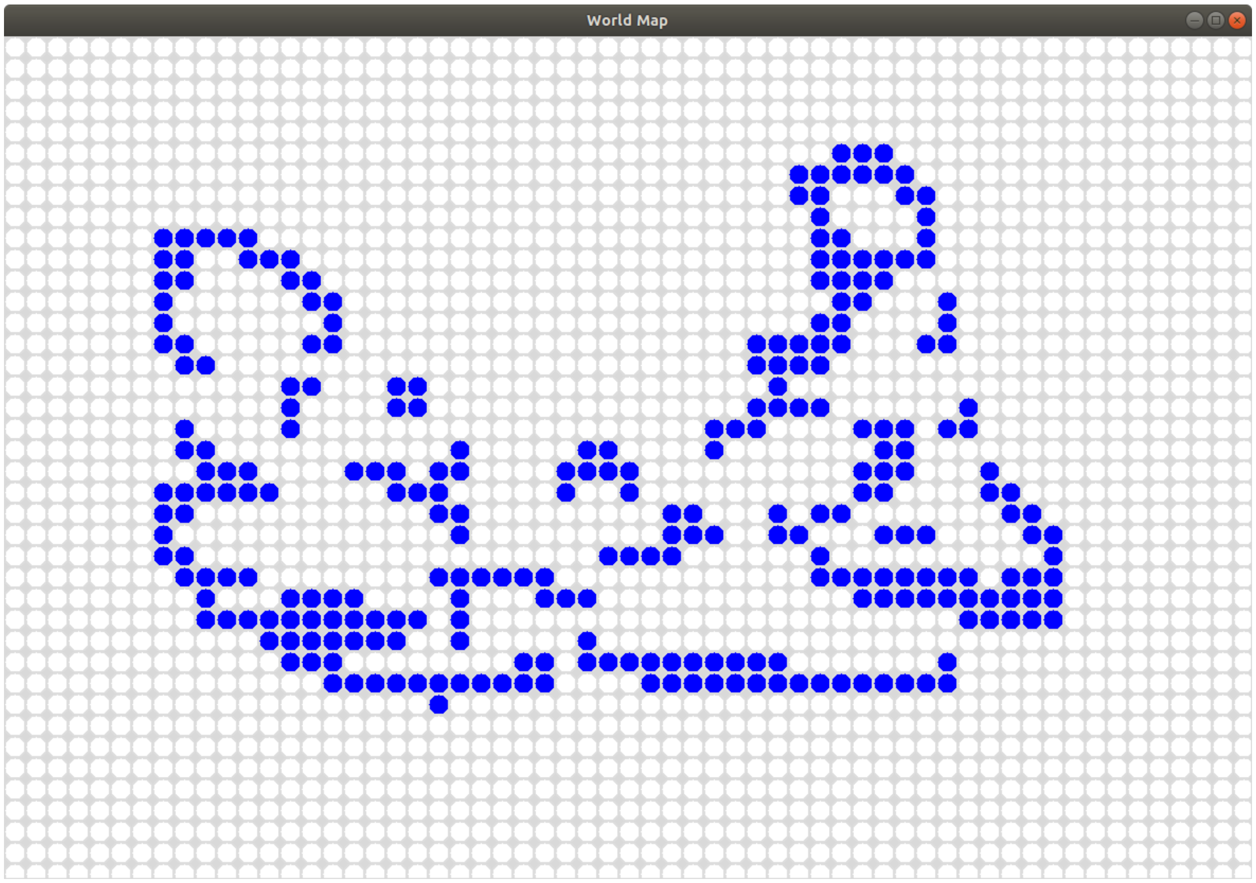
Buff Em' Up
Angel, Miguel, Nebiyu, Stefano
Lab 4: Mapping
March 8th, 2020

1. The drawback of a data structure that stores distances between every possible pair of nodes in the graph and that it does a blind search which will waste significant time. Also, for example, using Dijkstra's Algorithm isn't as beneficial because it doesn't check for any negative edges and can also cause the algorithm to not be able to find the best shortest path. The implementation in 4 -2 addresses this problem by providing a cost-effective way that will check the move between two different cells, rather than looking at every possible node. In comparing
2. Subscribers and Publishers
    A. The subscribers in our code are the:
        i. Subscriber_odometry
            1. This subscribes to the Pose2D messages. This contains information for x, y, and theta.
        ii. Subscriber_state
            1. This subscribes to the messages that has the dictionary containing the info theta (ultrasonic sensor), ping distance, and the 5 value array. The types of messages it is subscribing to is type String.
3. The publishers our code has:
        i. Publisher_motor
            1. This publishes 2 values in an array, hence type Float32MultiArray. Those values are between [-1.0, 1.0] to determine the left wheel and the right wheel speed.
        ii. Publisher_odom
            1. This publishes the odometry for the Sparki. It publishes a type of Pose2D with x, y, theta.
        iii. Publisher_ping
            1. This publishes an empty type message. It tells the Sparki to ping. It includes the result from ping in the next subscriber_state broadcast.
        iv. Publisher_servo
            1. This publishes a type of Int16 where you set the servo angle between [-80, 80].
        v. Publisher_render
            1. This publishes the simulation. It takes a type of empty to render after it pings.

4. Resulting Final Map:



Although you can kind of see the path along the edge, there is somewhat of a mismatch between our resulting map and the given map because it doesn't fully detect the obstacle during execution. This is because we can only measure an approximation of the obstacle. A way we can possibly make it better is by having the robot stop every time it starts detecting an obstacle. When it is stopped we can have the robot rotate some X degree to make sure it detects the whole obstacle. From there, it would continue along the path.

5. We like ROS. It gives us an inside on how A.I and human input works to make an object do what we want and be able to get information from the way it moves. There is a lot of functionality with ROS and sometimes it may be complicated to learn.
6. We spent about 18 hours within the last 2 weeks programming. We had more trouble in being able to implement the right cell matching index when we want it to print to the map. But we got it to print close enough. We also had some trouble at the beginning of the lab trying to begin everything in terms of the loop and messages (msg.data).