Individual Lab Report 5

Progress Review 4

Pulkit Goyal

November 22, 2017

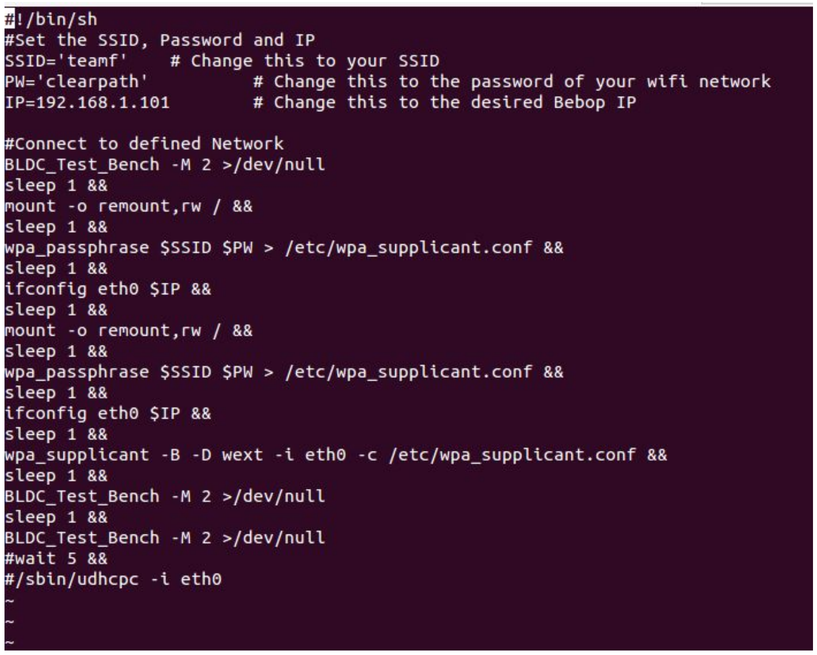
Team F - Falcon Eye
Pulkit Goyal
Pratibha Tripathi
Yuchi Wang
Rahul Ramkrishnan
Danendra singh

Individual Progress

After last PR review I was mainly working on different issues we faced with the Husky and Beebop which includes configuring WiFi network for Bebop, solving the network issue with Husky, and changing the dev rules for plugging in different serial components on the Husky. Making the Husky's communication stable at last. We also tested the network bandwidth and range after connecting the Husky and Bebop on the same network. Started with EKF localization.

1. Wifi Network connection with Bebop

I worked on this task along with Rahul. Bebop is configured by default to work as a Host of SSID of a network, where in any laptop(client) can connect to it, one at a time. Decision of making it work as a client was difficult as we needed to change the in-built functionality of Bebop without much support. Bebop runs a very lighter version of unix operating system having packages which are little different from the ubiquitous ubuntu system. We enabled the telnet on the bebop by reading the different instructions given on the internet. After getting the access to the drone Rahul and me went through various network files that were present on the Bebop. As there wasn't much support available for the network package of the unix OS installed on Bebop on the internet. We posted queries to Bebop's developer's forums which helped us understand the network stack udhcpc, instead of dnsmasq or dhcp which is present on normal ubuntu system. After having a firm understanding of the network stack. It also took us time to realize that the storage system of Bebop is mounted as read-only so we had to mount the storage system as read and write in order to edit any file. We disabled the hosting network and wrote a script which now runs at startup with help of upstart services to join the hosted network as a client instead of hosting it's own network. We provided different functionalities for startup for this service as per convenience.



```
#!/bin/sh
#Set the SSID, Password and IP
SSID='teamf'    # Change this to your SSID
PW='clearpath'          # Change this to the password of your wifi network
IP=192.168.1.101        # Change this to the desired Bebop IP

#Connect to defined Network
BLDC_Test_Bench -M 2 >/dev/null
sleep 1 &&
mount -o remount,rw / &&
sleep 1 &&
wpa_passphrase $SSID $PW > /etc/wpa_supplicant.conf &&
sleep 1 &&
ifconfig eth0 $IP &&
sleep 1 &&
mount -o remount,rw / &&
sleep 1 &&
wpa_passphrase $SSID $PW > /etc/wpa_supplicant.conf &&
sleep 1 &&
ifconfig eth0 $IP &&
sleep 1 &&
wpa_supplicant -B -D wext -i eth0 -c /etc/wpa_supplicant.conf &&
sleep 1 &&
BLDC_Test_Bench -M 2 >/dev/null
sleep 1 &&
BLDC_Test_Bench -M 2 >/dev/null
#wait 5 &&
#/sbin/udhcpc -i eth0
~
~
~
```

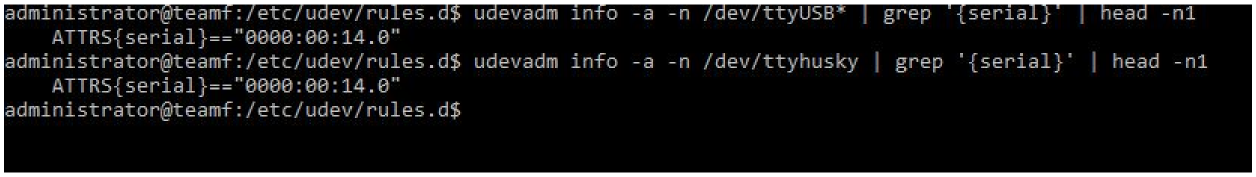
Shell script to connect bebop as a client to teamf network

2. **Network range & bandwidth test with Bebop and Husky's components**

After successful connection of the Bebop to the hosted network, we connected the Husky, remote base station and tested the network with the both the systems moving with Bebop streaming live video to the base station computer. Using this we did range test to verify if the range is as per the commitment in the FVE.

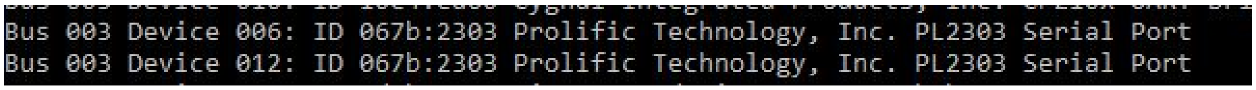
3. **GPS**

TIll previous ILR we were running Husky communication along with the IMU connected to Husky and reading the GPS on a separate laptop. As presented in the demo everything was working fine. But as we shifted the GPS module on the Husky's mini-pc the entire system started mis-behaving. I was working on this issue along with Rahul. After looking into the mounting of the serial devices to the ubuntu operating system we observed that clearpath has defined its own dev rules to create the link names for the mounted serial devices. Surprisingly, all the details including VendorID, ProductID and Serial number were identical for the Serial cable we were using for GPS and Husky communication. Details of the FTDIs that were present in both the serial cables were completely identical, some issue from the manufacturer. As per the rules defined by the clearpath for the serial devices GPS was being mounted with the same name as that of Husky communication cable, i.e. ttyclearpath. Debugging this issue ate up a lot of time as it involved looking into the serial package on system level.



```
administrator@teamf:/etc/udev/rules.d$ udevadm info -a -n /dev/ttyUSB* | grep '{serial}' | head -n1
    ATTRS{serial}=="0000:00:14.0"
administrator@teamf:/etc/udev/rules.d$ udevadm info -a -n /dev/ttyhusky | grep '{serial}' | head -n1
    ATTRS{serial}=="0000:00:14.0"
administrator@teamf:/etc/udev/rules.d$
```

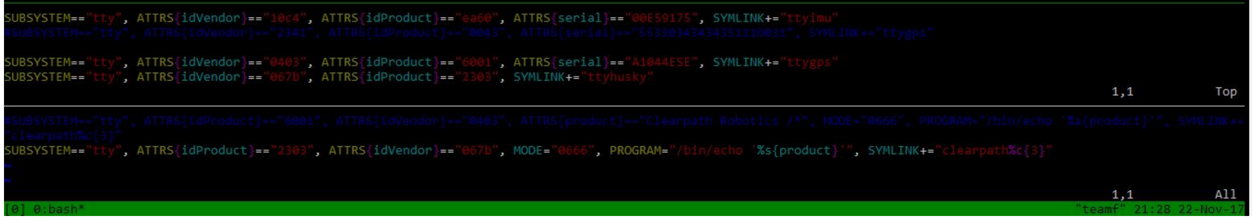
Same serial number for both the FTDIs



```
Bus 003 Device 006: ID 067b:2303 Prolific Technology, Inc. PL2303 Serial Port
Bus 003 Device 012: ID 067b:2303 Prolific Technology, Inc. PL2303 Serial Port
```

Same product and Vendor ID

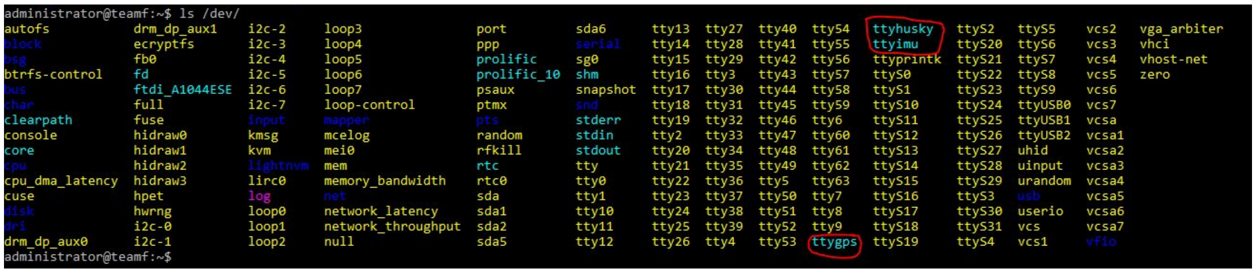
Finally we decided to change the Serial cable for GPS, we have also ordered some backup Serial cables to prevent any kinds of issues in future. We defined our own dev rules over-riding those of clearpath according to which we are creating the following links to keep the things working without any collision: GPS - **ttygps**, IMU - **ttyimu** and Husky communication - **ttyhusky**. This helped us clearly attaching these serial devices with their drivers in the launch files.



```
SUBSYSTEM=="tty", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60", ATTRS{serial}=="00E59175", SYMLINK+="ttyimu"
#SUBSYSTEM=="tty", ATTRS{idVendor}=="2341", ATTRS{idProduct}=="0043", ATTRS{serial}=="55330343434351110031", SYMLINK+="ttygps"

SUBSYSTEM=="tty", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", ATTRS{serial}=="A1044ESE", SYMLINK+="ttygps"
SUBSYSTEM=="tty", ATTRS{idVendor}=="067b", ATTRS{idProduct}=="2303", SYMLINK+="ttyhusky"
                                                                                          1,1           Top
#SUBSYSTEM=="tty", ATTRS{idProduct}=="6001", ATTRS{idVendor}=="0403", ATTRS{product}=="Clearpath Robotics /*", MODE="0666", PROGRAM="/bin/echo '%s{product}'", SYMLINK+=
"clearpath%c{3}"
SUBSYSTEM=="tty", ATTRS{idProduct}=="2303", ATTRS{idVendor}=="067b", MODE="0666", PROGRAM="/bin/echo '%s{product}'", SYMLINK+="clearpath%c{3}"
~
~
                                                                                          1,1           All
[0] 0:bash*                                                                      "teamf" 21:28 22-Nov-17
```

Dev rules defined for serial devices

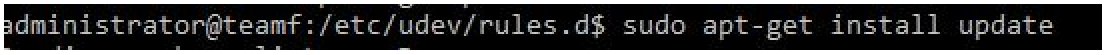


```
administrator@teamf:~$ ls /dev/
autofs           drm_dp_aux1    i2c-2     loop3              port         sda6      tty13  tty27  tty40  tty54  ttyhusky   ttyS2   ttyS5    vcs2   vga_arbiter
block            ecryptfs       i2c-3     loop4              ppp          serial    tty14  tty28  tty41  tty55  ttyimu     ttyS20  ttyS6    vcs3   vhci
bsg              fb0            i2c-4     loop5              prolific     sg0       tty15  tty29  tty42  tty56  ttyprintk  ttyS21  ttyS7    vcs4   vhost-net
btrfs-control    fd             i2c-5     loop6              prolific_10  shm       tty16  tty3   tty43  tty57  ttyS0      ttyS22  ttyS8    vcs5   zero
bus              ftdi_A1044ESE  i2c-6     loop7              psaux        snapshot  tty17  tty30  tty44  tty58  ttyS1      ttyS23  ttyS9    vcs6
char             full           i2c-7     loop-control       ptmx         snd       tty18  tty31  tty45  tty59  ttyS10     ttyS24  ttyUSB0  vcs7
clearpath        fuse           input     mapper             pts          stderr    tty19  tty32  tty46  tty6   ttyS11     ttyS25  ttyUSB1  vcsa
console          hidraw0        kmsg      mcelog             random       stdin     tty2   tty33  tty47  tty60  ttyS12     ttyS26  ttyUSB2  vcsa1
core             hidraw1        kvm       mei0               rfkill       stdout    tty20  tty34  tty48  tty61  ttyS13     ttyS27  uhid     vcsa2
cpu              hidraw2        lightnvm  mem                rtc          tty       tty21  tty35  tty49  tty62  ttyS14     ttyS28  uinput   vcsa3
cpu_dma_latency  hidraw3        lirc0     memory_bandwidth   rtc0         tty0      tty22  tty36  tty5   tty63  ttyS15     ttyS29  urandom  vcsa4
cuse             hpet           log       net                sda          tty1      tty23  tty37  tty50  tty7   ttyS16     ttyS3   usb      vcsa5
disk             hwrng          loop0     network_latency    sda1         tty10     tty24  tty38  tty51  tty8   ttyS17     ttyS30  userio   vcsa6
dri              i2c-0          loop1     network_throughput sda2         tty11     tty25  tty39  tty52  tty9   ttyS18     ttyS31  vcs      vcsa7
drm_dp_aux0      i2c-1          loop2     null               sda5         tty12     tty26  tty4   tty53  ttygps ttyS19     ttyS4   vcs1     vfio
administrator@teamf:~$
```

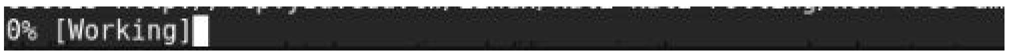
Links to the devices

4. Solving the network issue with Husky

We could use the internet but could not install any packages connecting to the Cmu-Secure. Actually it was pretty unpredictable, as sometimes we could install the packages but most of the time it got stuck on 0%. Me and Rahul spent a lot of time debugging this issue as even after connecting the mini-pc with the LAN we were unable to install any packages. As there was not much support from the internet we made a fresh installation for the ubuntu on the mini-pc thinking we would have corrupted the OS. But even after the fresh installation we faced the same issue. We concluded that there is some issue with installing the packages using Cmu-secure, we are using the personal mobile hotspot to install the packages, which is working fine.



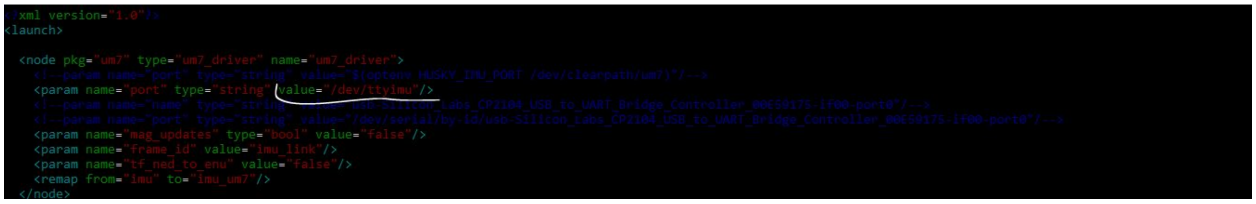
```
administrator@teamf:/etc/udev/rules.d$ sudo apt-get install update
```



```
0% [Working]
```

5. Making the Husky's communication stable at last.

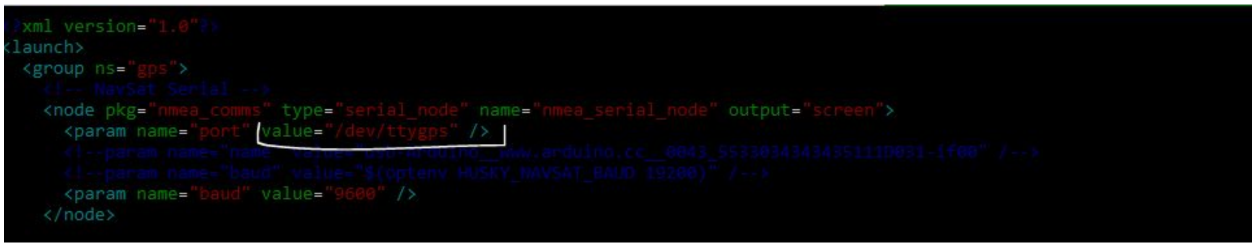
Finally after everything was running stably on the Husky-pc/mini-pc, we conducted several tests to check if the communication goes does for any of the components when connected at the same time via serial communication. We also called the launch files which was made by pratibha at the startup of the operating system using upstart services.



```
<?xml version="1.0"?>
<launch>

  <node pkg="um7" type="um7_driver" name="um7_driver">
    <!--param name="port" type="string" value="$(optenv HUSKY_IMU_PORT /dev/clearpath/um7)"/-->
    <param name="port" type="string" value="/dev/ttyimu"/>
    <!--param name="name" type="string" value="usb-Silicon_Labs_CP2104_USB_to_UART_Bridge_Controller_00E59175-if00-port0"/-->
    <!--param name="port" type="string" value="/dev/serial/by-id/usb-Silicon_Labs_CP2104_USB_to_UART_Bridge_Controller_00E59175-if00-port0"/-->
    <param name="mag_updates" type="bool" value="false"/>
    <param name="frame_id" value="imu_link"/>
    <param name="tf_ned_to_enu" value="false"/>
    <remap from="imu" to="imu_um7"/>
  </node>
```

Imu launch file pairing with serial link



```
<?xml version="1.0"?>
<launch>
  <group ns="gps">
    <!-- NavSat Serial -->
    <node pkg="nmea_comms" type="serial_node" name="nmea_serial_node" output="screen">
      <param name="port" value="/dev/ttygps" />
      <!--param name="name" value="usb-Arduino__www.arduino.cc__0043_55330343434351110051-if00" /-->
      <!--param name="baud" value="$(optenv HUSKY_NAVSAT_BAUD 19200)" /-->
      <param name="baud" value="9600" />
    </node>
```

Gps launch file pairing with serial link

6. **Started with EKF localization.**

After getting all the data from encoders(husky's communication), IMU and GPS, we have started working on fusing all this data to have a stable localization which can be used for multi waypoint navigation which is one of our major requirements for FVE.

Challenges

1. There is a very major misconception that husky is a famous platform with a lot of support so it should run with typing a set of commands or instruction straight out of the manual. I would say we faced a lot of problem with this misconception. As the work we did from the last PR wasn't actually contributing to any requirement or any visible test case but it was actually important for running the complete system. It was very difficult to make everyone understand that even though we weren't actually progressing with debugging these issues, but solving them will help us do higher level development of algorithms or localization or processing of lidar data without any deadlocks.
2. Systems are unpredictable at the start, specifically systems like Husky, where a lot of things are configured by the clearpath as a blackbox for anyone. If even a single module written by them starts misbehaving it's difficult to look into their module to look for errors. But time spent on solving these kinds of deadlocks is generally not visible and can't be shown in form of any FVE requirement check.
3. Expecting these integration issues we kept a buffer time and planned to complete everything 1 week before fve. Getting stuck in these issues also affected our timeline a lot. Initially we planned to show Bebop GPS waypoint navigation and teleoperation of Husky, while parallely working on the issues with the integration of different sensors. After feedback from multiple stakeholders, we added the GPS waypoint navigation for the FVE as a test. We are working really hard to meet the goals on time, as issues that we solved from the last PR only surface after a lot of testing. I am hoping we will have enough time to test all the module together with each other.
4. As this is the end of semester we also have a lot of assignment deadlines. We are finding it little tough to balance our time for everything. Specifically the team members who are enrolled in CV course.

Teamwork

I and Rahul worked on all the issues explained above in the ILR. Pratibha worked on writing launch files for the different modules to smoothen the integration once these issues were solved. Danendra and Yuchi were working together on the GPS waypoint navigation of the Bebop2. Yuchi was also working on the April tag detection and improving it’s accuracy by applying different filters.

Future Plans

We plan to successfully perform the mentioned tests in FVE. We specifically plan to work on the following tasks.

1. GPS waypoint navigation of the Husky Robot.
2. Stabilizing the GPS waypoint navigation of the Bebop2.
3. Integrating the entire system and running it together.
4. We also have to route the wiring of the robot so that we don't face any issues with random disconnection at the time of final demo.
5. Conducting the tests as specifically mentioned in the FVE document.