Progress Review 2

Individual lab report – 03 || October 30, 2015

Team Daedalus

Pranav Maheshwari

Team Members:
Mohak Bhardwaj
Dorothy Kirlew
Richa Varma
Shivam Gautam

1. Individual Progress

The tasks assigned to me were:

- Interfacing actuator control board with ROS
- Interfacing bluetooth serial communication with ROS

For interfacing actuator control board with ROS, I worked with Shivam. We had two options 1) simple serial communication or 2) setting up the board as a ROS node. We went with option 2 as, even though it required extra effort initially, it made the task of integration with ROS much simpler. Our system, as we envision it now, will be totally ROS based therefore it is important to have everything on the same framework. This helps in mitigating possible conflicts and incompatibility issues while integrating the various subsystems. To achieve this we used the rosserial package. Using this package, the Arduino can run in sync with rest of the system as it is connected to the clock of roscore. Also, unlike regular serial messages, ROS ensures that the ROS messages being sent and received are the same by transmitting and verifying the MD5 checksums. This makes the entire process of serial communication much more reliable.

At present, we have setup a test script that constantly publishes data from Arduino to the laptop. This data is accepted by another node and is printed on the screen. Further we are working on setting up an Arduino node that fetches data from Sharp IR sensors and lets the ros node know when emergency stop needs to be initiated. This will help us in achieving our functional requirement of having onboard emergency stop.

```
/*
 * rosserial Publisher Example
 * Prints "hello world!"
 */

#include <ros.h>
#include <std_msgs/String.h>

ros::NodeHandle nh;

std_msgs::String str_msg;
ros::Publisher chatter("chatter", &str_msg);

char hello[13] = "hello world!";

void setup()
{
  nh.initNode();
  nh.advertise(chatter);
}

void loop()
{
  str_msg.data = hello;
  chatter.publish( &str_msg );
  nh.spinOnce();
  delay(1000);
}
```

Arduino Code for ROS Interfacing

For interfacing Bluetooth serial communication with ROS, Mohak and I did some online research to understand the basic protocol followed by Bluetooth and make ourselves aware of basic networking terminology such as sockets, ports, and various protocols such as RFCOMM and L2CAP. Previously, Mohak had created a bash script which opened a COM Port for Bluetooth communication and attached a fetch service to it. Although a good proof of concept, this approach wasn't efficient as we need to publish the incoming data on a ROS based architecture. Therefore we decided to create a ROS node using Python which publishes the data being received through Bluetooth.

We used PyBluez, a library which provides python wrappers around system Bluetooth resources to facilitate the process of creating Bluetooth Applications. We further plan to make this node both receive and send data. Also, the node should be able to detect when a device gets disconnected and try to re-establish the connection. Once I decide the protocol for communication with Dorothy and Richa, the parsing of the strings will also be done in this node so that data gets published to relevant topic. For example, if the command to "Return" is received, the node will parse the string and send a message to the path planning node to calculate the path to the nearest exit.

```
#!/usr/bin/python


from bluetooth import *
import rospy
from std_msgs.msg import String

rospy.init_node("bluetooth_comm")
pub1 = rospy.Publisher("Bluetooth_recv", String, queue_size=10)

server_sock=BluetoothSocket( RFCOMM )
server_sock.bind(("",PORT_ANY))
server_sock.listen(1)

port = server_sock.getsockname()[1]

uuid = "94f39d29-7d6d-437d-973b-fba39e49d4ee"

advertise_service( server_sock, "SampleServer",
                   service_id = uuid,
                   service_classes = [ uuid, SERIAL_PORT_CLASS ],
                   profiles = [ SERIAL_PORT_PROFILE ],
#                  protocols = [ OBEX_UUID ]
                   )

print("Waiting for connection on RFCOMM channel %d" % port)

client_sock, client_info = server_sock.accept()
print("Accepted connection from ", client_info)

try:
    while True:
        data = client_sock.recv(1024)
        if len(data) == 0: break
        pub1.publish(data)
except IOError:
    pass

print("disconnected")

client_sock.close()
server_sock.close()
print("all done")

```

Code for Bluetooth Serial Interfacing with ROS

2. Challenges

While working on ROS Arduino interfacing, Shivam and I faced an unexpected error which was very difficult to trace. The linux kernel on Shivam's workstation, ver. 65, had difficulties communicating with external devices. After some online research, we found out that this was a general issue being faced by the Linux community and we rolled back from ver. 65 to ver. 63 which resolved the issue. ROS Bluetooth interfacing was challenging as it required us to figure out the proper configuration for our device before we could establish communication. This involved researching online and trying out multiple combinations.

3. Teamwork

As per the distribution of tasks, Mohak and I worked on setting up a ROS node to take care of the Bluetooth communication from laptop's end. Shivam and I worked on setting up the Arduino as a ROS node. Dorothy and Richa are working on the backend functionality of the app. Mohak and Shivam worked together to setup the environment for Kinect by installing all the libraries and their dependencies. They managed to visualize the incoming data and the Kinect is now ready to be integrated with the ROS architecture.

4. Plans

The Preliminary Design Review is next week and we wish to lock down on important factors affecting each subsystem.

1) Communication Subsystem: The components required for setting up a mesh network have been acquired and Shivam is actively work on setting it up. After that, I'll work with him to interface that with ROS so that we can start working on the "collaborative" aspects of the project. To do this I plan to use the multimaster library available in ROS which enables multiple roscores to actively exchange data.
2) Mobile Application Subsystem: Richa and Dorothy are actively working on completing the work on the mobile application. I'll collaborate with them to decide the protocol that'll be followed while communication is being done via Bluetooth. This'll enable easy integration of the mobile app with the ROS architecture.
3) Vision Subsystem: After completing preliminary testing, Shivam, I and Mohak will actively work on implementing Obstacle Detection using Kinect. We've already done a literature survey for the same.
4) Mobile Platform: Once we acquire the mobile platform, Mohak and Richa will setup the Odroid XU4 SBC to operate the platform. They will also mount the Kinect on the platform so that we can start logging some useful test data for our algorithms.

Apart from this, I'll be designing the overall ROS architecture of our system, by dividing processes into nodes and specifying the various publishers, subscribers, services, etc. This will help in task distribution within the team and getting a good high level understanding of the overall functionalities of our system.