05 to send the maze wirelessly to your laptop*/
23. maze_print();
24. }
25.
26. void loop()
27. {
28. }
29.

```

Upload the code into your Arduino. To visualize the maze download [Putty](#), open it and the select the following options

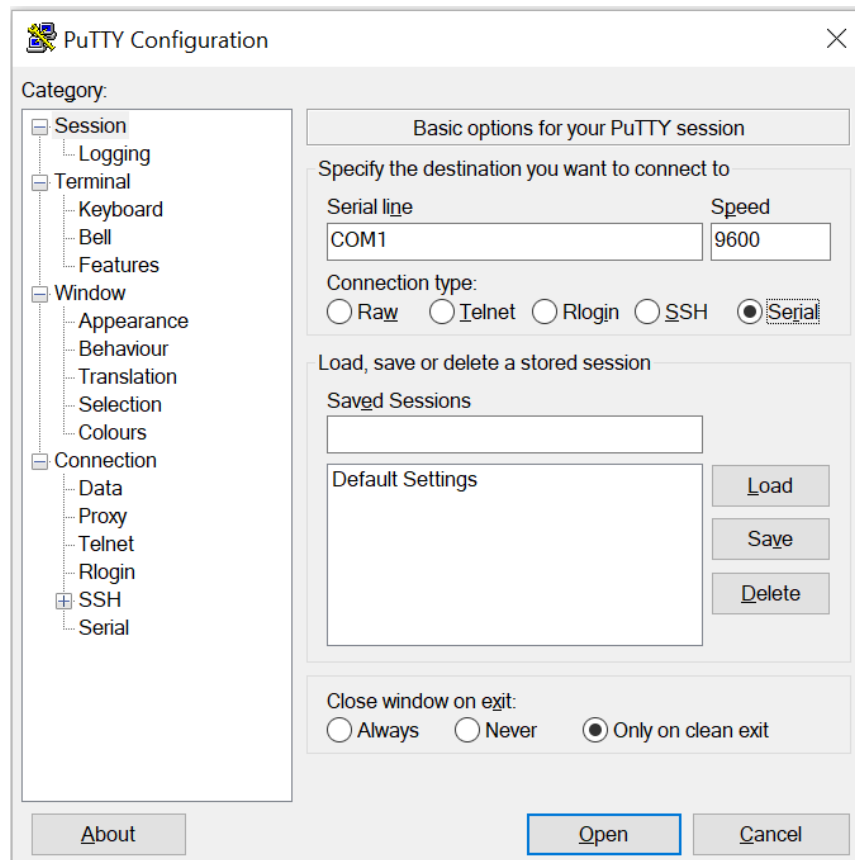


Figure 2 Putty setting for maze print output

Change the COM1 to your Arduino's COM port number. Once you open the serial terminal, you should see the maze with the mouse at the bottom left corner.

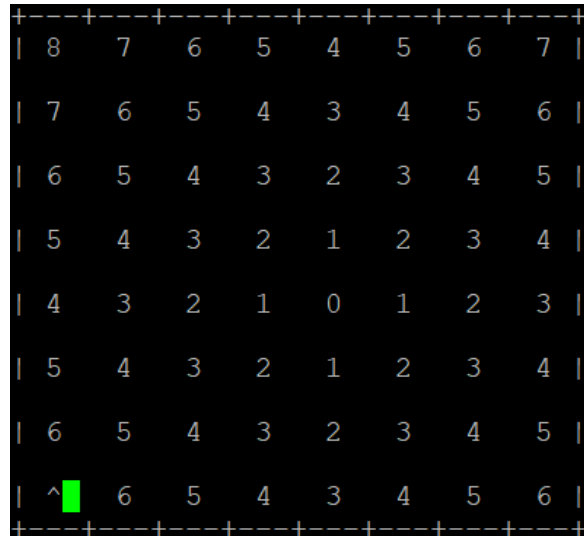


Figure 3 Maze and Mouse visualization. The mouse is represented by a caret symbol

Setting Walls on the Maze

You can set walls on the maze by using the function

```
1. int8_t maze_set_wall(uint8_t x, uint8_t y, uint8_t wall);
```

Where x and y are the row and column number of the maze and wall specifies which wall is to be set. For example, to set row 7 column 0 EAST wall, we can call the function with the following parameters

```
1. maze_set_wall(0, 7, EAST);
```

Write the following program to initialize and set the EAST wall from the initial position of the mouse

```
1. #include "micromouse.h"
2. #include "maze.h"
3.
4. void setup()
5. {
6.   Serial.begin(9600);
7.
8.   motors_init();
9.   ir_init();
10.  gyro_init();
11.
12.  /*Intialize maze with goal position at the center*/
13.  maze_init(MAZE_WIDTH/2, MAZE_HEIGHT/2);
14.
15.  /*Initailize mouse position at the bottom
16.  left corner of the maze. Initialize heading to NORTH*/
17.  mouse_set_heading(NORTH);
18.  mouse_set_x(0);
```

```

19. mouse_set_y(MAZE_HEIGHT-1);
20.
21. /*Set a wall*/
22. maze_set_wall(0, 7, EAST);
23.
24. /*Print maze. This is where you could use the
25.  * HC-05 to send the maze wirelessly to your laptop*/
26. maze_print();
27. }
28.
29. void loop()
30. {
31. }
32.

```

The output of the maze should look as follows:

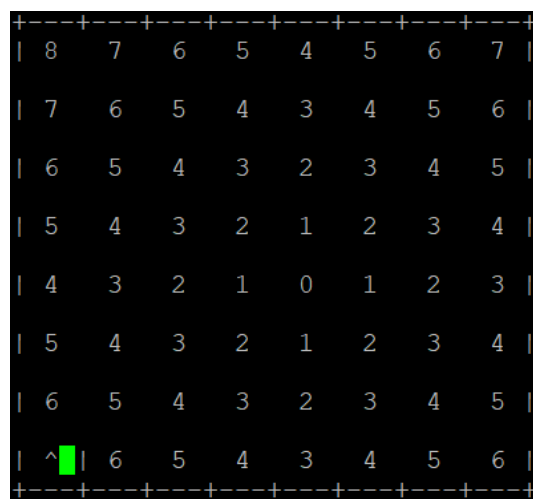


Figure 4 Maze with EAST wall at 0,7

Setting Mouse position and heading

We can also set the mouse position and heading by using the following functions,

```

1. void mouse_set_heading(uint8_t heading);
2. void mouse_set_x(int8_t x);
3. void mouse_set_y(int8_t y);

```

As an example, for the micromouse competition the mouse starts at one of the corners of the maze, so we can set the mouse initial position in $x = 0$ and $y = 7$ facing NORTH. Write the following code to move the mouse to a different position

```

1. #include "micromouse.h"
2. #include "maze.h"
3.
4. void setup()
5. {
6.     Serial.begin(9600);
7.
8.     motors_init();
9.     ir_init();
10.    gyro_init();
11.
12.    /*Initialize maze with goal position at the center*/
13.    maze_init(MAZE_WIDTH/2, MAZE_HEIGHT/2);
14.
15.    /*Initialize mouse position. Initialize heading to WEST*/
16.    mouse_set_heading(WEST);
17.    mouse_set_x(4);
18.    mouse_set_y(5);
19.
20.    /*Set a wall*/
21.    maze_set_wall(0, 7, EAST);
22.
23.    /*Print maze. This is where you could use the
24.     * HC-05 to send the maze wirelessly to your laptop*/
25.    maze_print();
26. }
27.
28. void loop()
29. {
30. }
31.

```

The output of the maze should be as follows

```

+---+---+---+---+---+---+---+---+
| 8   7   6   5   4   5   6   7 |
| 7   6   5   4   3   4   5   6 |
| 6   5   4   3   2   3   4   5 |
| 5   4   3   2   1   2   3   4 |
| 4   3   2   1   0   1   2   3 |
| 5   4   3   2   <  2   3   4 |
| 6   5   4   3   2   3   4   5 |
| 7   6   5   4   3   4   5   6 |
+---+---+---+---+---+---+---+---+

```

Figure 5 Changing mouse position and heading

Exercise:

Have the mouse move accept input from your computer keyboard so that it can move and head NORTH, EAST, SOUTH or WEST. You can accomplish this by using the `Serial.read()` and `Serial.available()` functions. Print the maze output into putty to visualize the position and heading of the mouse.

```
1. #include "micromouse.h"
2. #include "maze.h"
3.
4. void setup()
5. {
6.     Serial.begin(9600);
7.
8.     motors_init();
9.     ir_init();
10.    gyro_init();
11.
12.    /*Initialize maze with goal position at the center*/
13.    maze_init(MAZE_WIDTH/2, MAZE_HEIGHT/2);
14.
15.    /*Initailize mouse position at the bottom
16.    left corner of the maze. Initialize heading to NORTH*/
17.    mouse_set_heading(WEST);
18.    mouse_set_x(4);
19.    mouse_set_y(5);
20.
21.    /*Set a wall*/
22.    maze_set_wall(0, 7, EAST);
23.
24.    /*Print maze. This is where you could use the
25.     * HC-05 to send the maze wireleslly to your laptop*/
26.    maze_print();
27. }
28.
29. void loop()
30. {
31.     while(!Serial.available());
32.     int c = Serial.read();
33.     /*if c == 'w' move NORTH*/
34.     /*if c == 'd' move EAST*/
35.     /*if c == 's' move SOUTH*/
36.     /*if c == 'a' move WEST*/
37.     maze_print();
38. }
39.
```