**TRIBHUVAN UNIVERSITY**
**INSTITUTE OF ENGINEERING**
**PULCHOWK CAMPUS**

A
PROJECT REPORT
ON

# AUTOMOBILE AND ADVANCED CONTROL SYSTEMS THAT USE MULTIPLE SENSORS, CLOSED FEEDBACK LOOPS AND MACHINE LEARNING FOR ADAPTIVE BEHAVIOR(FIRST HALF OF AUTONOMOUS VEHICLE PROJECT)

**SUBMITTED BY:**
BIBEK PARIYAR (PUL076BEI012)
KRISHBIN PAUDEL (PUL076BEI018)
ASMIN SILWAL (PUL076BEI040)

**SUBMITTED TO:**
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
PULCHOWK CAMPUS, INSTITUTE OF ENGINEERING
TRIBHUVAN UNIVERSITY
LALITPUR, NEPAL

**April 9, 2025**

# Page of Approval

TRIBHUVAN UNIVERSIY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled **"AUTOMOBILE AND ADVANCED CONTROL SYSTEMS THAT USE MULTIPLE SENSORS, CLOSED FEEDBACK LOOPS AND MACHINE LEARNING FOR ADAPTIVE BEHAVIOR"** submitted by **Bibek Pariyar**, **Krishbin Paudel**, **Asmin Silwal** in partial fulfillment of the requirements for the Bachelor's degree in Electronics & Computer Engineering.

|  |  |
|---|---|
| ............................. | ............................. |
| Supervisor | External examiner |
| **Nischal Acharya** | **Dakshina Shrestha** |
| Assistant Professor | Principal / Associate Professor |
| Department of Electronics and Computer Engineering, | Sagarmatha Engineering College, IOE, TU. / Sagarmatha College of Science and Technology |
| Pulchowk Campus, IOE, TU. |  |

Date of approval: **April 18, 2024**

# Copyright

# Acknowledgements

We would like to express our heartfelt gratitude to all those who have encouraged and inspired us in the work on this project.

We would like to express our sincere gratitude to the **Robotics Club** of IOE-Pulchowk Campus for providing us with the necessary resources and space to build the mechanical and electronics system for the Vehicle. The club's commitment to fostering a culture of innovation and experimentation has been a significant factor to encourage us to continue with this project. We appreciate their continued support and encouragement.

We would also like to thank the **Department of Electronics & Computer Engineering** of IOE-Pulchowk Campus for approving proposal of this project to be our final year Major Project. Similarly, We would like to express our sincere appreciation to the **NIU Corporate Team Nepal**, for encouragement and providing us with the equipment necessary for our project, **Tinymovr** and **ODrive** for sending us motor drivers, **TinkerBugRobotics** for sponsoring us with RTK GPS system, **SEGGER** for providing us necessary embedded debugging tools.
We would also like to thank our other classmates who provided valuable feedback for our project concepts. Their willingness to share ideas and perspectives helped us to develop a more nuanced understanding of the subject matter.
Lastly, we are thankful to our seniors as well who guided and give feedback to our concept paper. Their experience and knowledge were crucial to us for designing this project.

# Abstract

The recent trends and advancements in the field of artificial intelligence and autonomous systems has motivated us to work on a similar project of our own. The project required much considerations and effort in both hardware and software aspect due to which this project is separated into first half and second half. The first half deals with the physical system of the vehicle which involves all the hardware and the control system for those hardware. The second half primarily focuses on the implementation of autonomy, i.e. computer vision and model training.

This half of the project involves building a system that integrates all the sensors and actuators along with processing units to provide a platform for the model to control the vehicle. The physical system uses a stereo camera to observe the surrounding and also uses other sensors which communicates to the core system and each other using the CAN bus protocol. The vehicle is a large model capable of driving a single person which also utilizes a hardwired switch to take over the controls to manual mode in case of emergency.

# Contents

**Page of Approval** **ii**

**Page of Approval** **ii**

**Copyright** **iii**

**Copyright** **iii**

**Acknowledgements** **iv**

**Abstract** **v**

**Contents** **viii**

**List of Figures** **ix**

**List of Abbreviations** **xi**

**1 INTRODUCTION** **1**

1.1 Background . . . . . 1

1.2 Problem Statements . . . . . 1

1.3 Objectives . . . . . 2

1.3.1 Collaboration Objectives . . . . . 2

1.3.2 Objectives of This Project . . . . . 2

1.4 Scope . . . . . 3

1.4.1 Collaboration Scope . . . . . 3

1.4.2 Scope of This Project . . . . . 3

**2 Literature Review** **4**

2.1 Related work . . . . . 6

2.2 Related theory . . . . . 6

2.2.1 Stereo vision . . . . . 6

2.2.2 CAN frames . . . . . 7

2.2.3 SLAM . . . . . 9

2.2.4 Isolated Redundant Network . . . . . 9

2.2.5 Real Time Kinematics (RTK) Global Positioning System (GPS) . . . 10

**3 Methodology** **13**

3.1 Project Management . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

3.1.1 Quality Assurance . . . . . . . . . . . . . . . . . . . . . . . . . 13

3.2 Scale of the Project . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

3.3 Autonomous Region and Environment . . . . . . . . . . . . . . . . . . 15

3.4 Absolute and Relative Positioning of the Device . . . . . . . . . . . . . 15

3.5 Team and Work Division . . . . . . . . . . . . . . . . . . . . . . . . . 15

3.5.1 Collaboration Team . . . . . . . . . . . . . . . . . . . . . . . . 16

3.6 Equipment, Tools and Devices . . . . . . . . . . . . . . . . . . . . . . 17

**4 Experimental Setup** **21**

4.1 Node Level Isolated System . . . . . . . . . . . . . . . . . . . . . . . 21

4.2 Communication Protocol Design . . . . . . . . . . . . . . . . . . . . . 21

4.3 PID Tuning and System Calibration . . . . . . . . . . . . . . . . . . . 22

4.4 Exposed System Parameter For Varied System Response . . . . . . . . . . 24

4.5 RC Based Initial Design . . . . . . . . . . . . . . . . . . . . . . . . . 24

4.6 Mechanical Based System Design . . . . . . . . . . . . . . . . . . . . 25

**5 System Design** **26**

5.1 How does the system work? . . . . . . . . . . . . . . . . . . . . . . . 26

5.1.1 Layer 1: Actuator Sensor Transducer Layer . . . . . . . . . . . . . 26

5.1.2 Layer 2: Node Control Layer . . . . . . . . . . . . . . . . . . . . 26

5.1.3 Layer 3: CAN Layer . . . . . . . . . . . . . . . . . . . . . . . . 26

5.1.4 Layer 4: Data Distribution Layer . . . . . . . . . . . . . . . . . . 27

5.1.5 Layer 5: System Control and Master Initiator Layer . . . . . . . . . 27

5.1.6 Layer 6: Environment Data Collection Layer . . . . . . . . . . . . . 27

5.1.7 Layer 7: Computing and Processing Layer . . . . . . . . . . . . . . 27

5.2 Nodes of the Vehicle . . . . . . . . . . . . . . . . . . . . . . . . . . . 28

5.2.1 Master Controller . . . . . . . . . . . . . . . . . . . . . . . . . 28

5.2.2 Acceleration Node . . . . . . . . . . . . . . . . . . . . . . . . . 28

5.2.3 Brake Node . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29

5.2.4 Battery Management Sytem(BMS) Node . . . . . . . . . . . . . . . 29

**6 Results and Discussion** **31**

6.1 CAN Layer . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33

6.2 Sensor Layer . . . . . 34
6.2.1 LiDAR . . . . . 34
6.2.2 GPS . . . . . 34
6.2.3 Motor Feedback . . . . . 34
6.2.4 Depth Camera . . . . . 35
6.3 Motor Control and Tuning . . . . . 35
6.4 Data Distribution Layer - ROS2 . . . . . 35
6.4.1 Software Description of Vehicle . . . . . 35
6.5 ROS Nodes . . . . . 36
6.6 ROS Topics . . . . . 37

**References** **38**

# List of Figures

2.1 Standard CAN Frame Layout . . . . . . . . . . 8

3.1 A bed slinger 3D printer . . . . . . . . . . 17
3.2 Micro-controllers . . . . . . . . . . 18
3.3 CAN Transceiver . . . . . . . . . . 18
3.4 Main System Board . . . . . . . . . . 20

4.1 PID Controller . . . . . . . . . . 23

5.1 Master Controller of CAN BUS . . . . . . . . . . 28
5.2 Acceleration Node connected to CAN Bus . . . . . . . . . . 29
5.3 Brake Node connected to CAN Bus . . . . . . . . . . 29
5.4 BMS Node connected to CAN Bus . . . . . . . . . . 30

6.1 CAN bus overview . . . . . . . . . . 33
6.2 Software Description of the Vehicle . . . . . . . . . . 36

# List of Abbreviations

**API** Application Protocol Interface . . . . . . . . . . . . . . . . . . . . . . . . . . 14

**CAN** Controller Area Network . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

**CRC** Cyclic Redundancy Check . . . . . . . . . . . . . . . . . . . . . . . . . . 8

**GPS** Global Positioning System . . . . . . . . . . . . . . . . . . . . . . . . . . vii

**I2C** Inter Integrated Circuit . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

**IMU** Inertial Measurement Unit . . . . . . . . . . . . . . . . . . . . . . . . . 15

**LIN** Local Interconnect Network . . . . . . . . . . . . . . . . . . . . . . . . . 5

**ML** Machine Learning . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

**MOST** Media Oriented Systems Transport . . . . . . . . . . . . . . . . . . . . 5

**PCB** Printed Circuit Boards . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

**PMBOK** Project Management Body of Knowledge . . . . . . . . . . . . . . . . . . 13

**PMI** Project Management Institute . . . . . . . . . . . . . . . . . . . . . . . 13

**RTK** Real Time Kinematics . . . . . . . . . . . . . . . . . . . . . . . . . . . vii

| | | |
|---|---|---|
| **RTOS** | Real Time Operating System | 3 |
| **RTR** | Remote Transmission Request | 8 |
| **SBC** | Single Board Computer | 19 |
| **SCI** | Serial Communications Interface | 5 |
| **SDK** | Software Development Kit | 14 |
| **SLAM** | Simultaneous Localization and Mapping | 15 |
| **SOF** | Start-of-frame | 7 |
| **TSN** | Time-Sensitive Networking | 6 |
| **QA** | Quality Assurance | 13 |

# 1. INTRODUCTION

## 1.1 Background

The recent trend in robotics has encouraged a class of vehicles which are autonomous and controllable to an extent by the user. Such intelligent system are able to identify their surroundings and react to different conditions that can occur in the surrounding when they are assigned to a task. These class of vehicle need a robust, reliable and highly reactive system to control them. Such systems are capable of processing loads of data in real time and adjusting to the required state in real time which requires a sophisticated set of devices within the system. Currently these systems are not that easy to find and are based on proprietary solutions. In context of Nepal, the vehicle industry has not taken off, there has not been any significant efforts in this industry to make it happen either. The environment is not predictive and previously employed and tuned controlled system may not be able to adapt to changing conditions. With advancement in edge AI it has allowed the device to compute sophisticated task real time on device which are up to the task of computing the changing environment around them. These mobile autonomous vehicle are able to explore and perform tasks that are not possible for normal human being.

## 1.2 Problem Statements

The problem statements we tackle for our half of the two-team collaboration are as follows:

- Can the system adapt to changes in the environment around it?
- How can we develop multiple feedback loops within a network?
- How can the system be reliable as to work on its own?
- Does the interface encounter all the exceptions and errors that can occur?
- Can we change the response of the system on demand?

These questions are integral to the challenge of building this project. The first two question relate on how we should implement the system and the third part reflects on what consideration needs to be taken while building such systems. The last two questions are based on how the communication system should react based on the input to the system.

## 1.3 Objectives

### 1.3.1 Collaboration Objectives

The collaboration objectives state the goals of the two-team collaboration, covering the entire project including the hardware, electronics, control systems and software. The can be stated as follows:

- To build an autonomous vehicle capable of driving a passive passenger from one location to another given the locations.
- To have the vehicle deal with obstacles, stay on route, and make decisions about changing routes both on a small and large scale in order to cause least damage to itself and discomfort to the passenger.
- To create a modular, layered and abstracted system that on principle opens possibilities of the modules' usage in a much wider variety of applications in the field of robotics and AI agents.

Further specifics from this point on can be found in the respective project objective sections for either half of the collaboration.

### 1.3.2 Objectives of This Project

An autonomous system includes a network of devices working in conjunction to achieve a system level goal.

1. System level design of every system nodes
2. Controller Area Network (CAN) protocol design and communication standard
3. Implementing on device Machine Learning (ML) on micro-controller and embedded systems
4. Establish a defined error reporting and error handling sequences
5. Higher Level abstracting command and hierarchy of command signals
6. Building a vehicle that can accommodate all the components of the system
7. Takeover constraints to discard the higher level automation commands
8. Tuning the system to different reactive circumstances

## 1.4 Scope

### 1.4.1 Collaboration Scope

The collective scope of the entire project refers to both the problem statements and the objectives to define the coverage and limits set for our goals in order to make the project well defined and achievable. They can be stated as follows:

- Regarding the transportation of a passenger from one location to another
  - The number of passengers will be limited to one
  - The area for which the locations may be defined will be confined within Pulchowk Campus
  - Obstacles will be within the context of the campus (hence, some common elements present in the usual context of vehicles may not be considered, such as traffic lights)
  - The speed off the vehicle will be limited to 20 km per hour
- Regarding layering and modularity
  - The number of layers and modules will be kept low to accommodate the workflow for our small team and low resources

### 1.4.2 Scope of This Project

For this half of the two-team collaboration, the following scope will be considered:

- Regarding the capabilities of the system
  - The system will have knowledge about localization and layout of campus
  - The system will be able to cope up with failure in any part of the system
  - User can override the control of the system
  - All the sub systems are isolated and have specific contracts with each other
  - The system will be based on deadline and use Real Time Operating System (RTOS)
- Regarding the information provided by the AI agent
  - Defined commands will be provide to a middleware
  - Middleware translates those information to required data packets

# 2. Literature Review

The autonomous vehicle was once thought of as a dream but it is closer and closer to become a reality. The consequences of vehicle automation are already being outlined on global mobility, on traffic efficiency, on competitiveness, etc [1]. This requires advances in many aspects of vehicle autonomy, ranging from vehicle design to control, perception, planning, coordination, and human interaction. Autonomous vehicles, which operates in complex dynamic environments, require methods that generalize to unpredictable situations and reason in a timely manner in order to reach human-level reliability and react safely even in complex urban situations [2].

The SAE International (Society of Automotive Engineers) has define various levels of autonomy ranging from level 0 up to level 5. The daily utility vehicles belongs to level 0 and vehicles where no human interaction is required belongs to level 5. Level 5 automation is yet to be achieved due to limitations in understanding and decision making. Continuous industry and academic effort can facilitate the achievement of level 5 automation system in the near future [3].

## Overview of Autonomous Vehicle Technology

Autonomous vehicle technology combines sensors, perception algorithms and advanced control systems to enable vehicles to navigate and make decisions without human intervention. With the integration of artificial intelligence and mapping technologies, these vehicles can perceive their environment, plan routes and execute actions with precision. Safety features and redundant measures ensure reliable and secure autonomous operations.

- **Sensor Integration**: Autonomous vehicles utilize a range of sensors, such as cameras, LIDAR, RADAR, and ultrasonic sensors, to gather comprehensive data about their surroundings.
- **Perception Algorithms**: Cutting-edge perception algorithms process sensor data to understand the environment and identify objects. For instance, computer vision and machine learning techniques are employed to classify pedestrians and detect obstacles.
- **Advanced Control Systems**: Autonomous vehicles employ sophisticated control systems to execute actions in real-time. These systems integrate sensor data and vehicle dynamics to make decisions such as acceleration, braking, steering, and lane

changes. Examples include adaptive cruise control, lane-keeping assist, and collision avoidance.

- **Artificial Intelligence (AI)**: AI plays a crucial role in autonomous vehicle technology, allowing vehicles to learn from data and improve performance. Machine learning and deep learning algorithms enable tasks such as object recognition, behavior prediction, path planning, and decision-making in complex environments.

- **Mapping and Localization** : High-definition mapping and precise localization are essential for autonomous navigation. Mapping involves creating detailed maps of the environment, while localization techniques like GPS and SLAM algorithms help vehicles determine their position accurately.

- **Safety Features** : Autonomous vehicles prioritize safety and safety features like automatic emergency braking and blind-spot detection provide additional protection for passengers and pedestrians.

## Node-to-Node Communication Protocol

The Controller Area Network (CAN) has become the standard communication protocol in autonomous node-based systems. It facilitates reliable and efficient data exchange between electronic control units (ECUs), enabling seamless coordination among subsystems. CAN's widespread adoption has made it a key component in the development of advanced autonomous capabilities.

Before the widespread adoption of the CAN, several communication protocol advances shaped the automotive industry. Serial Communications Interface Serial Communications Interface (SCI) provided a simple asynchronous serial communication interface, while Local Interconnect Network (LIN) offered a cost-effective alternative for less critical systems. Media Oriented Systems Transport (MOST) focused on high-speed multimedia and infotainment communication, utilizing fiber optics. Inter Integrated Circuit (I2C) facilitated short-range communication within electronic systems, and SAE J1850 served diagnostic and data communication purposes. Although these protocols were not as prevalent as CAN, they paved the way for advancements in automotive communication, contributing to the evolution of more sophisticated and standardized protocols.

Recent achievements in the CAN protocol include the introduction of CAN FD, enabling higher data rates and larger payload sizes for faster and more efficient communication. Enhanced security features, such as Secure CAN (CANcrypt), have been developed to ensure authentication and data integrity within CAN-based systems. Integration with

Time-Sensitive Networking (TSN) enables real-time and synchronized communication, while integration with Automotive Ethernet enhances compatibility with modern vehicle architectures. Additionally, improvements in fault tolerance and robustness have bolstered the reliability and resilience of CAN-based systems.

### Practical Challenges

Advancements in vehicle technology and traffic administration have led to a decrease in road deaths in developed countries, while developing countries like India are experiencing the opposite trend. Globally, the number of traffic-related deaths remains high. Autonomous driving can significantly reduce accidents, with a high penetration rate of fully autonomous vehicles and effective traffic management strategies being crucial. However, the increase in vehicle-kilometers traveled may offset the decrease in accidents. Other risks include passenger overconfidence and reckless pedestrian behavior.

## 2.1 Related work

Robotics and Artificial Intelligence Laboratory researchers at the Tongji University of China have published an article on the architecture design and implementation of autonomous vehicle [4]. The article discussed a practical framework of hardware and software. It describes three typical sensor plans and introduces a general autopilot for the vehicle. The final report contains autonomous driving test implemented using the proposed architecture.

## 2.2 Related theory

### 2.2.1 Stereo vision

Stereo vision, also known as stereo vision or binocular vision, is a technique used in computer vision and robotics to extract depth information from a pair of images captured by two cameras, commonly referred to as a stereo vision camera setup. The basic principle behind stereo vision is triangulation, which involves using the displacement or disparity between corresponding points in the two images to estimate the depth or distance of objects in the scene. The stereo vision processing pipeline is as follows:

- **Stereo vision Camera Setup**: Two cameras are positioned horizontally or slightly diverged from each other, mimicking the separation of human eyes. These cameras capture two slightly different views of the same scene simultaneously.

- **Image Acquisition**: Both cameras capture the scene, resulting in a pair of images called the left image and the right image. These images represent the scene from two different viewpoints.

- **Correspondence Matching**: Correspondence matching is performed to find corresponding pixels or features in the left and right images. This process involves comparing the intensity or feature descriptors of pixels in one image with the corresponding pixels in the other image.

- **Disparity Calculation**: Once the corresponding pixels are identified, the disparity or the horizontal displacement between the pixels in the left and right images is calculated. Disparity represents the apparent shift of an object between the two views due to the cameras' baseline separation.

- **Triangulation**: Using the disparity information and known camera parameters (such as the baseline distance and focal length), triangulation is performed to estimate the depth or 3D coordinates of the scene points. Triangulation involves finding the intersection point of two rays originating from the camera centers and passing through the corresponding image points.

- **Depth Map Generation**: By applying triangulation to all the corresponding points in the stereo image pair, a depth map or a disparity map is created. The depth map represents the scene's 3D structure, where each pixel corresponds to the estimated distance or depth of the corresponding object point in the scene.

- **3D Reconstruction and Applications**: With the depth map or the 3D information, it becomes possible to reconstruct the scene in three dimensions. This information can be used for various applications such as object detection, tracking, obstacle avoidance, 3D mapping, and robot navigation.

The accuracy of stereovision-based depth estimation depends on factors like camera calibration, image rectification, the quality of correspondence matching, and the baseline distance between the cameras. Disparity maps can provide valuable information about the scene's depth structure and help in perceiving the 3D environment, enabling more advanced and precise analysis in computer vision and robotics applications.

### 2.2.2 CAN frames

A CAN frame consists of several fields that together form the message being transmitted or received. The most common type of CAN frame is the Standard Frame, which has the following components:

- **Start-of-frame (SOF)**: This is a single dominant (logic 0) bit indicating the beginning of a CAN frame.

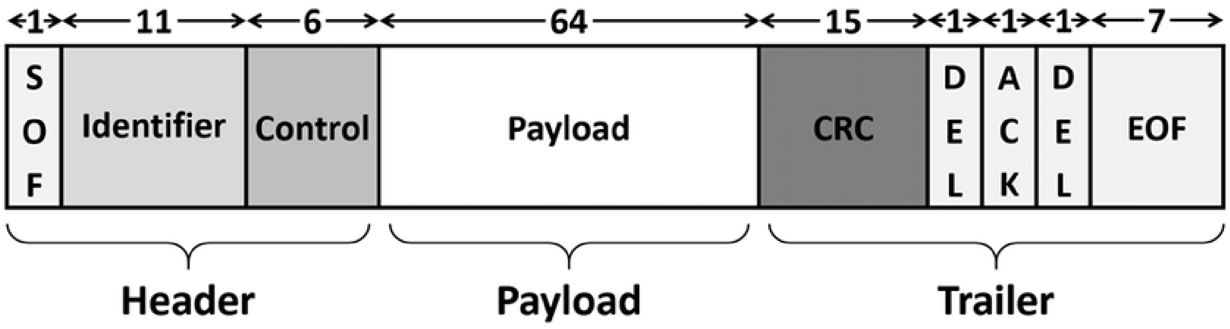


Figure 2.1: Standard CAN Frame Layout

- **Arbitration field**: It contains the identifier or message priority, which is used for determining message precedence when multiple nodes attempt to transmit simultaneously. The arbitration field consists of the identifier (11 bits for standard CAN) and the Remote Transmission Request (RTR) bit.
- **Control field**: This field includes bits for specifying the frame type, such as data frame or remote frame, as well as additional control information.
- **Data field**: This field carries the actual data being transmitted, ranging from 0 to 8 bytes in length. The data field is optional for remote frames.
- **Cyclic Redundancy Check (CRC)**: It is a 15-bit or 17-bit field used for error detection. The transmitting node calculates the CRC based on the data and appends it to the frame, while the receiving node performs the same calculation to verify the integrity of the received data.
- **Acknowledge field**: It consists of an acknowledge delimiter and an acknowledge slot. The acknowledge field serves to acknowledge successful receipt of a frame by other nodes.
- **End-of-frame (EOF)**: A sequence of seven recessive (logic 1) bits that mark the end of the frame.

CAN frames can be transmitted either in a ”broadcast” mode, where all nodes on the bus receive the frame, or in a ”point-to-point” mode, where the frame is specifically addressed to a single node.

In addition to the Standard Frame, there is also an Extended Frame format that allows for longer identifiers (29 bits) and more data payload (up to 64 bytes).

CAN frames enable reliable and efficient communication between various electronic control units within a system, facilitating tasks such as sensor data exchange, control signal transmission, and fault diagnosis.

### 2.2.3 SLAM

Simultaneous Localization and Mapping (SLAM) is a technique used by robots to build a map of an unknown environment while simultaneously estimating their own position within that map. The main goal of SLAM is to allow a robot to navigate autonomously in an unknown environment by constructing a map of the surroundings and estimating its own pose (position and orientation) relative to that map. This process involves several key steps:

- **Data Acquisition**: Robots collect sensor data from range sensors like lidar or depth cameras.
- **Feature Extraction**: Distinctive features are extracted from the sensor data.
- **Data Association**: Features are matched across frames to establish correspondences.
- **Mapping**: Robots incrementally build a map by incorporating observed features.
- **Localization**: Robots estimate their pose (position and orientation) relative to the map.
- **Loop Closure**: Revisited areas are detected, allowing for error correction and map refinement.
- **Optimization**: Global optimization improves the map and pose estimates by minimizing inconsistencies.

SLAM is used in various applications such as robotic navigation, autonomous vehicles, augmented reality, virtual reality, and 3D mapping. It enables robots to explore and understand unknown environments, performing tasks autonomously.

### 2.2.4 Isolated Redundant Network

An isolated redundant network, also known as a redundant network architecture or a redundant network design, is a system configuration that incorporates redundancy and isolation to enhance reliability and fault tolerance. It is commonly employed in critical infrastructure, industrial control systems, and mission-critical applications where system failures can have severe consequences.

The primary objective of an isolated redundant network is to ensure continuous operation even in the presence of failures or disruptions. This is achieved through the following key features:

- **Redundant Components**: Duplicate components and alternative paths are used to provide backup and redundancy.
- **Isolation**: The network is physically or logically separated from other systems to prevent failures from spreading.
- **Failover Mechanism**: Automatic switching to redundant components or alternate paths ensures uninterrupted operation in case of failures.
- **Network Monitoring**: Robust monitoring tools detect issues early and facilitate proactive measures.
- **Load Balancing**: Traffic distribution optimizes network utilization and prevents congestion.

An isolated redundant network provides increased reliability, fault tolerance, improved performance, enhanced security, and scalability. It ensures continuous operations, minimizes downtime, optimizes resource utilization, reduces the risk of unauthorized access, and accommodates future growth.

### 2.2.5 RTK GPS

Real-Time Kinematic (RTK) GPS is an advanced satellite-based positioning technology that provides highly accurate and precise location information for various applications. Unlike standard GPS, which typically offers meter-level accuracy, RTK GPS significantly enhances accuracy down to the centimeter or millimeter level. This level of precision is achieved through a combination of a fixed base station and a rover receiver. The base station, placed at a known location, continuously tracks satellite signals and calculates the errors in GPS measurements. The rover receiver, often mounted on a moving object like a vehicle or a drone, receives these corrected signals from the base station in real-time, allowing it to determine its precise position with exceptional accuracy.

RTK GPS relies on a technique known as carrier-phase tracking, where it measures the carrier phase of the GPS signals rather than just the code phase. This carrier-phase information is highly sensitive and can provide accurate distance measurements between the receiver and the satellites. By resolving the integer ambiguity associated with carrier-phase measurements, RTK GPS eliminates most sources of errors, including atmospheric disturbances and satellite clock inaccuracies. This makes RTK GPS ideal for applications that demand pinpoint accuracy, such as land surveying, precision agriculture, construction site management, autonomous vehicles, and geodetic research.

RTK GPS with both the base and rover in motion is a specialized application of Real-Time Kinematic (RTK) GPS technology, where both the reference base station and the mobile rover are in motion simultaneously. This configuration introduces additional complexities and challenges compared to traditional static or kinematic RTK setups. Here's how it works and some considerations for such a scenario:

In a typical RTK GPS setup, a stationary base station accurately determines its location and calculates corrections for the satellite signals it receives. These corrections are then transmitted to the mobile rover, which uses them to improve the accuracy of its position calculations.

In the case of RTK GPS with moving base and rover:

- Dynamic Base Station: The base station itself is mobile and is moving alongside the rover. Both the base and rover receivers continuously track satellite signals and exchange correction data, essentially creating a relative RTK system where the base's position is used as a reference for the rover's position.

- Continuous Communication: A robust and low-latency communication link is essential between the base and the rover to ensure that real-time corrections are transmitted effectively, despite the movement of both units.

- Relative Positioning: Since both the base and rover are in motion, the RTK GPS system calculates the relative positions between the two units in real time. This can be useful in scenarios where the exact separation and relationship between the two units need to be precisely known, such as in vehicle platooning, robotic swarms, or dynamic surveying applications.

- Accuracy and Ambiguity: The dynamic nature of both units introduces challenges in resolving the integer ambiguity associated with carrier-phase measurements. This can affect the system's ability to provide centimeter-level accuracy consistently, particularly during rapid movement or changes in direction.

- Application Scenarios: RTK GPS with moving base and rover has applications in areas like precise navigation of multiple vehicles in formation, accurate mapping of changing landscapes, coordinated movement of autonomous drones, and more.

- Complexity and Calibration: This setup requires careful calibration and synchronization between the base and rover units. Any mechanical or timing discrepancies between the two units can impact the accuracy of the relative positioning.
- Integration with Inertial Sensors: In some cases, inertial sensors (such as accelerometers and gyroscopes) may be integrated with the RTK GPS system to improve accuracy and robustness, especially in scenarios with rapid movements and changes in orientation.

# 3. Methodology

## 3.1 Project Management

This project was expected to be on a scale large enough to require quick and efficient task division, work tracking, careful planning in all time scales, etc. As a result, we decided to build a robust project management scheme for our two-team collaboration, giving the role of Project Manager to one of the six members. The project manager conducted all of the aforementioned tasks as well all levels of integration, planning, controlling, and monitoring, etc. as mentioned in and in accordance with **Project Management Institute (PMI)'s Project Management Body of Knowledge (PMBOK)**. The team member with the project manager role assigned had drafted a Project Management Plan laying out the planning, controlling, and monitoring schemes that were used during the development and progress of the project. The Project Management Plan document was attached to this proposal. The project management aspect of the project was made as official and as close to the actual industry as possible in order to provide legitimate project managing and team-work experience holding proper weight for applications in the industry.

### 3.1.1 Quality Assurance

With a robust project management scheme, a robust Quality Assurance (QA) scheme was also required. The purpose of this scheme was to estimate, enforce, and monitor the quality of entities and modules built, having a well-defined plan for measuring and tracking the quality. This information was used to alter the plan for the project to keep expectations realistic as well as the workflow to keep quality acceptable. For example, a 3D map of the campus was required for building the virtual environment for early training and testing (see below), the workflow for which was for the most part unique and improvisational. QA was responsible for defining the quality of the 3D mapping required, and monitored the work efficiency, noting successes and failures, to help in the future. Since a large portion of the project involved defining layers, modules, their abstraction, their interfacing, etc., their quality was defined as how flexible they were with their abstraction, how universal their inputs and outputs were, etc. All these definitions were managed by QA.

QA was integrated with the project management activities, and the role of *Quality Assurance Manager* was given to one of the six team members. This role (not necessarily the assignment) has been separated from the project manager role to stress its importance and build a separate abstracted scheme. Common practice is to have a dedicated team for QA, however, since our team is small, this was limited to one of the roles of a single member, with flexible coordination with other members for any problem solving and information gathering involved.

## 3.2 Scale of the Project

It was necessary to define our project assumptions. We created a go-cart sized (around 78" long, 25" tall, and 52" wide) vehicle which is controlled by the use of Brushless Hub motors like those found on electric scooters. The rear of the vehicle held two of these motors, and the front wheel was coupled for steering action. The steering implemented a drive-by-wire concept utilizing a system to decouple the steering from the user. Battery Packs were stored in the vehicle. The vehicle also had an infotainment system to provide real-time feedback of what the system was doing.

When we were behind a vehicle, we had a perspective of the environment around the vehicle and could adapt to changes within the environment instantly. This is difficult to achieve in an autonomous system and all of the system and environment perception needed to be calculated and handled by the system itself. The system was able to counteract the changes instantly by itself. Vehicle design took a lot of research and work. The difficult aspect before any mass production was creating the vehicle itself. We were not looking for mass producing this vehicle and neither did we want to spend time on any aesthetics of the system. So taking that out of the equation, still the main aspect of any vehicle was the chassis. The chassis needed to be designed from scratch which added about 1 month of work hours for a team of six like ours.

As listing our working domain, we had to complete tasks from Printed Circuit Boards (PCB) designing, circuit layout, sensor placement, communication design, protocol interface, embedded firmware development, real-time operating systems, linux socketCAN, CAN protocols, system errors and exceptions, Software Development Kit (SDK) design or Application Protocol Interface (API) design, mapping, slam, mechanical gears and tools, welding, CAD, simulations, machine learning, multi-layer model deployment and much more to be expected. It was sure that this was a very engaging project and would populate most of our effort in it.

The system design required a thorough introduction to how each component would play a role in the system. Since this was a node-based isolated system, each node was a system by itself. This contributed to a lot of work for the team but brought a trade-off. The system was modular and could be replaced at node if any error occurred. Also about the system-level exception and reporting protocol that needed to be defined. For a team like ours, we estimated this alone could take around 2-3 months of active work. And we were not even in the tuning part of the system yet, as each node needed to be tuned to the sensor, motors, or transducers they were connected to. This alone could take 2 months of active work for optimizing the system.

Now the system can be controlled but we needed a way to send command to the system and show its status on a dashboard. Some UI design, control tweaking would take some more amount of our time. After this was the main part of the system, making it autonomous and providing an abstracting SDK to build on top of our hardware or creating an API to which the higher level software could connect and send commands. We needed to expose all of our real-time data to the upper layers of Machine Learning models. This was the part to combine the work of the Machine Learning part of the system. So both of the team needed to run in parallel to make sure that they achieved the goals on time and none of them acted as throttling each others job. The collaboration alone could take 3 months to complete. Therefore, the hardware team could expect active work of around 8 months for the system to be complete. Of course, many tasks could run in parallel and multiple tasks could make it easy for other tasks to be complete.

## 3.3 Autonomous Region and Environment

The world for the vehicle is where it is able to provide its services. Mapping a large area requires a lot of time and effort. For this vehicle, we have chosen the world to be Pulchowk Campus CIT block. We utilize all the roads present in there, where the vehicle is free to roam about provided that instructions are given to it. A mapping needs to be done of Pulchowk Campus to provide a digital environment to the vehicle.

## 3.4 Absolute and Relative Positioning of the Device

We need to keep track of where the vehicle is so that we can direct the vehicle towards where it needs to go. For this, we need to know the position of the vehicle. We are using a GPS-based system with Simultaneous Localization and Mapping (SLAM) to pinpoint its exact location. The directive constraint is provided via an Inertial Measurement Unit (IMU) that is on the device.

## 3.5 Team and Work Division

We are a team of three individuals with a diverse set of skills. The work was divided on the basis of our individual skill sets. With this classification, we divided our work under multiple categories so that it is easy to evaluate each work.

1. **Mechanical Work**

One of our team members has some experience in mechanical work and with the assisted tools in the Pulchowk Robotics Club, all of our mechanical work was completed under the supervision of that team member. The mechanical parts include the frame and all the actuators that we need to build on our own.

2. **Electronics and Embedded Systems**
   We are students of this very domain and are quite familiar with the system we need to develop, so this work was divided equally if not based on the amount of work each team member has.

3. **Programming Based Work**
   This is similar to that of the before as it was divided among all of the members.

4. **Protocol and Contract Defining Work** This lays a foundation on how the system should collaborate, so all need to be there to lay the groundwork for this work.

5. **CAD Based Work**
   One of our members has previous experience in CAD-based work using Fusion 360, Solid Works, and Free Cad and another member has experience in Blender. So the CAD-based work was divided among these individuals.

6. **PCB Design and Fabrication Based Work**
   One of our members has an excellent record of PCB design during their time in Pulchowk Campus, so this work was issued to them.

### 3.5.1 Collaboration Team

The hardware and software parts of the system are vague by themselves and require a lot of work in each aspect. Two teams are present which represent each of the hardware and software parts of the system. Here hardware refers to the controllable system design and software refers to the controlling system design. Both teams are not only to complete each of their tasks but collaborate throughout the process about each and every decision that is to be made throughout the development, as every single change was expected to affect both teams. So a collaborative effort is to be made between the teams defining what they are doing and pitching in the necessary developments to the other team so that they can bring those changes in their domain. Both teams worked independently with a timely update from each side. The integration part of the system started as early as the project initiation and was active throughout the development. This is why we chose a project management-based approach so that each system is documented and each member is aware of what is going on in the system.

## 3.6 Equipment, Tools and Devices

These are the devices that we used in the system or was used to design the system.

- **3D Printer** One of our team members has access to a 3D Printer, so all of our CAD-based work was brought to life using this machine. This enables us to create parts that are unimaginable with handwork.

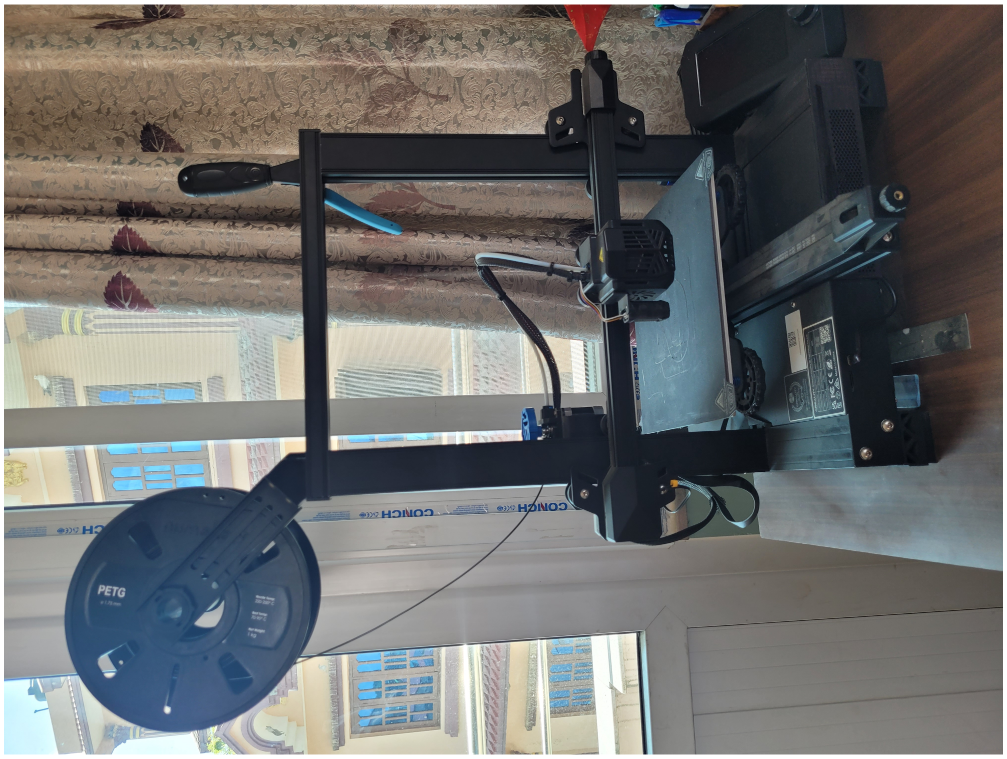
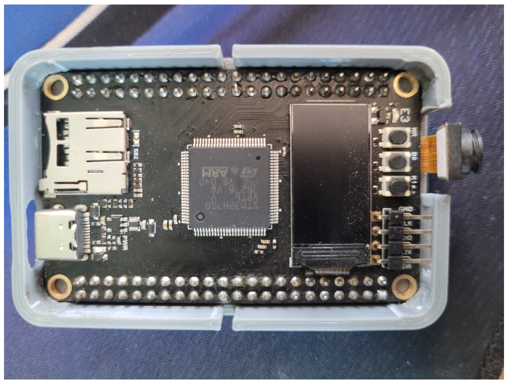
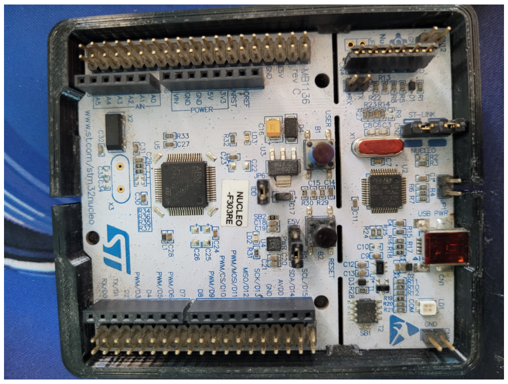


Figure 3.1: A bed slinger 3D printer

- **Micro controller** We have a range of micro-controllers from STM32 like F4,F3,M0,H7 series. With a total of 10 micro-controllers, we have the ability to create isolated redundant systems.


(a) Master Controller


(b) Node Controller A

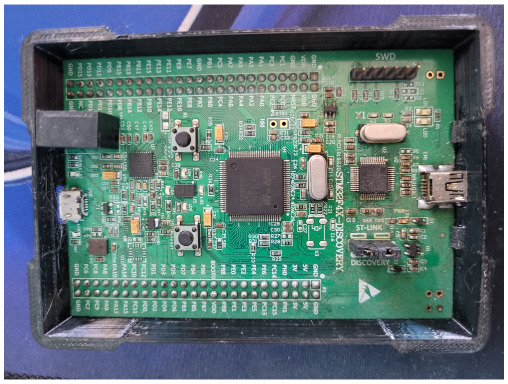

(c) Node Controller B

Figure 3.2: Micro-controllers

- **CAN Transceivers** These form the physical layer of our CAN Bus protocol.

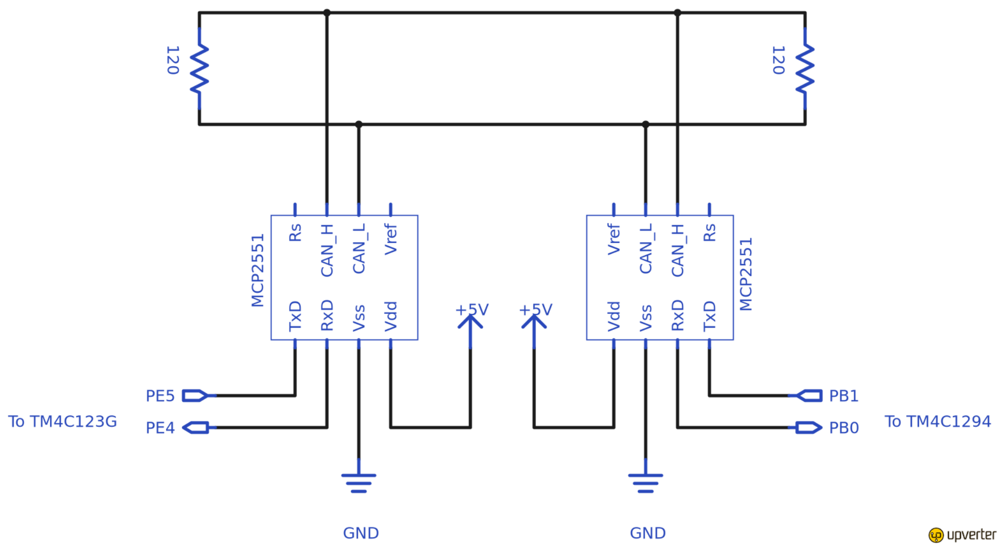
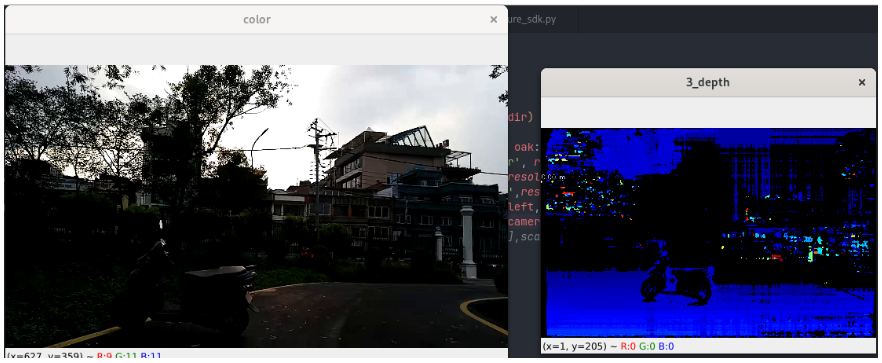


Figure 3.3: CAN Transceiver

- **Stereo-Vision Camera** A stereo-vision camera works on the basis of disparity and triangulation to produce depth feedback from the environment. We are utilizing the depth sensor to provide feedback to the system about the path, roadways, and obstacles that are present in front of the vehicle.

(a) Camera



(b) StereoVision of Camera

- **Main Board ( Logic Board )** A real-time mapping has been provided to the vehicle using the sensor in the vehicle. There is a requirement to process multiple camera feeds and data from the sensors in the vehicle to provide feedback to the vehicle. These enormous processing requires a lot of calculation and needs to process multiple ML models. With this in mind, we have got a hold of a board that is capable of processing this information in real time. The board is based on ARM architecture and is one of the fastest Single Board Computer (SBC) available till date. We use linux with SocketCAN to communicate with our low-level CAN BUS. This has a contract with the CAN Master Controller and can provide instruction to the vehicle.

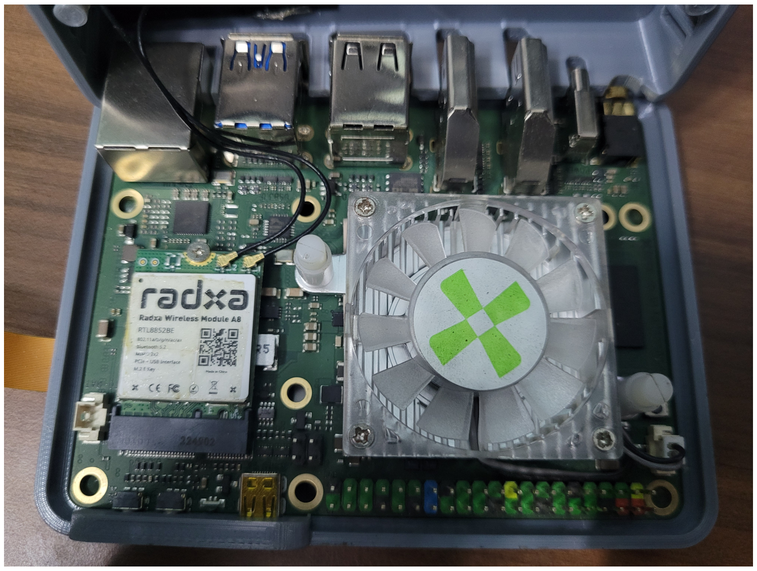


Figure 3.4: Main System Board

# 4. Experimental Setup

## 4.1 Node Level Isolated System

Each node of the CAN bus, such as the acceleration node or braking node, is carefully isolated from one another to ensure optimal performance. This isolation is achieved by assigning each node its own micro-controller, allowing operations specific to that node to be carried out independently. By operating in isolation, the functions of one node do not interfere with the operations of other nodes, thereby enhancing the speed and efficiency of the entire system. Each node is capable of receiving message packets from other nodes or the master controller, and based on the content of these messages, it performs the necessary tasks. The isolation of nodes not only contributes to reliable and robust communication but also necessitates the use of a custom CAN protocol. Designing this custom message communication protocol involves the meticulous debugging of packets using tools such as oscilloscopes and signal analyzers, ensuring the smooth and accurate transmission of data across the CAN bus.

## 4.2 Communication Protocol Design

Designing a custom CAN bus communication protocol requires careful consideration and adherence to certain principles. Here are some steps to design a custom CAN bus protocol:

1. **Define Communication Requirements:** We start by clearly defining the communication requirements for our specific application. We identify the types of data to be transmitted, their priorities, message formats, and any specific timing constraints. We consider the desired network topology and the number of devices that will be connected to the bus.

2. **Identify Message Structure:** We determine the structure of the messages that will be transmitted over the bus. This includes the identification of message IDs, data fields, and any additional control or status information that needs to be included. Decide on the data formats, such as binary, hexadecimal, or ASCII.

3. **Define Message Arbitration:** CAN bus uses a priority-based message arbitration scheme to determine which message gets transmitted when multiple devices attempt to send data simultaneously. We then define our custom arbitration scheme, considering

factors such as message priorities, message IDs, and the algorithm for resolving bus contention.

4. **Error Handling and Recovery:** We plan how your custom protocol will handle errors and recover from them. We define the error detection and error handling mechanisms, such as checksums, CRCs (Cyclic Redundancy Checks), and acknowledgments. Then, we determine the actions to be taken when errors occur, such as re-transmission or error notification.

5. **Frame and Bit Timing:** We then specify the frame format and bit timing parameters for your custom protocol. The length of the data frame, the number of bits for the message ID, and any additional fields required are determined. Then, the bit timing parameters, such as bit rate, sample point, and synchronization requirements are also determined.

6. **Protocol Implementation:** We then implement our custom CAN bus protocol in software or hardware, depending on the needs. Then the necessary firmware or drivers to handle the protocol on the transmitting and receiving devices are developed. We then test the implementation thoroughly to ensure its correctness and compliance with the desired specifications.

7. **Verification and Validation:** We need to perform rigorous testing and validation of our custom CAN bus protocol. we have to test it under different scenarios, including normal operation, high-load conditions, and error conditions. we need to verify its performance, reliability, and adherence to the defined specifications.

## 4.3 PID Tuning and System Calibration

The system may have errors, you may not achieve the required output in the system due to mechanical deformities and environmental factors. This variations in the required output of the system and actual output of the system can be minimized and system can be calibrated using the technique called PID(Proportional, Integration, Derivative) tuning. PID tuning uses PID controller which has three constants(($K_p$, $K_i$ and $K_d$) that can be manipulated.

PID controller is one of the most popular closed-loop controllers which is used in the automation industry. By fine-tuning 3 constants, you are able to achieve a system which is almost free from any errors. [5]

In a PID controller, we calculate an error $e(t)$ as the difference between the desired set-point and the current value(process variable) and pass it as a feedback signal. The error $e(t)$ is then corrected based on the proportional(p),integral(i) and derivative(d) terms.

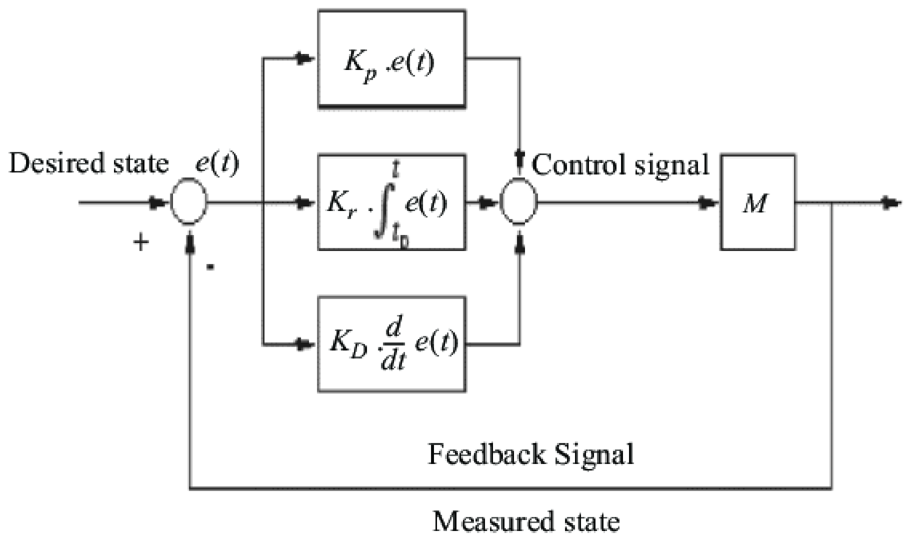


Figure 4.1: PID Controller

We'll use a genetic algorithm to find the 3 constant values($K_p$, $K_i$ and $K_d$).

As the name implies, the PID controller provides three separate actions on the error signal to produce the controller output, proportional, integral, and derivative. The proportional action produces an output proportional to the error signal $y(t) = K_p * e(t)$. Thus, no proportional action will result when the plant output is equal to the set point, since the error is zero. As the proportional gain $K_p$ is increased the controller will produce a larger signal in response to an error. This amplification of the error signal will result in a faster response to errors, but may also add instability. These effects are quantified by the response having a shorter rise time, and an increase in the percent overshoot. Conversely, reducing the proportional gain will reduce overshoot but will increase the steady state error.

The derivative action provides a control signal proportional to the time rate of change of the error signal. $y(t) = K_d * de(t)/dt$. The derivative term provides an anticipatory element to the controller, providing larger controller response to rapidly changing error signals, and smaller response to slower changes. This permits a faster system transient response without increasing the percent overshoot. The derivative action alone has little effect on the steady state behavior of the system. It cannot remove a fixed error (constant steady state error) since the derivative of a constant is zero. Therefore, the derivative element of the PID controller would produce zero output to a non-changing error.

The integral action produces an output proportional to the accumulated (integral) of the error signal, allowing the controller to zero the steady state error between setpoint and system output. $y(t) = k_i \int e(t)$. From a control system theory perspective the integrator term improves the steady state performance because it increases the "system type" by one by adding a zero at the origin. Intuitively, the integral element insures that even small errors will eventually amass to produce significant controller output. The combination of the three actions is summarized as

$$y(t) = K_p * e(t) + K_d * de(t)/dt + k_i \int e(t) \tag{4.1}$$

In equation 4.1, $y(t)$ is the system output, $e(t)$ is the error signal, $K_p$, $K_i$,and $K_d$ are the proportional, integral, and derivative gain constants. The selection of the three constants $K_p$, $K_i$,and $K_d$, to achieve a specific response is known as tuning. Manual tuning consists of adjusting the three gain factors in a systematic manner using multiple trials.

After obtaining the mathematical model of our system, we can use MATLAB software to obtain the best possible value of $K_p$, $K_i$,and $K_d$. Using these values our system shall be calibrated.

## 4.4 Exposed System Parameter For Varied System Response

Our system offers the flexibility to tune various parameters of the vehicles according to specific preferences and requirements. Parameters such as brake bias, acceleration gain, battery current limit, throttle gain, speed limit, acceleration curve, motor sync parameters, steering motor gain, steering motor curve, and steering motor limits can all be adjusted.

All these parameters are exposed and can be updated via the controller with a set of administrative commands in the upper layer of the system in real-time, allowing for adaptive tuning. which means that the lower layer of the system can have a simpler design, as the complex calculations and feedback can be handled by the upper layer. By providing a mechanism to update these parameters, the system becomes more manageable, eliminating the need for firmware modifications on individual nodes. This indeed means calibration of the system is essential before use but we can store these parameters permanently in the memory of the main controller so when it wakes up it updates all of these parameters to a default state.

## 4.5 RC Based Initial Design

Our end system will pose different challenges when it comes to modifications, as each change becomes increasingly expensive as we scale up. We do not have the option to completely abandon the current idea or design if it doesn't work out, especially since we are working on the end product. This situation carries a huge amount of risk. Therefore, to adopt a more manageable approach is followed. We have decided to create a small-scale 1:1 Remote controlled vehicle which will be representative of the final product. This approach allows us the flexibility to make changes as we progress in our project. While it may not be possible to achieve a perfect replica of the end system, this strategy will effectively eliminates most of the design issues that arise in the early stages, thus preventing wastage of the time and resources

## 4.6 Mechanical Based System Design

In order to achieve our desired outcome, we must carefully design the individual mechanical components that will facilitate the translation of motion from various actuators or motors to trigger the mechanical system. Each aspect of the system will be given individually designed ensuring that we establish a clear understanding of how the electrical feedback can be effectively translated into mechanical feedback.

# 5. System Design

## 5.1 How does the system work?

The system works with the combination of different nodes and controller that can be divided into functional layers. Various data packets, messages, control signals are passed between the different layers for proper functioning of the system. The different layers are:

### 5.1.1 Layer 1: Actuator Sensor Transducer Layer

This layer acts as a physical foundation to connect the mechanical system of the vehicle to the electronically controlled systems. A sensor acts in reporting the current state of the mechanical system, a transducer acts in bidirectional communication to the mechanical parts, an actuator acts in helping the system achieve a given command.

### 5.1.2 Layer 2: Node Control Layer

A node represents an isolated system that can react on its own in the universe that is the vehicle. A node contains a range of sensors, actuator or transducer. A node consists of the following components, micro-controller(CPU), CAN controller, and CAN transceiver that is able to read the current status of the system and report its information to the higher layer. Making an independent node allows the system to be robust to include multiple nodes if required as a redundant system. A node is also able to receive commands from the upper layer to perform a given set of tasks. A ML network is initiated within the micro-controller which is able to identify anomalies of the system or update its parameter on the fly. A simple example would be correcting the power level fed to the motor by the battery management system in a range of voltage fluctuation as per the road conditions or the type of the road terrain.

### 5.1.3 Layer 3: CAN Layer

A Controller Area Network is what we are using for the communication within the nodes. A CAN Bus is a multi-master bus that runs parallel to each other in real time. The Bus is shared and all the nodes are not able to send data at the same time to other nodes. The Shared Bus allows the system to communicate with only two wires throughout the system.CAN itself has multiple layers which define things like cable type, electrical signal, node requirement, cable impedance e.t.c. Multiple types of CAN Bus are available, the one we implemented is a

High Speed CAN which has a bandwidth of 1 Mbits/s.The CAN Bus as two layers. They are:

- **Physical Layer of CAN:** CAN layer offers two wire communication, the wires are twisted and are a differential pair represented as CAN_H (can high) and CAN_L (can low). Termination resistors are present at each end which matches the nominal impedance of the wire.
- **Controller Layer of CAN:** A controller layer is responsible for the transmission and reception of messages between nodes. It performs activities like arbitration of messages and the error detection and handling.

### 5.1.4 Layer 4: Data Distribution Layer

The microcontroller acquires data from multiple sensors, which might be essential for other interconnected systems. To seamlessly distribute the information from these sensors and data from other device such as cameras and gps. We have opted for the Robotics Operating System (ROS). ROS functions on the principle of message passing by publishing and subscribing between nodes, thereby transmitting and distribuiting data more effectively.

### 5.1.5 Layer 5: System Control and Master Initiator Layer

Node Status are read by this control layer, it can control the other nodes in the network, handle system exceptions, reset the system to the default state, handle system errors and compensate for the errors, and translate the user interaction to the system. The system runs a higher level ML network which is able to identify the current system status and adjust to various environmental factors in real time. This is the controller that provides all the information on how the nodes should react.

### 5.1.6 Layer 6: Environment Data Collection Layer

Multiple cameras are employed within the system to understand the environment the system is in. The cameras we are using are stereoscopic cameras which understands the depth of the objects around it. Also the lower layer sensor information from the depth sensors are accessed and transformed. A SLAM based environment is employed to locate the system. The system also employs a GPS for real time location mapping.

### 5.1.7 Layer 7: Computing and Processing Layer

This layers runs application which takes all the feedback from the layers below and provides a general procedure on what the system need to do next.

## 5.2 Nodes of the Vehicle

Humans can take over the control of the vehicle whenever required, so every node of the vehicle can take inputs from both humans and AI. Both input mechanisms are different and separated from one another. Every node shall be designed in such a way that switching between human control and AI control can be achieved by clicking a button.

Following are the nodes of the vehicle:

### 5.2.1 Master Controller

Master Controller is the master of the CAN Bus system, attached with all the nodes, it initiates the communication, sends the message packets for operations in nodes and receive the information regarding state, operation and health of the node. It takes input from the AI computation, and transducers that generates signals on the basis of human input.

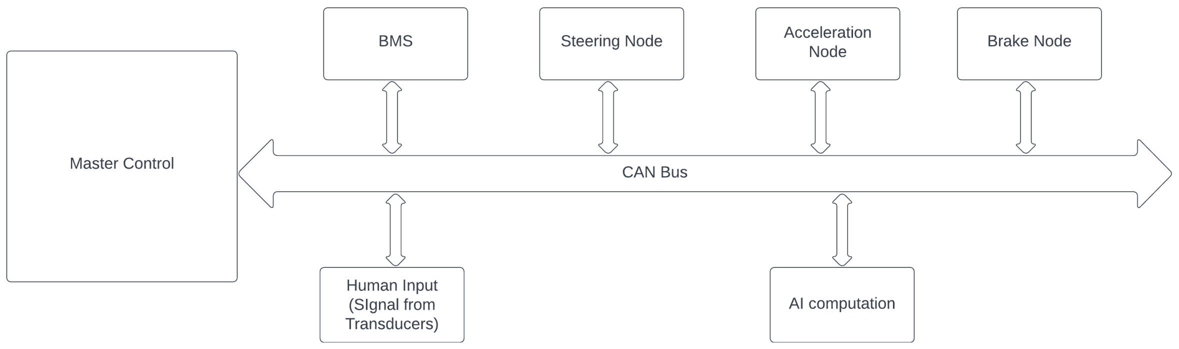


Figure 5.1: Master Controller of CAN BUS

### 5.2.2 Acceleration Node

This node is responsible for acceleration and de-acceleration of the vehicle. it operates when the master controller sends the message packets requesting for acceleration or de-acceleration of the vehicle. When it receives the signal from the master controller, it then sends signals(PWM signal) to the motor controller. This node sends the message packets to the master controller which includes the acceleration value of the vehicle and other operations of node.

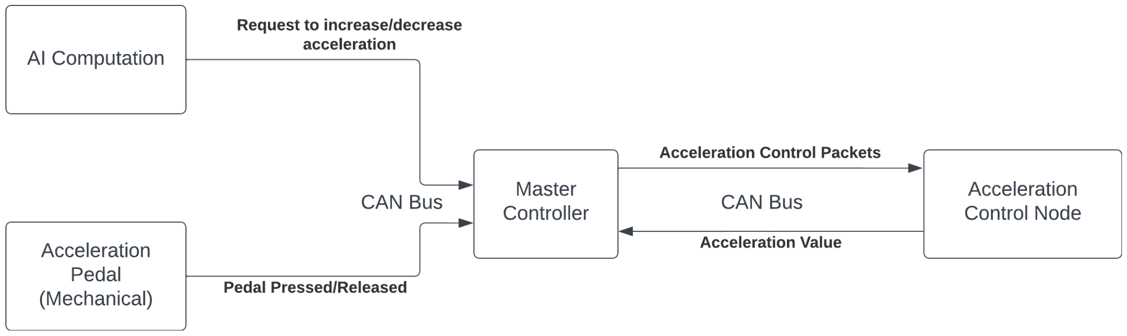

AI Computation
Request to increase/decrease acceleration
Acceleration Pedal (Mechanical)
Pedal Pressed/Released
CAN Bus
Master Controller
Acceleration Control Packets
CAN Bus
Acceleration Value
Acceleration Control Node

Figure 5.2: Acceleration Node connected to CAN Bus

### 5.2.3 Brake Node

This node is responsible for applying the brake of the vehicle. it operates when the master controller sends the message packets requesting to apply brake or release the brake. When it receives the signal from the master controller, it then sends signal to the actuator which applies physical brake. This node sends the feedback message packets to the master controller which includes the amount of brake pressed and other health of the node. When the vehicle is turned off, the brake is applied.

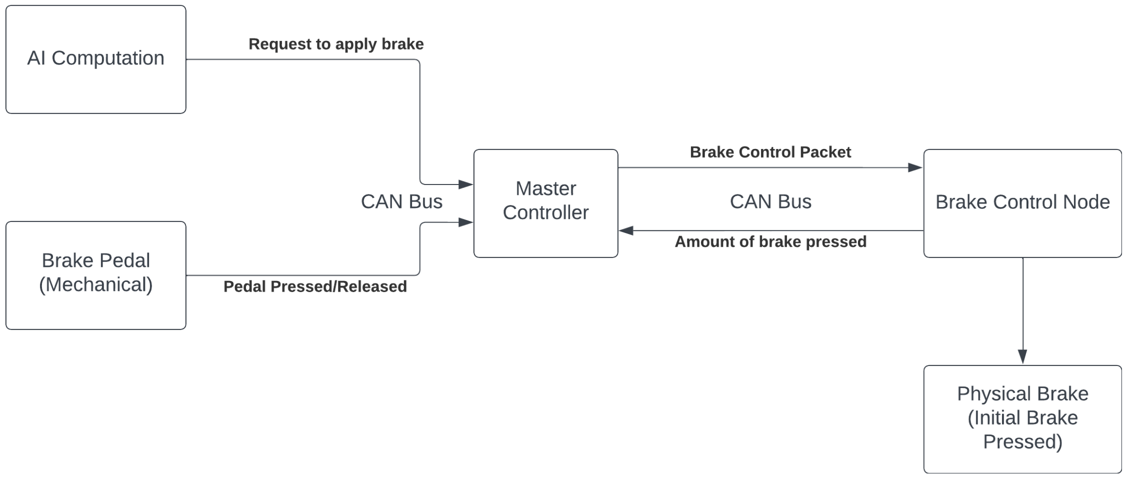


Figure 5.3: Brake Node connected to CAN Bus

### 5.2.4 Battery Management Sytem(BMS) Node

This node is responsible for charging and discharging of the battery packs of the vehicle. During charging of the vehicle this nodes balances the charge among the each cell of the packs avoiding any hazardous situation. It discharges the battery packs according to the need as requested by the master controller. This node sends the feedback message packets which includes the State of Health(SOH), State of Current(SOC) and State of Temperature(SOT) of the cells to the master controller.

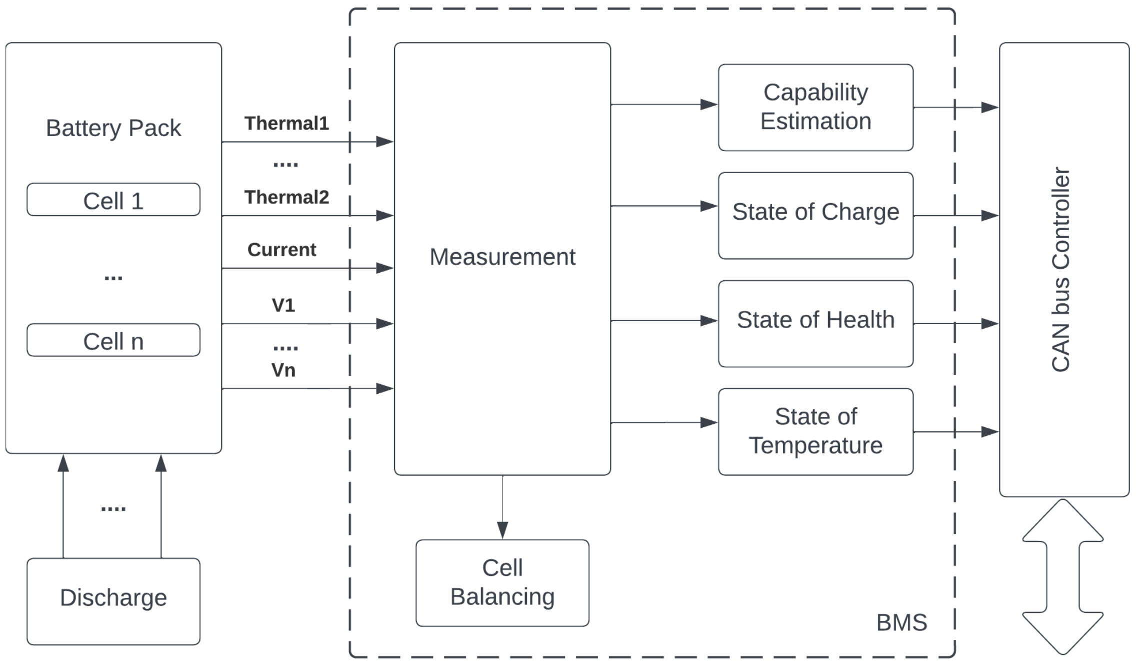
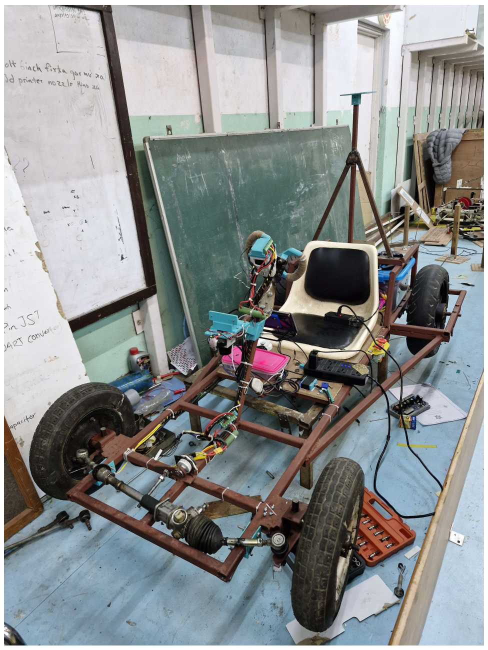


Figure 5.4: BMS Node connected to CAN Bus

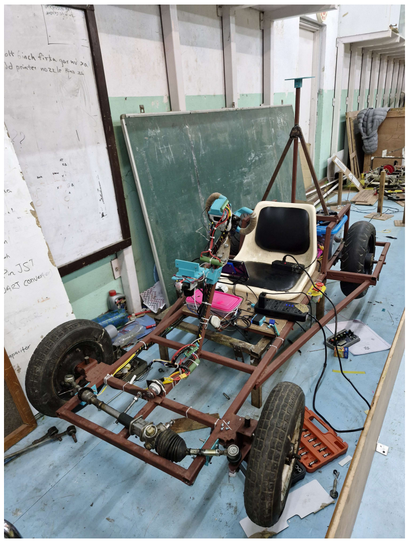
# 6. Results and Discussion

The current results of our project focus on the complete integration of our vehicle with the sensors, battery system, motor drivers connected as CAN nodes and operable from a display unit connected to the RockPi. For our battery system, we have rewired the existing 60V lithium ion battery provided to us by Niu as per our needs.

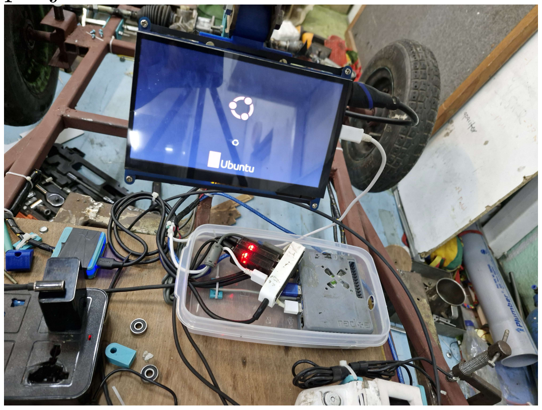


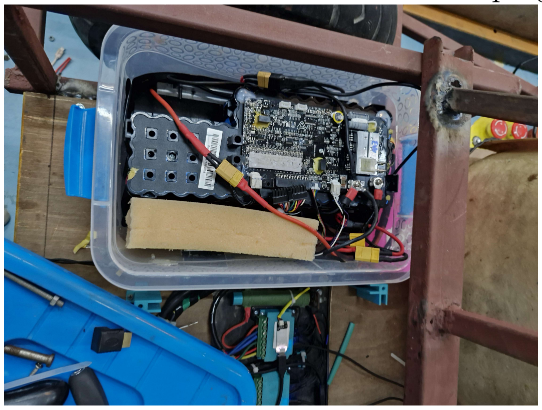
The integration of an RTK GPS system has also been completed, with the base station mounted at the Robotics Club and configured to broadcast correction data from multiple satellites.

We've diligently worked on establishing a robust vehicle communication network. Node-level testing with CAN transceivers, alongside the development of CAN wiring harnesses and message filtering, has been completed. Furthermore, inter-node communication has been successfully tested through packet transmission within the system. Firmware development at the node level is complete, with a focus on GUI-based display feedback, node addressing, and API layer system design.



In parallel, efforts have been directed towards vehicle simulation and pre-assembly tasks, incorporating ML-based feedback control, chassis design refinement, wiring and cable management, early system tuning, Matlab simulations for parameter optimization, and the development of a GUI for system command input. Through these coordinated efforts, our project continues to advance towards its goals with a seamless flow of tasks and progress.





Our embedded system design has already undergone testing, focusing on micro-ros and ROS2 integration for data distribution. Our system requires field testing, which has been not possible yet due to the delay in delivery of the front motor controller. This has consequently delayed the design of our steering system. This will be followed by human-controlled vehicle operation. These include implementing SLAM-based localization around Pulchowk to ensure precise positioning, setting PID constraints for individual components to optimize performance, and conducting thorough command-based movement testing to validate functionality and responsiveness.

Different systems in the vehicles and corresponding results from those systems are discussed below.

## 6.1 CAN Layer

In CAN physical we tested the use of multiple wire available to us, our findings led us to using a 24AWG single core aluminium wire throughout the length of the car. We tested communication of devices on the bus and this offered no packet loss for 2 meter in wire length.

Our plan was to isolate each node as much as possible, so we have settled on this CAN bus layout. This provides us good robustness in the vehicle. There is always a trade-off for redundancy and cost. We cannot duplicate each node but can confirm the vehicle will not run down in failure of a single node in the bus. The overview of the CAN bus is:

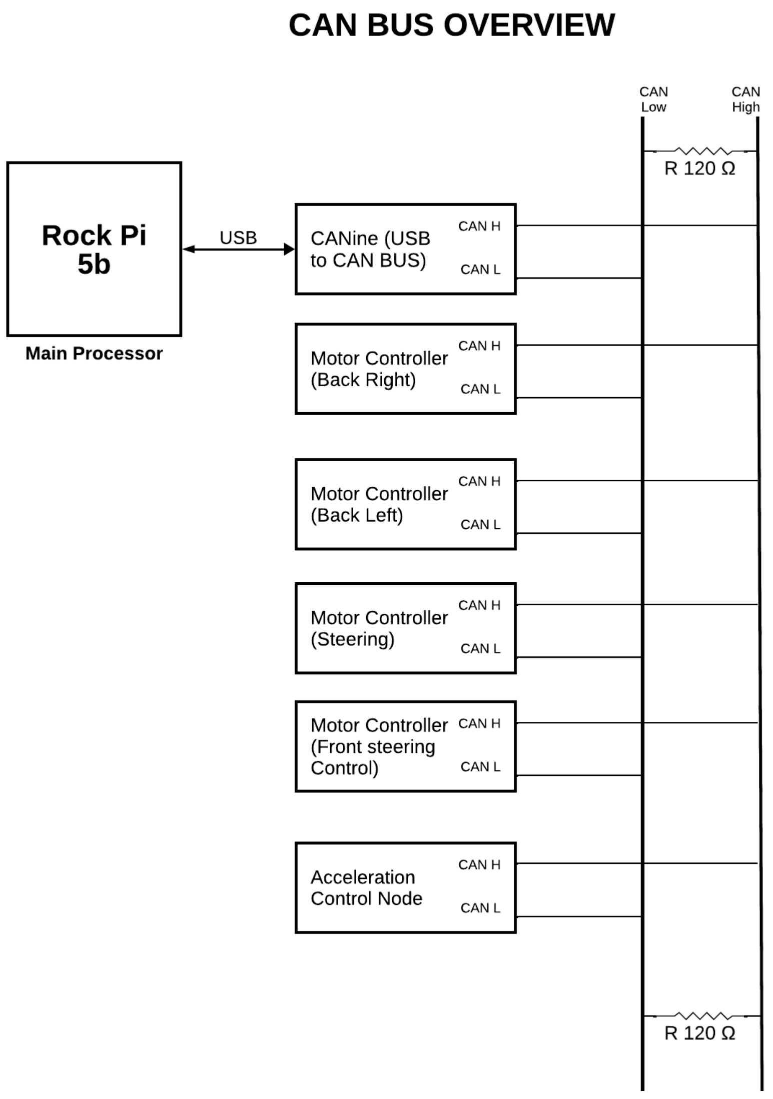

CAN BUS OVERVIEW
CAN Low
CAN High
R 120 Ω
Rock Pi 5b
Main Processor
USB
CANine (USB to CAN BUS)
CAN H
CAN L
Motor Controller (Back Right)
Motor Controller (Back Left)
Motor Controller (Steering)
Motor Controller (Front steering Control)
Acceleration Control Node
R 120 Ω

Figure 6.1: CAN bus overview

## 6.2 Sensor Layer

Our system is not an end-to-end solution but a platform for other applications integrate with a robust base such as ours. We have integrated multiple sensors in our design for navigation as well as obstacle detection. The different sensors used in the system and the outputs obtained from them is mentioned below.

### 6.2.1 LiDAR

The laser scan output of a LiDAR (Light Detection and Ranging) system typically consists of a collection of data points representing distances measured by the LiDAR sensor. Each data point, often referred to as a ”point cloud,” includes information about the distance from the LiDAR sensor to a particular object or surface in its field of view. Some of the important data in the point cloud are:

1. Coordinates: Each data point in the point cloud is represented by its spatial coordinates (x, y, z), indicating its position relative to the LiDAR sensor's reference frame.

2. Distance: The distance measured by the LiDAR sensor to the object or surface corresponding to each data point. This distance information is typically represented in meters or another unit of length.

3. Intensity: Some LiDAR systems also provide information about the intensity of the returned laser pulse for each data point. This intensity value can be used to infer properties of the reflecting surface, such as its reflectivity or material properties.

4. Scan Angle: The angle at which the laser beam was emitted and received by the LiDAR sensor. This information helps in understanding the orientation of the detected objects relative to the LiDAR sensor.

### 6.2.2 GPS

The RTK (Real Time Kinematics) GPS we used provided us the latitude, longitude and altitude of any point around the campus with few centimeter level precision. It used the output of both base station and mobile GPS system to calculate the precise coordinate of any point on the campus.

### 6.2.3 Motor Feedback

The Hall Sensor is used to track the position and calculate the speed of the Hub Motors. The three channel Hall Sensor tracks the 54 Pole pairs of the Hub motors and provided us with the feedback with which the motor driver calculated the velocity and tracked the position of the rotating motor.

### 6.2.4 Depth Camera

Like most depth cameras, the OAK-D depth camera that we are using provides a depth map, which is a 2D array where each pixel corresponds to a distance value from the camera to the corresponding point in the scene.

## 6.3 Motor Control and Tuning

The Hub motor we used had no available datasheet so it had to be manually tuned. With the available interface of Odrive Motor driver, we performed following steps.

- Set vel_integrator_gain gain to 0
- Make sure you have a stable system. If it is not, decrease all gains until you have one.
- Increase vel_gain by around 30
- Back down vel_gain to 50
- Increase pos_gain by around 30
- Back down pos_gain until you do not have overshoot anymore.
- The integrator can be set to $0.5 * bandwidth * vel\_gain$, where bandwidth is the overall resulting tracking bandwidth of your system. Say your tuning made it track commands with a settling time of 100ms (the time from when the setpoint changes to when the system arrives at the new setpoint); this means the bandwidth was $1/100ms = 100Hz$. In this case you should set $vel\_integrator\_gain = 0.5 * 10 * < vel\_gain >$

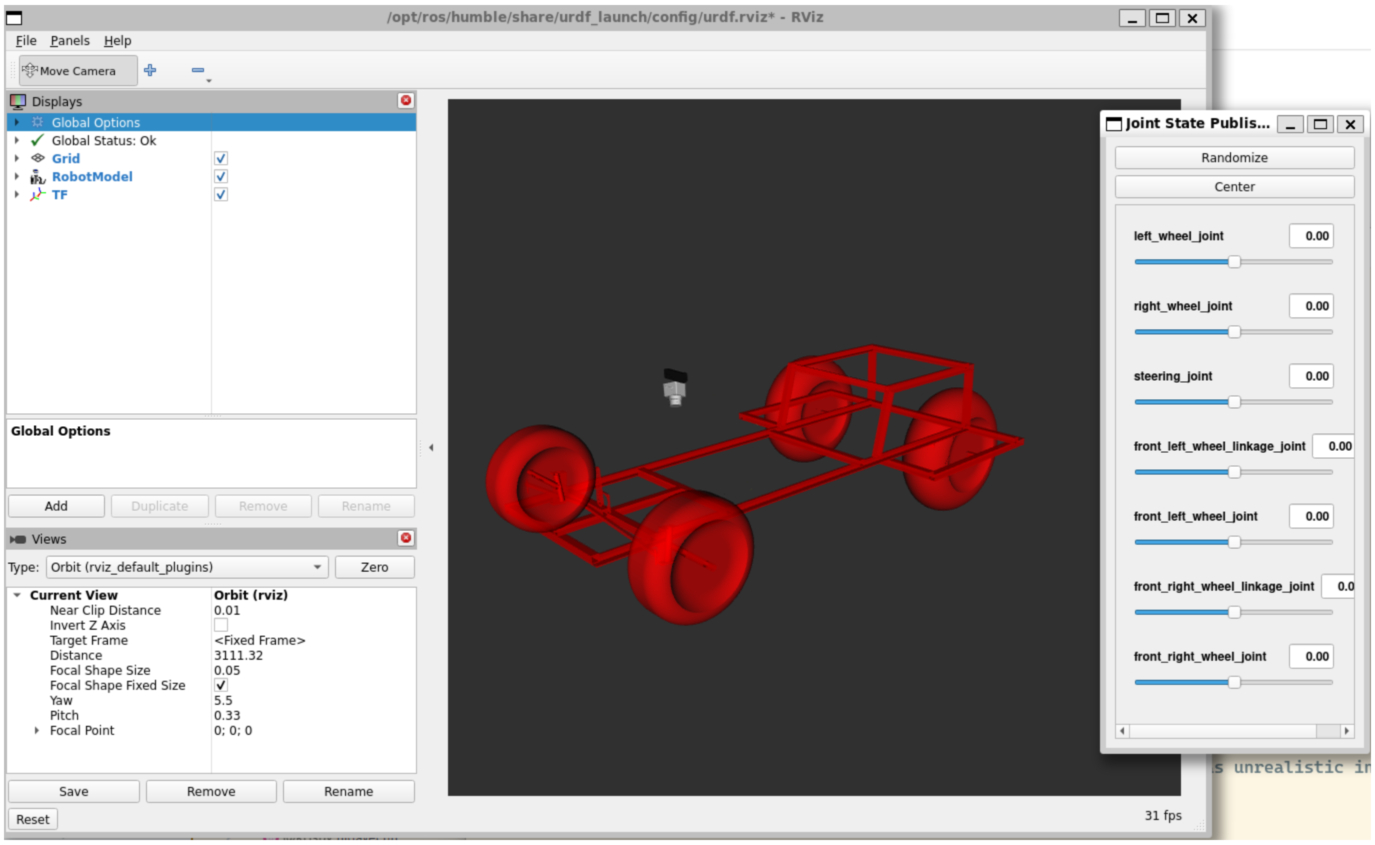
The end value for our motors were.
**vel_gain=0.52**
**vel_integrator_gain=0.31**

## 6.4 Data Distribution Layer - ROS2

The default fast-DDS implementation in ROS 2 ensures interoperability, reliability, and performance.

### 6.4.1 Software Description of Vehicle

To visualize our vehicle in the simulation environment we used the URDF format and load it into the Rviz and Gazebo. URDF (Unified Robot Description Format) is an XML-based file format used in ROS (Robot Operating System) to describe the physical properties and structure of a robot. URDF files define the robot's kinematics, dynamics, visual appearance, and collision properties, among other aspects. The final design and view of the vehicle in simulation environment is



Type: Orbit (rviz_default_plugins)

```
Current View                Orbit (rviz)
  Near Clip Distance        0.01
  Invert Z Axis
  Target Frame              <Fixed Frame>
  Distance                  3111.32
  Focal Shape Size          0.05
  Focal Shape Fixed Size    ✓
  Yaw                       5.5
  Pitch                     0.33
  Focal Point               0; 0; 0
```

```
left_wheel_joint                0.00
right_wheel_joint               0.00
steering_joint                  0.00
front_left_wheel_linkage_joint  0.00
front_left_wheel_joint          0.00
front_right_wheel_linkage_joint 0.0
front_right_wheel_joint         0.00
```

Figure 6.2: Software Description of the Vehicle

## 6.5 ROS Nodes

The nodes in the ROS system are:

- joint_state_controller: This node publishes joint states, such as positions, velocities, and efforts, to the ROS graph. It typically subscribes to feedback from the robot's sensors (e.g., encoders) and publishes joint state information that can be used by other nodes for various purposes, such as visualization or higher-level control.

- cmd_vel:cmd_vel is used to send velocity commands to a robot's base. The message published to the /cmd_vel topic are of type geometry_msgs/Twist. This message type contains two main components: linear velocity (linear.x, linear.y, linear.z) and angular velocity (angular.x, angular.y, angular.z). These components define the desired linear and angular velocity of the robot.

## 6.6 ROS Topics

ROS messages are transmitted through ROS Topics between the different nodes. The ROS topics in our system are:

- cmd_vel: This topic is used to send velocity commands to control the robot's motion. It typically publishes messages of type geometry_msgs/Twist, specifying linear and angular velocities.

- odom: The odometry topic provides estimated pose and velocity information about the robot's motion. It typically publishes messages of type nav_msgs/Odometry, containing information such as position, orientation, linear velocity, and angular velocity.

- joint_states: This topic provides information about the robot's joint states, such as positions and velocities of its wheels or other movable parts. It typically publishes messages of type sensor_msgs/JointState.

- scan or lidar_scan: If the robot is equipped with a lidar sensor for distance measurement, it may publish laser scan data on this topic. It typically publishes messages of type sensor_msgs/LaserScan, containing distance measurements at various angles.

- tf: The tf topic publishes the transform tree, which represents the spatial relationship between different coordinate frames in the robot's environment. This is essential for coordinating sensor data and robot motion in a consistent reference frame.

- battery_state: If the robot has a battery or power supply, it may publish information about its state, such as voltage, current, and remaining capacity. This data is typically published on a topic like sensor_msgs/BatteryState.

- diagnostics: This topic may be used to publish diagnostic information about the robot's health and status. It can include data such as error messages, warnings, or other diagnostic information to facilitate troubleshooting and maintenance.

# References

[1] M. Martínez-Díaz and F. Soriguera, “Autonomous vehicles: Theoretical and practical challenges,” *Transportation Research Procedia*, vol. 33, pp. 275–282, 2018, XIII Conference on Transport Engineering, CIT2018, ISSN: 2352-1465. DOI: `https://doi.org/10.1016/j.trpro.2018.10.103`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S2352146518302606`.

[2] W. Schwarting, J. Alonso-Mora, and D. Rus, “Planning and decision-making for autonomous vehicles,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018. DOI: `10.1146/annurev-control-060117-105157`. [Online]. Available: `https://doi.org/10.1146/annurev-control-060117-105157`.

[3] M. Silverio, *How to make a vehicle, autonomous.* Dec. 2019. [Online]. Available: `https://towardsdatascience.com/how-to-make-a-vehicle-autonomous-16edf164c30f`.

[4] W. Zong, C. Zhang, Z. Wang, J. Zhu, and Q. Chen, “Architecture design and implementation of an autonomous vehicle,” *IEEE Access*, vol. 6, pp. 21 956–21 970, 2018. DOI: `10.1109/ACCESS.2018.2828260`.

[5] K. Manohar, *PID tuning using Machine Learning — maohar502*, `https://medium.com/@maohar502/pid-tuning-using-machine-learning-6cf6f7fe5690`, [Accessed 13-Jun-2023].