

Report Date: 07/01/2022

To: [ematson@purdue.edu](mailto:ematson@purdue.edu), [ahsmith@purdue.edu](mailto:ahsmith@purdue.edu), [lhiday@purdue.edu](mailto:lhiday@purdue.edu) and [lee3450@purdue.edu](mailto:lee3450@purdue.edu)

From: FarmVroong

- Seongil Heo ([tjddlf101@hufs.ac.kr](mailto:tjddlf101@hufs.ac.kr))
- Jiwon Park ([overflow21@khu.ac.kr](mailto:overflow21@khu.ac.kr))
- Jueun Mun ([cindy4741@khu.ac.kr](mailto:cindy4741@khu.ac.kr))
- Jiwoong Choi ([jwtiger22@khu.ac.kr](mailto:jwtiger22@khu.ac.kr))

## Summary

SLAM code was developed almost all and being revised to execute well at the ROS. Searched the way how to calculate the unknown GPS. Lastly, we reviewed BIT\* code and decided the point which is going to be developed.

## What FarmVroong completed this week:

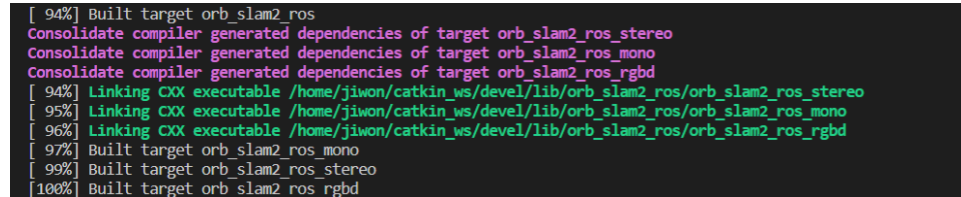
- Build SLAM code in Jetson.

This week, The SLAM code with the GPS data was built in Jetson. At first, the code was built successfully. However, GUI was not applying to the code.

For testing in the outdoor, visualizing the actual data is needed. For this reason, GUI part was also implemented in the SLAM.

Showing the windows was succeeded; however, the camera did not fit with the code.

The next steps of the SLAM code are connecting SLAM part and GUI part.



```
[ 94%] Built target orb_slam2_ros
Consolidate compiler generated dependencies of target orb_slam2_ros_stereo
Consolidate compiler generated dependencies of target orb_slam2_ros_mono
Consolidate compiler generated dependencies of target orb_slam2_ros_rgbd
[ 94%] Linking CXX executable /home/jiwon/catkin_ws/devel/lib/orb_slam2_ros/orb_slam2_ros_stereo
[ 95%] Linking CXX executable /home/jiwon/catkin_ws/devel/lib/orb_slam2_ros/orb_slam2_ros_mono
[ 96%] Linking CXX executable /home/jiwon/catkin_ws/devel/lib/orb_slam2_ros/orb_slam2_ros_rgbd
[ 97%] Built target orb_slam2_ros_mono
[ 99%] Built target orb_slam2_ros_stereo
[100%] Built target orb_slam2_ros_rgbd
```

Fig. 1. SLAM building process

- Motor control in ROS environment

Last week, we controlled DC motor with python3. However, to integrate motor control in ROS SLAM package, it was necessary to get keyboard input as a ROS message (geometry\_msgs/Twist) and send it to the motor.

Also, we used a joystick controller by using /joy topic.

- Receive GPS data in ROS environment

Last week, we checked GPS data with gpsd package. However, to integrate GPS into our ROS package, we used nmea\_navsat\_driver and received message (sensor\_msgs/NavSatFix).

- Searched how to calculate the unknown GPS of map point by known GPS and depth information.

By the Haversine formula, simultaneous equations were made. Two equations find two intersections of the circles, and another circle decides the exact one intersection. However, it still shows some errors, so it is necessary to find a different way to calculate GPS.

- Bit\* path planning code review

Reviewed Bit\* code [1] and decided which part to revise and improve. GPS will be added, and the stable points marked by GPS will get more weight. This process is still being done.

### **Things to do by next week**

- Connect SLAM part and GUI part.
- Finish developing BIT\* code
- Find the way how to calculate GPS.

### **Problems or challenges:**

- Solve the system of trigonometrical equations in C++.
- Import PCA9685 package in ROS environment
- What makes the work more complicated is that we are using the PCA9685 pwm control board. When we were dealing with the python environment, there was a package named adafruit\_pca9585 that helped us to use DC and servo motors easily.
- However, there was no official PCA9685 package that runs in ROS c++, and it was not easy to build a new rospy package to reuse the python code.
- At last, we found one repository that uses servo by roserial and implemented that to our project successfully.

### **References**

[1] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," GitHub, Dec. 01, 2012. <https://github.com/ompl/ompl>