ILR 06 – Progress Review 7

Dorothy Kirlew

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

January 27, 2016

1. Individual Progress

For the progress review on January 27, I worked on three aspects of the project – the app, the mock parking lot, and the communication subsystem.

a. App

Integrating subsystems for FVE uncovered two main problems in the Bluetooth communication between the app and the associated script. The first problem is that when the Bluetooth connection between the script and the app was initiated, no port was specified in the script, but the app initiated a connection on port 1. Although this did not cause problems during preliminary testing, integration called for starting and stopping the connection frequently. Sometimes the connection would not close completely, so the script chose the next open port, port 2. However, the app was set to connect to port 1. Obviously, no connection could be made. This was solved by hard-coding the script to connect to port 1, as well as adding signal catching to the script to completely close the connections when the user sends "Ctrl+C" or "Ctrl+Z" to end the script.

When the app is closed, it closes the Bluetooth connection. Here lies the second problem; the script was created to accept a connection and continually poll for messages. Closing the connection crashes the script because there is no longer a connection to be used to poll for messages. This situation also caused the port errors mentioned above. I modified the script so that in the case of the connection being closed, it would wait for the connection to open again, rather than throwing an error and terminating the script.

I also worked with Pranav to do some extensive testing of the app. We determined that the app is "good enough". That is, neither the app nor the script crash and both send, receive, and parse messages correctly. Although they are not as sophisticated as they could be, more pressing aspects of the project require our attention. Pranav and I also did preliminary testing of multiple apps running at the same time. Because we plan to use three platforms, there is a small risk of interference in the Bluetooth communication or of two apps accidentally connecting to the same platform due to user error. To test these scenarios, we initiated the connection between one SBC and one phone. We confirmed that messages were correctly sent and received from both the app and the script. We then connected another phone to the SBC and attempted to send messages to and from the second app. The script did not show any received messages from the second phone, nor did the second phone receive any of the messages sent from the script. This confirmed that two phones connected to the same platform will not cause any errors. More testing should be done in different scenarios, but these preliminary tests lessened the likelihood of future problems.

b. Parking Lot Grid

Richa and I worked together to create a grid representing the mock parking lot. After several iterations of the design, we decided on the parking lot seen in Figure 1: 5m X 5m Parking Lot Grid. The initial conditions for the mock parking lot were to have a 10m X 10m testing area. Given that the mobile platform is only 30cm X 30cm, this would either create too many parking spots to be feasible, or the cells in the parking lot would be so large as to be completely unrealistic. To create a more practical lot, we decided that a 5m X 5m testing area would better suit our testing needs. Each cell in the parking lot, which can be a lane or parking spot, is 50cm X 50cm. This leaves a 10cm gap on any side of the centered mobile platform. Once preliminary testing with the Xtion is complete, I will be able to more accurately

gauge if this is a feasible buffer amount, or if the cells will need to be resized. Regardless of the size, the number of cells will likely remain the same. The parking lot has 24 spaces, which will provide ample space to test many different scenarios.

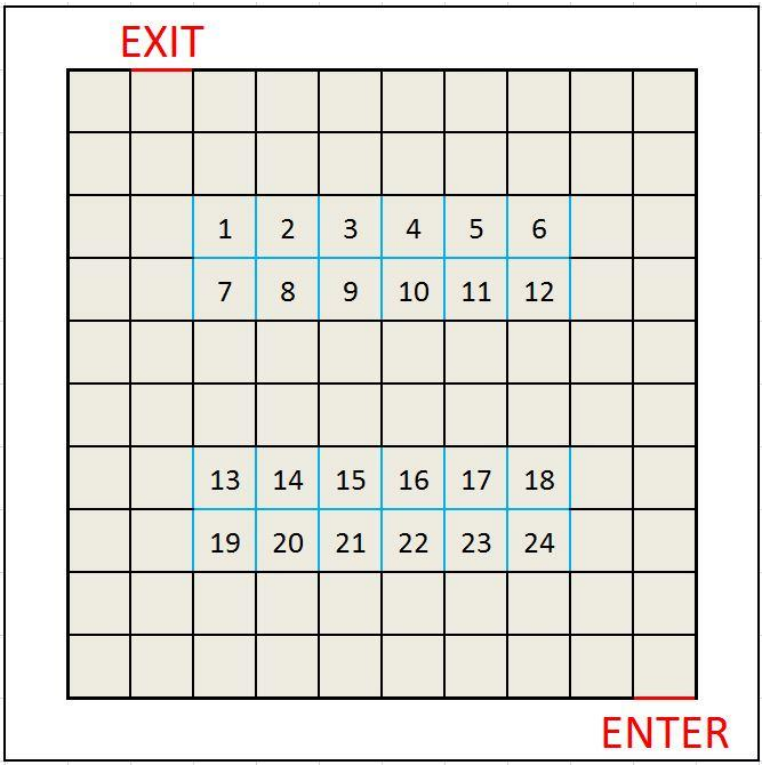

Figure 1: 5m X 5m Parking Lot Grid

c. XBee Communication

Richa and I worked on two aspects of the communication subsystem – the routine definition and coding and testing the sequence to initiate communication.

i. Routine Definition

There are two parts in the communication routine. The first is to initiate communication (Figure 2: Communication Initiation Sequence) and the second is message handling (Figure 3: Message Handling Sequence). Before these are discussed, the data stored in each XBee must be defined. Each XBee has a vehicle ID (vcl_id) and a vehicle dictionary (vcl_dict). The vehicle id is a unique integer that identifies the vehicle. The vehicle dictionary stores the vehicle ID of all the XBees in the area and the ID of the parking spot that vehicle is in, which is 0 if the vehicle has not yet chosen a spot. Each XBee also stores if it is the "last in queue" (liq), which means that it is the last vehicle to have joined the network. The last in queue is determined to be the vehicle with the highest vehicle ID.

When the script first starts, it is a new vehicle and initiates communication with other vehicles in the network by sending a HELLO message. Upon receiving the HELLO, the last in queue vehicle will send an INTRO message. The new vehicle sets its vehicle ID to be one more than the ID of the vehicle it received the INTRO message from. If no INTRO message is received, the vehicle waits 5 seconds and

sends another HELLO. If there is still no response, then it assumes it is the only vehicle in the network, sets its vehicle ID to 1, and adds itself to its vehicle dictionary.

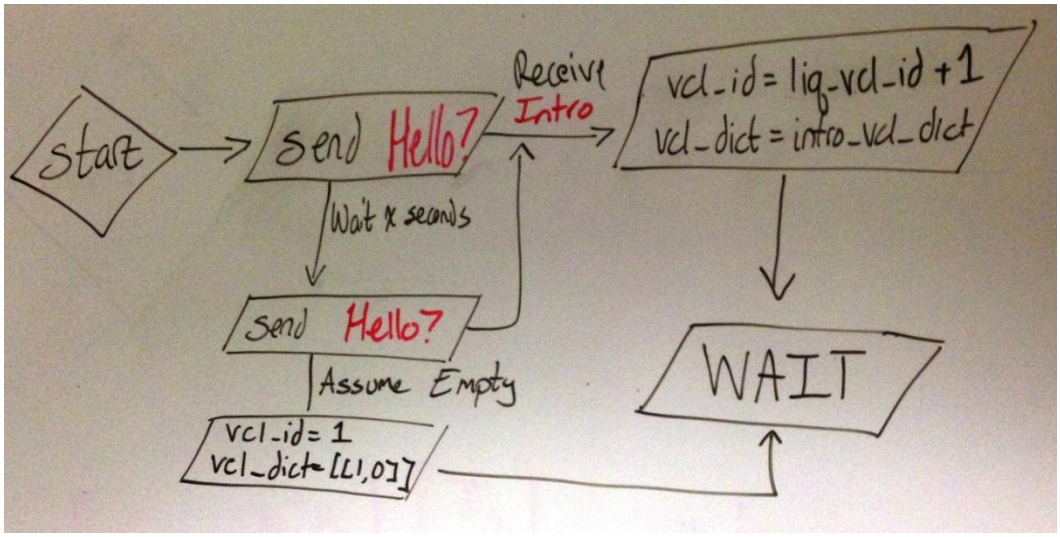

Figure 2: Communication Initiation Sequence

There are seven different types of messages that an XBee can receive:

1. INTRO – This is a message sent over XBee. When an XBee (Car A) receives an INTRO message outside of the communication initiation sequence, it performs two checks to ensure that the message can be ignored. First, it checks that the XBee that sent the INTRO message (Car B) has a different vehicle ID than itself. If Car A and Car B have the same vehicle ID, it is assumed that Car A did not know that Car B arrived in the parking lot due to network error. This means that Car B has more up to date information that Car A does. Car A will then replace its vehicle dictionary with Car B's and update its vehicle ID to be one more than Car B's.
   If the INTRO message passes the first check, Car A then checks if Car B's vehicle ID is greater than the last in queue vehicle ID. This would again mean that Car B has more updated information than Car A. If so, Car A would replace its vehicle dictionary with that of Car B. If the INTRO message has passed both checks, it is ignored.
2. HELLO – This is a message sent over XBee. When an XBee (Car A) receives a HELLO message, it takes note of the new XBee (Car B). If Car A is the last in queue, it will send an INTRO message. Otherwise, it will ignore Car B. However, if this is the second time Car B has sent HELLO and Car A is the second-to-last in queue, Car A will assume that the real last in queue XBee is suffering from network problems and Car A will send the INTRO message in its stead.
3. UPDATE – This is a message sent over XBee to notify other vehicles in the network that it has either chosen a parking spot or it is exiting the parking lot. In either case, the receiving XBees update their vehicle dictionary to reflect this.
4. GOODBYE – This is a message sent over XBee when a vehicle is leaving the parking lot. When an XBee receives this message, it removes the XBee that sent the GOODBYE from its dictionary. It will also check to see if it is now last in queue so that it can respond to HELLO messages appropriately.
5. PARK – This is a message sent over ROS when the user presses Park on the mobile app. When the XBee receives this command, it converts the vehicle dictionary into a binary list of 24 numbers that represent the 24 parking spots, where 0 indicates a free spot and 1 indicates an

occupied spot. It then passes this binary list to the Planner, which uses this to choose the optimal parking spot. The Planner then publishes this spot in a ROS UPDATE message (below).

6. UPDATE – This is a message sent over ROS by the Planner subsystem to indicate the optimal spot for this vehicle. When an XBee receives this, it updates its vehicle dictionary and sends a UPDATE message to the XBee network to let other XBees know which spot it has selected.
7. RETURN – This is a message sent over ROS when the user presses Return on the mobile app. When the XBee receives this command, it sends the exit coordinates to the Navigation subsystem. The Planner is bypassed because the exit is a known location and the Planner does not need to plan a spot. The XBee will also send a UPDATE message to other XBees to let them know that the spot is now vacant. It will follow up with a GOODBYE message to alert the other XBees that it is exiting the parking lot.

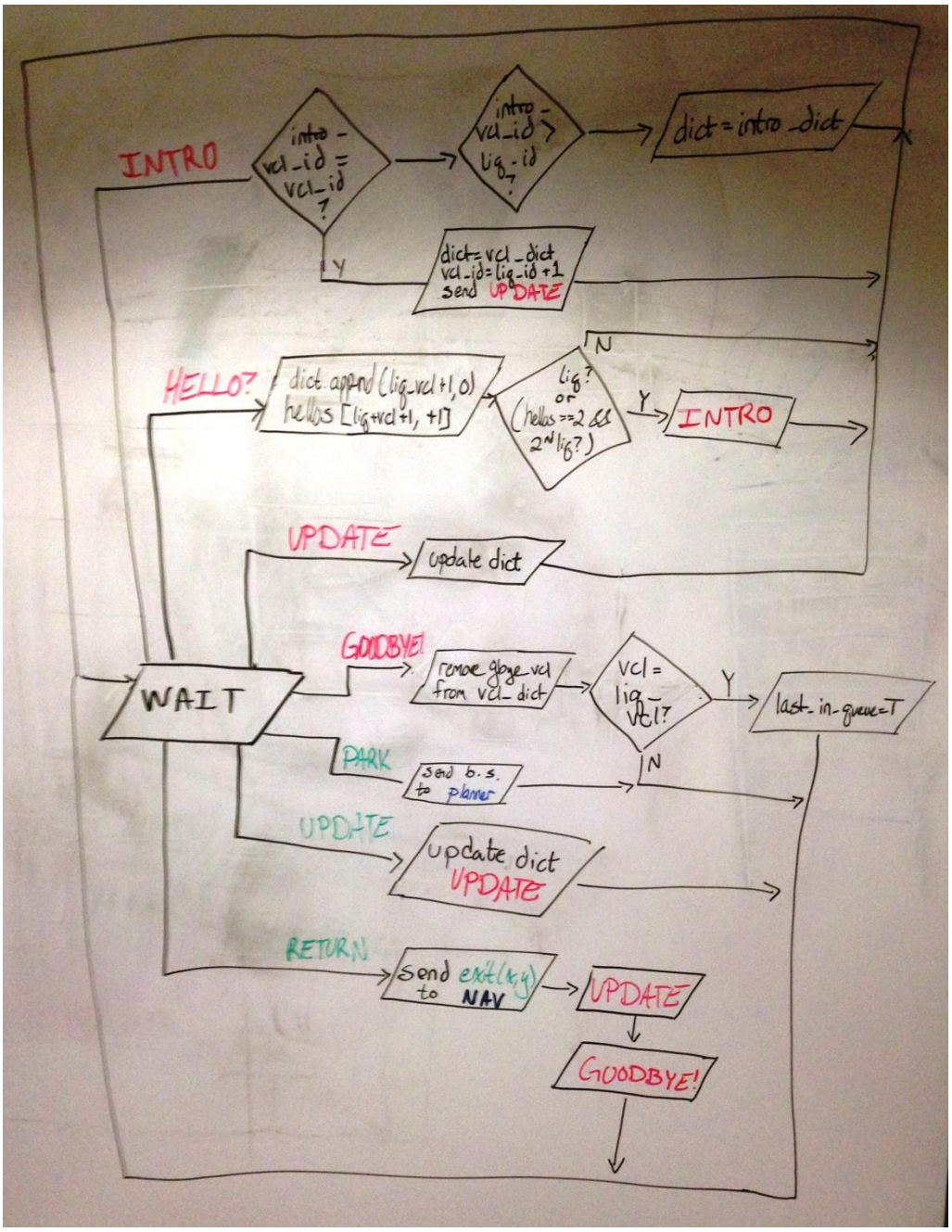

Figure 3: Message Handling Sequence, where green denotes ROS messages and red denotes XBee messages

i. Initial 3-Way XBee Testing

Once Richa and I defined the routine that the XBees would follow, we coded communication initiation sequence. This was used to ensure that three XBees can communicate with each other, without creating complex debugging scenarios by including all other types of messages.

Richa and I created the following scenario to test communication initiation: two vehicles (Car 1 and Car 2) are parked in the lot when Car 3 joins the network. Car 3 sends a HELLO message, which is

received by both Car 1 and Car 2. Because Car 2 is the last vehicle to have entered the lot, it is last in queue. Thus, Car 2, and only Car 2, sends an INTRO in response to Car 3. Car 1 simply ignores the HELLO and INTRO messages. To confirm that each message is being received and responded to accordingly, the output can be seen in Figure 4**: HELLOs and INTROs, photo by Richa Varma**. As this is a testing scenario, Car 3 continues to send HELLO and Car 2 continues to respond with INTRO. When implemented, Car 3 would only send HELLO once and Car 2 would only respond with an INTRO message once. The repeated messages were sent to ensure a) that no messages are lost between the XBees and b) repeatability.

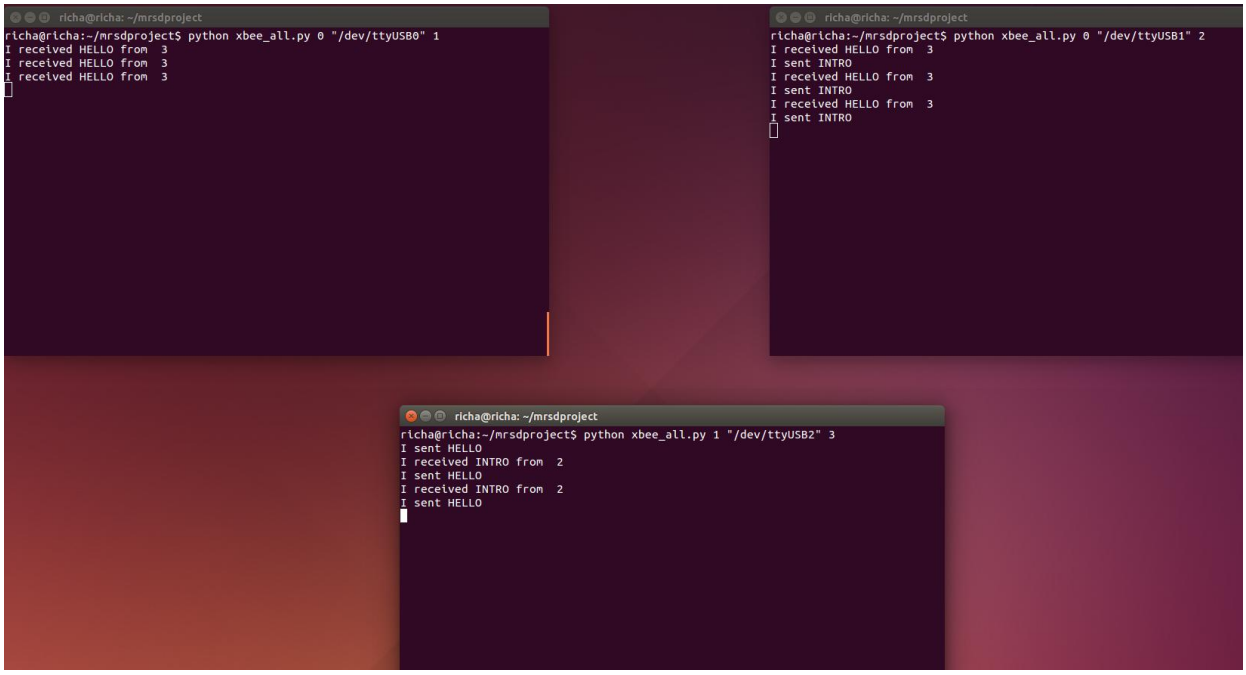

Figure 4: HELLOs and INTROs, photo by Richa Varma

The script for Car 2 is closed, which sends a GOODBYE message to Car 1 and Car 3. Car 1 then becomes last in queue and begins sending the INTRO messages in response to Car 3's HELLO messages (Figure 5**: Car 2 Leaves and Car 1 is Last in Queue, photo by Richa Varma**).

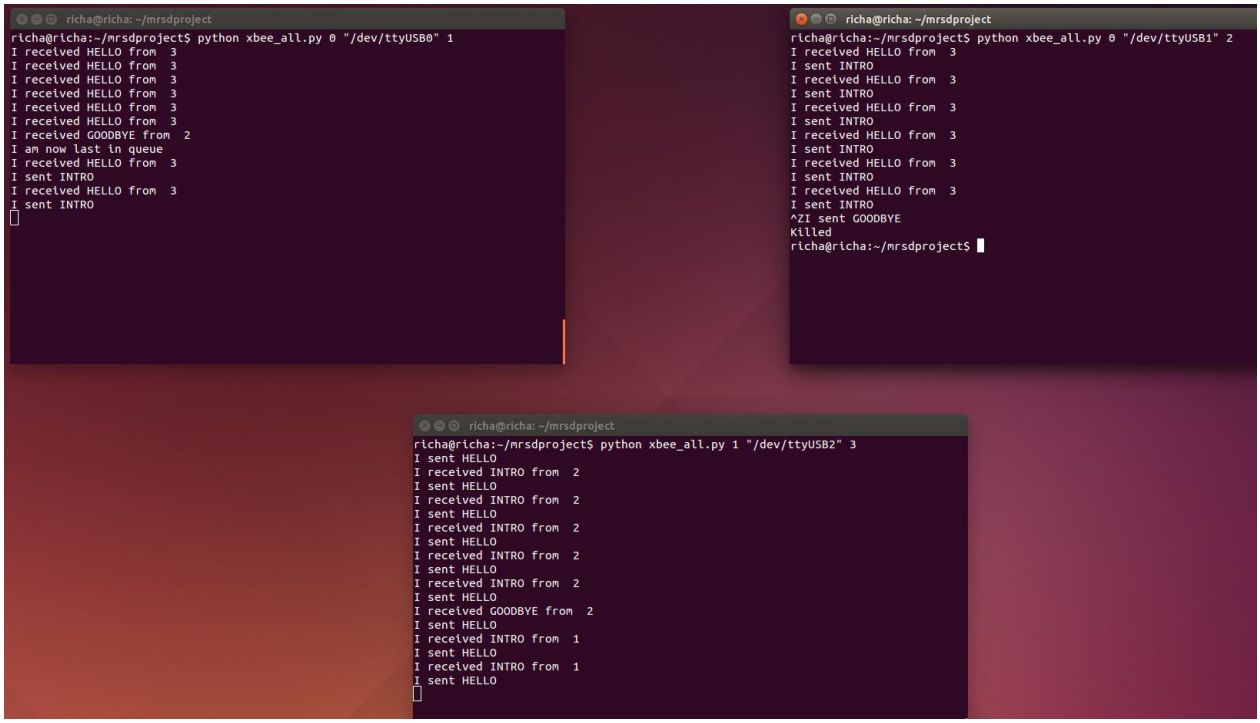

Figure 5: Car 2 Leaves and Car 1 is Last in Queue, photo by Richa Varma

This scenario demonstrates bidirectional communication between multiple XBees. It also shows the initial routine being followed by the XBees.

8. Challenges

The work that I did for this progress review all went fairly smoothly. My past few ILRs have mentioned resource issues, which have unfortunately continued into this semester. Because I am the only team member without an Android phone and the only team member to work on the Android app, the team ordered a burner Android phone over break to be used for testing purposes. This would enable me to test the app without using someone else's phone. Unfortunately, the phone seems to need Verizon to activate it in order to use it. This may prove problematic, but, again, there will be little to no work done on the app throughout this semester.

9. Teamwork

For the Spring semester, our team will be focusing on nine subsystems. Most of the subsystems are expansions of work done in the Fall semester, which enables more time to be spent on integration. For this PR, I worked on the App and worked with Richa on the XBee communication subsystem. Pranav worked on the UI and began work on the Locomotion subsystem, specifically studying waypoint navigation and integrating the emergency stop with the Master node. Due to the Xtion arriving late, Mohak was unable to work on the Mapping subsystem as much as he planned but instead worked on the Multi-Agent Planning subsystem. Shivam was unable to work on the Obstacle Detection and Perception subsystem for the same reason, so he instead redesigned the PCB that will be on each platform.

10. Plans

The team has planned to complete each subsystem by February 7, at which point integration will begin.

1. Android App (Dorothy)
    a. This is already completed
2. P2P Communication (Richa and Dorothy)
    a. Completely code the routine
    b. Test code with 2 and 3 XBees in multiple scenarios
3. Locomotion (Pranav)
    a. Define routine
    b. Test localization
4. Mobile Platforms
    a. CAD and print IR mounts (Dorothy and Pranav)
    b. Completely assemble platforms will all project additions (Shivam)
5. Obstacle Detection and Perception (Shivam)
    a. Extensive testing with IRs and ROS
6. Mapping of Parking Lot (Mohak)
    a. Map test area
    b. Map mock parking lot
7. Creating Parking Lot (Dorothy)
    a. Plan parking lot with respect to Xtion requirements and restrictions
    b. Integrate parking lot plan and Xtion restrictions to fully design parking lot
    c. Fabricate and assemble parking lot
8. Multi-Agent Planning (Mohak)
    a. Code and test
9. User Interface (Pranav)
    a. Completely code and test in multiple scenarios
    b. Integrate with XBees to receive appropriate updates