Progress Review 3

Individual lab report – 04 || November 12, 2015

Team Daedalus

Pranav Maheshwari

Team Members:

Mohak Bhardwaj

Dorothy Kirlew

Richa Varma

Shivam Gautam

1. Individual Progress

- ❖ Setting up the DigiMesh XBee adapters to establish communication
- ❖ Assembling Oculus Prime
- ❖ Integrating the Single Board Computer with Oculus Prime

Communication Subsystem

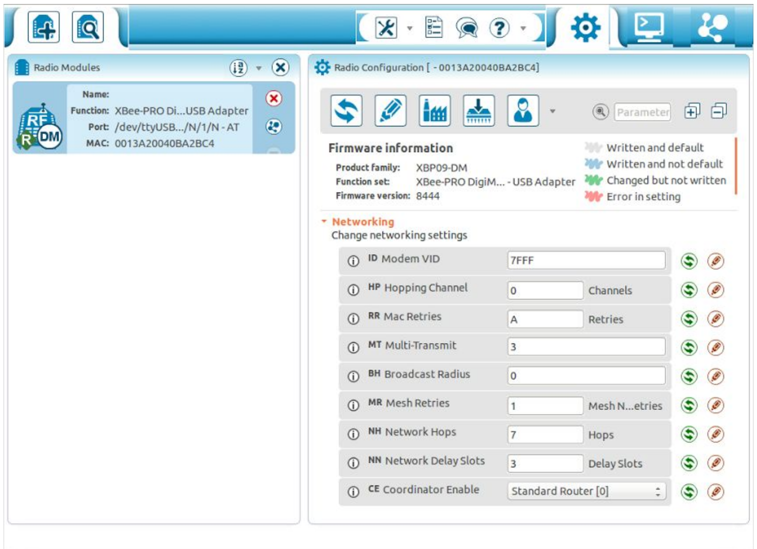
Setting up the XBee adapters:

The XBee adapters can be configured through X-CTU (Figure 1). It detects the XBee adapter connected to the system and lists down the relevant networking settings which can be modified. In our case, some important parameters are:

> Multi-Transmit (MT) to set/read number of additional broadcast retransmissions. All broadcast packets are transmitted MT+1 times to ensure it is received.

> Broadcast Radius (BR) to set/read the transmission radius for broadcast data transmissions. Set to 0 for maximum radius. If BR is set greater than Network Hops then the value of Network Hops is used.

> Mesh Retries (MR) to set/read maximum number of network packet delivery attempts. If MR is non-zero, packets sent will request a network acknowledgement, and can be resent up to MR times if no acknowledgements are received.



Radio Modules

Name:
Function: XBee-PRO Di...USB Adapter
Port: /dev/ttyUSB.../N/1/N - AT
MAC: 0013A20040BA2BC4

Radio Configuration [- 0013A20040BA2BC4]

Firmware information

Product family: XBP09-DM
Function set: XBee-PRO DigiM... - USB Adapter
Firmware version: 8444

Written and default
Written and not default
Changed but not written
Error in setting

Networking
Change networking settings

ID Modem VID	7FFF	
HP Hopping Channel	0	Channels
RR Mac Retries	A	Retries
MT Multi-Transmit	3	
BH Broadcast Radius	0	
MR Mesh Retries	1	Mesh N...etries
NH Network Hops	7	Hops
NN Network Delay Slots	3	Delay Slots
CE Coordinator Enable	Standard Router [0]	

Figure 1: X-CTU interface to configure the XBee adapter

Once the two modules were configured, they were ready to communicate serial data with each other.

Create the protocol:

The work on formulating all the required protocol and packet definitions is still in progress. The few things that have been decided are that:

1) Cars will not continuously publish data; to avoid congestion in the network.
2) Location of cars will be an important factor in deciding their role in the communication.
3) The data being sent shall be timestamped and always contain a unique identifier to ascertain the source of the data.

Create the ROS node:

The serial communication between multiple ros-masters is taking place through the collaboration node.

> **Structure of the node**

The node automatically searches for the COM port to which the XBee is connected. Once it finds the address, it opens the serial port and flushes the buffer values stored on the XBee module. After this initialization routine, the node starts two threads which run in conjunction to take care of sending and receiving the data. These two threads are capable of re-establishing connection to the XBee module in case access to serial port is lost.

> **Functionality of the node**

All the data that is read through serial gets published to a ROS topic which can be accessed by any other active node. The node has a subscriber running in a separate thread, which any other active other node can use to send data to a different ros-master on the network. Both the threads stay alive till the time ros-master is active. The node is also responsible for encoding and decoding all the data being sent and received respectively. For example, adding the value of unique identifier as prefix and attaching a custom EOL character to specify end of message.

> **Testing**

The collaborate node was rigorously tested to check the appropriate rate at which data should be read and written so that no packets are lost. Two laptops were made to exchange a timestamped list of 100 random integers using the Serial COM port. The results from testing can be seen in Figure 2. This list of integers is bigger in size than any data structure that we might potentially use during the project. Therefore, this acts as a good proof of concept to show that the hardware and the network configuration are appropriate and capable.

The result from this testing helped in fine tuning various parameters and also refining our serial protocol and packet definition.

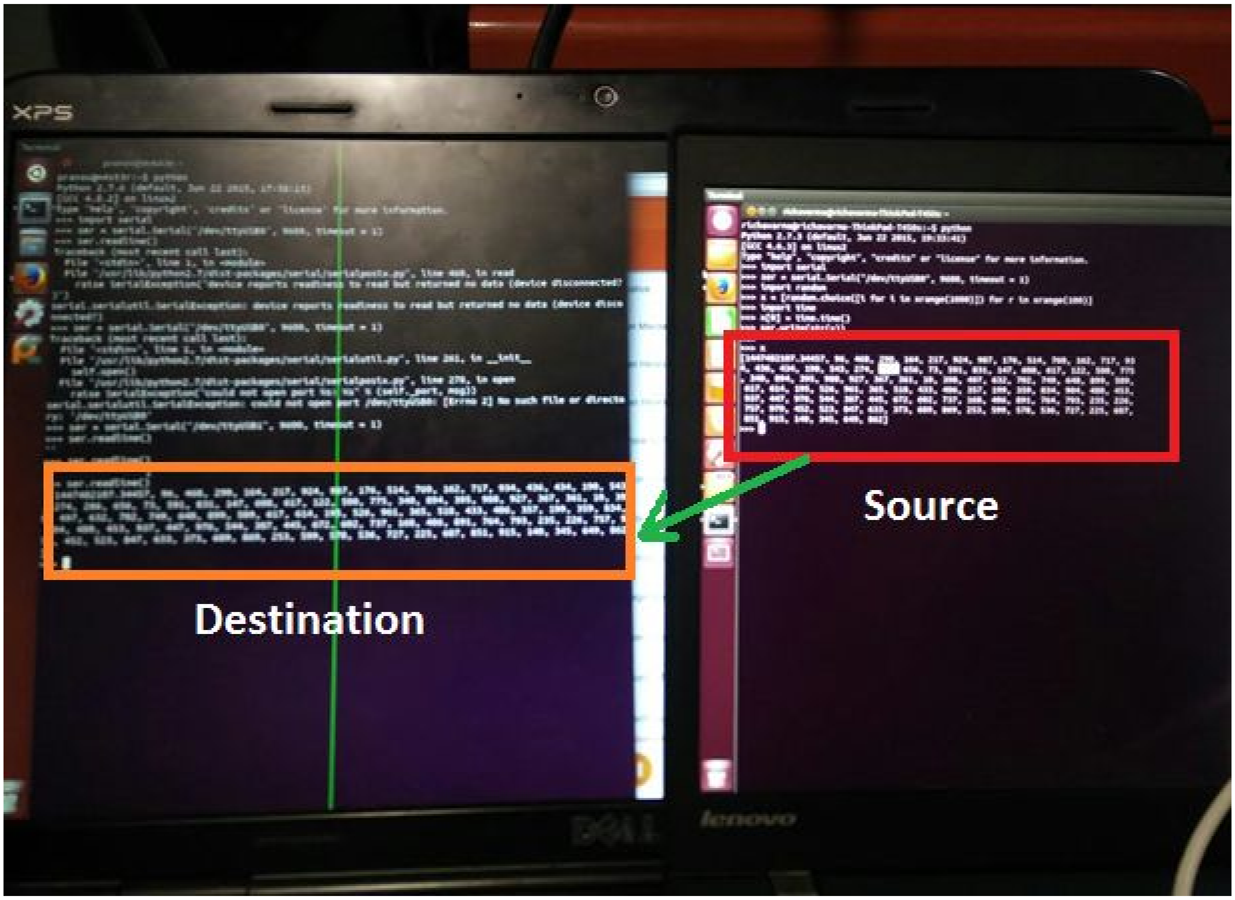


Figure 2: Data being transmitted between two laptops via Serial XBee

Mobile Platform Subsystem

Mechanical Assembly

The package, after experiencing some delays, finally arrived on 11/12/2015. The work on assembling it began that very day. The online guide provided by the manufacturer was followed and the platform was assembled up to a point where the electrical assembly was taken over by Richa and Shivam.

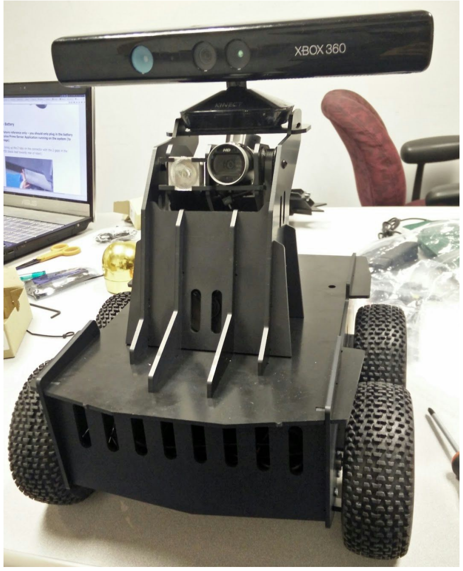


Figure 3: Partially assembled Oculus Prime

2. Challenges

With Fall Validation Experiment approaching, the project is currently in a transient phase, where all the work on various subsystems is drawing close to completion and work needs to be done on integrating it all. Since I am the risk owner and responsible to integrate all of this over ROS, I have to get involved with everyone's work to ensure that certain standardised procedures are being followed which facilitates the final integration. The major challenge in this is that everyone is running code on their personal machines with different packages and environment settings. This leads to a lot of issues whenever cross subsystem compatibility testing needs to be done. Apart from this, the Single Board Computer that were ordered came with Ubuntu 15.05 which doesn't support ROS Indigo. Our platform currently only support ROS Indigo and due to this we had to spend some time trying to flash all the SBCs with the correct OS Image and setting everything else up.

3. Teamwork

As per the distribution of tasks, Mohak and Dorothy worked on setting up the SBCs. Richa and I worked on setting up the XBee adapters and creating the protocol for communication. Dorothy is working on refining the functionality of the app and interfacing IR proximity sensors with Arduino Mega and set it up as a ROS node. Shivam and Mohak worked together to implement the obstacle detection algorithm using PCL. They managed to visualize the obstacle using the data provided online and are presently working to. adapt the code for the obstacles complying with the standards set by our performance requirements. Everyone in the team contributed towards mechanical and electrical assembly of Oculus Prime.

4. Plans

A lot of progress has been made on individual subsystems in the past few weeks. Once the Single Board Computers are completely set-up, we can start working on testing and integrating these subsystems.

1) Mohak and I shall set-up the platform to work with and be controlled by the Single Board Computer via ROS.
2) Dorothy is the task owner for the Mobile App and is working on refining and testing the app to ensure that it works robustly. I shall be working with her to test the bi-directional communication between the Single Board Computer and the Android Phone.
3) Richa is the task owner for the Communication Subsystem and I shall be actively working with her to ensure that all the capabilities of our communication architecture can be executed and work well with ROS. Richa and Shivam our working together to define the data packets and other transmission protocols.
4) Mohak is the task owner for Vision system and will collaborate with Shivam to implement and test the Obstacle Detection algorithm on Kinect.
5) Shivam and Dorothy are working on the IR proximity sensors + Arduino Nano for close range obstacle detection. Once their setup is ready, I'll create a CAD of the mounting base needed to attach all of this to Oculus Prime and get it 3D printed.