### **CSCI 3302 - Introduction to Robotics**

### **12 October 2017**

### **Nikolaus Correll**

Cora Schneck, Kylee Bennett, Max Messenger Bouricius, Anthony Hauger, Douglas Allen

# **Lab 3.3: Navigation**

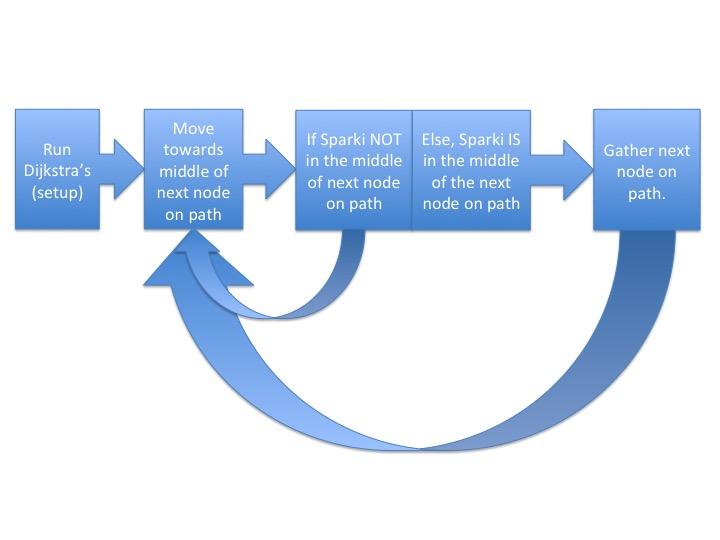
[*Github Link to Live Repo*](https://github.com/cschneck/Sparki_CSCI3302/tree/master/Labs/Lab2)

**INSTRUCTIONS:**

1. **Write code that calculates an X-Y coordinate in the real world based on the I-J coordinates of your 2D map. This coordinate should be the center of each grid cell.**
2. **Implement a state machine that pulls an entry from the route, calculate its X-Y coordinate and then use feedback control to drive there. Pull the next coordinate of the stack once the robot is close enough to the last waypoint.**

**Deliverables  
Provide a write-up describing your approach including a picture of the state machine you implemented. Answer the following questions: (1) How did you chose the distance at which you pull the next waypoint? What happens if you do this too early or too late? (2) What do you need to do should an unforeseen obstacle appear? Try to use tools/algorithms from previous exercises to solve this problem.**

**Picture of State Machine:**



**Answers:**

1. **The distance we chose to pull the next waypoint is the center of the next grid. We decided upon this because our “goal node” as we travel the path to the actual goal node is the center of the next grid on the shortest path. Once we have reached the center of the next grid on our shortest path, we can tell Sparki to move to the next grid’s center along the shortest path. Sparki is guaranteed to reach that point as our inverse kinematics will ensure that Sparki hits that center and will therefore pull the next waypoint. If you do this too early or too late, it won’t be able to turn fast enough to continue on its current path, so Dijkstra’s algorithm will need to be run in order to recalculate the path from our current position (chances are if we’ve done this too late we’ve run into a new grid space).**
2. **As Sparki is “looking” for objects using the Ultrasonic Sensor, Sparki will be able to detect if an object is placed in the map space. Once detected, we would put the obstacle in the correct grid space and we will rerun Dijkstra’s algorithm in order to find the new shortest path (if a new shortest path is needed or if a new shortest path is possible).**