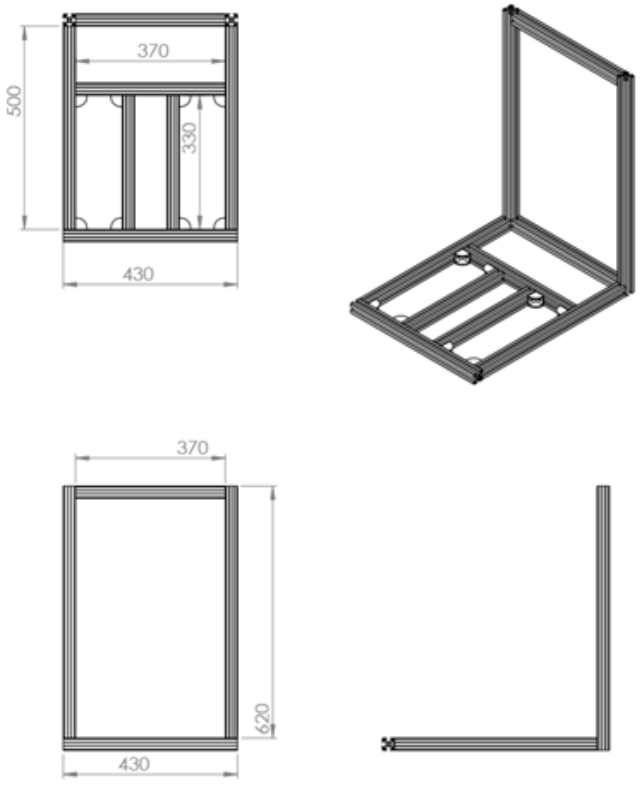
Data Collection Trolley (Data collection and visualization)

- **Setup**
  - **Frame Setup**
  - **Electrical Circuitry**
  - **Sensor Integration**
  - **Extrinsic Sensor Calibration**
- **Data Collection**
- **Projection/Visualization**
- **Mapviz**
- **GPS coordinates visualisation using KML file on Google Earth**

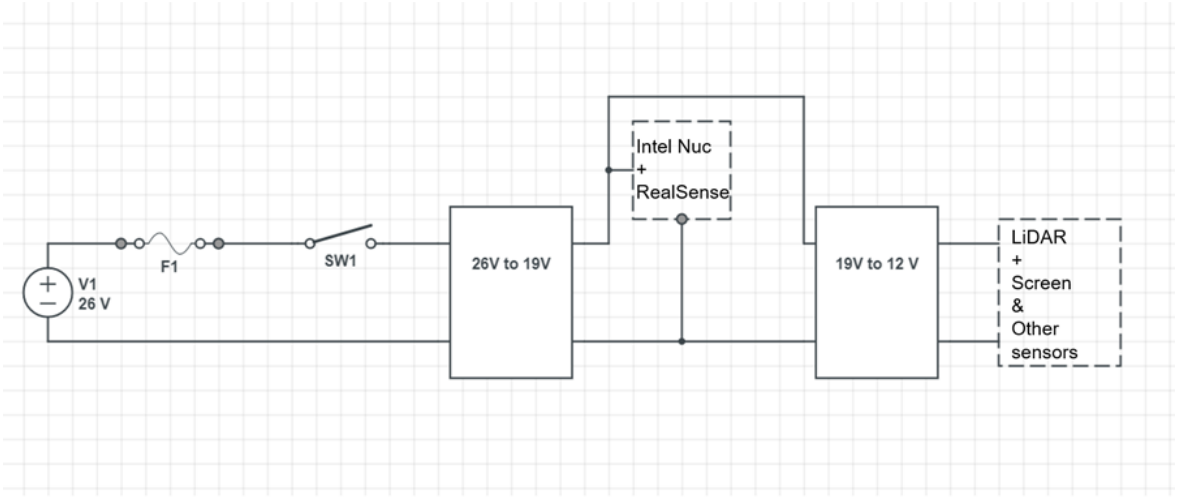
a) Frame Setup



Extrusion Length	No. of Pieces
620mm	2
500mm	2
430mm	2
370mm	2

Other Parts
Aluminium Extrusion L Joint (x15)
Fixed Wheel (x2) Castor Wheels: 8mm Pneumatic Wheels
Swivel Wheel (x2) Castor Wheels: 8mm Pneumatic Wheels

b) Electrical Circuitry



Parts needed:	**Description:**
12 AWG wire **S$75.80** Manufacturer: ALPHA WIRE Manufacturer Part No: 3080 RD005	12 AWG 30.5m reel
Heat shrinks **S$20.79** Manufacturer: PRO POWER Manufacturer Part No: 13633	1.6 mm Black 100 m

Toggle switch **S$6.33** Manufacturer: CARLING TECHNOLOGIES Manufacturer Part No: 2FC53-73/TABS	10A 250V
Inline fuse *Buy from Sim Lim	10A
Converter *Use existing converter	26 to 19V
Cable ties **S$3.33** Manufacturer: HELLERMANNTYTON Manufacturer Part No: T18R-BLACK(100 PACK)	100 Pieces
Electricals box **S$42.18** Manufacturer: HELLERMANNTYTON Manufacturer Part No: AS55	220mm, 300mm, 120mm
Total Cost:	$148.43

c) Sensor integration

Sensor	Instructions
LiDAR	From https://github.com/RoboSense-LiDAR/ros_rslidar/tree/develop-curves-function: 1. Prerequisites (1) Install a ubuntu PC. We suggested Ubuntu 14.04 or Ubuntu 16.04. Please do not use virtual machine. (2) Install ros full-desktop version. We tried Indigo and Kinect. (3) Please also install libpcap-dev. 2. Install (1). Copy the whole rslidar ROS driver directory into ROS workspace, i.e "~/catkin_ws/src". cd ~/catkin_ws/src git clone https://github.com/RoboSense-LiDAR/ros_rslidar/tree/master (2). Check the file attributes: cd ~/catkin_ws/src/ros_rslidar/rslidar_drvier chmod 777 cfg/* cd ~/catkin_ws/src/ros_rslidar/rslidar_pointcloud chmod 777 cfg/* (3). Then to compile the source code and to install it: cd ~/catkin_ws catkin_make 3. Configure PC IP By default, the RSLIDAR is configured to 192.168.1.200 as its device IP and 192.168.1.102 as PC IP that it would communicate. The default MSOP port is 6699 while DIFOP port is 7788. So you need configure your PC IP as a static one 192.168.1.102. For this part, check the interfaces file by doing: cd /etc/network sudo nano/emacs/vi interfaces Within the interfaces file, you should change something from 'dynamic' to 'static'.

	4. Run as independent node We have provide example launch files under rslidar_pointcloud/launch, we can run the launch file to view the point cloud data. For example, if we want to view RS-LiDAR-16 real time data: (1). Open a new terminal and run: cd ~/catkin_ws source devel/setup.bash roslaunch rslidar_pointcloud rs_lidar_16.launch (2). Open a new terminal and run: rviz Set the Fixed Frame to "rslidar". Add a Pointcloud2 type and set the topic to "rslidar_points". 5. Run as nodelet We can also run the driver node and cloud node as a nodelet. Open a new terminal and run: cd ~/catkin_ws source devel/setup.bash roslaunch rslidar_pointcloud cloud_nodelet.launch Then we can run view the pointcloud via "rviz" 6. About the lidar calibration parameters Under "rslidar_pointcloud/data" directory, you can find the lidar calibration parameters files for the exact sensor. By default the launch file "rs_lidar_16.launch" load the three files: • rslidar_pointcloud/data/rs_lidar_16/angle.csv • rslidar_pointcloud/data/rs_lidar_16/ChannelNum.csv • rslidar_pointcloud/data/rs_lidar_16/curves.csv. If you have more than one RSLIDAR, you can create new sub-directories under the "rslidar_pointcloud/data/", and put the data files into it. Then you need rewrite the launch file to start you lidar. We have put an example launch file "two_lidar.launch" to load two lidars together for reference. Begin from 2018.09.01 In the launch file, we could set the MSOP port and DIFOP port. MSOP port is used for receive the main point cloud data, while DIFOP port is used for receive the device information data. If we set the right DIFOP port, we will get the lidar calibration parameters from the DIFOP packets not from the files, so can ignore the local lidar calibration files. About how to check the DIFOP port, please reference the Appendix G in RS-LiDAR manual
RealSense Camera	From https://github.com/IntelRealSense/librealsense/blob/development/doc/distribution_linux.md: **Installing the packages:** • Register the server's public key: `sudo apt-key adv --keyserver keys.gnupg.net --recv-key C8B3A55A6F3EFCDE || sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-key C8B3A55A6F3EFCDE` In case the public key still cannot be retrieved, check and specify proxy settings: `export http_proxy="http://<proxy>:<port>"` , and rerun the command. See additional methods in the following link. • Add the server to the list of repositories: Ubuntu 16 LTS: `sudo add-apt-repository "deb http://realsense-hw-public.s3.amazonaws.com/Debian/apt-repo xenial main" -u` Ubuntu 18 LTS: `sudo add-apt-repository "deb http://realsense-hw-public.s3.amazonaws.com/Debian/apt-repo bionic main" -u` • Install the libraries (see section below if upgrading packages): `sudo apt-get install librealsense2-dkms sudo apt-get install librealsense2-utils` The above two lines will deploy librealsense2 udev rules, build and activate kernel modules, runtime library and executable demos and tools. • Optionally install the developer and debug packages: `sudo apt-get install librealsense2-dev sudo apt-get install librealsense2-dbg` With `dev` package installed, you can compile an application with **librealsense** using `g++ -std=c++11 filename.cpp -lrealsense2` or an IDE of your choice. Reconnect the Intel RealSense depth camera and run: `realsense-viewer` to verify the installation. Verify that the kernel is updated : `modinfo uvcvideo | grep "version:"` should include `realsense` string **Tips:** If the system is unable to detect the Realsense camera, go to https://www.intel.sg/content/www/xa/en/support/articles/000028171/emerging-technologies/intel-realsense-technology.html and follow the PDF instructions. You may need to disable secure boot on the Intel NUC.
GPS	To download the drivers: https://www.globalsat.com.tw/en/features-10593/BU-353S4.html For additional reference: http://docs.ros.org/melodic/api/robot_localization/html/integrating_gps.html Look at mnea_navsat_driver from ROS.

IMU	UM7 Driver to be installed: https://github.com/ros-drivers/um7.git Ros Wiki: http://wiki.ros.org/um7
Radar	Contact Shanoop for more information. Remember to do PTP time sync. On the Intel NUC, alias has been set as `ptptime1` and `ptptime2`.

d) Extrinsic Sensor Calibration (LiDAR and RealSense)

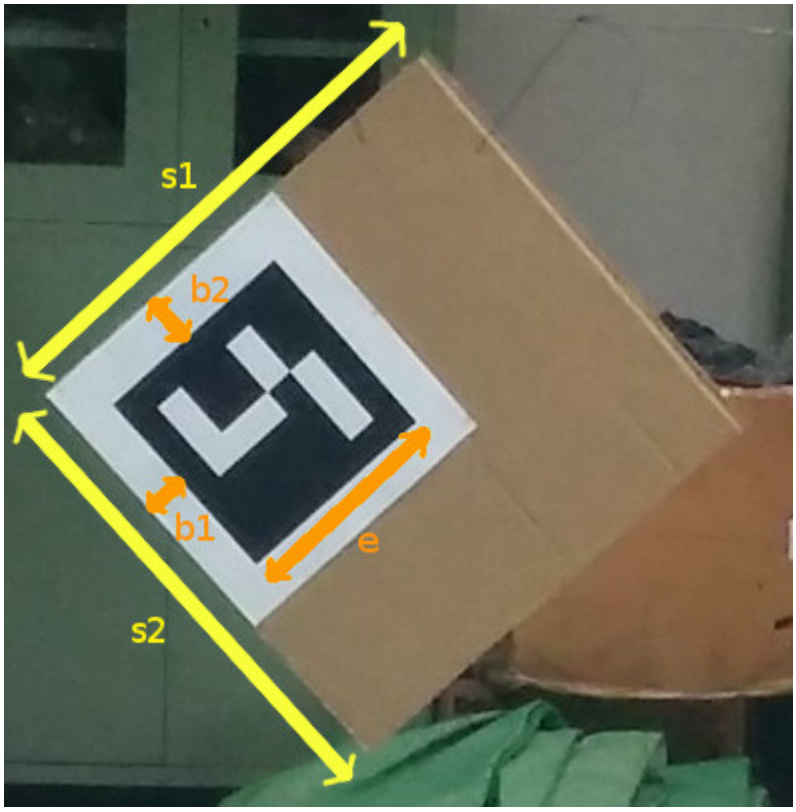
(If no changes have been made since the prevous calibration, skip to step 7)

http://github.conti.de/CorpSnT-AirRobotics/lidar_camera_calib

Youtube Video:

(To uplaod edited package to conti github once internet for NUC is up)

Step no.	Instructions
1	Install **aruco_ros** and **aruco_mapping** into the catkin workspace first before setting up the calibration package. (Both are dependencies, links available in the git repo)- do not use the original aruco_mapping, the one provided via the repo link above has been modified slightly for this intended purpose and should be the one used instead.
2	Print and stick on bigger sized cardboard ArUco Markers: 26 and 582 (Original ArUco) http://chev.me/arucogen/
3	Hang the two arUco markers in ascending order from left to right(26 left, 582 right), suspended and in the frame of the RealSense camera (try not to have any other objects in the same plane as the markers)

4	Edit config_file.txt(refer to git) and marker_coordinates.txt (below) 1.  The first line specfies 'N' the number of boards being used. Followed by N*5 lines with the following information about the dimensions of the board: *length (s1)* *breadth (s2)* *border_width_along_length (b1)* *border_width_along_breadth (b2)* *edge_length_of_ArUco_marker (e)* All dimensions in `marker_coordinates.txt` are in centimeters. (Measurements vary depending on your print and cardboard backing size)
5	Edit lidar_camera_calibration.yaml file to the following (DONE) *lidar_camera_calibration:* *camera_frame_topic: /camera/color/image_raw* *camera_info_topic: /camera/color/camera_info* *velodyne_topic: /rslidar_points*
6	Edit find_transform.launch file params as follows; • calibration_file(.ini format) • num_of_markers • marker_size(in meters)
7	Calibration Procedure*** • Ensure that all the nodes are running (Realsense and Lidar), or you could just use the launch file to launch all the nodes: datacollection.launch • After which, use the calibration package launch file: roslaunch lidar_camera_calibration find_transform.launch • Three windows will pop up, the camera view, together with two other windows named 'cloud' and 'polygon' • In the polygon window, you will be able to make out two distinct rectangular shapes (the aruco boards). Starting with the left board, mark out each edge at a time in the order: top left, top right, bottom right, bottom left (keypress after every mouse click) • Do the same for the rectangular shape on the right • You will see the rectangles you have marked out on the 'polygon' window. There should be 8 rectangles in total, 4 for each board. • The calibration data will be shown on the terminal where the calibration package launch file was launched.

Data Collection:

- Run the nodes: "roslaunch data_collection.launch"
    - Launchfile is located in home directory.
    - Feel free to edit the launchfile to include/exclude certain nodes, such as IMU and GPS (Currently excluded).
- Run the IMU node: "rosrun um7 um7_driver _port:=/dev/ttyUSB0"
- Run the GPS node: "rosrun nmea_navsat_driver nmea_serial_driver _port:=/dev/ttyUSB1 _baud:=4800"
    - Take note: The baud rate for the GPS BU353-S4 is 4800, change accordingly.
    - **Important**: The port in which the IMU/GPS is connected to might change arbitrarily, so make sure to rostopic echo /fix or rostopic echo /imu/data to ensure they are working before starting rosbag.
- Start the rosbag script (located in ~/ROSBAGS/rosbags): "./record_jackal_sensors.sh"
    - Feel free to edit the script by adding on new topics to record at the end of the line OR removing topics not required.

Remember the good practice of making <file>_backup files before editing anything though!

Projection:

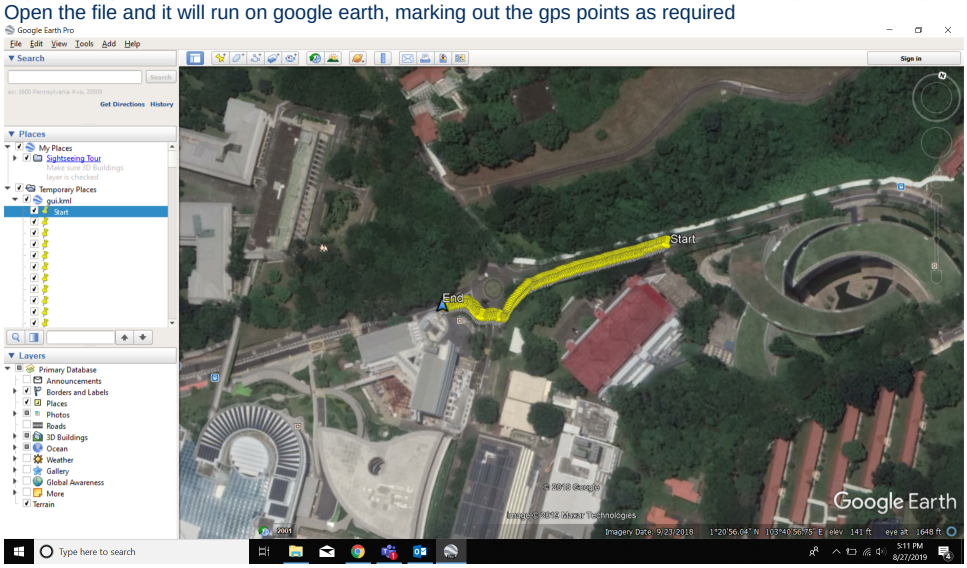
- Run the nodes: "roslaunch data_collection.launch" (launch nodes when wanting to see live projection) OR run a ROSBAG
- Launch the projection package: "roslaunch lidar_to_camera_projection project_lidar_to_camera.launch"
- Alternatively, edit the launch file "roslaunch lidar_to_camera_projection project_lidar_to_camera_ntutest.launch" for the respective ROSBAG (Rosbag can be copied into the 'test' folder in the same directory as the 'launch' folder)
- Open a new terminal and run "rosrun rqt_gui rqt_gui -s reconfigure" to open the dynamic configure tool. Adjust the values of x, y, z, R, P, Y using the slider or manually type in the values.

Projection Pictures:

MAPVIZ

- Rosbag play the bag file (rosbag play....bag)
- Open a new terminal and run "rosrun mapviz mapviz" (for Bing map using API key, zoom in to the location to check on the path taken by trolley /robot)

GPS coordinates visualisation using KML file on Google Earth

- Make a new directory and add bag2csv.py (https://gist.github.com/marc-hanheide/4c35796e6a7cd0042dca274bf9e5e9f5) & the bag file to be converted to csv into this new directory
- In the terminal, cd to this directory and run the command 'python bag2csv.py'
- A new directory will appear with csv files of all topics of the rosbag (/fix will be of concern here)
- cd into Csv-to-KML and run 'python3 csvtokmlgui.py' (https://github.com/anandhere8/Csv-to-Kml)
- A gui will appear, browse for the /fix.csv file made earlier and click generate. A file called gui.kml will be generated
- Open the file and it will run on google earth, marking out the gps points as required
- 

Things to note:
Coordinate system for Realsense point cloud differs from the RSLidar pointcloud. (Realsense: x=right, y=down, z=forward) Documentation: https://dev.intelrealsense.com/docs/projection-in-intel-realsense-sdk-20#section-point-cloud