Progress Review 4

Individual lab report – 05 || November 24, 2015

TEAM DAEDALUS

Richa Varma

Team Members:

Mohak Bhardwaj

Dorothy Kirlew

Pranav Maheshwari

Shivam Gautam

INDIVIDUAL PROGRESS

1. Electrical assembly of Oculus Prime mobile platform
2. V2V serial protocol definition, implementation and testing with two XBee Pro 900 adapters.

Electrical assembly of Oculus Prime

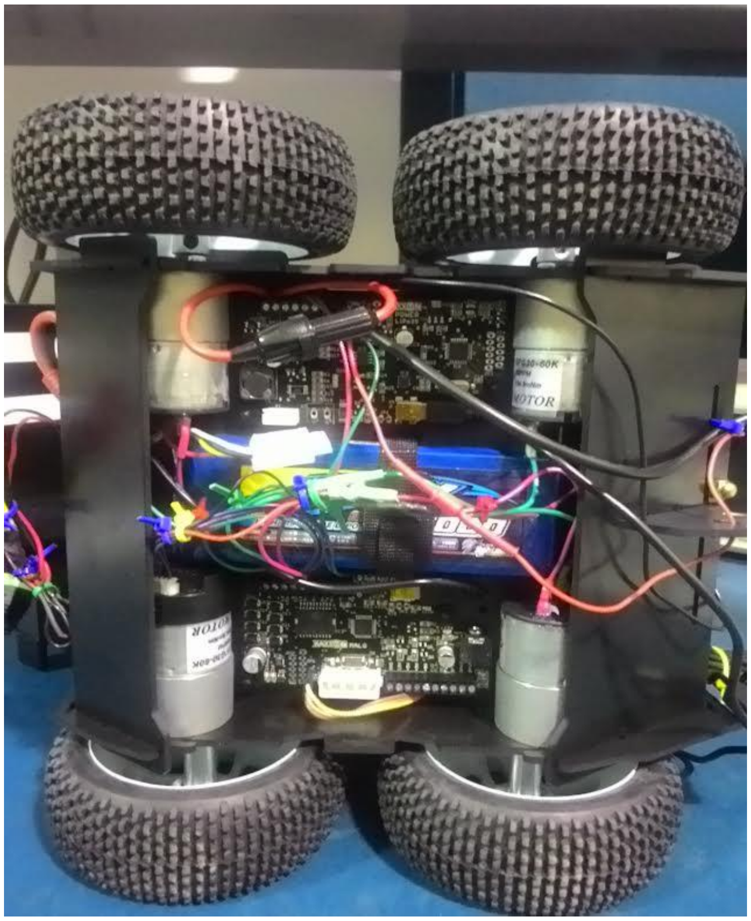


Figure 1. Power PCB, MALG PCB and LiPo battery on the base of Oculus Prime

The platform arrived on the day of Progress Review 3 and we started the assembly soon after. Pranav, Mohak and Dorothy worked on the mechanical assembly. Thereafter, Shivam and I began setting up the Power Distribution Board and the MALG PCB on the platform. This involved following the wiring instructions provided by the manufacturer and ensuring correct polarity of connections and the appropriate types of wiring. We divided the work to save time: I worked on the power PCB and Shivam on the MALG PCB for motor connections. After the connections were made, we plugged in the battery, the LED on power PCB lit up and all seemed fine. But in a short while, we found that the LED was no longer glowing and upon inspecting with a multi-meter, found that the fuse on the power PCB had blown out.
The documentation had suggested that the “battery should not be plugged in until the operating system and Oculus Prime Server Application is set up and running reliably on reboot. The server app contains the necessary battery and power monitoring code to help ensure the battery doesn’t get damaged.”
Thus, we replaced the fuse and decided to wait until the SBC is set up with the operating system and Oculus Prime server application is up before powering on the system again.

Serial protocol definition, implementation and testing

I have been working on the communication subsystem for the past three weeks. After the setup and testing phase was over, it was time to define a protocol for serial messages in the vehicle network.
The module UART performs tasks, such as timing and parity checking, that are needed for data communications. The two UARTs are configured with compatible settings of baud rate, parity, start/stop bits, etc. But the application layer protocol had to be defined for our system to determine syntax rules at the application level and ensure agreement at both ends about error recovery procedures and data integrity.

I found that the Python construct library was a good resource for serialization and reliable building and parsing. In particular, some useful features are:

- Enumerations
- Support for byte-oriented and bit-oriented fields
- Arrays of data with specified length
- Embedded structs

I worked on defining a custom format with construct library features. This format can be used for arbitrarily complex data formats. I realized this was very important at this stage to ensure scalability at later stages.
The sending process is:

1. Serialize all the fields into a packed string using the message format object
2. Compute the CRC and insert it into the frame
3. Wrap the frame with the protocol

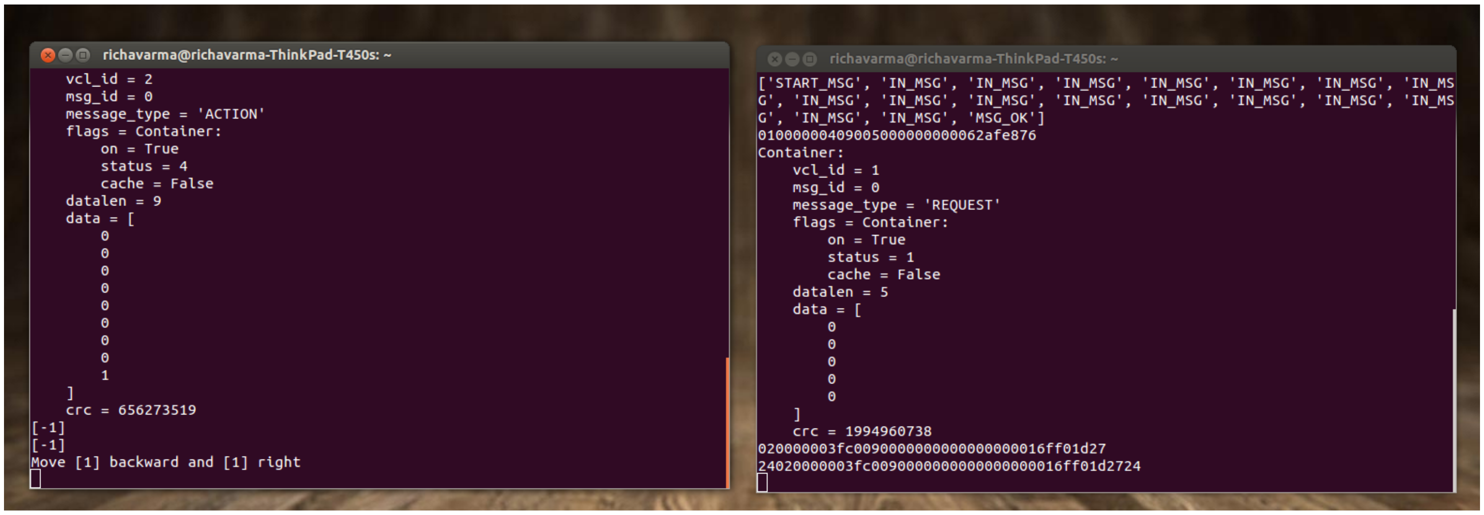
The receiving process is:

1. Unwrap the protocol to receive a frame
2. Unpack the frame into separate fields using the frame format
3. Compute the CRC and compare it to the one received

For our current purpose, the following fields constitute the frame, which can be added upon as per future requirements:

a. Vehicle ID
b. Message ID
c. Message Type
    (i) REQUEST
    (ii) ACTION
d. Flags
    (iii) On flag : Indicates if the vehicle is active
    (iv) Status flag : Indicates one of the four states – Searching, Parking, Parked, Returning
e. Data Length
f. Data
    A binary occupancy map indicating allowed destinations
g. CRC

For testing, I started off with writing two python scripts for two devices communicating with each other. One of the devices sends a REQUEST message to the other. The second vehicle receives this message, interprets it and responds with an ACTION message, providing the first vehicle with an occupancy map in its data field. Upon receiving this information, the first vehicle decides the direction to move depending upon its current coordinates, which will be known and pre-determined.


richavarma@richavarma-ThinkPad-T450s: ~
richavarma@richavarma-ThinkPad-T450s: ~

```
    vcl_id = 2
    msg_id = 0
    message_type = 'ACTION'
    flags = Container:
        on = True
        status = 4
        cache = False
    datalen = 9
    data = [
        0
        0
        0
        0
        0
        0
        0
        0
        1
    ]
    crc = 656273519
[-1]
[-1]
Move [1] backward and [1] right
```

```
['START_MSG', 'IN_MSG', 'IN_MSG', 'IN_MSG', 'IN_MSG', 'IN_MSG', 'IN_MSG', 'IN_MS
G', 'IN_MSG', 'IN_MSG', 'IN_MSG', 'IN_MSG', 'IN_MSG', 'IN_MSG', 'IN_MSG', 'IN_MS
G', 'IN_MSG', 'IN_MSG', 'MSG_OK']
0100000040900500000000000062afe876
Container:
    vcl_id = 1
    msg_id = 0
    message_type = 'REQUEST'
    flags = Container:
        on = True
        status = 1
        cache = False
    datalen = 5
    data = [
        0
        0
        0
        0
        0
    ]
    crc = 1994960738
020000003fc009000000000000000000016ff01d27
24020000003fc009000000000000000000016ff01d2724
```

Figure 2. REQUEST and ACTION messages exchanged between devices and interpreting them to determine direction

I was successful in implementing this functionality and am currently working on creating ROS nodes for the same. The node on the mobile platform will listen for the user commands and accordingly send out a REQUEST or ACTION messages. It will publish to a topic such that the decision node is informed about the intended direction of movement of the platform, which will be relayed to the locomotion node.

Challenges

The main challenges have been:

1. Deciding the format of the occupancy map: For simplicity and to meet the FVE goals, a binary map is sufficient to convey occupancy of cells, and decide the direction of movement. For the actual application, the map would be updated real time with sensor readings and it is very difficult to implement a generalized format at this point, which will be used to convey the state of the environment, without complete understanding of the localization methods that will be used in the future.
2. CRC32: I have used the crc32 checksum in my implementation and have been having issues with when trying to send more than a few bytes of data.

   The error that shows up is:

   “Integer value too big for L format code.”

   I have been trying to resolve this issue but have not found a solution for it yet. I might try using MD5 checksum and see if it resolves the problem.

3. My plan was to test mesh network functionality using three Xbees. We had ordered the third Xbee two weeks back but it is out of stock and will be shipped only after December 8. Without the equipment to test, I am not confident that the current implementation will be suitable with more than two devices communicating simultaneously. There might be a need for fragmentation and reassembly of frames, but that can only be dealt with after the equipment arrives.
4. The entire team faced the challenge of the Odroid XU4 being incompatible with the mobile platform's requirement of a 64-bit SBC. This was a critical risk as all the subsystems depended it. Fortunately, we were able to borrow a MinnowBoard from another team and this risk has been eliminated. We have also ordered another to keep as a backup.

Teamwork

The team has been working to a plan for the past week, with each person taking full responsibility of their designated subsystem and collaborating with others whenever necessary.
Pranav has been actively involved with me in the communication subsystem, providing help with Python and ROS. We have been discussing possible configurations of the occupancy map to narrow down on one suitable and simple enough for our current requirements. I have had individual discussions with each member about their ideas on this, and am trying to incorporate them in my work. Mohak was working on setting up the platform with the Minnowboard, and I worked with him to sort out some electrical issues before he could test locomotion.

Future plans

The Fall Validation Experiment is just round the corner and we are gearing up to integrate all our subsystems. For my part, the future plan is to get the ROS node running and test the setup on the mobile platform and a second SBC. I plan to use one of the ODROIDs as the SBC for the second platform, which is stationary and only needs the communication system running on it. I plan to rigorously perform tests and fine tune my implementation to ensure reliability.
Apart from this, once all the PCB components arrive, I will work with Shivam on populating our PCB. This will allow us to mount IR proximity sensors on the platform and integrate them with the system using Arduino Nano. Mohak and Pranav will work towards testing 2D locomotion. Dorothy will continue working on the app's integration with the ROS architecture.
The entire team will work on integration tasks and ensure that their subsystems are performing well as a part of the whole system.