**Lab 5 Report**

1. Darrien Lee, Daniel Kim
2. We start with the “get\_path” state machine to figure out a path from current location to target location. Once it figures out the path, it will move on to “get\_waypoint” to get the next location to move to, which is given from the path. After figuring out the next location from the path, it will transition to “move\_to\_waypoint” state machine which makes the epuck move to that next location, and the return to the “get\_waypoint” state to figure out the next location to move to. The state machines “get\_waypoint” and “move\_to\_waypoint” will continue to repeat until the epuck reaches the target location. Once it reaches the target location, it will return to the “get\_path” machine state.
3. If an object is found in the path of the robot, then it would mark the location as being “filled” or having an object there, and plan again using dijkstra’s algorithm to find the new path around this object from the current location.
4. We could use the RRT algorithm to generate a viable path as well. The path would look a lot more fluid and smooth over the dijkstra’s algorithm as the Dijkstra’s algorithm will try to find the greediest path to the target point, whereas RRT would be able to find more sub-optimal routes that get better over many iterations. However, with the RRT algorithm running for long enough, it will find a path that would take less major turns (90 degree turns) than Dijkstra's algorithm, and instead be a more gradual slope around objects and to the target.
5. We spent roughly 5 hours on this programming lab.