The Linux Foundation
GSoC 2021 Proposal

TABLE OF CONTENTS

Shankho Boron Ghosh

Contact Information

Full Name	Shankho Boron Ghosh <resume>
Email	shankhoghosh123@gmail.com <primary> sghosh_be18@thapar.edu <secondary>
Country	India
College **Academic Programme**	Thapar Institute of Engineering & Technology B.E. Electronics & Communication Engineering
Current Year	Undergraduate - Junior / 3rd Year
Expected Graduation	June 2022
Postal Address	C215, Hostel A, Thapar Institute of Engineering & Technology, Patiala, Punjab, India - 147004
Phone	+91-9819674639
Github Username	growupboron
IRC Nick	boron
LinkedIn Profile	www.linkedin.com/in/shankho-ghosh/
Telegram Handle	growupboron
Preferred Communication	Email, Video Conference
Website	growupboron.github.io/

0. Introduction:

I am Shankho Boron Ghosh, a junior at Thapar Institute of Engineering & Technology, Patiala (India). I am pursuing my majors in Electronics & Communication Engineering with minors in Computer Science.

I have project-based experience in Python, C, C++, Robot Operating System (ROS), tensorflow, OpenCV and IoT using Arduino, esp8266 and Raspberry Pi. Being a quick learner and a natural improviser, and having applied this skill of problem-solving to win multiple national level hackathons including the prestigious Smart India Hackathon.

I recently interned as a Robotics Research Intern at the Indian Institute of Information Technology, Allahabad. I am also the On-Board Computer engineer, for the Student Satellite Team of our university, ThapSat and the Data Acquisition Lead for the Formula Student Team, FSAE Team Fateh at my university.

I have a good understanding and experience of software and hardware engineering related development methodologies, tools and usage.

1. Organization Information: The Linux Foundation

- **Group Interested: Automotive Grade Linux**
- **Mentors: Jan-Simon Möller, Walt Miner**

The Linux Foundation is the nonprofit consortium dedicated to fostering the growth of Linux. Automotive Grade Linux is an open source project hosted by The Linux Foundation that is building an open operating system and framework for automotive applications.

2. Project Interested: <idea page>

- **LIDAR visualization application on AGL using ROS2**
  - Integration of the meta-ros layer with Automotive Grade Linux (AGL), to support Robot Operating System (ROS2) which is an open source robotics middleware suite.
  - Develop an application using ROS2 on AGL to visualize LIDAR sensor data streams to accurately map 2D depth points with the surrounding environment.
  - This project would benefit the Automotive Grade Linux (AGL) platform by making the platform development ready for Advanced Driver Assistance systems (ADAS) and Autonomous Driving systems using the powerful Robot Operating System (ROS) framework.

3. Why this project?:

- Having first encountered Automotive Grade Linux when we (Team Fateh) had to decide on a robust platform that could handle low latency data communication and concurrently be reliable enough for automotive data acquisition (on the Raspberry Pi prototyping board), part of my duties as Data Acquisition Engineer at the Formula Student Team, FSAE Team Fateh at my university. Subsequently, the principle on which Automotive Grade Linux worked fascinated me.
- Having previously used AGL (on Raspberry Pi) for designing a Data Acquisition System, it's very clear to me, the minute specificity and the refinement needed from a user's perspective and concurrently I possess the relevant desired development experience.
- I have also been participating and actively interacting and engaging in the AGL community, regularly attending the Weekly Developer Calls, and further understanding the organization's development and production pipeline along with their developer tools and practices.
- Local development system specifications :
    - Lenovo Thinkpad L470 - Intel i5 7th Gen, 16 GB RAM, 512 GB SSD
    - Operating System -
        - Primary - Ubuntu LTS
        - Secondary - Windows 10
- I truly believe Open Source is the future and the best technique of learning is by doing. The perks, opportunities and prestige that are associated with Google Summer of Code (GSoC) are just added benefits to this.
- This project would give the necessary experience and rigor to contribute meaningfully and efficiently as Data Acquisition Engineer to FSAE Team Fateh.

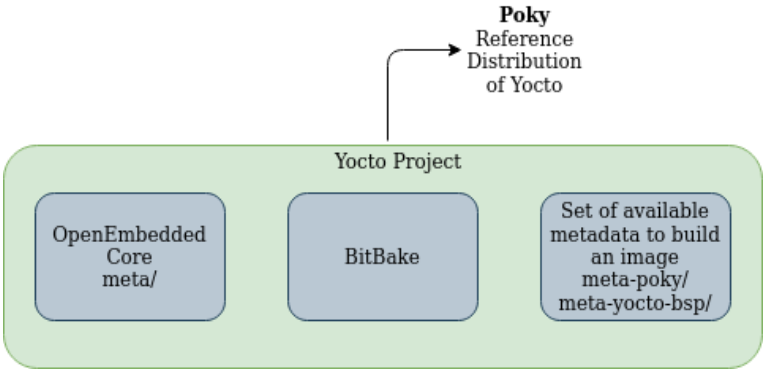
4. Deliverables:

- Integrate yocto based meta-ros layer with Automotive Grade Linux (AGL) layers to support Robot Operating System (ROS 2) infrastructure.
- Create custom recipes for ROS modules to support LIDAR drivers and other unmet dependencies that might arise.
- Develop a visualization application for AGL by implementing hector mapping using LIDAR using ROS2.
- Ultimately documenting the whole implementation methodology, so that the developer community can easily carry on with and independently develop relevant ROS applications on AGL.

5. Project Description:

Objective: To integrate, deploy and test meta-ros with AGL layers using the Yocto based Poky and OpenEmbedded Build system to support Robot Operating System (ROS 2) infrastructure.

1. Automotive Grade Linux is an open operating system and framework for automotive applications which is developed using the Yocto Project. Benefits of using Yocto are :
    a. Develop using common linux OS for all major architecture and boards.
    b. Flexible framework allowing reuse of software.
    c. Changing hardware platforms is easy due to the presence of BSPs.
    d. Access to wide collections of layers and recipes.
2. Yocto maintains the OpenEmbedded core and poky is the reference distribution of the Yocto project.
    a. **Poky** = OpenEmbedded Build system + metadata
        i. **Metadata**: Task definitions / Set of instructions. This contains the files that OpenEmbedded build system parses when building an image. It includes recipes, configuration files and instructions on how to build an image.
    b. **Open Embedded Build system** = BitBake + OE-Core (meta/ directory)
        i. **BitBake**: A task executor and scheduler. It is a build engine that works through recipes written in a specific format in order to perform sets of tasks.
        ii. **OpenEmbedded-Core**: OpenEmbedded-Core (OE-Core) is a common layer of metadata (i.e. recipes, classes, and associated files) used by OpenEmbedded-derived systems, which includes the Yocto Project. It consists of foundation recipes, classes and associated files that are meant to be common among many different OpenEmbedded systems.

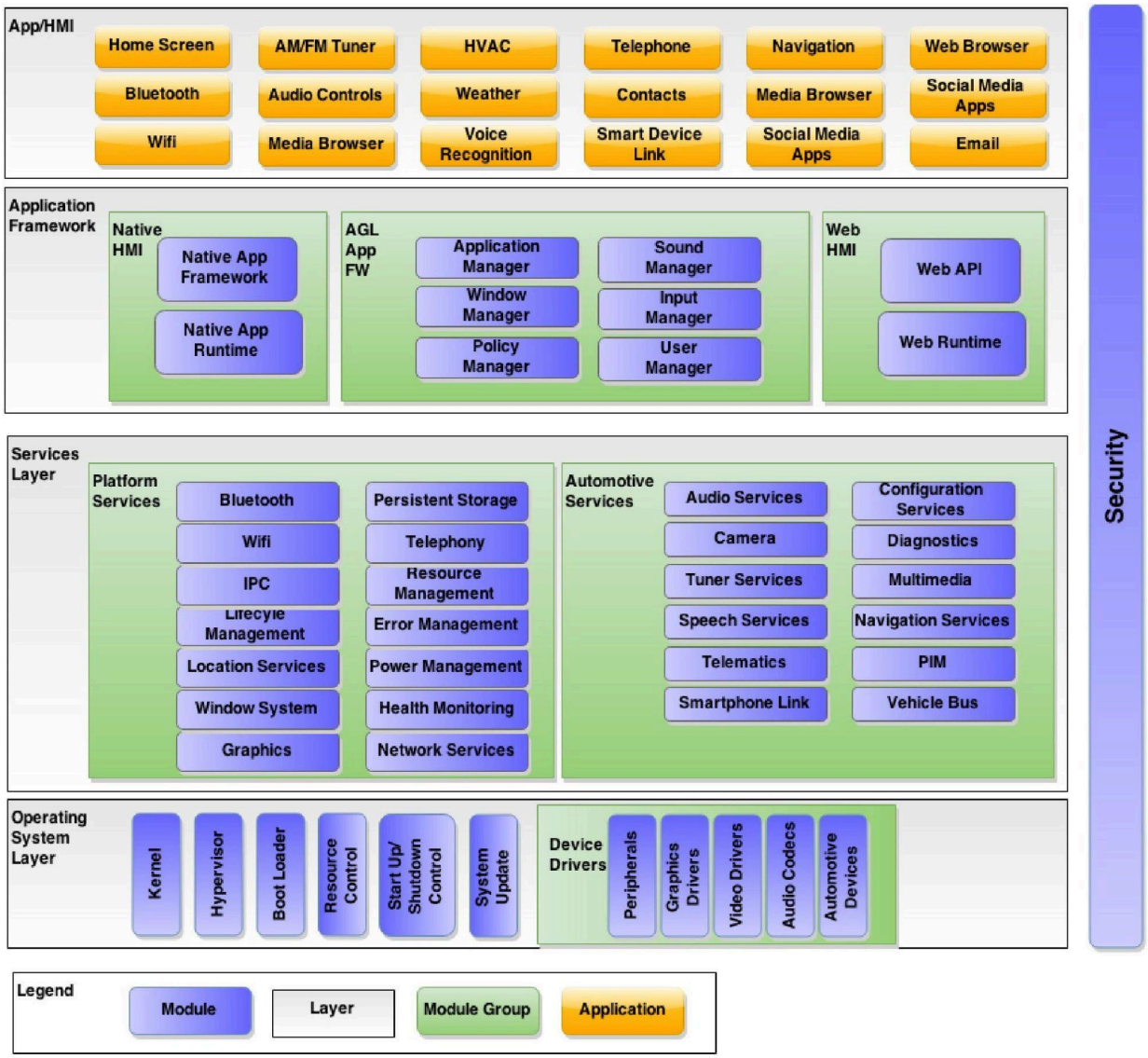


3. Particularly in Yocto infrastructure, packages are grouped into layers and recipes are part of layers which are nothing but individual pieces of software. Layers are different kinds of repositories or folders and multiple recipes can be present within each layer. Different components of the Yocto project :
    a. **Configuration files (.conf)** : Global definition variables [types of machine architecture, global variables, build path, compiler flags]
    b. **Classes (.bbclass)** : Encapsulation and inheritance of build logic [defines how we build linux kernel, how to generate RPM package, how to create root file system image]
    c. **Recipes (.bb)** : Logical units of SW / Images to build [individual piece of SW to be built, what packages get included in final file system image. Recipes includes meta data for the SW such as where we download upstream sources from, build or runtime dependencies, what feature we enable in our application, configuration, compilation options, define what files goes in to what output package]
    d. **Layers (bblayers):** Repositories that contain related metadata, set of recipes.
4. Due existence of multiple yocto layers in AGL like :

- poky
- meta-agl
- meta-agl-cluster-demo
- meta-agl-demo
- meta-agl-devel
- meta-agl-extra
- meta-agl-telematics-demo
- meta-openembedded
- etc … … …
- meta-security
- meta-virtualization
- meta-qt5
- meta-updater
- meta-spdxscanner
- meta-clang
- BSP layers :
    - meta-raspberrypi4
    - meta-intel
    - meta-ti
    - meta-renesas-rcar-gen3
    - meta-sancloud, etc.

It is possible to implement AGL as per the requirement specifications of the architecture. This leads to the formation of the AGL Unified Code Base (UCB) which is a Linux distribution built from the ground up through a joint effort by automakers and suppliers to deliver a modern in-vehicle infotainment and connected car experience for consumers. The goal of the UCB platform is to provide 70-80% of the starting point for a production project. This enables automakers and suppliers to focus their resources on customizing the other 20-30% to meet their unique product needs. The Automotive Grade Linux

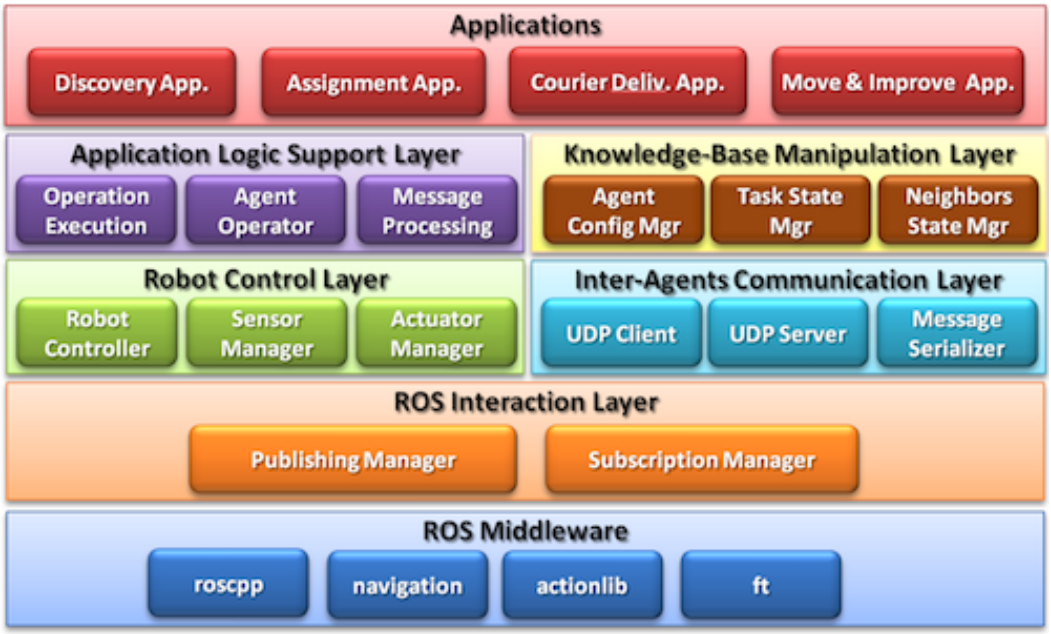
Software Architecture diagram is as mentioned below. The App/HMI layer contains applications with vendor associated business logic and HMI.



5. Robot Operating System is an open source robotics middleware suite. Although ROS is not an operating system but a collection of software frameworks for robot software development, it provides services designed for a heterogeneous computer cluster such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management.
6. Despite the importance of reactivity and low latency in robot control, ROS itself is not a real-time OS (RTOS). It is possible, however, to integrate ROS with real-time code. The lack of support for real-time systems has been addressed in the creation of ROS 2.0, a major revision of the ROS API which will take

advantage of modern libraries and technologies for core ROS functionality and add support for real-time code and embedded hardware with the introduction of meta-ros layer.

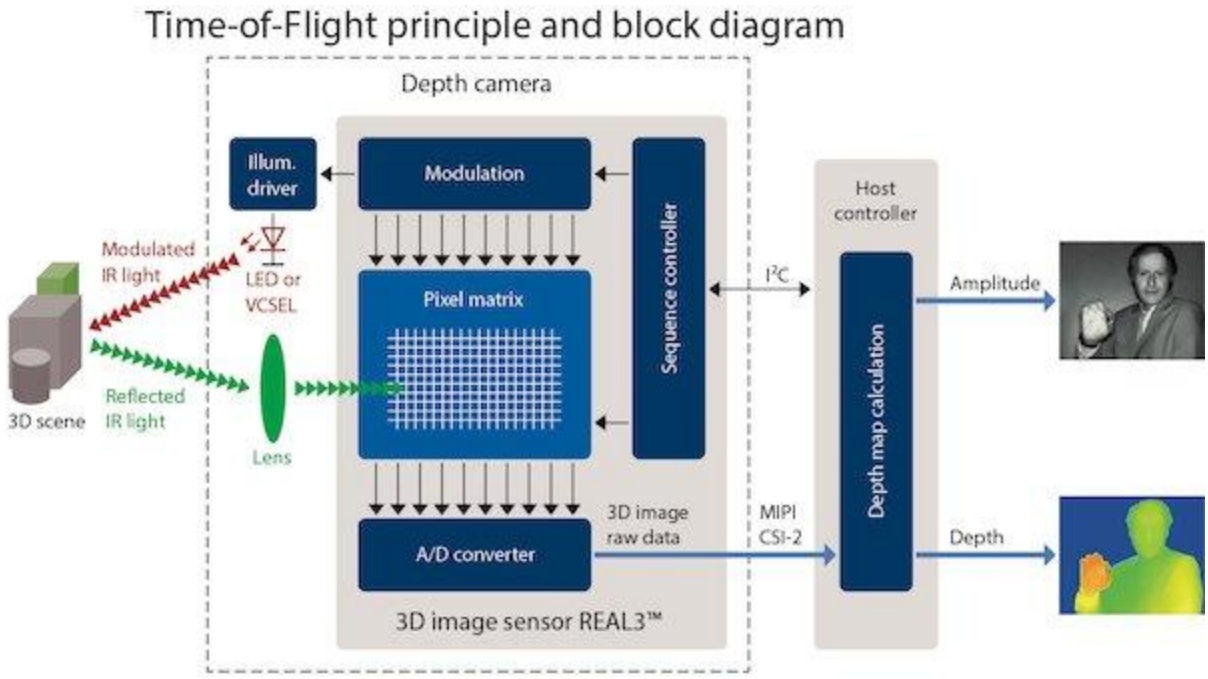
7. ROS allows you to stop reinventing the wheel and focus on code for new implementation due to the presence of countless support packages and active developer community. Software in the ROS ecosystem can be separated into three groups:
    a. language-and platform-independent tools used for building and distributing ROS-based software.
    b. ROS client library implementations such as roscpp rospy, and roslisp.
    c. packages containing application-related code which uses one or more ROS client libraries.



8. Although the meta-ros layer takes care of most language and platform-independent tools used for building and distributing ROS-based software and ROS client library implementations such as roscpp rospy, and roslisp. There might be some missing packages containing application-related code which uses one or more ROS client libraries. These can be resolved by creating custom recipes to port required unmet dependencies (like YDLIDAR driver) and make it AGL layer compatible.
9. Finally, the integrated system would be tested on Raspberry Pi 4 hardware board to finalize the AGL + ROS 2 system and debug corresponding issues that might arise.

Objective: Develop a visualization application for AGL by implementing hector mapping using LIDAR using ROS2.

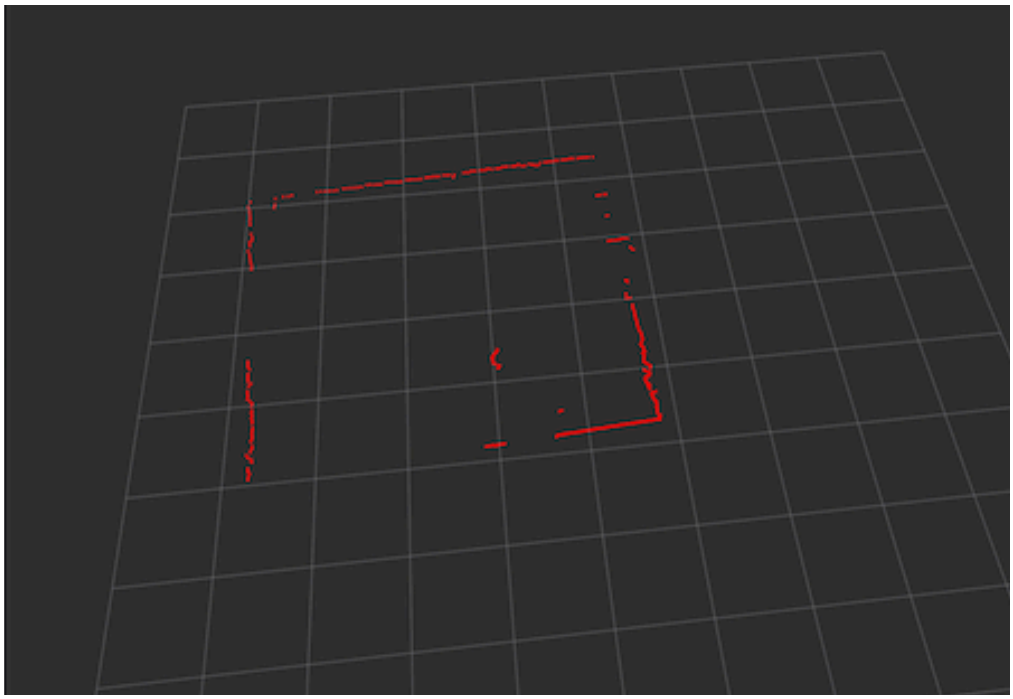
1. Light detection and ranging (LiDAR) is a remote sensing technology. It usually has a source of laser (a source of optically amplified light) pulse and a receiver that accepts the reflection. It uses different computational techniques to find the distance between the source and the target. The output is usually a point cloud carrying range/distance information and orientation of the corresponding line-of-sight.
2. The basic principles for traditional LiDAR have been seen in time-of-flight-based distance measurement and triangulation-based measurement. The ToF concept uses the delay time between the signal emission and its reflection to compute the distance to the target. The time delay is not measured for one particular beam's round trip. Instead, the phase shifts for multiple signals are used to indirectly obtain the ToF and then compute the distance. The triangulation method is used to measure the distance between the source and the obstruction.



3. YDLIDAR X4 is a low cost 2D LIDAR solution developed by Shenzhen EAI Technology Co. The X4 is a triangulation-principle-based LiDAR with a rotating platform that carries a laser emitter and a receiver lens. It is a 360° 2D rangefinder with up to 5kHz frequency and 6Hz-12Hz motor speed, and has a USB interface. It also has a fine angular resolution of 0.48°-0.52° and a range of

0.12-10m.The output of YDLIDAR is very suitable to build maps, do SLAM, or build 3D models.

4. Hector mapping is a SLAM approach that can be used without odometry as well as on platforms that exhibit roll/pitch motion (of the sensor, the platform or both). This method leverages the high update rate of modern LIDAR systems.
5. The implementation would be initially performed and tested on a desktop based conventional ROS 2 system and later be shifted to the Raspberry Pi 4 based AGL + ROS 2 system. The corresponding LIDAR depth map would look like :



6. The possible ways for developing the AGL application to process and visualize real time LIDAR sensor data can be Qt or HTML 5 based because AGL supports them well. The probable methods for this are :
    a. Port existing Qt based rviz application to AGL by creating custom recipes. Since rviz application is well supported in ROS platform and is still experimental with respect to ROS2 platform, substantial efforts would be required to port rviz to ROS2 and later to AGL platform.
    b. If the above approach fails, develop Qt application from scratch, something very similar to github.com/malichao/XV11-LIDAR-Visualizer.

    The approach would be finalized after deliberation with mentor (Jan-Simon Möller) at a later stage of GSoC.

6. Timeline:

Time Period	Milestones
Pre GSoC **14th Apr - 17th May**	❏ Engage within the AGL community. ❏ Set up a personal blog at https://growupboron.github.io/.

Community Bonding Period
18th May - 6th Jun
	❏ Research and gain experience with relevant tech stacks :
	  ❏ OpenEmbedded / Yocto Project
	  ❏ Robot Operating System (ROS2)
	  ❏ Application development on AGL.

Week 1 & 2 **7th Jun - 20th Jun**	❏ Setup development repository. ❏ Integrate, deploy and test meta-ros with AGL layers.
Week 3 & 4 **21st Jun - 3rd Jul**	❏ Setup YDLIDAR X4 and experiment with hector mapping algorithm and test on a desktop based conventional ROS 2 system.
Week 5 & 6 **4th Jul - 17th Jul**	❏ Create custom recipes to port required unmet dependencies and YDLIDAR driver to make it AGL layer compatible and test on Raspberry Pi 4 based AGL + ROS 2 system.
Week 7 & 8 **18th Jul - 1st Aug**	❏ Develop AGL application to process and visualize real time LIDAR sensor data on Raspberry Pi 4 based AGL + ROS 2 system.
Week 9 & 10 **2nd Aug - 16th Aug**	❏ Buffer for debugging code / unintentional delays ❏ Compile resource list that future developers can follow. ❏ Write down and submit the final project report.

7. Commitment & Availability:

- I have always been keen to learn more, especially about Open Source Software Development. An opportunity like Google Summer of Code with The Linux Foundation seems like the perfect path for it. The mentors have been very kind, patient and helpful to all my queries. I would be honored to continue working with them.
- I am sure that I would be able to devote 20+ hours per week to this cause. My work timings are very flexible, but I usually start working after 18:00 all the way till 23:00 in UTC+5:30.
- I expect to give my full participation during the GSoC period and have no prior engagements during this period, except that I'm a junior, so there might be time when I would be unavailable due to academic commitments.
- I would suggest starting work earlier than the original GSoC timeline in order to avoid any unforeseen notices by my university regarding academic examinations.

8. Support Required:

- Raspberry Pi 4 as an implementation and testing platform.
- LIDAR (YDLIDAR X4) sensor for developing ROS2 visualization application.

9. Post-GSoC Period:

- I will add to Automotive Grade Linux's ROS2 integration documentation and demo application repository for the foreseeable future and work on continually improving it.
- I will keep contributing and assist other new contributors to explore and learn projects with this organisation.

10. Open Source Contributions & Experiences:

- Contributions to Open Source :
  - **Automotive Grade Linux** : Reworked documentation hosting & generation and restructured getting started pages and developer guides as a part of Google Season Of Docs (GSoD'20) program.
  - **Data Version Control** : Renamed a function and correspondingly updated the documentation.
  - **The Turing Way** : Wrote and improved some chapters and existing documentation.

    - **Veritas : Engineering Logbook Project** : Maintaining and leading, university's official logbook portal.
- I have actively developed as well as contributed to several projects; for more details please refer to my resume and GitHub.
- Being an avid hackathon hunter, I have participated in over a dozen hackathons collecting many laurels, helping me to become a quick learner and a natural improviser.
- I can truly say that my experiences taught me the value of patience, perseverance, and to constantly strive for honor and excellence.
- I believe in learning by doing and have been exposed to a lot of things and I will always treasure the insights that I've earned both in and outside the portals of the university.