GSoC 2025 Proposal

The Linux Foundation

Project: Automotive Grade Linux (AGL)

Idea: meta-ros integration (Robotic framework)

Project Size: 350 hours

Candidate Name: Mohammad Saalim

Candidate Email: saalimquadri2@gmail.com

Github ID: http://github.com/danascape

Mentors: Jan-Simon Möller, Walt Miner

Table of Contents

Introduction

- I am Mohammad Saalim, a final-year student at Dayananda Sagar College of Engineering, India, pursuing a Bachelor's degree in Electronics and Communication Engineering. My curiosity about low-level systems, from the initial Boot ROM code to running user-space applications, has driven my passion for embedded systems and Linux kernel development.
- Over the years, I have gained extensive experience working with **embedded Linux**, particularly in **bootloaders**, linux **kernel** development, and working on board **BSP** bring-up.
- I have previously participated in The **Linux Foundation Mentorship Program**, where I got an opportunity to work under Shuah Khan to deepen my understanding of kernel development workflows. Linux Kernel introduced me to the diverse world of working on Hardwares and I got to study a lot about how low-level systems work.

Programming Languages	**C/C++, Bash, Python**
Hardware	**Raspberry Pi 3B/4B Plus/5, TI AM625 EVM**
Tools	**Git, GDB, QEMU, GNU Coreutils, Make, Vim**
Experience With	**SPI, I2C, Yocto, AGL, IIOs**

Organization Information: The Linux Foundation

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

The Linux Foundation is the non-profit 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. They are designed to help developers and professionals to develop their skills and advance in their careers.

Project Interested

- meta-ros Integration
  - Goal is an AGL demo image built with the meta-ros layer, used to represent a simulated automobile.
  - The automobile will be able to use ROS to collect sensor data.
  - Write an AGL/Flutter Application built with meta-ros (agl-devel-ros) feature to visualise and simulate the collected sensor data.

Why this Project?

- The integration of the Robot Operating System (ROS) Framework as an option within Automotive Grade Linux (AGL) through the meta-ros yocto layer is an important step in upgrading vehicle technology using automotive systems. This project is important for several reasons:
  - Bridging the Gap Between Robotics and Automotive Software: AGL is widely used in the automotive industry for IVI (In-Vehicle Infotainment) and telematics adding native support for ROS, is essential for robotic applications in ADAS (Advanced Driver Assistance Systems), sensor fusion, and autonomous driving. By integrating meta-ros with AGL, this project enables deployment of robotics-based functionalities on embedded platforms.
  - Expanding AGL's Capabilities: Modern vehicles require AI-driven automation and sensor-based intelligence. ROS integration will allow AGL to support robotics applications, making it a versatile platform for future self-driving and AI-assisted vehicles.
  - Advancing Open-Source Development: Since, AGL is the most wide open-source platform, this project aligns with the Linux Foundation's vision of expanding open-source contributions in automotive systems.
  - Enhancing Real-World Use Cases: ROS has been successfully integrated in several automotive applications, like the Autoware Foundation has an open-source software stack that powers autonomous vehicles, Apex.AI focuses on creating safety-critical automotive-grade systems, offering Apex.OS, which is a version of

ROS2. Similarly F1Tenth, uses ROS to enable real-time path planning and obstacle avoidance in high performance scenarios. These examples show that ROS is increasingly being adopted in the automotive sector.
    - Providing a Scalable and Reproducible Solution: The meta-ros layer will be modular, making it easy to extend and maintain for future AGL versions and different automotive hardware. The demo image created from this project can serve as a reference implementation for future developments in robotics within AGL.
- 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 and developer's tools and practices.
- Local development system specifications :
    - PC Setup - Ryzen 7 7700X , 32 GB RAM, 1 TB SSD
    - Operating System -
        - Primary - Ubuntu LTS 22.04
        - Secondary - Windows 11
- 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.

Deliverables

- The primary aim of the project is to ensure a successful integration of the meta-ros yocto layer with Automotive Grade Linux (AGL), resulting in a functional demo image that supports ROS applications.
    - A fully functional AGL feature demo image that includes the meta-ros layer, enabling the execution of ROS-based applications within AGL.
        - AGL devel feature flag patchsets to enable meta-ros layer recipes and packagegroups.
        - Ensured compatibility with AGL's stack and dependencies.

        - Tested and validated ROS package compilation and execution within AGL.
    - Bitbake Recipes and Layer Customization
        - Write Yocto recipes to package ROS components for AGL.
        - Modify and override meta-ros recipes to meet AGL's requirements.
        - Ensure correct dependency resolution for ROS and AGL components.
    - Demo Application: AGL + ROS Sensor Integration
        - Develop a demo AGL + ROS application (e.g., sensor data processing), with real-time data collection and visualisation in the application.
        - Showcase real-time ROS node communication within AGL.
    - Documentation and Contribution to AGL Community
        - Step-by-step installation and build guide for setting up ROS in AGL.
        - Troubleshooting guide for dependency issues and build errors.
        - Contribution of patches and modifications to the AGL/meta-ros repositories.
- Write a Getting Started page and Developer guides documentation along with writing new documentation into the following clear and bullet-proof structure :
    - Tutorials: a lesson that allows the newcomer to get started with ROS.
        - Example : teaching a small child how to cook
    - How-to Guides: a series of steps that show how to solve a specific problem.
        - Example : a recipe in a cookery book
    - Reference: a technical description describing the appropriate workflow processes.
        - Example : a reference encyclopedia article
    - Explanation: discursive explanation explaining code / workflow structure in detail
        - Example : an article on culinary social history
- Ultimately, this project will bridge the gap between Automotive Grade Linux (AGL) and robotic frameworks by successfully integrating meta-ros, enabling seamless deployment of ROS applications on

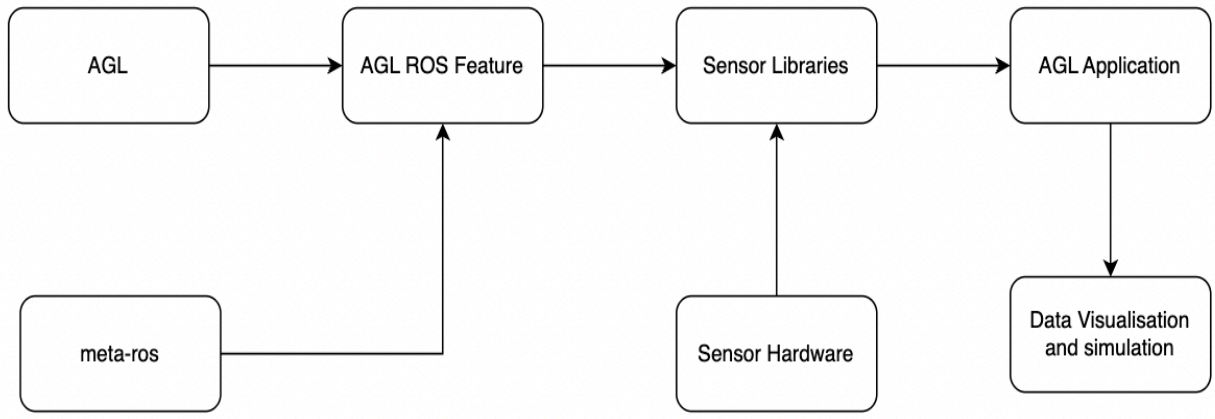
automotive platforms and laying the foundation for future advancements in autonomous driving, ADAS, and AI-driven vehicle systems.

Project Description

- Abstract:
  - The integration of robotic frameworks with Automotive Grade Linux (AGL) marks a significant upgrade in enabling autonomous and vehicle solutions. This project focuses on integrating the meta-ros layer into AGL to create a demo image that integrates the Robot Operating System (ROS) libraries with AGL.
  - The meta-ros layer provides the necessary components, libraries and dependencies to compile, deploy, and run ROS-based applications on embedded automotive platforms. This project will involve a detailed analysis of the build system of AGL, modification and integration of the meta-ros layer, resolution of dependency and compatibility issues, and extensive testing to ensure proper functionality.
  - By the end of the project, the primary deliverable will be a fully functional AGL image capable of running ROS-based applications, including Demo Applications to test ROS Components.
- Analysis:
  - The rapid advancement in autonomous sectors and AI-driven applications, has highlighted the need for an ecosystem that can integrate real-time data processing, sensor fusion, and decision-making. Automotive Grade Linux (AGL) has emerged as a leading open-source platform for in-vehicle infotainment (IVI), telematics, and automotive systems, it currently lacks native support for robotic middleware such as Robot Operating System (ROS). This project aims to bridge this gap by integrating meta-ros yocto layer into AGL, unlocking new capabilities for robotic applications, ADAS (Advanced Driver Assistance Systems), autonomous navigation, and AI-driven perception systems within vehicles. Whilst people can write complex applications to perform the same algorithms, having a dedicated layer will help trim down various pre-requirements.

- The meta-ros layer is a Yocto layer that enables cross-compilation and deployment of ROS packages on embedded Linux platforms.
- Importance:
  - ROS has a modular architecture, middle-ware capabilities and real-time sensor data processing.
  - Integrate support for cameras, LiDAR, and radar for environmental awareness.
  - A wide range of libraries and components for making ROS integration.
- Challenges:
  - Dependency Management – Handling different components and system libraries between ROS and AGL.
  - Cross-Compilation – Ensuring that ROS packages are correctly compiled and optimized for automotive hardware.
  - Performance & Stability – Testing real-time execution of ROS applications within the AGL ecosystem on embedded platforms.

- This proposed workflow will help AGL seamlessly integrate ROS functionalities, enabling advanced robotics and automation capabilities within AGL applications while maintaining compatibility with the existing build system.



- Block Diagram:
  - Inside AGL, ROS layer can be optionally enabled via a feature during the initialisation of the environment, to enable the packagegroups listed in meta-ros layer and compile them alongside AGL image.

Timeline

- Community Bonding Period - 8th May - 1st June :
  - Engage more with mentors, attent weekly calls.
  - Setup development environment, define an AGL and meta-ros layer release to use as a base (possibly super salmon release for AGL and scarthgap for meta-ros).
  - Explore ROS documentation, and identify the possible hardware for testing (petalinux ROS hardware and match with AGL supported farms).
  - Decide on an emulation environment to ease testing of builds (possibly qemu-aarch64).
  - Identify hardware and a sensor unit for integration and testing.
  - Draft a high level design document outlining the integration strategy.
  - Design a workflow for the AGL application, involving the GUI of the app along with sensor integration (a hardware or a mock sensor).
  - Identify the AGL and ROS integration strategies to involve, like packagegroups, image recipes for minimal test setup.
  - A validation of development setup before coding starts.
- Week-1 : 2nd June - 8th June :
  - A buffer period for the hardware units to arrive (Unit as well as sensors).
  - Write scripts for ease of building the distro.
  - Resolve build failures and missing dependencies.
  - Test out the initial build on emulator, and check any dependency issues.
- Week-2 : 9th June - 15th June :
  - Ensure the arrived hardware is validated and working properly.
  - Setup and compile AGL minimal image for the hardware and validate the images.
  - A buffer period for the validation of the hardware units.
  - Identify and resolve any library conflicts with AGL.
  - Modify and override recipes if necessary.
  - Setup the initial repository for AGL ROS feature under agl-devel.

    - Ensure ROS2 packagegroups are included along with libraries, when AGL ROS feature is enabled.
    - Identify and resolve any library conflicts with AGL.
- Week-3 : 16th June - 22nd June :
    - Modify and override recipes if necessary.
    - Test AGL + meta-ros packagegroups on the decided emulator environment along with the identified hardware.
    - Verify if ROS nodes are able to communicate in AGL distribution.
    - Initial submission of meta-ros integration patches on gerrit.
- Week-4 : 23rd June - 29th June :
    - Take and understand the reviews from the AGL community about the initial patchsets about ROS AGL devel feature.
    - A buffer period for validating and improving the patchsets on gerrit.
- Week-5 : 30th June - 6th July :
    - Increase the variety of AGL development image recipes enabled with AGL ROS feature to include more ROS packagegroup and validate the same.
    - Verify if all meta-ros recipes are resolved and compilable.
- Week-6 : 7th July - 13th July :
    - Conduct thorough functionality testing of ROS libraries (including ros1/ros2) inside AGL.
    - Fix any issues blocking ROS nodes from running inside AGL.
    - Update patchsets on gerrit for inclusion of more ROS recipes.
    - A buffer period for improvement of patchsets and testing on hardware environment.
- Week-7 : 14th July - 20th July :
    - Submit Phase 1 Deliverables, involving final patchsets for AGL ROS feature, a working build with the devel feature and a test report.
    - A buffer period for discussing the feature-set with the community.
- Week-8 : 21st July - 27th July :
    - Write a base AGL Application and include recipes with ROS Feature.
    - Update the Application to collect and visualise ROS data.
    - Verify if ROS nodes are able to communicate towards the frontend of the AGL Application.
    - Ensure a successful communication between ROS nodes inside AGL.

- Week-9 : 28th July - 3rd August :
  - A buffer period to ensure successful communication of the distro with the ROS nodes within the application.
- Week-10 : 4th August - 10th August :
  - Implement real-time data collection and visualisation of the frontend of the application.
  - Test sensor data node calls within application in AGL.
- Week-11 : 11th August - 17th August :
  - Optimize demo ROS + AGL application for efficiency.
  - Conduct extensive performance testing of the Demo AGL Application.
  - Discuss techniques for multiple sensor integration in meta-ros.
- Week-12 : 18th August - 24th August :
  - Update the resource list that future documenters and developers can follow to write and improve documentation.
  - Write down and submit the final project report.
- Week-13 : 25th August - 1st September :
  - Submit the personal evaluation on success of the project and experience working with AGL mentors and community.

Commitment & Availability

- I have always been keen to learn more, especially about Embedded Linux and 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 40+ hours per week to this cause. My work timings are very flexible, but I usually start working after 14:00 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 in my final year, so there might be time when I would be unavailable due to academic commitments.

Post GSoC Period

- I will maintain the meta-agl-ros integration framework and also add to Automotive Grade Linux's documentation repositories for the foreseeable future and work on continually improving it.
- I will keep contributing and help other new contributors to explore and learn projects (especially the documentation) in this organisation.

Why am I the right Person?

- With experience in embedded Yocto, AGL, BSP development, and Linux kernel, I am confident in my ability to handle the meta-ros integration with Automotive Grade Linux (AGL). My background in Yocto, AGL, cross-compiling, and hardware integration directly aligns with the project's requirements.
- I aim to deepen my expertise in Embedded Linux Systems to work with and adapt to a range of hardware devices. Working on robotic framework will allow me to gain a deeper understanding of hardware and sensor interactions, which are crucial for real-time and automotive applications.

Open Source Contributions & Experiences

- Contributions to Open-Source:
  - Automotive Grade Linux (AGL): Added support for TI AM62X series BSP in AGL, and working on drm-lease to improve display rendering.
  - Linux Kernel Mainline: Contributed 15+ patches to the Linux kernel in dt-bindings and Industrial I/O (IIO) subsystems, involving device tree fixes and driver upstreams.
  - StormBreaker Project: Founded StormBreaker, an open-source Linux kernel optimization project with 500,000+ downloads. Worked on upstreaming OEM-supplied kernels with stable Linux updates and integrating features from higher upstreams, such as: vDSO32, s-random, schedulers and TCP.
  - PostMarketOS (pmOS): Mainlined MSM8937 and SM6350 chipsets in PostMarketOS, enabling support for a broader range of smartphones.

Developed basic framebuffer and USB debugging support for these platforms. Worked on Global Clock Controller (gcc-clk) drivers and MDSS-DSI panel drivers to enable display support.

- Experiences:
  - **Raptee Energy - Embedded Firmware Developer:** Led the development of a custom embedded Linux distribution using Automotive Grade Linux (AGL). Integrated RAUC (Robust Auto-Update Controller) with Zoho Cloud Suite, enabling secure OTA updates for embedded devices. Developed Yocto recipes for Embedded Flutter to support graphical interfaces in automotive applications.
  - **Vispero - BSP Engineer:** Worked on U-Boot for Raspberry Pi 4B/5, integrating AB partitioning for Android-based embedded devices. Developed a custom U-Boot environment, improving boot reliability and redundancy in embedded Android systems.
  - **The Rainmakers - Linux Kernel Developer:** Upstreamed the Linux Memory Extractor (LiME) driver for newer Android kernels, enabling volatile memory acquisition for forensic and debugging purposes. Developed a custom Linux kernel driver to extract the physical offset of a file within a disk, improving low-level file system analysis.
  - **Nimo Planet - Linux Kernel Developer:** Developed and upstreamed a Linux kernel driver for the Azotek IQS5xx touchpad. Participated in the development of a 6-Axis Inertial Motion Unit (IMU) sensor driver, supporting accelerometer, gyroscope, pedometer, and tilt detection.