EE 491 Senior Design May 2018 Group Meeting

September 2017~May 2018
Client: Vishal Mahulkar
Advisor: Dr. Hegdey Chinmay

Safe Communication Between Lead and Following Vehicle

Week 4 Report

Team Members:

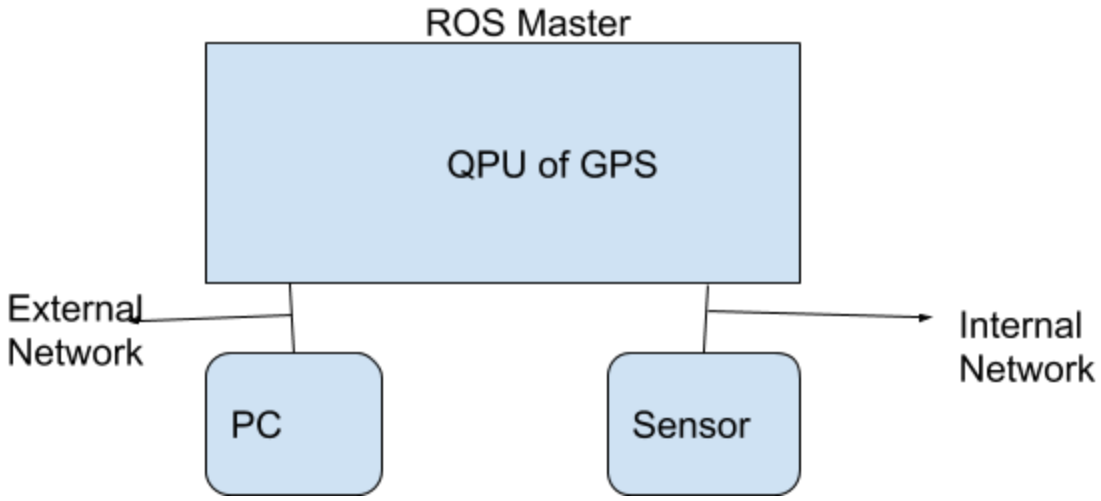
Bradley Stiff- Software Lead, Project Lead
Justin Wheeler- Software Lead
Sanguk Park- Scribe Lead, Communication Lead
Zhize Ma- Scribe Lead, Hardware Lead
Junho Chun- Hardware Lead
Yifan Lu- Hardware Lead
Jose Candelario- Project Lead, Communication Lead

This Week's Accomplishments:
For this meeting, we had a big talk about the software part of the project. The software aspect of our project is key in making sure that we receive and transmit the data in a format that can be suitable for the teams to observe and predict what certain actions will go through the following car. Our software team: Justin and Brad, are responsible of receiving that information from each of the two cars and making sure that the data communication between the cars are consistent to the data output by the sensors. For the first three weeks, our group has been going over the hardware aspects and all of the sensors that are included within our project. We also needed a big briefing on the software portion of the project.

For this meeting, we met up with one of the lead controls members to have us walk through the software role of this project. Our software team were to use an ROS (robot

operating software) do observe the data between the cars and make sure that each car was transmitting and receiving the correct data. From the controls group lead our software group was given the following instructions:

- For PC, the computer most have dual operating systems with the second OS being Ubuntu 16.04.3 LTS
- ROS > Lunar Loggerhead
- Install ROS at ros.org > download> Ubuntu 16.014.3 > Follow the steps listed on the website
- Open sensors' files and create an archive. Make sure to upload them on sharepoint.
- When using the ROS, it uses commands used on the robot parts: Python, C++



The control groups lead gave us a brief explanations on the technical roles that the ROS plays in the communication between the computer and the sensors. From his briefing, we found out that the robot operating system functioned as a middle ground for both the lead and following vehicle. The data that was transmitted by the lead vehicle would be sent to both the ROS and the following vehicle and the controls team can observe the data that is transmitted between them. Although the car itself has its own eyes and ears utilizing the hardware sensors, the software that communicates between them.

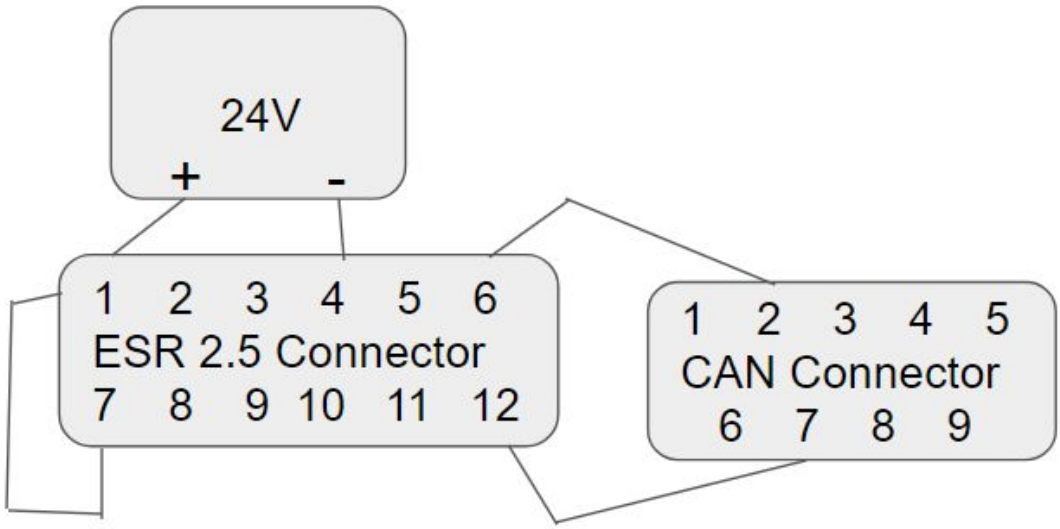
The Hardware Specs:

The Radar

For this week, we also looked into the specs of the sensors that we were working with in order to gain a better idea of what format of output and the function that they each served. The data sheet for the radar shown below:

We were initially working with wiring the DB9 cable of the radar which would enable the observation of its output data. From this spreadsheet, we were to base the wiring off these specifications.

Pin Number	Singal	Port Color
1	Battery (+24V)	Red
2	USB D+ (Green Wire)	Green (USB)
3	USB D- (White Wire)	White (USB)
4	Ground	Black
5	USB Ground (Black Wire)	Black (USB)
6	PRVCANL	Green
7	Ignition (+24V)	White
8	USB +5V (Red Wire)	Red (USB)
9	VEHCAN L	Blue
10	VEHCAN H	Brown
11	VEHCAN Shield	
12	PRVCANH	Orange



The ESR has a long range of 174 meters with a view of 20 degrees but it also has a medium range of 60 meters with a view of 90 degrees. Both modes can update at the same speed.

The Camera

The current specs of the camera are shown below, but currently our camera is still in the ordering progress, but we do know that the specs will be consistent with the ones shown above

	BFS-U3-51S5M	BFS-U3-51S5C
Firmware Version	**1605.1.3.0**	**1605.1.3.0**
Resolution	**2448 x 2048**	**2448 x 2048**
Frame Rate	**75 FPS**	**75 FPS**
Megapixels	**5.0 MP**	**5.0 MP**
Chroma	**Mono**	**Color**
Sensor	**Sony IMX250, CMOS, ⅔”**	**Sony IMX250, CMOS, ⅔”**
Readout Method	**Global Shutter**	**Global Shutter**
Pixel Size	**3.45 um**	**3.45 um**

Lens Mount	C-mount	C-mount
ADC	10 bit / 12 bit	10 bit / 12 bit
Minimum Frame Rate	1 FPS	1 FPS
Gain Range	0 to 47 dB	0 to 47 dB
Exposure Range	6 us to 30 s	6 us to 30 s
Acquisition Modes	Continuous, Single Frame, Multi Frame	Continuous, Single Frame, Multi Frame
Partial Image Modes	Pixel binning, decimation, ROI	Pixel binning, decimation, ROI
Image Processing	Gamma, look up table, and sharpness	Color correction matrix, gamma, look up table, hue, saturation, and sharpness
Sequencer	Up to 8 sets using 2 features, exposure and gain	Up to 8 sets using 2 features, exposure and gain
Image Buffer	240 MB	240 MB
User Sets	2 user configuration sets for custom camera settings	2 user configuration sets for custom camera settings
Flash Memory	6 MB non-volatile memory	6 MB non-volatile memory
Opto-isolated I/O	1 input, 1 output	1 input, 1 output
Non-isolated I/O	1 bi-directional, 1 input	1 bi-directional, 1 input
Auxiliary Output	3.3V, 120 mA maximum	3.3V, 120 mA maximum
Interface	USB 3.0	USB 3.0
Power Requirements	8-24V via GPIO OR 5V via USB 3.0 interface	8-24V via GPIO OR 5V via USB 3.0 interface
Power Consumption	3 W maximum	3 W maximum
Dimensions/Mass	29 mm x 29 mm x 30 mm / 36g	29 mm x 29 mm x 30 mm / 36g
Machine Vision Standard	USB3 Vision v1.0	USB3 Vision v1.0

The Lidar

Power supply
- 24VDC
- 1.5A(max)
- Protection
  - Short Circuit
  - Over Current
  - Over Voltage
- Ripple
  - 1-2%Vpp

Use Quanergy Processing Unit (QPU)
-pre configured at the factory as a complete solution that includes necessary source code, library, and third party applications.

We need also(to make it work)

Power Source
- To power the sensor, we need to do this efficiently and take the other sensors into account.(mobile battery required for all the sensors)

Mouse+keyboard+Monitor
- Support computing environment.

Mounting surface
- Affix the sensor(I believe the Mechanical team will probably handle this)

Ethernet switch + Power adaptor
- To handle multiple sensors, Netgear ProSafe GS108 recommended

GPS/IMU module
- Report position and supply the NMEA/PPS timing signals(OXTS RT3003 and VectorNav200 are supported by Quanergy)

Lidar has multiple returns(3)
- Maximum, Second Strongest, and Last

Need to connect sensor to Ubunta Host computer
- Page 26 really talks about the process in how to connect it

Laser Firing:
- Sensor spins at 10Hz
- Lasers fire at 53,828Hz
- They fire at 8 different angles
  - (+3.2 to -18.25 degrees)

For debugging:
(get a power adapter with blinking light for debugging)

Get Infrared (IR) scope to see the rapid, rhythmic laser flashes.
The Q-view application's Dashboard lets you check a sensor's health and any error codes that might pop up.
The M8 Sensor Settings Management application's Versions tab reports versions of the software associated with the M8 sensor.

Individual Contributions (9/15~9/22)

Team Member	**Contribution**	**Weekly Hours**	**Total Hours**
Brad Stiff	Researched C++/Python support with ROS Lunar. Began to learn Python by completing online tutorials. Also looked into working with the ROS software in Ubuntu terminal.	4	15
Jose Candelario	DId some more research on Lidar. Started looking into DSRC as a solution.	4	12
Junho Chun	Did research on the radar sensors and looked up datasheet and spec for radar.	3	13
Justin Wheeler	Completed ROS tutorials to get more familiar with how it works. Also, refreshed myself with C++ and Python.	3	9
Sang Uk Park	Did research on the lidar sensors and researched data on the wiring of the radar cables to enable the observation of the output data	3	15
Yifan Lu	Looked over GPS data sheet and specs. Looking into cellular platform as an alternative communication method	3	12
Zhize Ma	Worked on camera search and datasheet, discussed about data transmission method	3	14