HW2: Individual Assignment
Manuel Agraz Vallejo

I built a total of 4 ros nodes. A waypoint publisher, a simple waypoint follower, the VFH follower, and a map publisher. The waypoint publisher reads a YAML file containing the waypoints and publishes them as a Pose Array only once. Then the simple waypoint follower has a proportional controller to control both linear and angular velocity. The map publisher isn't actually publishing a map; it creates a static transform at the world's origin so that the waypoints are properly positioned. Finally, the VFH follower is implementing a simplified version of VFH without the Cartesian histogram. I used the polar histogram as well as the VFH cost function to determine which sector to follow. This took some tuning, especially for the safety distance and to consider the robot radius by "inflating" the obstacles and essentially choosing sectors further away from the current best (which would often be too close to the obstacle).

The VFH algorithm worked pretty well on my first world, where the obstacles were spread out. However, when testing it on the more structured world with rooms, it would struggle if the waypoint was close but right behind a wall. Since it wasn't completely trapped, the robot would keep oscillating along the wall to find a better route. To deal with this, I added an intermediate waypoint that would let it get out of the room and continue to the other waypoints. In the end, I learned that obstacle avoidance is very much local, which means that without any notion of a global plan, it is hard to escape situations like being trapped in a room. When testing on the real robot, since we had no world simulation, it was also quite difficult to set up waypoints, as I had no idea how far was too far until I saw the robot try to go into a waypoint inside an obstacle. Finally, I learned to be aware that a lot of the parameters that work in the simulation will not necessarily suit the actual robot; some examples of this are the robot radius, safety radius, linear and angular speeds. Which meant that I had to set slower speeds on the actual robot than in simulation.