Report Date: 06/24/2022
To: ematson@purdue.edu, ahsmith@purdue.edu, lhiday@purdue.edu and lee3450@purdue.edu
From: FarmVroong

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

Summary

The code of ORB-SLAM2 [1] is developed by adding and changing functions using GPS data. Building the actual car with John Deere toy vehicle was started. In the process, the circuit of the rear wheels was soldered together. The data from GPS sensor was tested.

What FarmVroong completed this week:

- Developed the ORB-SLAM2 code.

  In ORB-SLAM2, there are three main steps: Tracking, Local Mapping, and Loop Closing. The source code consists of over 10 source files due to the Classes operating the three main threads, such as 'Frame.cc', 'KeyFrame.cc', and 'MapPoint.cc.' Therefore, first of all, the fundamental source files were revised, adding functions which get and set GPS data. Next, 'Tracking.cc', 'LocalMapping.cc', and 'LoopClosing.cc' were revised.

  At the Tracking stage, GPS data discerns whether the car moves or not, and if the car does not move, the Tracking thread breaks for saving bootless computing. Also, it is decided whether to make a new keyframe by three conditions at the ORB-SLAM2. However, one new condition was added that makes keyframe be made only when the current frame moves certain distance from the latest keyframe.

  At the LocalMapping stage, the local map was made by the similarity of map points shared between each frame, and the degree of similarity were assigned as a weight, which makes the priority. However, the GPS location between each map made new weight and were added to the previous weight, ultimately making total weight. The closer the distance between keyframes is, the bigger the GPS weight is.

  At the LoopClosing stage, the loopclosing thread was changed to operate only when the car makes the new keyframe which are near the one of the 'KeyframeDataBase', which is a set of previous keyframes. The degree of the near will be defined as a threshold. The exact real number of the threshold will be decided by future experiments.

- Started building the actual car with John Deere toy vehicle.

  We made the circuits using PCA9685 (GPIO extension module for Jetson Nano or Raspberry Pi) and checked that the motors (DC encoder motor, Servo motor) worked well.

  The python code was written with class "drive," which helps control the motors by keyboard inputs. However, when it comes to ROS environment, the cpp code will be written to employ the system to ROS.

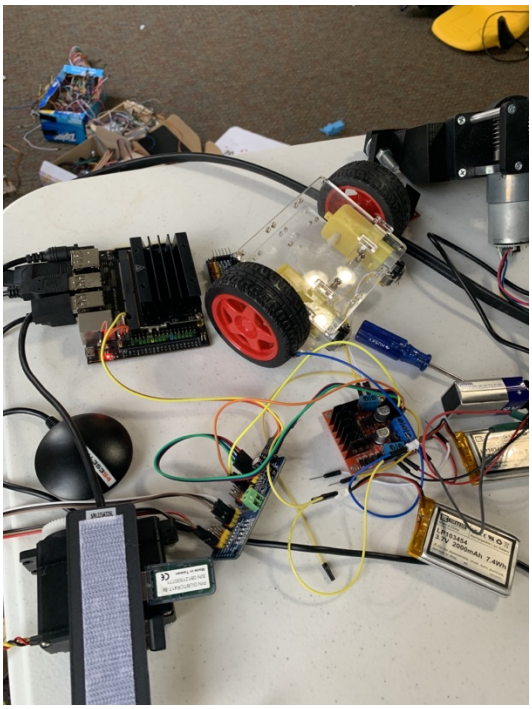

Fig 1. Building the circuits

- Soldered the rear wheels together.

  There are two ways to steer the car. One is to use the difference of the velocity between 2 wheels, and the other is to use the steering motor in front of the car. We decided to use the latter way.

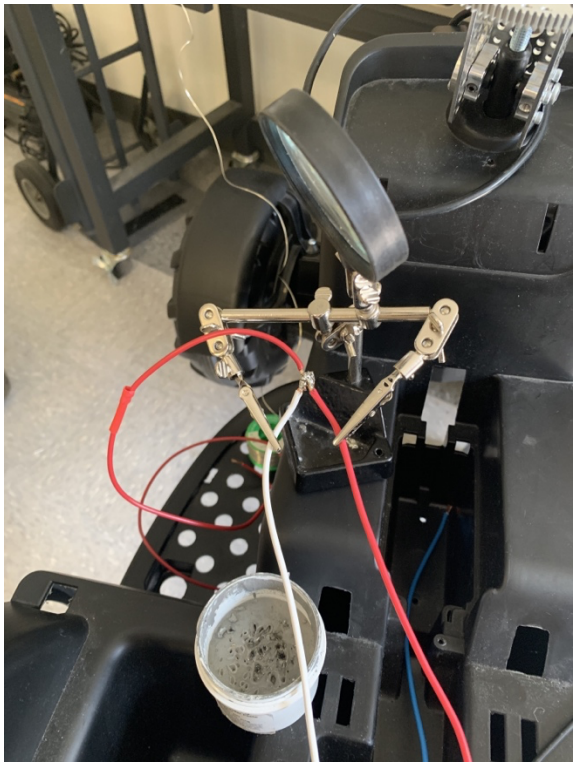

Fig 2. Soldering the rear wheels each other

- GPS data test

We checked whether the GPS data is accurate and integrated the code with ROS environment. So now we can get the GPS data in the form of ROS topic.

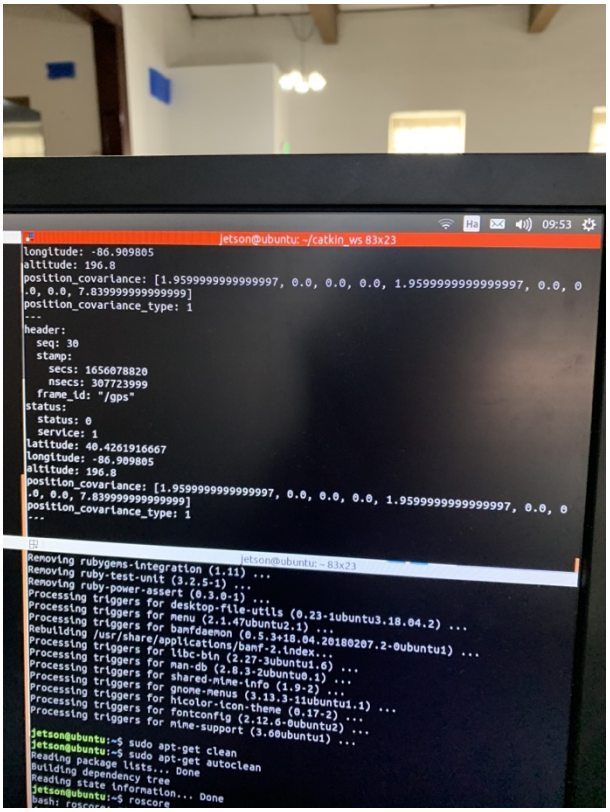

Fig 3. Getting the GPS data in the form of ROS topic

Things to do by next week

- Develop the off-road Path Planning code.
- Indoor Test with the car.

Problems or challenges:

- Coming up with the idea of controlling the car

At first, the steering system using DC encoder motor was made. We also have made 3D printed stand for that motor. However, we discovered that the DC motor does not move exactly what we ordered (a little more or less), so we tested servo motors.

Nevertheless, as the discussion continued, our thought changed. Even if the motor moves correctly what we have ordered, the car cannot move in the right way because we will test the car in an outdoor environment.

Finally, we decided to make a feedback loop motor control system and keep correcting our route while driving. Also, we will use DC motors to get more substantial power.

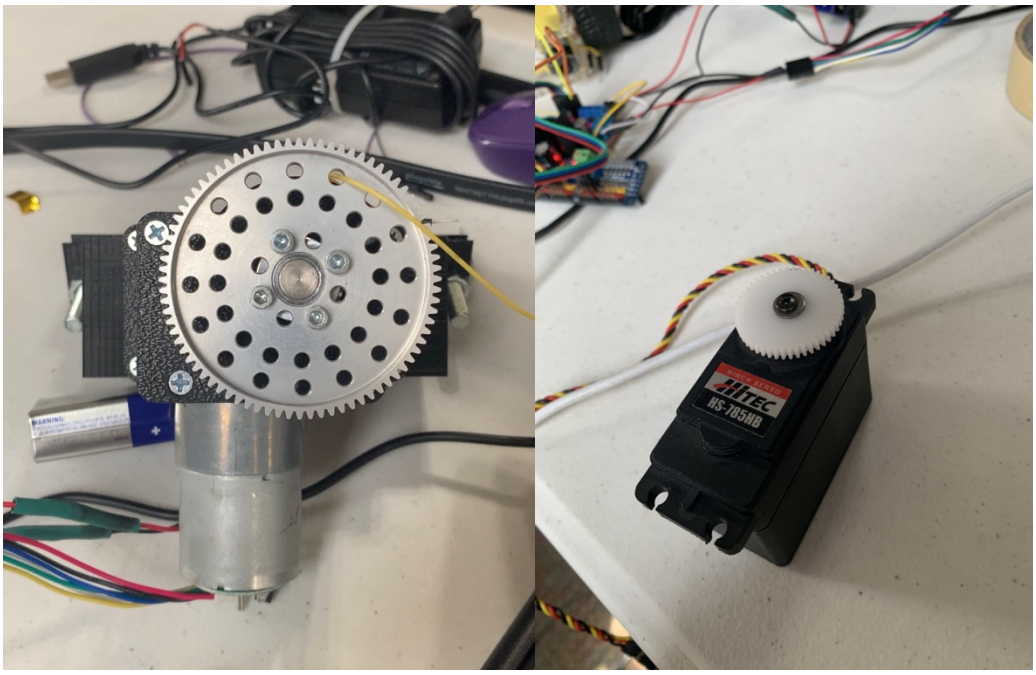

Fig 4. (a) DC encoder motor (b) Servo motor

References

[1] R. Mur-Artal and J. D. Tardos, “Orb-slam2: An open-source slam system for monocular, stereo, and RGB-D cameras,” IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255–1262, 2017.

[2] J. Gammell, S. Srinivasa, and T. Barfoot, “Batch Informed Trees (BIT*): Sampling-based Optimal Planning via the Heuristically Guided Search of Implicit Random Geometric Graphs.”

[3] raulmur, “raulmur/ORB_SLAM2,” GitHub, Oct. 11, 2017. https://github.com/raulmur/ORB_SLAM2