**Techfest** **IIT Bombay** **e-Yantra**



# Techfest 2021 - International Micromouse Competition

## gmaze - Gazebo Micromouse Maze Generator

**Gmaze** is a tool which can be used to generate Micromouse maze for Gazebo from a text file image of the maze.

To design and generate your own maze follow the following steps,

1. Navigate to `gmaze` folder.
2. In order to make your own maze make a copy of `sample_maze.mz` and name it let's say `my_maze.mz`. Open this file in a text editor. The file contains ASCII text drawing of the maze. You can edit this file as per your requirement.
3. Open up terminal in the `gmaze`folder and type the following command.

   ```
   ./setup.sh
   ```

4. Now `Gazebo` will open up. On top right corner of your Gazebo instance you will be able to see three buttons.
   1. First button is to load your maze text file.
   2. Second button is to generate the maze in Gazebo from the text file.
5. Now press the first button and load `my_maze.mz` which you just created in step 2.
6. Press the second button to generate the maze in Gazebo.
7. Once maze is generated click on the robot at the start of the maze and press `delete` to remove the robot from arena.
8. After deleting the robot, click on File -> Save World As and save the world as `my_maze.world`.
9. Now go to `pkg_techfest_imc/world` folder and move the `my_maze.world` there.
10. Now go to `pkg_techfest_imc/launch` folder and open `gazebo.launch` in a text editor.

11. Change the following line,

```
<arg name="world_name" value="$(find
pkg_techfest_imc)/world/arena.world"/>
```

to

```
<arg name="world_name" value="$(find
pkg_techfest_imc)/world/my_maze.world"/>
```

# pkg_techfest_imc - ROS Micromouse Template Package

This ROS Package consists of four main folders which you can edit as per the rules of the competition.

1. launch folder : In this folder there are 3 launch files, one for launching rviz visualization environment which is **rviz.launch**, the other one is to launch gazebo simulation environment wihch is **gazebo.launch** and the last one is to launch both the enviornment simultaneously which is **final.launch**.

2. scripts folder : In this folder we have to keep all our pythons codes in which we are going to write our algorithms. For example **cmd_vel_robot.py** is the file where all the sensors are actuated.

3. src folder : In this folder you can keep all the cpp files where if you dont want to write your algorithms in python you can use cpp language as an option as well.

4. urdf folder In this folder we have kept the main urdf_file with the maze included in it. In this urdf file you can edit the file to customize the robot within the competition rules.

## To run the rviz visualization run this command:

```
roslaunch pkg_techfest_imc rviz.launch
```

## To run the Gazebo simulation run this command:

```
roslaunch pkg_techfest_imc gazebo.launch
```

## To run the both visualization and simulation simultaneosuly run this command:

```
roslaunch pkg_techfest_imc final.launch
```

## To run any python file run this command:

```
rosrun pkg_techfest_imc cmd_vel_robot.py
```

If you are creating new python file make it executable first before running only once after creating the file, this is the command, First go to the directory in the terminal,

```
cd <your_ws>/src/scripts
```

Then make the particular python file executable for example you created a new file named as "control_robot.py",

```
chmod +x control_robot.py
```

### To run any Cpp file run this command:

We have to make cpp also executable by adding few lines in **CMakeLists.txt** file, for reference visit this link, "http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29#roscpp_tutorials.2FTutorials.2FWritingPublisherSubscriber.Building_your_nodes-1"

For example you create control_robot_cpp.cpp file in src then,

```
rosrun pkg_techfest_imc control_robot_cpp
```

### If you are using python code we have provided sample codes for your reference,

1. cmd_vel_robot.py : In this file we have implemented 2 sensors which are odometry and Laser scanner,

   ```
   position_ = msg.pose.pose.position
   # gives x and y distance of the bot
   x_dist = position_.x
   y_dist = position_.y
   ```

   In this file we have first stored the position in **position_** variable and then we have extracted x and y distance travelled from the starting position(starting position is not at (0,0)).

   ```
   quaternion = (
       msg.pose.pose.orientation.x,
       msg.pose.pose.orientation.y,
       msg.pose.pose.orientation.z,
       msg.pose.pose.orientation.w)
   ```

```
euler = transformations.euler_from_quaternion(quaternion)

yaw_ = euler[2]
print(x_dist, y_dist)
```

Now we have just stored the quaternion values and then converted it to euler angles but we only wanted yaw angles thus we have stored yaw angles in **yaw_** variable.

```
region = {
'p' : msg.ranges[:],
}
# region['p'][0] represents the 0 degree and 0the value start from
back and continues in anti-clockwise direction
for i in range(360):
    print region['p'][i]
```

Now in this clbk_laser function we are storing the laser scans in a region list and then printing each range value of the leaser scan. 0th degree is at the back of the robot, 90 degrees on the right and so on.

You can use these variables in any other function by making it global variable.

```
while not rospy.is_shutdown():
    msg1 = Twist()
#positive speed_z value represents clockwise angular velocity of the
bot and positive speed_x value represents forward linear velocity of
the robot
    speed_z = 0
    speed_x = 0.1
    msg1.linear.x = speed_x
    msg1.angular.z = speed_z
    pub.publish(msg1)
    rate.sleep()
```

Now to give command to the robot we have to set 2 variables that is speed in x direction and angular speed in z direction i.e. yaw. **speed_z** is the variable where you have to set the angular velocity and **speed_x** is the variable where you have to set the linear velocity. Other than this we have just created a message twist and then changed the linear and angular velocity and then we published the message.

There are other sample codes as well you can have a look at it, these are the fundamental functions which will be used in this competition.

You can also use ROS_CONTROL Package provided by ROS just read the urdf file and uncomment the plugin and use position_controller.py for further references.