ILR 10 – Progress Review 11

Dorothy Kirlew

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

March 30, 2016

1. Individual Progress

For the progress review on March 30, I designed and built the final parking lot. I worked with Richa on the software and hardware of the original platform. I also upgraded the XBees and, more significantly, the UI.

a. Parking Lot Design and Construction

The parking lot has been slightly modified from the original 5x5m design to have two 50x50cm protrusions for the exit and entrance. This is due the nature of the Oculus Prime platform; it moves forward slightly before starting its trajectory. Thus, if it begins or ends in a protrusion, the trajectory will both start and end inside the 5x5m lot. The design can be seen in Figure 1, with details below. The parking lot was created to be sturdy and collapsible, with the added requirement that it must be the same shape upon each reconstruction. Thus, the final parking lot is made of plywood with 2x4 supports. Instead of cardboard, which does not hold up well when it is moved or bent.

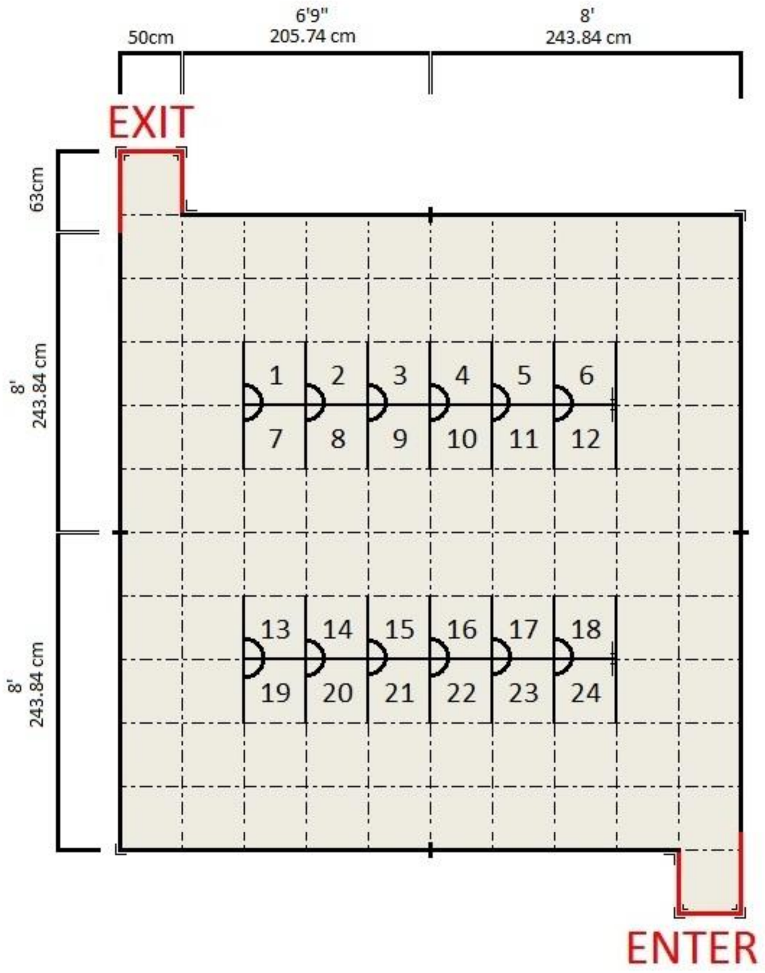


Figure 1: Parking Lot Design (Not to Scale)

i. Dimensions[1] and Parts

As mentioned above, the lot is 5x5m, excluding the protrusions. The outer walls are made from four sheets of 4’x16’ plywood cut into six 2’x8’ sheets and two 2’x6’9” sheets. These sheets can be seen in Figure 1, which is rotationally symmetric. Each protrusion is made from three pieces of plywood; two 2’x20” (~50cm) pieces and one 2’x24” (~63cm) piece. Because the two 8’ walls do not quite add to 5m, an extra 13cm was added to the one length of the protrusion to make up for this.

The parking spot walls are more straightforward. Each spot is 50x50cm. The shared wall between the rows of the spots (between 1 and 7, 2 and 8, etc.) is made of two 2’x150cm sheets of plywood. This splits each 2x6 spot section of spots into two 2x3 sections of spots. The shared wall between sequential spots (1 and 2, 2 and 3, etc.) is made of a single 2’x50cm sheet of plywood.

ii. Piece Assembly

Each wall of the parking lot is made from a single sheet of plywood with two supports, each using three pieces of 2x4s. The supports are spaced ¼ of the way in from each side for an evenly distributed system of support (Figure 2).

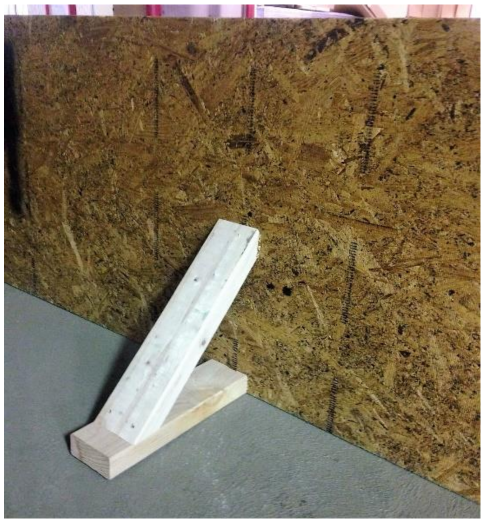


Figure 2: Wall Support

I originally designed the protrusions to use two inner-angle brackets spaced 6” from the top and 6” from the bottom of each corner (Figure 3). The brackets were intended to ensure a 90° angle. However, the inner angle brackets were of poor quality and caused the angles of the

[1] A key piece of information when pondering the dimensions of the lot is that it was designed using the metric system. Unfortunately, the materials were all purchased in America, so the imperial system was used during construction, thereby forming some dimensional oddities.

protrusion to be significantly less than 90°. To fix this, I attached an exterior angle bracket in the center of each corner (Figure 4). Although it is not perfectly 90°, it is significantly better than the original (Figure 5).

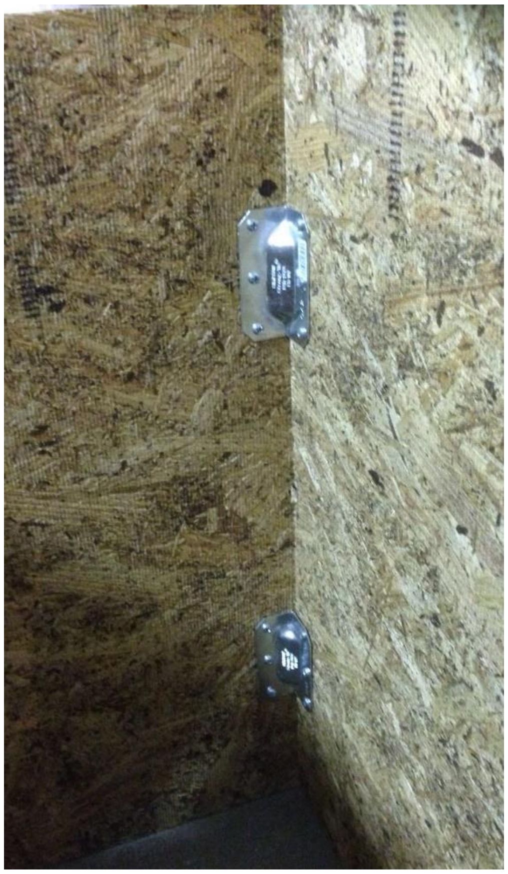


Figure 3: Interior Angle Brackets of Protrusion

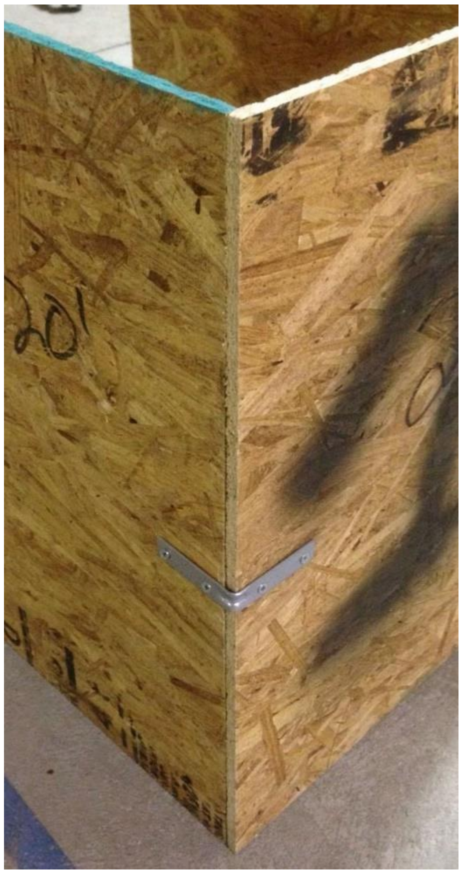


Figure 4: Exterior Angle Bracket of Protrusion

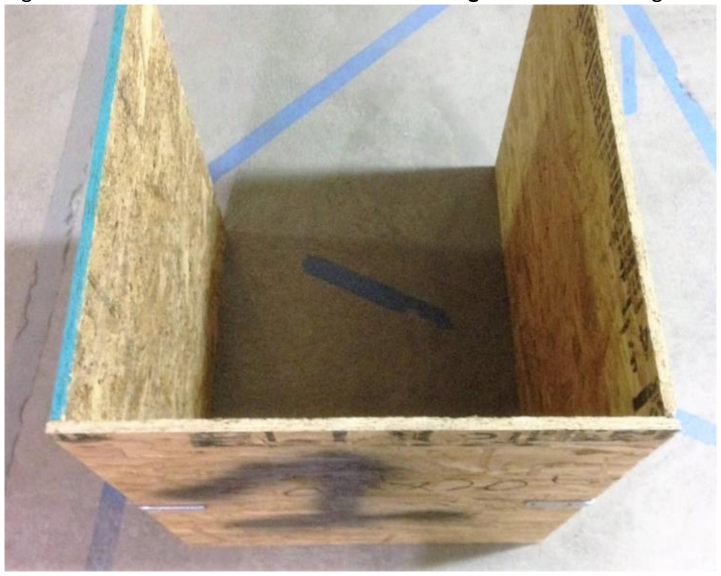


Figure 5: Entrance Protrusion Viewed from Above

As mentioned above, the walls for the 24 parking spots are made of 4 sections made of 2x3 spots (1, 2, 3, 7, 8, and 9, for example). The wall shared by the spots is one sheet of plywood. The left wall of the spot is attached to the shared wall by two hinges, spaced 6" from the top and 6" from the bottom (Figure 6). By hinging the spot walls, they can be folded against the shared wall to make for easy storage. Except for the four right-most spots (6, 12, 18, and 24), the right wall of the spot is the left wall of the next spot. For these four edge spots, two dowels are pushed through the shared wall, spaced 6" from the top and 6" from the bottom. The right walls of the spots slide onto the dowels using pipe brackets (Figure 7). When two sections of 2x3 spots are set up next to each other, they form 12 of the 24 spots (Figure 8). The spots can be collapsed and stored easily in order to occupy as little space as possible (Figure 9).

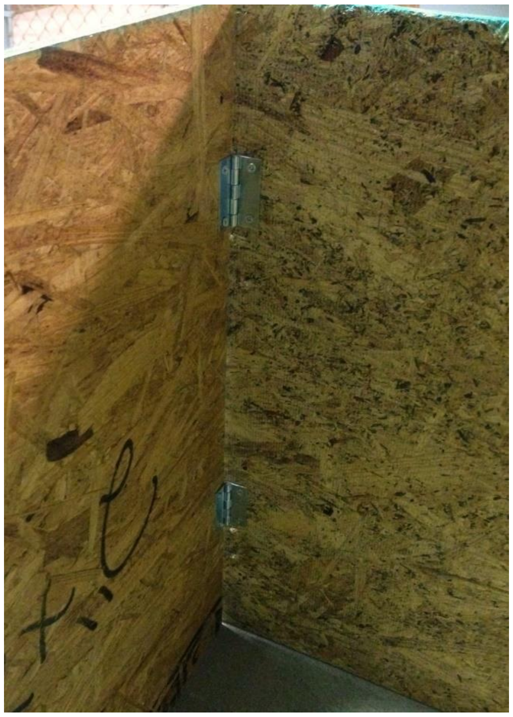


Figure 6: Spot Hinges

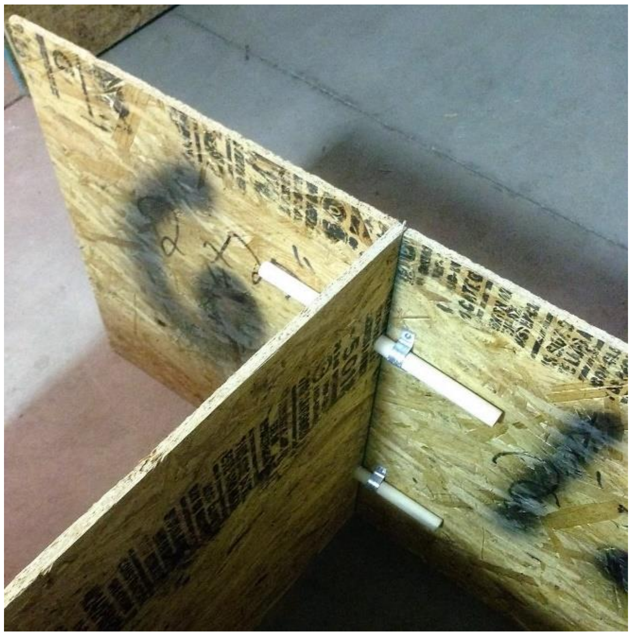


Figure 7: Spot Dowel Connectors

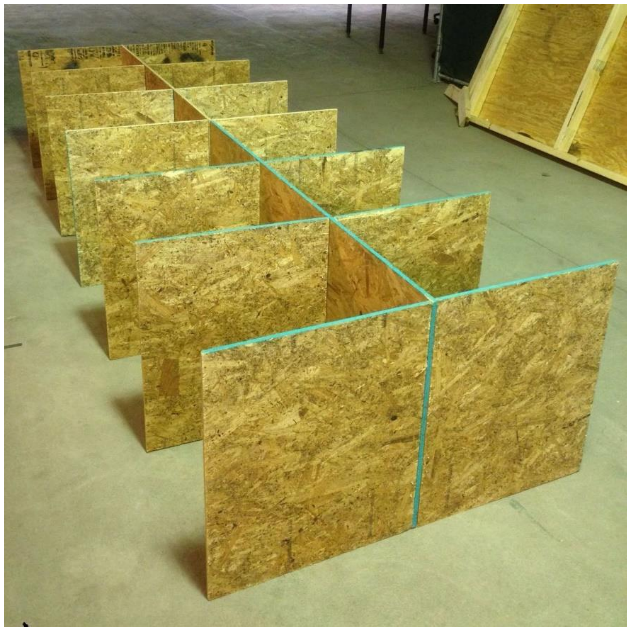


Figure 8: Assembled Spots (12)

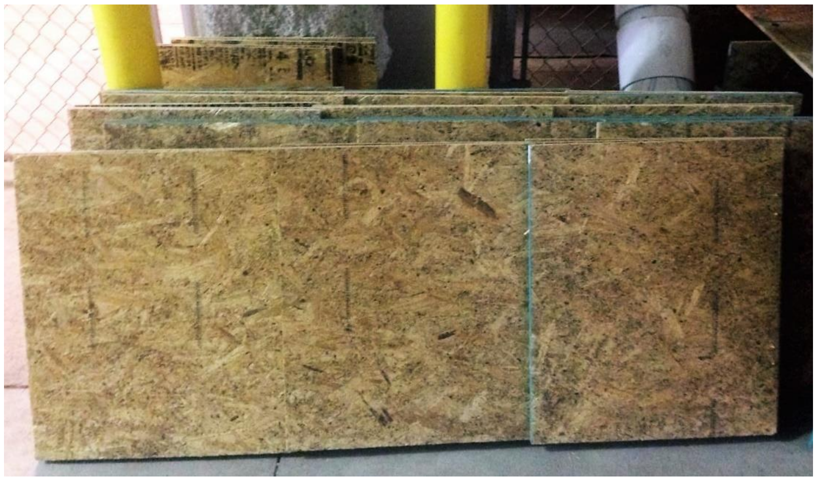


Figure 9: Stored Spots (All)

iii. Final Assembly

The final assembly requires walls to be joined in straight lines and at 90° angles. They must also be joined in a quick, easy manner, and dismantled in the same way to meet the requirement of easy storage. The walls joined in straight lines use two hook & eye latches, spaced 6" from the top and 6" from the bottom of each wall (Figure 10). These help counteract warps in the plywood and imperfections in the cement floor.

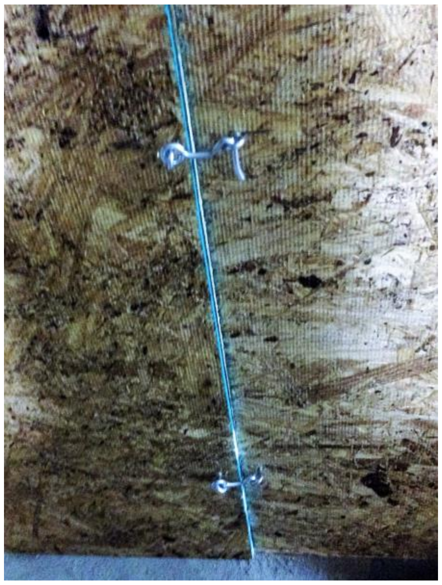


Figure 10: Walls Joined using Hook & Eye Latches

In a similar manner to the angles of the protrusion, walls joined at an angle use two angle brackets to ensure they are 90°. The angle brackets are placed 6” from the top and 6” from the bottom of each corner and are located on the exterior of the lot (Figure 11), regardless of whether it is a convex or concave angle. The bracket locations throughout the lot can be seen in Figure 1. To ensure that the brackets can be attached and removed as the parking lot is assembled and disassembled, one side of the bracket is permanently attached with screws. The other side uses bolts and wing nuts (Figure 12) that are easily added and removed.

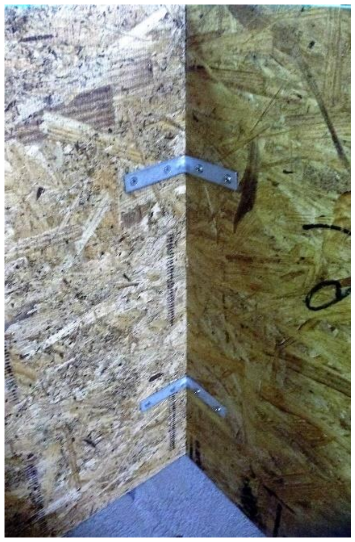


Figure 11: Exterior of Walls Joined using Angle Brackets, Bolts, and Wing Nuts

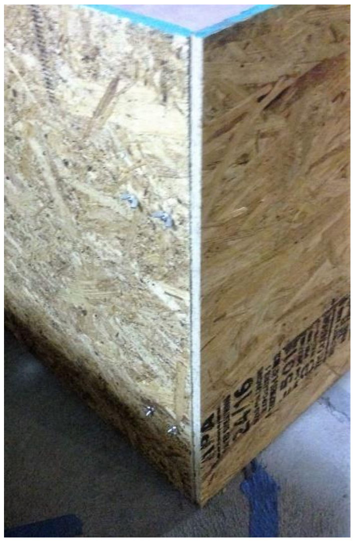


Figure 12: Interior of Walls Joined using Angle Brackets, Bolts, and Wing Nuts

The entire construction took approximately 24 hours. I completely the majority of it by myself, with occasional assistance from Astha Prasad and Eitan Babcock. Richa Varma assisted in the addition of the joining angle brackets detailed in this section. Fully assembling the parking lot by oneself takes approximately 30 minutes (Figure 13). Disassembling takes approximately 20 minutes (Figure 14).

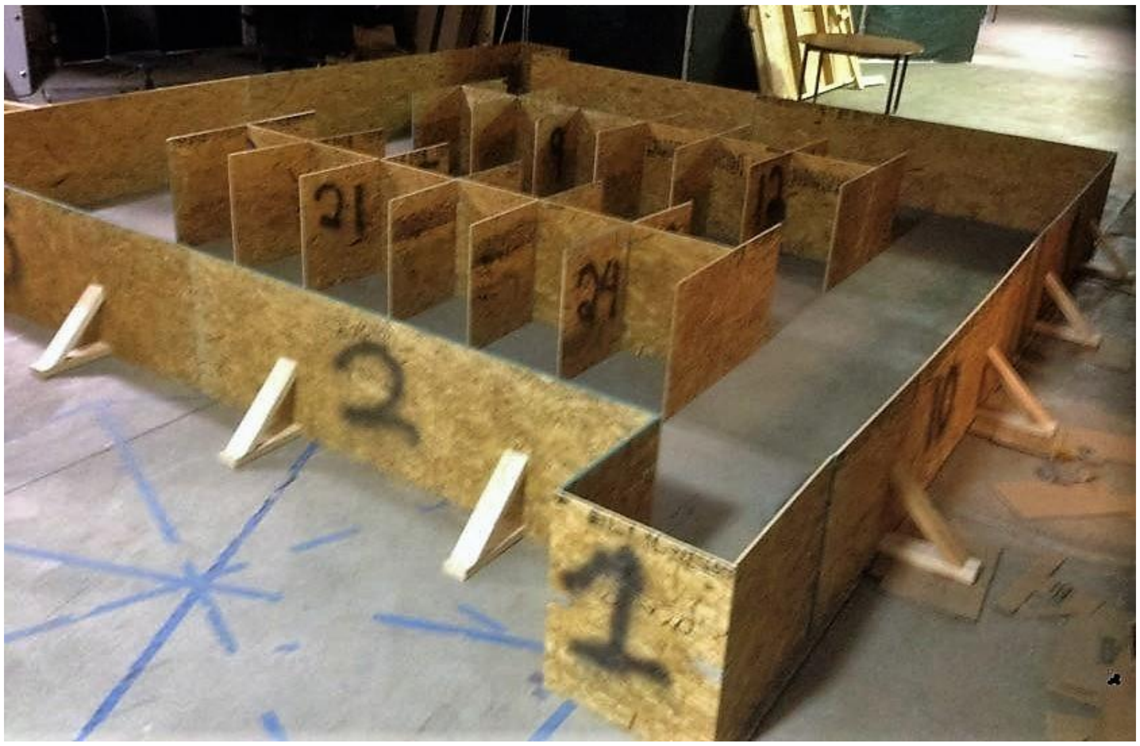


Figure 13: Completed Parking Lot

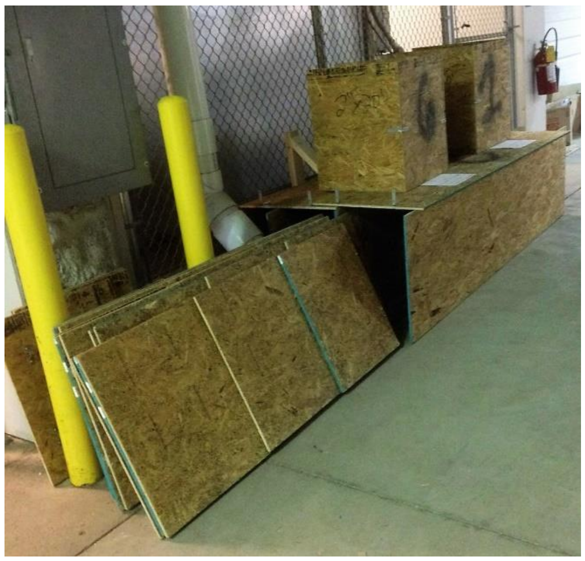


Figure 14: Stored Parking Lot

b. Platform Hardware and Software Progress

We finally received the male-to-female 4-pin Molex cable necessary to connect the platform's SSD to the MALG PCB. This completed the hardware on the platform. Unfortunately, powering on the platform was still not working; the light on the motherboard would come on for 3-4 seconds, then power off. The manual instructed us to jump pins 5 and 7 on the front header panel during start-up in the event of such a power problem. After multiple variations of this, we were still unable to fix this problem. We consulted with a Xaxxon representative, who suggested jumping pins 6 and 8 instead. This worked and we were able to start the platform and install Xubuntu. We downloaded the necessary Oculus Prime packages from their website and were able to teleoperate the platform from the web interface once. We have not been able to replicate this and will be in contact with Xaxxon for support.

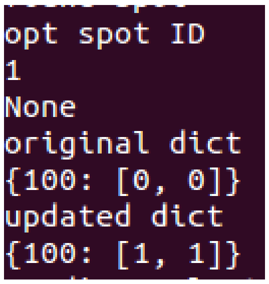
c. XBee Improvements

The XBee and UI have been upgraded as a basis for improvements on spot selection and path planning. Richa and I decided that the vehicle dictionaries, previously including only vehicle IDs and spot IDs, will now also include a flag indicating whether the vehicle is in motion or not. This, combined with the spot ID, indicates the state of the vehicle, as seen in Table 1.

Table 1: The Vehicle States, Determined by Spot ID and Motion Flag

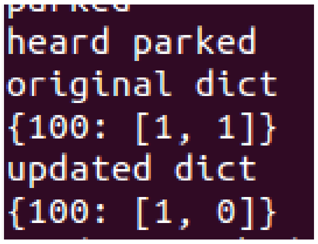
Spot ID	Motion Flag	Vehicle State
0	0	In Queue
1-24	1	Parking
1-24	0	Parked
25	1	Returning
25	0	Returned

We also decided to change the virtual vehicle and physical vehicle IDs. Previously, only four virtual vehicles were allowed, using IDs 1-4. Now, virtual vehicles use IDs 1-99 and physical vehicles use IDs greater than or equal to 100. In addition to allowing more virtual vehicles in the lot, this creates an easy way to check if a vehicle is a virtual or physical vehicle. Combining these two new aspects results in the communication below, which shows the vehicle dictionary with one physical vehicle transitioning from in queue to parking in spot 1 (Figure 15) and from parking to parked (Figure 16).



```
opt spot ID
1
None
original dict
{100: [0, 0]}
updated dict
{100: [1, 1]}
```

Figure 15: XBee Choosing Optimal Spot and Updating Spot and Motion in Dictionary



```
heard parked
original dict
{100: [1, 1]}
updated dict
{100: [1, 0]}
```

Figure 16: XBee Updating Motion in Dictionary

d. UI Improvements

Several large improvements were made to the UI – like the UI, it now saves the state of the vehicle as well as the vehicle location, the states of the virtual vehicles now include parking and returning, and the UI now shuts down smoothly after informing the XBees that the virtual vehicles have left the lot.

The ultimate goal of my work on the UI was to be able to add parking and returning vehicles to the lot. To do this, I first had to do an extensive re-work of the code. Previously, the vehicles in the lot and their respective spots were saved in a list. Not only is this a confusing way to structure the data, it created a bug whereby when a vehicle from a row of vehicles parked out of order (i.e. removing vehicle 4 from the row of vehicles 1, 4, 3, and 2 in spots 1, 3, 4, and 5, respectively (Figure 17)) is removed, the vehicle will be removed correctly, but one of the remaining vehicles jump spots to take the place of the removed vehicle (Figure 18). Creating a dictionary fixed this error, as data can be referenced by value in a dictionary, rather than by index in a list.

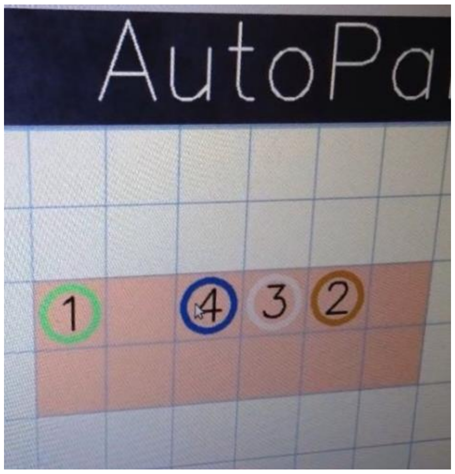


Figure 17: Virtual Vehicles Parked Out of Order

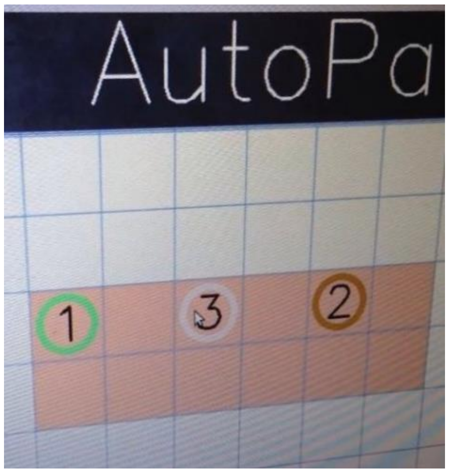


Figure 18: Vehicle 4 Removed and Vehicle 4 Incorrectly Changing Parking Spot

The next improvement I made was to modify the dictionary in the same manner than I changed it in the XBees – add motion in addition to the spot ID. This allows the UI to keep track of the state in addition to the location of all the vehicles. While making these improvements, I noticed that upon closing the UI script, the UI window would freeze and needed to be force quit in order to close. Additionally, the UI never communicated to the XBee that it was closing and would think there were still virtual vehicles in the lot. I created a signal handler in the script to pass messages to the UI from each virtual vehicle indicating that it has left the lot. The signal handler then closed the window.

Finally, I changed the functionality of the UI to be able to have parking and returning virtual vehicles. Initially, double-left-clicking on an empty parking spot would add a virtual vehicle to that spot and double-right-clicking on a spot occupied by a virtual vehicle would remove the vehicle from the lot. I modified this so that double-left-clicking on an empty spot would add a parking vehicle, indicated by a circled vehicle located at the entrance and the un-circled vehicle ID located in the parking spot (Figure 19). Double-left-clicking on a parking vehicle changes the vehicle state from parking to parked, indicated by the circled vehicle ID located in the spot (Figure 20). When a parked virtual vehicle is double-right-clicked, the vehicle state changes from parked to returning, indicated by a circled vehicle ID in the exit queue and an un-circled vehicle ID located in the parking spot (Figure 21). When a returning vehicle is double-right-clicked, the vehicle state changes from returning to returned, indicated by a circled vehicle ID located in the exit queue (Figure 22). After a one second pause, the vehicle is removed from the parking lot entirely (Figure 23).

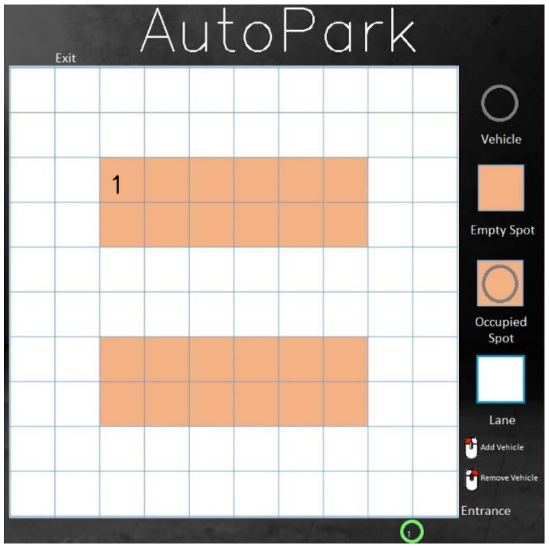


Figure 19: Parking Virtual Vehicle

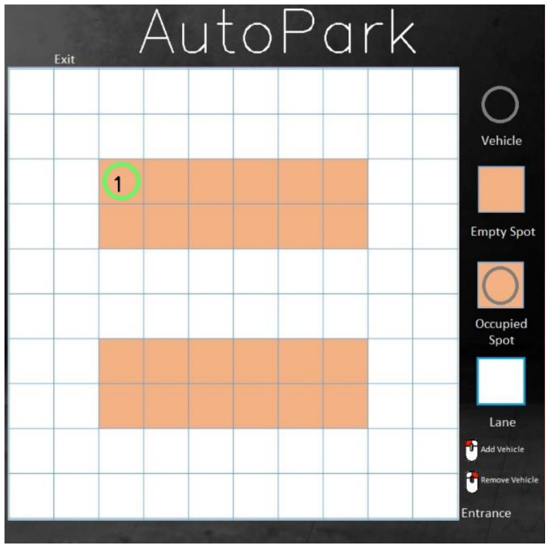


Figure 20: Parked Virtual Vehicle

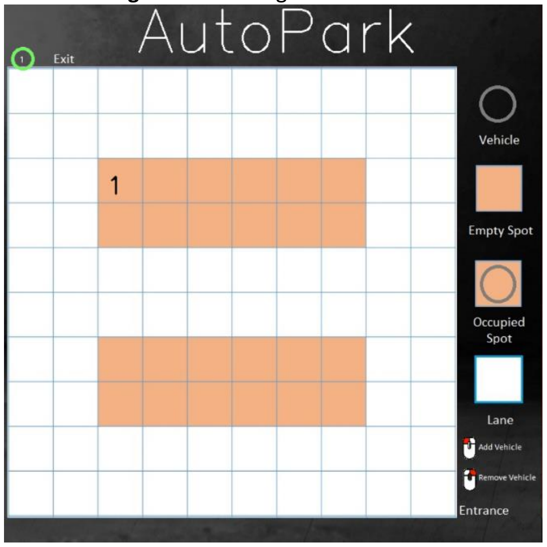


Figure 21: Returning Virtual Vehicle

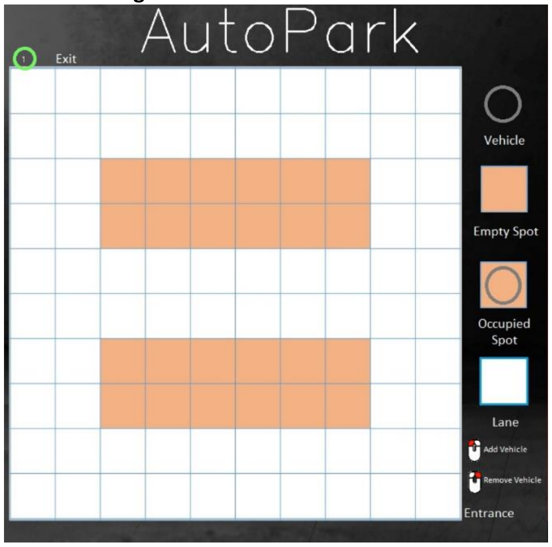


Figure 22: Returned Vehicle

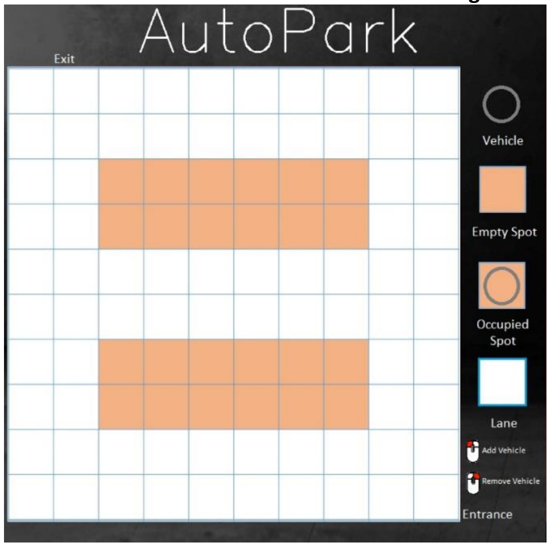


Figure 23: Removed Virtual Vehicle

Currently, these improvements to the XBees and UI have only been tested between one XBee and the UI. There are a few small bugs to iron out, but it is functioning well overall. The next step is to test with multiple XBees and multiple virtual vehicles, then to test with the navigation subsystem. After this is complete, we will implement an algorithm to more intelligently choose parking spots and paths based on the states and locations of the vehicles that are now saved in the XBees and in the UI.

2. Challenges

The largest challenge I faced for this PR is in regards to the platform. Although the hardware is now complete and it has the appropriate software, we are not able to teleoperate the platform. Although this is currently only a minor setback, it will impede on our ability to test with multiple platforms in the future.

Other teammates are experiencing difficult with accurate navigation within the physical parking lot. The new platform does not execute precise enough navigation to avoid hitting the walls of the parking spots. Additionally, although the new parking lot is significantly more feature-rich than the cardboard parking lot, it has difficulty localizing itself and planning appropriate trajectories.

3. Teamwork

I designed and built the parking lot. I worked with Richa to finalize the hardware and install Xubuntu on the original platform. Richa and I created a design for the XBees and UI to better communicate vehicle statuses. For the simulation, Mohak worked on the local planner, Shivam worked on the global planner, and Pranav worked on the visualization. Pranav and Shivam mapped the physical parking lot.

4. Future Plans

For the next progress review, we will show improvement in platform navigation in the parking lot, i.e. not hitting walls, less time localizing. The simulation will be integrated and using realistic data. We will be working towards meeting our performance requirements and will hopefully be able to perform our SVE Demo #1.