ILR 11 – Progress Review 12

Dorothy Kirlew

Team Daedalus Members: Mohak Bhardwaj, Shivam Gautam, Pranav Maheshwari, and Richa Varma

April 11, 2016

1. Individual Progress

For the progress review on April 11, Richa and I completed the first step in achieving our reach goal of deciding on the optimal route for a vehicle.

a. Optimal Route

Thus far, the optimal parking spot has been selected purely by choosing an empty parking spot with minimal distance to the exit. We have used the built-in path planner in the platform to move the vehicle throughout the lot based on waypoints. Richa and I implemented an algorithm that will choose the optimal spot based on the distance from the spot to the exit and based on the current and future traffic in the parking lot. That is, the selected route will avoid the routes of other vehicles in addition to taking the shortest route possible. This required modifications to the ROS and XBee communication infrastructure, as well as a thorough understanding of the A* planning algorithm.

i. Communication Modifications

In order for a vehicle to plan a route dependent on the routes of other vehicles, these routes must be known and communicated between the vehicles. I modified the actions taken when a ROS message is received and what information is sent between vehicles.

I first modified the actions taken when a ROS message is received (Figure 1). When a Park command is received, the XBee will now send the spot options and the waypoints of other vehicles to the A* Planner. When the optimal spot and waypoints (or route) to the spot are received, an update is sent to other XBees so that they know what waypoints the vehicle is traveling to. The waypoints are also sent to the Navigation node so that it can begin executing a trajectory. A similar course of action is followed when a return command is received, with the exception that the spot options are negligible, as the destination of the vehicle is known. When a Parked notification is received, the vehicle now deletes its personal waypoints from the waypoint dictionary, as it is no longer on that path.

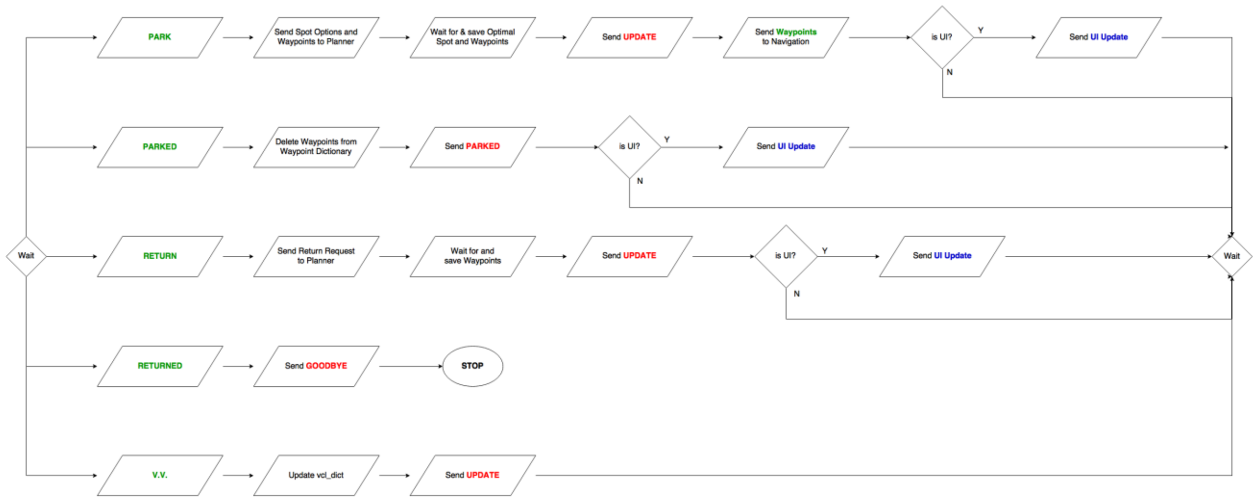

Figure 1: ROS Messages Communication Architecture

I then modified the XBee communication architecture (Figure 2). When an INTRO message is received from another XBee, the waypoint dictionary will be updated to include the waypoints that were received. The other significant change to the architecture is that when an UPDATE message is received, the XBee will update the waypoint dictionary only if the vehicle sending the message is moving, i.e. parking or returning. When the vehicle receives a PARKED or RETURNED message from another XBee, it deletes the waypoints of that XBee from the waypoints dictionary.

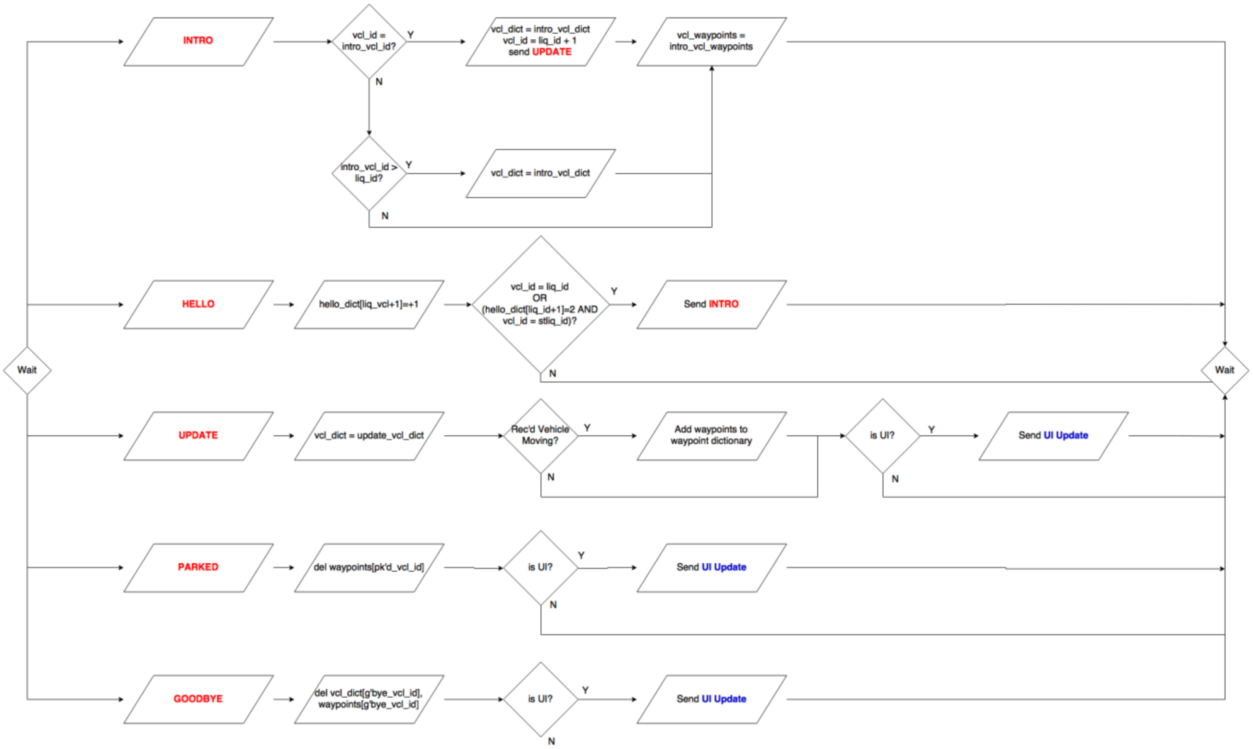

Figure 2: XBee Messages Communication Architecture

ii. A* Implementation

Richa and I created an algorithm to choose the best parking spot based on the available parking spots and the paths of other vehicles in the parking lot. The first step is to choose the 10 best available spots in the parking lot. This is based on proximity to exit. Then, the A* algorithm is run on each of these spots. The algorithm returns the waypoints to the spot and the f-score of the path. The spot with the smallest f-score is chosen as the optimal spot and the spot and the waypoints are returned.

The g-score and h-score of the A* algorithm were modified to fit the context of our system. Traditionally, the g-score is simply the distance from the start to the current node in the algorithm. Generally, the distance between each node is the Euclidian distance. With only 4 directions of movement, this can be modeled in a distance map of 1s (Figure 3). We modified this distance map by overlaying the paths of the vehicles, thereby increasing the cost by 1 for each location on the map that the path covers (Figure 4). This means that the distance map may be different, as it depends on the paths of the vehicles moving in the parking lot. We also added a corner cost. Turning can often take the platform some time and introduces error to the system. Because two paths can be of equal length, but with varying numbers of turns, we implemented a corner cost of 1 to encourage the algorithm to make only necessary turns.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 3: Traditional Distance Map

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2

Figure 4: Distance Map with Path Overlaid. Example Path is from (18,18) to (18, 2) to (4,2) to (4,4)

The h-cost of the A* algorithm is the distance of the spot to the exit. There is a subtle but significant difference here. Traditionally, an easy h-cost is the distance between the current node and the *goal*. Instead, we are using the distance between the current node and the *parking lot exit*. Although this may seem simple, it is an imperative part of the algorithm when selecting the best spot of 10. If the purpose of the algorithm was to simply choose the shortest path, regardless of the proximity of the spot to the exit, the algorithm would choose the parking spot closest to the entrance of the lot. Having the heuristic cost of distance to exit means that two spots with equal path length will have a different cost due to their proximity to the exit. The h-cost map can be seen in (Figure 5).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36

Figure 5: Heuristic Cost Map

When the algorithm completes, it has a path composed of every node that was passed through. We then "shorten" this path, i.e. reduce the path nodes to only the start, goal, and corners. This creates simple waypoints for the platform to navigate to, rather than telling the platform to move straight multiple times.

2. Challenges

I faced no challenges for this PR. However, there were difficulties faced on the physical side of the system. A wire was loose on the platform, which took some time to discover. The work on simulation created challenges typical to any software work.

3. Teamwork

Pranav, Mohak, and Shivam worked on the simulation environment, including global and local planning and the simulator. Shivam and Pranav worked on navigation with the physical platform. Richa performed the final fixes on the original platform to get it in working order. Richa and I created and implemented a planning algorithm on the XBees.

4. Future Plans

Pranav, Mohak, and Shivam will put finishing touches on the platform. Richa and I will integrate the UI with the new A* planner in order to simulate more complex environments. This will also include integrating the planner with navigation. We have received our third and final platform and will set that up to perform in the SVE.