Implement a BUG2 algorithm to move the GoPiGo from a designated starting point to a goal point in an environment. Your robot should start out along the straight line path from start to goal (the M Line). Using the ultrasound sensor, if it sees any obstacles along the way, it will invoke a perimeter following behavior until the M Line is reacquired and the robot again starts out for the goal point along the M Line. When the goal point is reached, the robot will stop.

Assume a trajectory from a start point to a goal point about 3 meters in front of the robot. Add 2-3 test obstacles (garbage cans and cardboard boxes work well here!) that impede the path, requiring the GoPiGo to invoke BUG2 behavior.

Your program should also map out your robot’s progress graphically, showing robot position and robot orientation (robot pose) as it moves along (odometry). You should also display on the map locations where the ultrasound sensor has seen an obstacle. A quick tutorial on python plotting using matplotlib (installed on your RaspberryPi) is here: http://matplotlib.org/users/pyplot_tutorial.html

Note: this mapping may slow down your communication and affect performance of your robot. If you are experiencing this, then cache the pose and obstacle information and map it out after the robot stops.

Video your robot as it does its movement from start to finish, avoiding obstacles. Post the video to youtube and include the link in your handed in README file.

Note: Your BUG2 algorithm should also be aware when it is trapped inside of an obstacle and report this. Make sure you test this part of your algorithm as well (no video required of this).

Hand in the usual code and README files, along with the link to the robot video on youtube.

Extra Credit: (10 points):

**Robotic Mapping:** Create a 2-D occupancy grid for your robot environment, and map it out with cells that are either occupied or empty. You might think of this as how a vacuum cleaner robot “learns” its environment so it can remember where to clean the floor and where obstacles are. Initially all cells in the grid are empty, and as the robot navigates the environment, using its ultrasound sensor, it fills in the grid spaces as free space or obstacles as they are sensed. The robot’s behavior should be such that it covers the room area. You might try combinations of behaviors such as: random, in that it goes in a direction that is randomly chosen until it hits an obstacle, at which point it changes its behavior to continue exploring the
environment; **Perimeter Following** of an obstacle, and **Spiraling** to cover an area. You can use any behavior you like, just output an occupancy grid map of the environment when the program finishes (see figures below). Some details:

1. Assume a grid cell size of the diameter of the create robot.
2. Make sure you create a bounded environment so the robot has a finite area to map (something like a rectangle 2-3 meters on a side). You can create your own obstacles or get an obstacle set of small plastic road cones at the lab to use as obstacles.
3. Stopping conditions: if your robot has not updated any new cells after a certain amount of time has passed, you should consider stopping.
4. Take a photo of the environment you are mapping and include it with your map so we can see how well your mapping went.
Output occupancy grid showing free and occupied cells