# ILR 09 – Progress Review 10

Dorothy Kirlew

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

**March 17, 2016**

# 1. Individual Progress

For the progress review on March 16, I worked with Richa to integrate the App, Master, XBee, and UI. I also incorporated the Planner into the XBee code to mitigate ROS communication risks allow for easy future planner improvements. I also worked with Richa on the platform hardware.

## a. Architecture Improvements

Below, Figure 1 shows the original architecture of the entire system, where the blue squares are the nodes, the green ellipses show the topics and what is the nodes are publishing on them, and the arrows indicate which node is publishing to the topic and which node is listening to the topic.

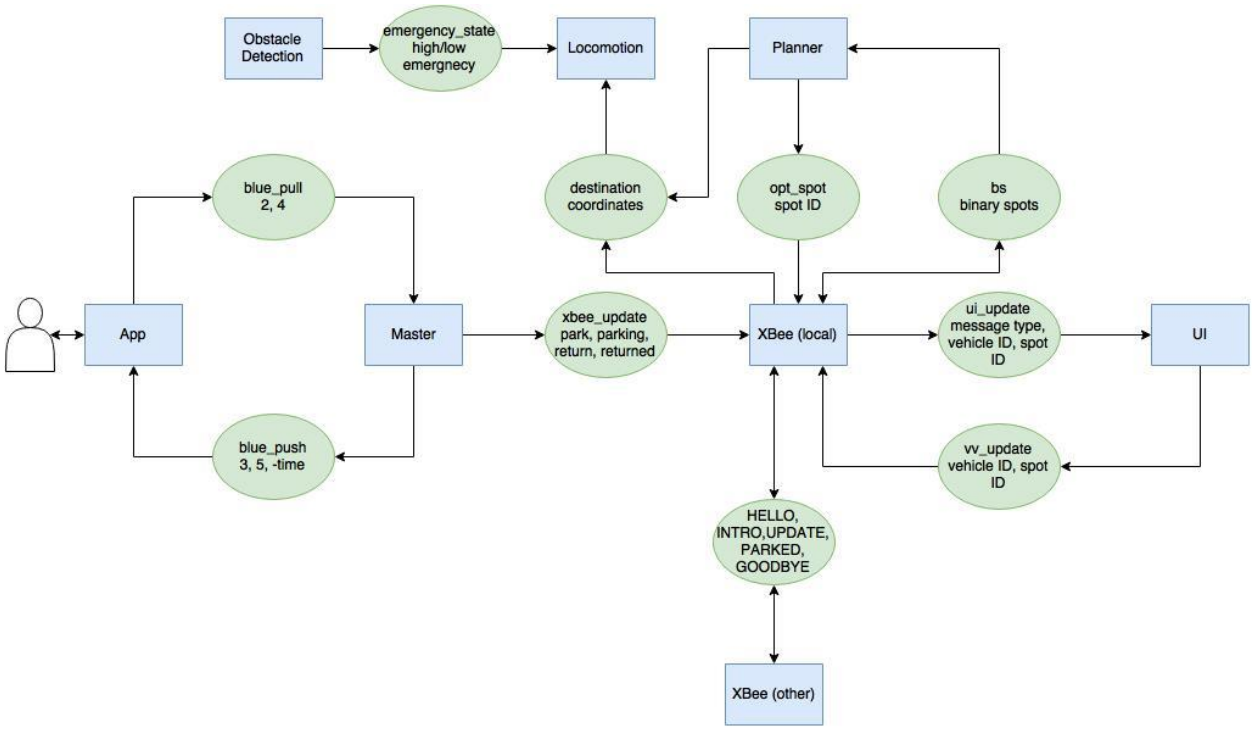


**Figure 1**: Original Software Architecture

Several changes have been made to the architecture, which can be seen in Figure 2. There are three changes that took place. First, as mentioned, I integrated the Planner with the XBees. Thus, an XBee will no longer publish the binary list to the Planner, and the Planner will no longer publish the chosen spot to Navigation and to the XBee. Instead, the XBee will compute the optimal spot and publish this to the Navigation. Not only does this make integration easier, this reduces the risk of problems in ROS communication. The second change involves the Navigation stack. It was using originally using coordinates to determine its goal location. Now, the Navigation stack requires the spot ID. In addition to modifying the XBee code to incorporate the

Planner, I also modified the Planner by removing the coordinates so that it contains only the IDs of the spots and the associated cost. The final change to the architecture is that obstacle detection is no longer needed. More accurately, there are built-in obstacle detection and obstacle avoidance algorithms, so there is no need for this to be done manually.

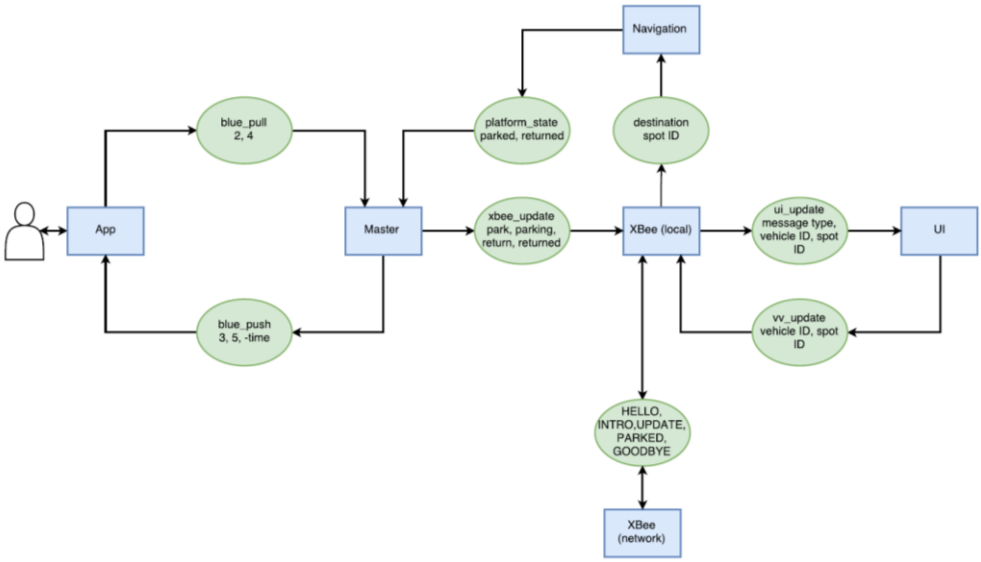


**Figure 2**: Updated Software Architecture

I updated the ROS communication structure within the XBees to reflect the modified ROS communication. As seen in Figure 3, the XBee has a listener that waits for an Update from the Planner. In Figure 4, this has been removed. Instead, when a Park command is heard, the XBee will send an UPDATE message to other XBees, publish the Spot ID for the Navigation node, and finally update the UI, if necessary.

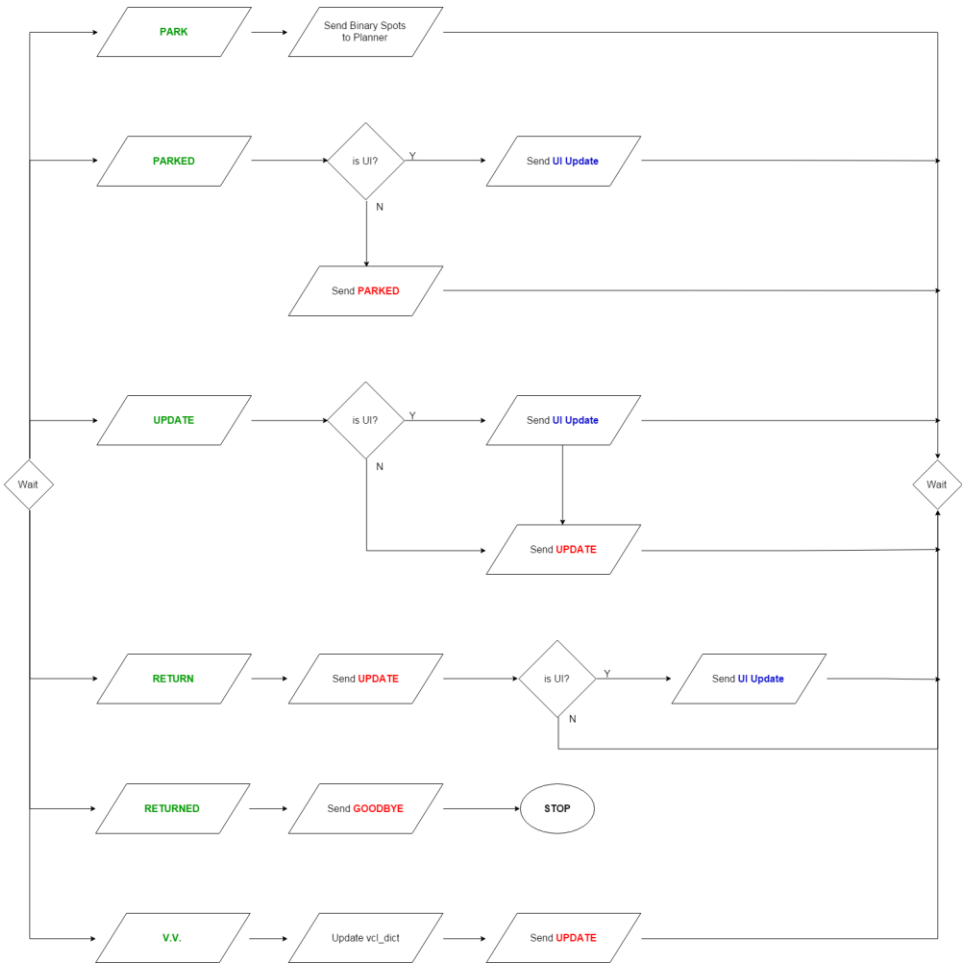


**Figure 3**: Original ROS Communication to XBee

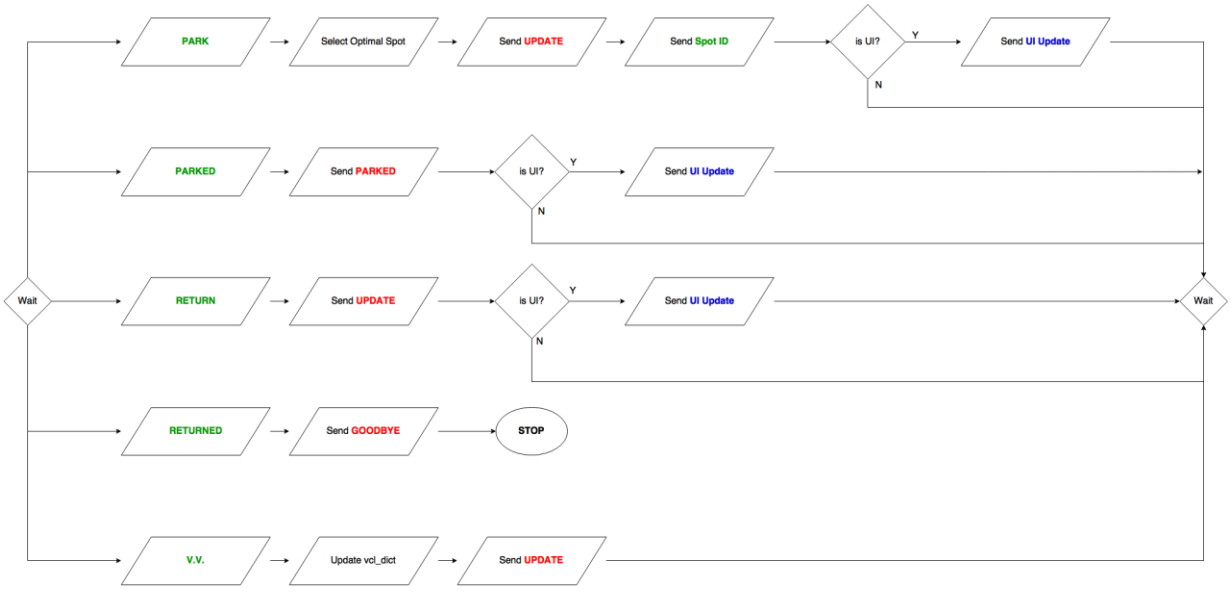


**Figure 4**: Updated ROS Communication to XBee

## b. Integration and Testing

Richa and I worked together to integrate and test the App, Master, XBee, and UI as a complete unit. As seen in Figure 2, the user presses the Park or Return button on the app, which is heard by the Bluetooth node. The Bluetooth node publishes this Park (2) or Return (4) command on the blue_pull topic, which the Master is listening to. When the Master hears either of these commands, it publishes the Park or Return command on the xbee_update topic. The XBee reacts as described in Figure 4; when the Park command is heard, it finds the optimal spot, sends an UPDATE message to the other XBees on the network, publishes the Spot ID to the destination topic, and, if necessary, updates the UI. We did not integrate the Navigation with the system, so we simulated it by publishing the appropriate messages to platform_status. When the Master hears that the platform has reached its destination, it publishes the Parked or Returned status to xbee_update. It also publishes this to blue_push. When the XBee hears the Parked update, it will update the UI if it is the UI. If not, it will update the other XBees that it has parked. When the XBee hears a Returned status, it will send a Goodbye message to the other XBees and then shut down. When the Bluetooth node hears the parked or returned status, it communicates this to the Android app, which updates the status appropriately.

Richa and I ran this process from start to end. We ran the UI on my laptop and had the Android app connected to her laptop. Our laptops both had XBees. We then started the launch script with mine modified to initiate the UI. Richa pressed Park on the app, which started the chain reaction that caused the UI on my laptop to display that her "vehicle" had chosen the most optimal spot and was "parking" in that spot. We then simulated that the vehicle had parked in the spot and the app and UI updated accordingly. Richa then pressed Return on the app. The UI removed her "vehicle" from the parking lot. We simulated that the vehicle had reached the exit and the scripts stopped themselves.

Overall, integration and testing was very successful. We tested a variety of scenarios and discovered a few minor bugs in various sections. Mostly, these were easily solved. We also worked to add descriptive commends to the code, create functions out of repetitive tasks, organize the code to improve readability, and overall improve the accessibility of the code.

## c. Oculus Prime Hardware

Richa has been working on fixing the original Oculus Prime platform. Specifically, this involves installing the motherboard on the platform and setting up the motherboard with XUbuntu and ROS to be able to run on the platform. We assumed that the newly-ordered platform would include all necessary components, but soon discovered that the necessary RAM was missing. We received the RAM on March 15, but were unable to power the platform using the benchtop power supply. Richa and I decided to move on from the software of the platform to the hardware in hopes that powering the motherboard through the correct channels would be successful. We attached the motherboard to the chassis (Figure 5). Unfortunately, we discovered that the platform was missing a vital male-to-female 4-pin Molex cable that would

connect the SSD (Figure 6) to the MALG PCB on the underneath of the chassis. We have ordered the Molex cable, which will hopefully arrive this week. This will allow us to complete both the hardware and the software of the platform.

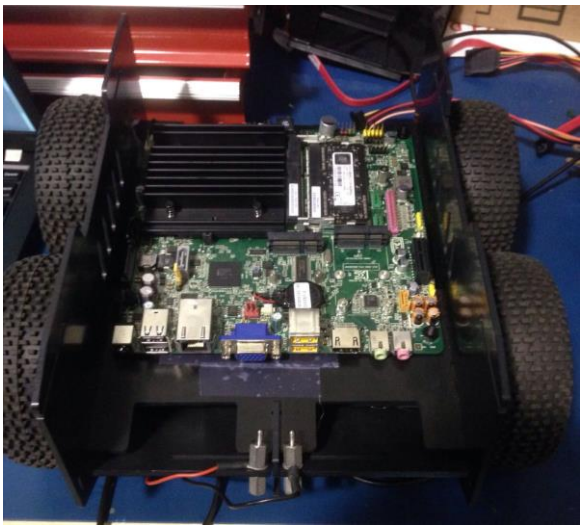

**Figure 5**: Motherboard on Top of Chassis


**Figure 6**: SSD Card on Bottom of Upper Assembly

## 2. Challenges

Due to conflicting schedules, it was difficult to find time to meet with teammates to integrate subsystems.

Both the Bluetooth node and the XBee node have signal handlers that smoothly shut down the script when a “ctrl+z” or “ctrl+c” is heard from the keyboard. Unfortunately, we discovered that when the launch script is used to start the scripts, sending a “ctrl+z” command will shut down all the scripts that were started, but it does not use the signal handlers. This means that the Bluetooth sockets that were opened are not closed and the XBee does not send the necessary GOODBYE message. We worked for several hours researching and testing work-arounds to this problem. Finally, we discovered that using “ctrl+c” *will* send the appropriate commands to smoothly shut down all the scripts.

As mentioned above, we were waiting to receive the RAM to continue work on the platform, which caused a significant delay. Similarly, we are now waiting on a Molex cable to continue work. This is delaying progress.

## 3. Teamwork

Mohak worked on the local planner and the simulation environment. Shivam worked on the global planner for the simulation. Pranav worked on the simulator for the simulation. Mohak, Shivam, and Pranav worked on reinforcing the parking lot and mapping and testing the parking lot. Richa and I worked on integration and testing of the real-world aspects of the project.

Richa worked on the software of the platform and we both worked on the hardware for the platform.

## 4. Future Plans

For the next progress review, there were be three platforms (original platform, new platform, and platform that is being shipped) will be operational and fully integrated. The final parking lot will be created, mapped, and the platforms will be able to navigate within it. The simulation will be able to show basic integration.