Autonomous UGV for Off-Road Navigation

Muhammad Abdullah

May 2025

Contents

List of Figures

List of Tables

Abstract

This thesis presents the design and development of an autonomous Unmanned Ground Vehicle (UGV) using a full-scale 110cc All-Terrain Vehicle (ATV) platform. The objective is to implement point-to-point autonomous navigation through a combination of mechanical, electrical, and software integration. A simulation-first approach was adopted using Gazebo, PX4, and ROS2 to validate control algorithms and sensor integration in a virtual environment. The Pure Pursuit algorithm was implemented for path tracking and validated in MATLAB with consideration for throttle nonlinearity and dynamic vehicle constraints.

The hardware setup includes the integration of Pixhawk 5X for low-level control, Arduino Nano for PWM conversion, and ROS2 running on an embedded computer for high-level planning. Key systems such as steering, throttle, and braking were automated using servos and an Electric Power Steering (EPS) module. Real-time GPS-based navigation was achieved using the Here 3 module, with commands relayed through QGroundControl and tested in both simulation and field trials.

Results demonstrate that the autonomous ATV can effectively follow predefined GPS waypoints, navigate off-road terrain, and adapt to control challenges posed by internal combustion dynamics. The project establishes a modular and scalable framework that serves as a foundation for future extensions involving SLAM, obstacle avoidance, and advanced payload integration.

Chapter 1

INTRODUCTION

1.1 BACKGROUND

Unmanned Ground Vehicles (UGVs) are becoming increasingly prominent in civilian applications, where they offer significant potential for automating complex tasks and enhancing safety in hazardous environments. These autonomous systems are designed for ground-based operations and are equipped with a wide array of sensors tailored for specific applications. The versatility of UGVs extends to applications in agriculture, where they can perform tasks such as plowing, seeding, fertilizing, and even removing snow in regions like Canada. In addition, UGVs are gaining traction in areas such as logistics, infrastructure maintenance, and industrial automation, reflecting the growing demand for these systems in multiple sectors. While some effort has been made to satisfy this demand, there still exists a huge gap for a generalized framework capable of simulation, path planning and autonomous navigation capabilities. This project aims to contribute to the field of UGV development by building a foundational software platform that supports autonomous navigation and simulation capabilities. In addition, the results of the framework are extensively tested on an all-terrain vehicle (ATV).

1.2 OBJECTIVES

This thesis project focuses on addressing the initial and critical need for a comprehensive platform that will serve three key purposes:

- Simulation of Vehicle and Environment: To create a virtual testing environment for simulating the UGV's behavior in various scenarios, facilitating early-stage development and validation of autonomous point-to-point navigation.
- Automate the ATV: To convert the All terrain vehicle (ATV) into an unmanned

ground vehicle through mechanical and electrical modifications.

  - Steering Automation: To automate the Steering, shifting from mechanical torque to control using a stepper motor and power steering.
  - Throttle Control : Automating Throttle Control using stepper motor
  - Brake Control: Automation of brake from spring based to control using stepper motor

- Testing of UGV's Low-Level Controller: To develop a full scale prototype on an All-Terrain Vehicle, that enables the testing of UGVs for waypoint navigation, ensuring that the vehicle follows the given trajectory efficiently.

Integrating PX4 with ROS 2 for off-road navigation represents a significant advancement in the field. By addressing the limitations of classical approaches through real-time performance, robust simulation, flexible architecture, and enhanced communication, this system is designed to navigate challenging terrains autonomously.

Through this project, we aim to develop a robust software platform for autonomous vehicle navigation using ROS2 and PX4 at its core. This platform will serve as a base for future work, such as integrating application-specific configurations like robotic arms, pesticide spraying systems, UAV launch pads, or agricultural tools for plowing, seeding, fertilizing, and snow removal.

Chapter 2

LITERATURE REVIEW

With rising demand in vehicle automation and autonomous capabilities, increasing amount of sensors and actuators are added resulting in an increase in vehicle complexity.[15] discusses the challenges and problems associated with integration of PX4 and Ros2. It highlights the importance of using ROS2 for high level and Pixhwak for low level control for autonomous navigation. They further proposed AUTOSTAR method for service-based communication over Ethernel, to better address the bandwidth limitations faced by communication over the CAN protocol. This is critical since huge amounts of data is often transmitted from sensors and bandwidth becomes a huge limit.

[24], performs a extensive evaluation comparing different control algorithms and their result in different terrains. It argues that while each has their own pros and cons, Pure Pursuit algorithm in average performs better than the rest and hence is the algorithm being adopted in this project. However, [2] places an emphasis on the limitations of the pure pursuit algorithm in cases of high-speed and sharp-turn scenarios, especially in cases of larger vehicles which might prioritize stability over efficiency. They further propose Model Predictive Control (MPC) along with dynamic curvature planning for more stability. This is further explored by [13] which considers scenarios of aggressive, high-speed driving especially in off-road scenarios where terrain is uneven. This places an emphasis on the importance of terrain in control algorithms ensuring safety and maneuverability.

In regard to steering automation, Coulter [5] introduced pure pursuit with Ackermann Steering. This laid down the foundation of what was adopted on several vehicles which participated in the DARPA challenge, with the vehicle winning podium positions.

For mechanical aspects of UGVs, [3] and [23] investigate challenges in ATV steering systems, particularly thermal issues in servos. They suggest using more robust PID based control along with larger motors to ensure efficient operation along

with higher heat resistance this has been adopted in this thesis project as well. [23], further discusses the automation of an ATV which has been partially adopted in this project. [17] builds up on the classical pure pursuit algorithm and introduces regulated pure pursuit, this enhances the traditional algorithm ensuring more stability for more constrained or offroad activities. To address obstacle avoidance and path planning under uncertain terrain conditions, RL-based methods are proposed in [16] and [10]. Similarly, [26] suggest combining Gaussian Processes with MPC for better adaptation to off-road terrains.

Chapter 3

PROBLEM DEFINITION

UGV developed traditionally are highly specific for operations and are often designed to work on reduced scale prototype without considering actual vehicle. These frameworks in some cases lack a simulation model as well as are highly specific to an operation, while real application require flexibility in the mission. The goal is to create a unified, human-in-the-loop mission system that is capable of interfacing with other autonomous vehicles and can support different payload. Furthermore, rather than developing a scaled prototype, the objective is to develop a final prototype that can in future be pitched to investors.

Chapter 4

METHODOLOGY

The methodology followed in this project is divided into distinct phases, each building upon the previous one to progressively achieve full autonomous capability.

4.0.1 Phase 1: Simulation in Gazebo

Objective: Simulate the UGV's off-road navigation capabilities in the Gazebo environment.

Approach: PX4 Software-In-The-Loop (SITL) simulation is used alongside ROS2 nodes to evaluate navigation strategies and obstacle detection. Virtual sensors such as LiDAR, GPS, IMU, and depth cameras are tested for environmental mapping and localization.

Expected Output: A robust simulation of the UGV navigating through off-road terrain, suitable for validating the path planner and control logic.

4.0.2 Phase 2: Remote Control Testing

Objective: Establish baseline remote-controlled navigation.

Approach: A remote-controlled vehicle is acquired and tested manually. Servo calibration is conducted to tune the turning radius and responsiveness. Electrical and mechanical systems are validated for reliable operation under manual control.

Expected Output: Reliable manual control of the vehicle, confirming readiness for autonomous system integration.

4.0.3 Phase 3: Low-Level Controller Implementation

Objective: Automate the vehicle's movement using onboard sensors and controllers.

Approach: Integration of the Pixhawk controller with GPS and IMU is performed. Configuration and tuning via QGroundControl (QGC) enables autonomous point-to-point navigation based on internal waypoints.

Expected Output: An autonomous vehicle capable of following GPS-based waypoints with low-level control handled by the Pixhawk.

4.0.4 Phase 4: ROS2-Based Trajectory Following

Objective: Enable trajectory following and dynamic behavior using ROS2.

Approach: ROS2 is deployed on an onboard embedded computer (e.g., Raspberry Pi) to manage high-level trajectory planning. Sensor data is processed in real time to adjust the path dynamically. Commands are sent to the Pixhawk through MAVROS or similar middleware.

Expected Output: A semi-autonomous UGV capable of following trajectories based on GPS and sensor inputs.

Note on Phases 5 and 6

The following phases are beyond the scope of the current Final Year Project. They are outlined here for completeness.

Phase 5: Obstacle Avoidance

Objective: Equip the UGV with object detection and avoidance capabilities.

Approach: Implementation of SLAM (Simultaneous Localization and Mapping) and integration of object detection algorithms such as YOLO to allow real-time obstacle awareness and path replanning.

Expected Output: A vehicle capable of autonomous navigation in complex environments, dynamically avoiding static and moving obstacles.

Phase 6: Feature Expansion for Surveillance

Objective: Enable the UGV to patrol and monitor bounded areas for surveillance applications.

Approach: Development of mission-level logic to allow autonomous patrolling of predefined regions, including GPS bounding box traversal and sensor payload management.

Expected Output: A fully autonomous surveillance-capable vehicle, capable of executing long-range missions within a specified operational area.

Chapter 5

DETAILED SYSTEM DESIGN

5.1 SIMULATION

Simulation of the vehicle plays a critical role in the design, development, testing and validation of the control system of the vehicle. The simulation environment replicates real world environment allowing testing of the framework before deployment on the actual interface. Furthermore, with software in loop (SITL) mission allowing for cross platform development, leading to the design of more complex functionalities. The simulation used involves the simulated environment in Gazebo, with control supported by PX4 and ROS2.

5.1.1 Gazebo

The choice of Gazebo [14] as a simulation platform was due to its features. Gazebo supports several functionalities:

- Cross Platform Support such as Linux and MacOS: This allows for deployment and testing across different Operating Systems (OS)
- Cloud Integration allowing for remote deployment and hosting simulation worlds on the Cloud hosted server.
- Integration with ROS: Robotics Operating System (ROS)[18] is one of the leading frameworks for robotic control, including navigation, sensor Input/Output, joint control. By supporting ROS, Gazebo extends its capabilities to the simulation domain allowing the scripts developed using ROS for any robotic project to be validated on the Gazebo Simulation world.
- Sensor and Noise Model: Gazebo supports several sensors that are often used in such applications. Supported sensors include: Monocular Camera, Depth

Camera, LIDAR, IMU, contact, altimeter and magnometer. These sensors allow for additional functionality both in simulation as well as deployment using ROS.

- 3D Graphics: Gazebo supports a rich variety of high definition graphics, for the environment, physical assets and vehicle.
- Precise Physics: The best feature about Gazebo is its real world environment, and the physics behind it. Gazebo takes in consideration several factors including forces such as gravity, friction, contact force etc. In addition, through camera visual information can be transmitted from the gazebo world to the Ros script. This allows good real world replication and hence is used as a suitable model to test software before deployment.

Through Gazebo the aim is to:

- Validate the control algorithms including steering, navigation and point to point control
- Testing sensor integration including IMU
- Debugging PX4-ROS2 communication instead of directly deploying on hardware and risking it
- Simulate different environment condition and cross check software's performance in different environments
- Calibrate accuracy and precision in way point navigation

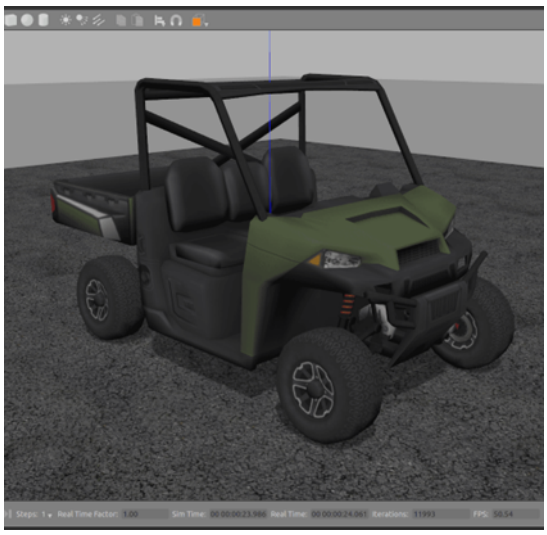

Figure 5.1: Gazebo simulation environment showcasing the UGV and sensor models

5.1.2 PX4

PX4 [8] is an open-source autopilot developed for unmanned vehicles. It is maintained by the drone code foundation and has a huge open-source community leading to diverse and robust, development and testing. This open-source framework provides a range of tools and features that can be used to develop autonomous capabilities in the vehicle. It further supports capabilities such as control of actuators and servos, path planning, perception etc. PX4 supports multiple vehicle types such as drones, UGV, USV with sensor fusion, mission planning and fail-safe feature such as Return to Launch as well.

Figure 5.2: PX4 logo[8]

Key Features of PX4 that differentiate it from other controllers include:

- Open Source: Being Open Source and Modular Pixhawk allows developers to easily customize and extend the framework according to their specific project and need. In addition, having a large open-source community allows for cross-institute collaboration as well as discussion to resolve issues and develop unified global solutions.
- Real time performance: PX4 is built to run on embedded devices operating Linux, hence supports real time performance
- Parametric System: PX4 supports a wide range of parameters that the developers can toggle and manipulate leading to a vast range of control and custom mission development. For instance, through the way_point_rad parameter, allows control over accuracy in reaching the waypoint. This is an essential parameter that varies based on the mission or the vehicle. For example, a Fixed wing UAV would have a larger way_point_rad value as it is not necessary for it to reach the waypoint exactly, however in the case of a quad copter this matters a lot and accuracy is critical for the mission. These parameters can be controlled using Q Ground Control (QGC) [9].
- Simulation Support: PX4 provides built-in Software in the loop (SITL) and hardware in the loop control (HITL). SITL allows PX4 control in the simulation environment such as gazebo while hardware in the loop control allows for real life deployment and testing of PX4 commands

5.1.3 ROS2

ROS 2 [18] is a famous middle ware framework for building modular and scalable robotics software. It incorporates real-time capabilities as well as allows for high-level control. Furthermore, it allows the integration of high-level elements such as perception and LIDAR and allows for more diverse missions to be executed on the robot. In addition, ROS 2 also supports multi-robot and distributed systems.

5.1.4 Key Features of ROS 2:

- It uses the **Data Distribution Service (DDS)** for real-time, decentralized communication using ORB. This allows multiple clients to join DDS and take advantage of the system by taking communication data as input and transferring

data as well.

- **Node Based Architecture**: ROS 2, like Pixhawk, also follows a modular structure where functionality is dependent on Topics and Nodes. Through each node, topics or sensor data can be transferred or received. Each new client can choose to subscribe or publish a node. With subscription, data can be obtained from the node, while through publication, data are transferred to the node as can be seen in Fig. 5.3

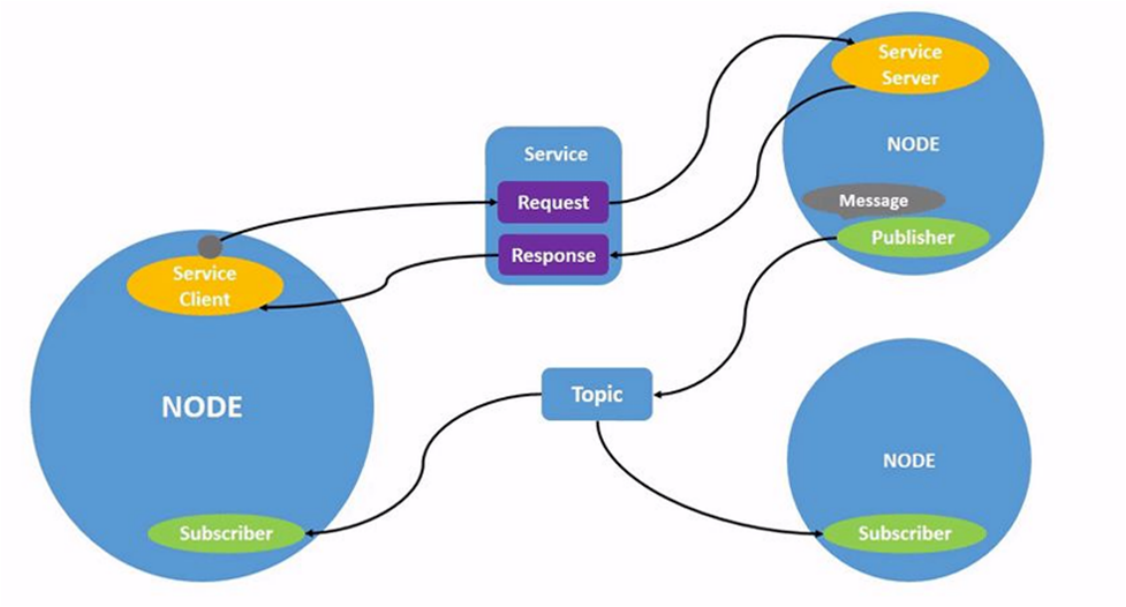

Figure 5.3: Ros working layout[22]

- **Real Time Capabilities**: ROS 2 supports real-time performance, which is critical for control purposes.
- **Cross Platform Support and Multi Language Support**: Supports C++, Python, and others in Linux, Windows, and macOS.

ROS 2 acts as a high-level controller for the UGV, making it more autonomous. In addition, it supports path planning and obstacle avoidance capabilities which can be integrated in the future.

Integration of Ros2 and PX4

Deep integration between ROS 2 and PX4 is made possible by the ROS2-PX4 architecture, that allows direct communication between ROS 2 subscribers or publisher

nodes and PX4 uORB topics.

Comprising a client operating on PX4 and an agent running on the companion computer, the uXRCE-DDS middleware facilitates bi-directional data exchange between them over a serial, UDP, TCP or bespoke link. In order to publish and subscribe to subjects in the global DDS data space, the agent serves as a stand-in for the client. In order to enable uORB messages to be broadcast and subscribed to on a companion computer as if they were ROS 2 topics, PX4 employs uXRCE-DDS middleware. This makes it considerably simpler for ROS 2 apps to retrieve vehicle information and issue commands, and it offers a quick and dependable interface between PX4 and ROS 2.

PX4 makes use of eProsima Micro XRCE-DDS in its XRCE-DDS implementation. The architecture and several choices for configuring the client and agent are explained in the tutorial that follows. It specifically discusses the settings that PX4 users care about the most.

A client running on PX4 and an agent running on the accompanying PC make up the uXRCE-DDS middleware, which allows bi-directional data exchange via a serial or UDP link. The client can publish and subscribe to subjects in the global DDS data space by using the agent as a proxy.

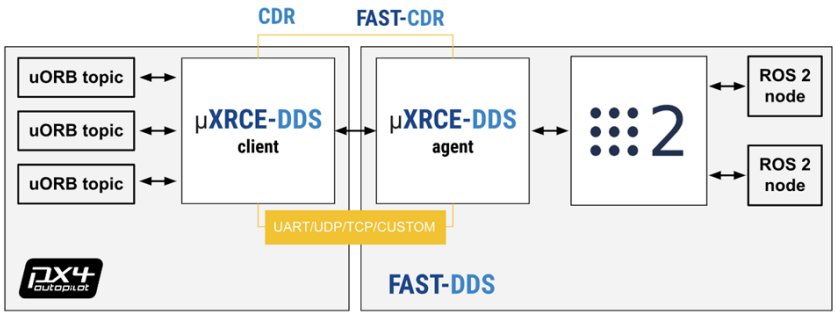

Figure 5.4: Ros2 and PX4 interface[19]

The uXRCE-DDS client must be installed on PX4 and linked to the companion's micro XRCE-DDS agent in order for PX4 uORB topics to be exchanged on the DDS network. A specified set of uORB topics are published to and from the global DDS data space via the PX4 uxrce dds client. In the DDS/ROS 2 network, the companion PC runs the eProsima mini XRCE-DDS agent, which serves as a client proxy.

The agent itself may be produced and/or installed without relying on PX4 or

ROS, and it is not dependent on client-side code. Client-side code is necessary for programming that want to subscribe or publish to PX4; it needs uORB message definitions that correspond to those used to build the PX4 uXRCE-DDS client in order to interpret the messages.

5.2 SYSTEM ARCHITECTURE

The overall system architecture of the autonomous ATV is designed to ensure modular separation between perception, planning, and control. Figure 5.5 shows the high-level block diagram of the system, comprising input sensors, processing components, control interfaces, and actuators.

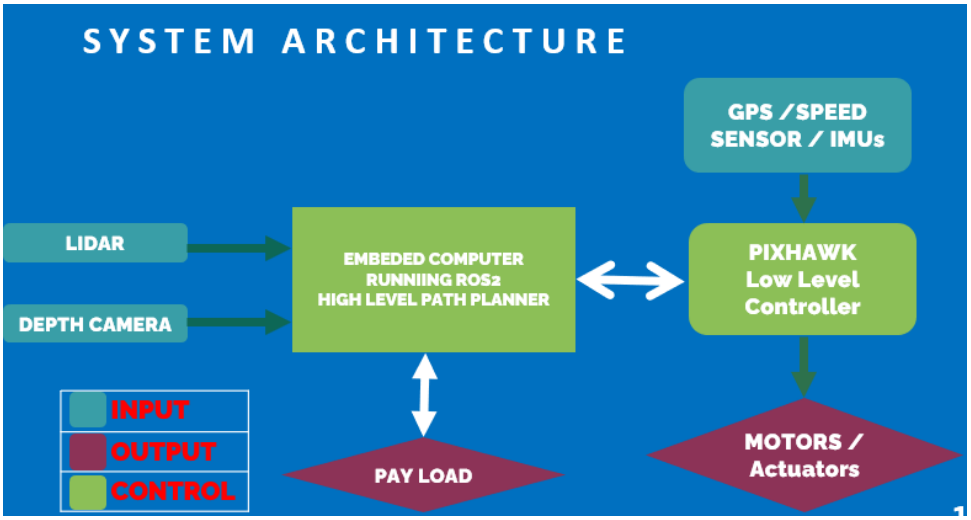

Figure 5.5: System architecture of the autonomous ATV showing perception, planning, and control modules.

Input and Perception

Sensor inputs include a LiDAR and a depth camera. These sensors provide 3D environmental information that is processed by the onboard embedded computer. Additional inputs such as GPS, speed sensors, and IMUs are connected to the Pixhawk flight controller to assist in localization and low-level state estimation.

Embedded Computer and High-Level Planning

The core of the autonomy stack is an embedded computer running ROS2, responsible for executing the high-level path planning algorithms. This system processes LiDAR and depth camera data to generate a navigable trajectory, and sends target motion commands to the Pixhawk. It also interfaces with and manages the payload system as required.

Low-Level Control via Pixhawk

The Pixhawk flight controller runs PX4 firmware and acts as the low-level controller. It receives commands from the embedded computer and translates them into motor and actuator-level signals for steering and throttle. The Pixhawk also integrates data from GPS, IMU, and wheel encoders to perform low-level stabilization and feedback control.

Actuation

Based on the processed control signals, the motors and actuators drive the physical ATV. This includes Ackermann steering and throttle/brake control using actuators interfaced with the Pixhawk.

5.3 MECHANICAL DESIGN

5.3.1 Body Design

Metal Body

Fig 5.6, shows the complete final CAD design of the vehicle. The body is made up of stainless steel, welded together. The body is connected to the vehicle chasis with nut bolt mechanical joints.

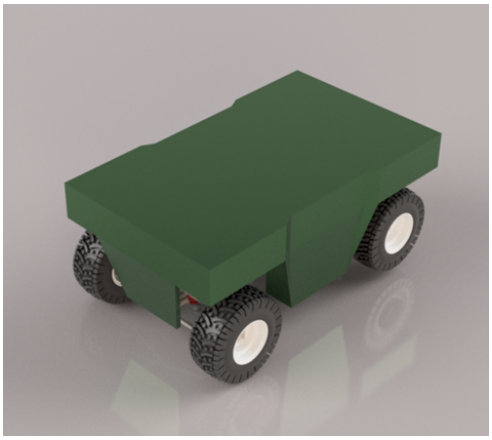

Figure 5.6: UGV Body

Fig 5.7, shows the Top view of the metal body, which will be mounted on top of the vehicle. The top is a flat structure, which is fastened to the vehicle's chassis through nut and bolt joint. The top of the metal body has double panel French door allowing for ease of access to the components inside.

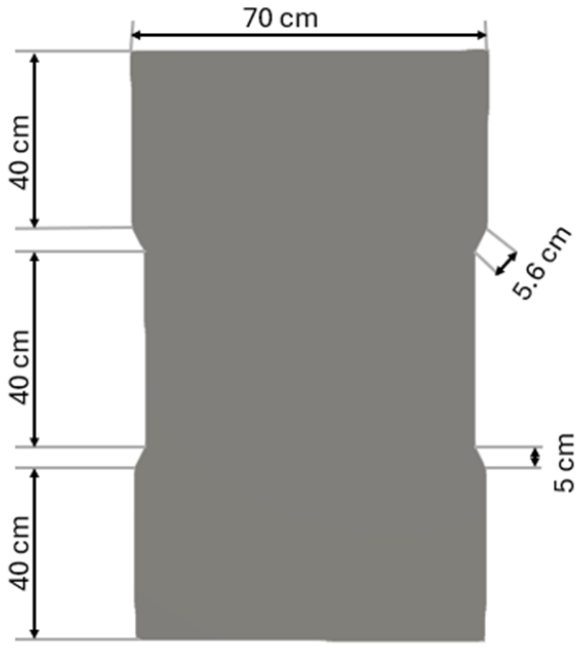

Figure 5.7: Top view of mechanical body

Fig 5.8,shows the side view of the same metal body. The entire body is made up of Stainless Steel to ensure robustness.

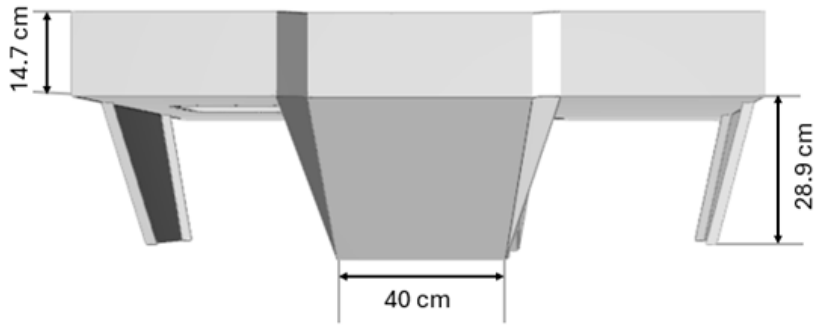

Figure 5.8: Side view of mechanical body

Servo Mount

Fig. 5.9, shows the servo mount used. The servo mount is made up of Mild Steel. The steering column passes through circular hole (radius 1.9 cm) and is welded together ensuring rigid connection. Servo (160 Kg/s) is mounted on the rectangular cavity and is fastened through nut and bolt connection with the servo mount.

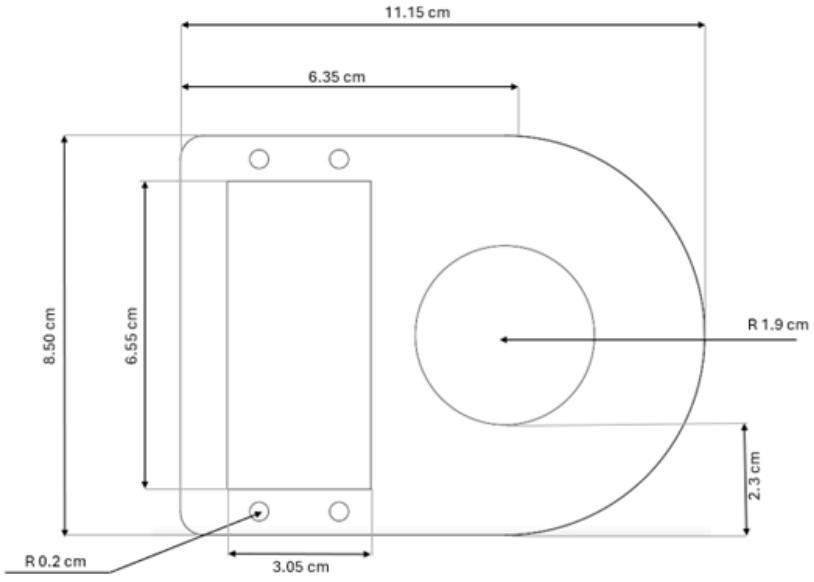

Figure 5.9: servo mount drawing

Gears

Gears were added to the vehicle to increase the amount of torque transmitted from the Servo to the Power Steering in a 1:3 Ratio. This allowed for increased torque

transfer between the two. The smaller gear is connected to the servo while the larger gear is connected to the Steering shaft, hence through this mechanical connection overall increased power is transferred from the servo to the steering column and in turn control the vehicle's direction.

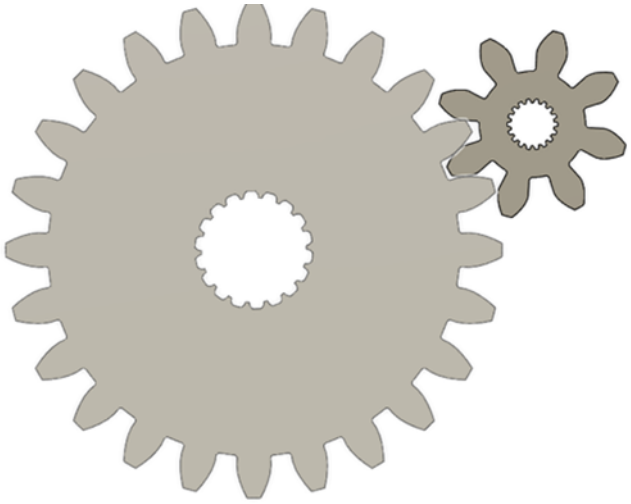

Figure 5.10: Gear connection

Servo Gear: The servo gear is the smaller gear, designed to effectively increase the torque transmitted through the gear ratio. It has 8 teeth.

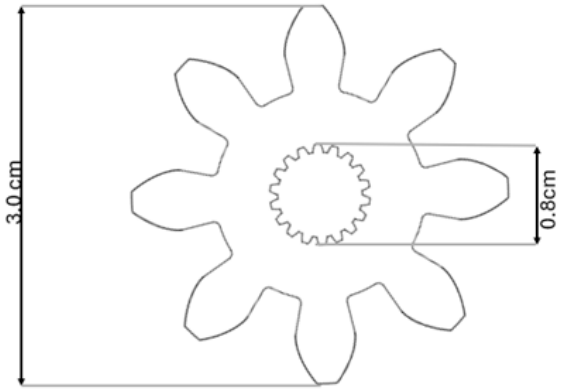

Figure 5.11: servo gear

Shaft Gear: The shaft gear is the larger gear and is welded to the Power Steering. Maintaining the 1:3 Ratio, it has 24 teeth.

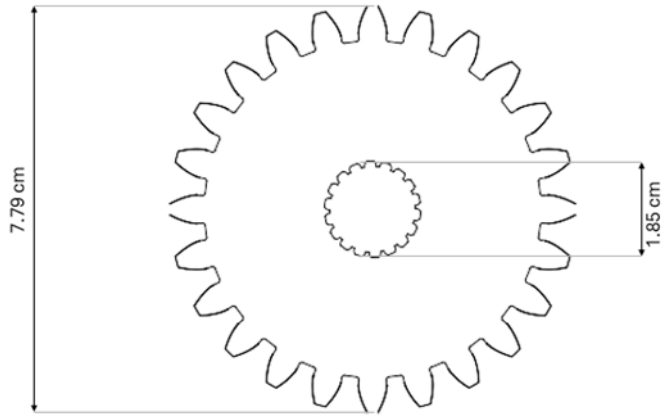

Figure 5.12: Shaft gear

Assembly of the gears on the Servo Mount Fig. 5.13, shows the assembly of the gears in Figure 2 on top of the servo mount of figure 1. This shows the relative positioning as explained earlier.

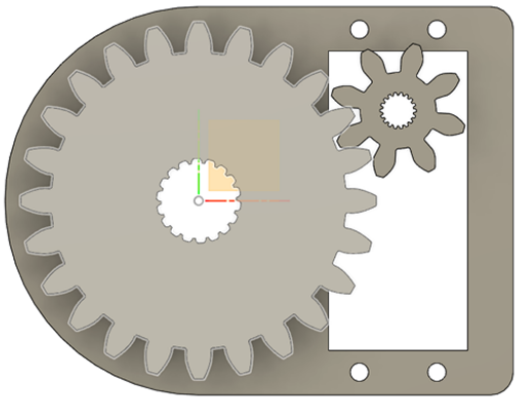

Figure 5.13: Servo Mount with gear placement

Power Steering

Power Steering: To substitute for the mechanical torque provided by the human rider and automate the ATV, power steering was added. Since the torque provided by the steeping motor is insufficient, power steering is used to reduce the required torque. This Power Steering is connected via mechanical linkage to the steering column.

Figure 5.14: Hyundai Power Steering

Hyundai Electrical Power Steering replaces traditional hydraulic power steering with electrical control mechanism enhancing fuel efficiency and reducing maintenance. The EPS System operates between 12V to 14 V and consists of a 3 phase brushless DC motor. The specifications are discussed later in Section 5.5.

The control unit in the EPS System processes signals from various sensors to determine the appropriate level of steering system. Hall effect sensor is used to sense to input torque applied allowing the system to control appropriately.

In short, flow chart. 5.15 below shows that the mechanical power is transferred from the Servo to the Mechanical Gears from which through the 1:3 Gear Ratio it increases and is transferred to the Honda Electrical Power Steering where it scale the torque and applies it on the steering column.

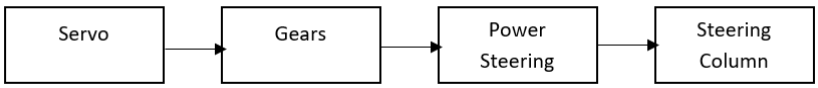

Figure 5.15: Mechanical Power Flow chart

Steering Calculations

Steering Angle: The vehicle has a steering angle of 45 degrees

$$\text{Turning Radius} = \frac{L}{\sin(\theta)} = \frac{L}{\sin(0.1444)} = \frac{0.98}{\sin(0.1444)} = 6.8\,\text{m}$$

Where:

L = wheelbase (distance between front and rear axles, in meters)

θ = steering angle (in radians)

5.4 CONTROLLER DESIGN

5.4.1 Kinematic Modeling of an Ackermann-Steered ATV

All-Terrain Vehicles (ATVs) typically employ an Ackermann steering configuration, where the front left and right wheels can steer independently to ensure proper turning geometry. The rear wheels are fixed and do not steer. To model such a vehicle, the non-holonomic constraints and turning radius differences between the inner and outer wheels must be taken into account.

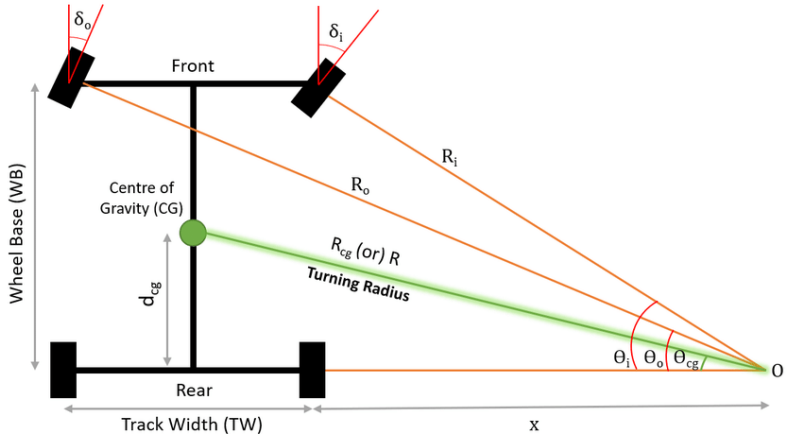

Figure 5.16: Ackermann steering geometry and inner/outer turning radii.

As illustrated in Figure 5.16, the vehicle turns about a common instantaneous center of rotation (ICR). The inner and outer steering angles δ_i and δ_o, respectively, are the angles between each front wheel and the forward x-axis, projected onto the horizontal plane.

Let:

- WB: Wheelbase (distance between front and rear axles)
- TW: Track width (distance between the left and right wheels)
- R: Turning radius from the center of gravity (CG)

The steering angles for the inner and outer front wheels are derived from simple geometric relationships:

$$\tan(\delta_i) = \frac{WB}{R - \frac{TW}{2}} \tag{5.1}$$

$$\tan(\delta_o) = \frac{WB}{R + \frac{TW}{2}} \tag{5.2}$$

This ensures that during a turn, all wheels trace circular paths that share a common center of rotation, minimizing tire slip and ensuring accurate path tracking.

To simplify control and trajectory planning, a kinematic bicycle model is often used. In this abstraction, the front wheels are replaced with a single equivalent wheel located at the midpoint of the axle. This model captures the essential turning behavior of Ackermann-steered vehicles while remaining computationally efficient.

The simplified bicycle model assumes:

- No slip conditions (ideal traction)
- All steering is concentrated at the front wheel
- The rear wheel follows the front wheel's trajectory

The state of the vehicle is given by its position and orientation (x, y, θ), and the control inputs are forward velocity v and steering angle δ. The governing equations are:

$$\dot{x} = v\cos(\theta) \tag{5.3}$$

$$\dot{y} = v\sin(\theta) \tag{5.4}$$

$$\dot{\theta} = \frac{v}{L}\tan(\delta) \tag{5.5}$$

Here, L is the wheelbase (same as WB). These equations provide a foundational model for vehicle motion, and are used in conjunction with path-following algorithms such as Pure Pursuit or Stanley controller.

This modeling approach is widely used in automated driving applications and is based on standard principles from vehicle dynamics. The derivation and assumptions follow the methodology described in [11].

5.4.2 Pure Pursuit Controller

This section explains the path following controller used. The pure pursuit algorithm [6] is used to efficiently track the given waypoints. It is a tracking algorithm that works by calculating the circular trajectory on which the vehicle will move from one waypoint to another. The controller takes a future waypoint just like humans see in front of them while driving to follow along the road. When a driver is driving a vehicle, he is basically chasing a point some distance ahead of the vehicle as chasing a point on the path some distance ahead of it - it is pursuing that moving point.

The Pure Pursuit path-following controller operates under the assumption that the reference path is represented by a sequence of 2D waypoints. The vehicle has access to this list of waypoints and is able to localize itself within a local coordinate frame.

Consider a robot R whose pose is at the origin of the local coordinate system, as shown in Figure 5.17. The position of the robot is $(0, 0)$, and its heading is ϕ along the local x-axis, which is defined as the forward direction. The local y-axis points to the right-hand side of the robot. Let (g_x, g_y) be the coordinates of a goal point on

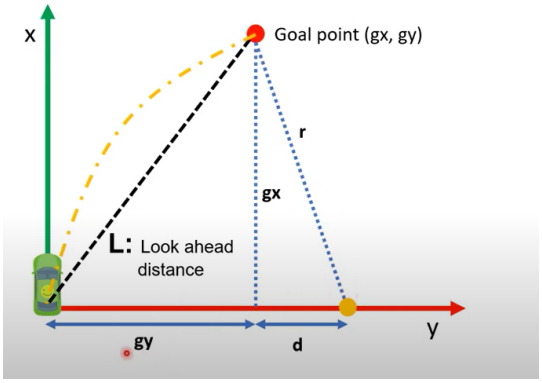

Figure 5.17: Pure Pursuit controller geometry.

the path, chosen such that it is a lookahead distance L away from the vehicle. The Pure Pursuit controller computes the curvature of a circular arc that starts from the vehicle's current location and ends at this lookahead goal point. This arc is used to guide the vehicle toward the goal.

To ensure the uniqueness of this arc, it is assumed that the center of the circular arc lies somewhere along the y-axis of the vehicle. Hence, the radius r of the arc can be computed as:

$$r = |g_y| + d \tag{5.6}$$

where d is the horizontal distance from the origin to the goal point along the x-axis, and $|g_y|$ is the vertical distance from the origin to the arc's center.

Using the Pythagorean theorem, the lookahead distance L is related to the radius r by:

$$d^2 + g_x^2 = r^2 \tag{5.7}$$

Substituting Equation 5.6 into Equation 5.7:

$$(r - |g_y|)^2 + g_x^2 = r^2 \tag{5.8}$$

Solving for r:

$$r = \frac{|g_y|^2 + g_x^2}{2|g_y|} \tag{5.9}$$

Since $L^2 = g_x^2 + g_y^2$, we can express the radius as:

$$r = \frac{L^2}{2|g_y|} \tag{5.10}$$

The curvature α is the inverse of the turning radius:

$$\alpha = \frac{1}{r} = \frac{2|g_y|}{L^2} \tag{5.11}$$

The steering angle δ is then made proportional to the curvature:

$$\delta = \tan^{-1}(L \cdot \alpha) \tag{5.12}$$

This controller results in smooth, arc-like turns toward the goal point, enabling

accurate path following.

5.4.3 Corner cutting logic

The acceptance radius of waypoints is scaled according to the angle between a line segment from the current-to-next and current-to-previous waypoints in order to facilitate a smooth trajectory. Reaching the next line segment with the heading pointing in the direction of the next waypoint would be the best trajectory. In order to do this, the rover's minimum turning circle is tangentially inscribed to both line segments as shown below.

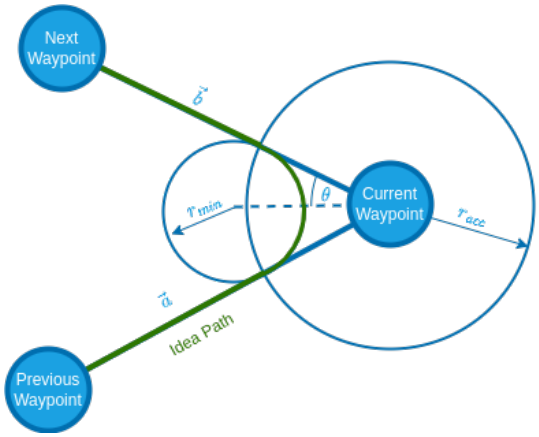

Figure 5.18: Corner Cutting logic [8]

Chapter 6

HARDWARE

This section will give deep insight into the selection and requirement of hardware components that are used in this project and how it is set up.

6.1 PIXHAWK 5X

Pixhawk 5X is a modern and upgraded flight controller developed in partnership with Holybro and the PX4 team [21]. It is a cost-effective option for both research and commercial users. The controller is built on the FMUv5X standard and includes the PX4 Autopilot system. It uses a powerful STM32F7 processor from STMicroelectronics, sensors from Bosch and InvenSense, and runs on a real-time operating system, ensuring strong performance, high flexibility, and stable control for autonomous vehicles. With 2MB of flash and 512KB of RAM, it can handle complex programming tasks and advanced models.

It has high-performance and low-noise IMUs, made especially for stable flight and control. The system includes triple redundant IMUs and dual barometers, placed on separate lines, allowing it to switch instantly if one sensor fails, keeping the flight safe. Each sensor set has its own power control, and a new isolation system reduces vibrations and noise, helping improve flight stability.

Pixhawk 5X also supports external SPI sensors with two chip select lines and high-speed data exchange. It includes an Ethernet port for faster connection with mission computers, smart battery ports using SMBus, and connectors for GPS, NFC, and other modules.

Pixhawk 5X is built with modular design and easy plug-in connections, making it simple to connect various external devices. Figure 6.1 below shows the hardware of Pixhawk 5x, while the table 6.1 below provides detailed specifications of Pixhawk 5X.

Figure 6.1: Pixhawk 5x[21]

Table 6.1: Pixhawk 5X Specifications[21]

Specification	**Electrical Data**	**Mechanical Data**
FMU Processor: STM32F765 (32-bit ARM Cortex-M7, 216 MHz)	Power module output: 4.9–5.5V	Dimensions (Controller): 38.8 x 31.8 x 14.6 mm
IO Processor: STM32F100 (32-bit ARM Cortex-M3, 24 MHz)	Max input voltage: 6V	Dimensions (Baseboard): 52.4 x 103.4 x 16.7 mm
Memory: 2MB Flash, 512KB RAM	Max current sensing: Not specified	Weight (Controller): 23g
Triple redundant IMUs & dual barometers on separate buses	USB Power Input: 4.75–5.25V	Weight (Baseboard): 51g
Advanced vibration isolation for accurate readings	Servo Rail Input: 0–36V	
External safety switch supported		
Support for high-speed Ethernet mission computer integration		
16 PWM outputs		

6.2 POWER SUPPLY MODULE: PM02D

The PM02D power module is used in our off-road autonomous navigation system to ensure a stable and reliable power supply to the Pixhawk 5X flight controller

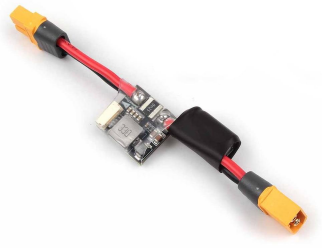

Figure 6.2: PM02D Power Moudule[20]

[20]. This module not only delivers regulated voltage to both the flight controller and electronic speed controllers (ESCs), but also facilitates real-time battery voltage and current monitoring via the I2C communication protocol.

Fully compliant with the Pixhawk FMUv5X standard, PM02D is a robust and efficient choice for high-demand vehicular systems. It is capable of handling continuous current loads of up to 60A and can support peaks of up to 120A for durations of less than 60 seconds. This capability makes it particularly well-suited to the high-power requirements associated with off-road autonomous operations.

One of the key features of the PM02D is its digital sensing capability, which enables precise and reliable measurement of current and voltage levels. The module uses the TI INA228 digital current sensor to ensure high accuracy in power monitoring. Additionally, its compact form factor and lightweight construction contribute to a more efficient and manageable system integration.

The choice of the PM02D power module is driven by its reliability in performance, seamless integration with the Pixhawk 5X platform, and the ability to maintain power demands under rugged terrain conditions, all of which are critical to the consistent and safe operation of the autonomous vehicle.

Table 6.2: PM02D Power Module Specifications [20]

Parameter	**Value**
Rated Current	60A
Maximum Current (short duration)	120A (<60 seconds)
Maximum Current Sensing	164A
Supported Battery Voltage	Up to 6S
Communication Protocol	I2C
Switching Regulator Output	5.2V, 3A max
Integrated Circuit (IC) Used	TI INA228
Weight	20g

6.3 GPS MODULE FOR NAVIGATION: HERE 3

To enable precise localization and navigation, the autonomous vehicle employs the Here 3 GPS module. This high-performance GNSS receiver supports Real-Time Kinematic (RTK) positioning, which enables centimeter-level accuracy under favorable conditions. Such precision is especially beneficial in off-road environments where accurate path following and obstacle avoidance are critical.

The Here 3 module is powered by the STM32F302 processor, while the Here 3+ variant upgrades to a dual-core STM32H757 running at 400 MHz[7]. This processing improvement ensures faster handling of GNSS data and enhanced responsiveness. Both variants utilize the DroneCAN protocol, supporting data rates up to 8 Mbit/s, which allows for robust and low-latency communication with other vehicle systems.

In addition to GNSS capabilities, the module integrates an onboard compass, gyroscope, and accelerometer (ICM20948), providing inertial measurements that improve heading estimation and vehicle stability, particularly in GPS-denied conditions. Firmware updates and system configuration are supported through ground control software such as Mission Planner, ensuring long-term compatibility and feature expansion.

In summary, the Here 3 GPS system delivers accurate, fast, and reliable navigation performance, making it a suitable choice for off-road autonomous vehicle applications.

Table 6.3: Here 3 GPS Module Specifications[7]

Parameter	**Value**
Model	Here 3
Receiver Type	u-blox M8P-2 high-precision GNSS
Satellite Constellations	GPS L1C/A, GLONASS L1OF, BeiDou B1I
Positioning Accuracy	3D FIX: 2.5 m, RTK: 0.025 m
Processor	STM32F302
IMU Sensor	ICM20948 (gyro, accelerometer, compass)
Navigation Update Rate	8 Hz
Communication Protocol	DroneCAN, 1 Mbit/s
Operating Temperature	$-40°$C to $85°$C
Dimensions	68 mm $\times$ 68 mm $\times$ 16 mm
Weight	48.8 g

6.4 FLYSKY TRANSMITTER & RECEIVER MODULE

To enable manual control and safety override in the autonomous vehicle system, the Flysky FS-i6X transmitter and FS-iA10B receiver are utilized. The FS-i6X is a 2.4 GHz handheld radio transmitter with a default of 6 channels, expandable to 10 channels [12]. It operates on the AFHDS 2A protocol, ensuring secure and interference-resistant communication through automatic frequency hopping. The transmitter includes a user-friendly LCD interface, 4096-level resolution joysticks, and provides an operational range exceeding 1 kilometer in open conditions. Its primary roles include throttle control, steering override, operational mode switching, and emergency stop activation during testing and manual intervention.

The FS-iA10B receiver is mounted on the vehicle to capture signals from the transmitter. It supports both PWM and i-BUS outputs, enabling compatibility with Pixhawk and Arduino-based systems[12]. Functioning within the 2.4–2.48 GHz spectrum, it incorporates dual antennas to ensure enhanced signal strength and communication reliability. With compact form factor, low power draw, and a high sensitivity of -105 dBm, the receiver ensures dependable control, which is crucial during calibration, testing, and real-time override operations.

FS-i6X Transmitter Specifications

Specification	Value
Channels	6–10 (Default 6)
Model Type	Fixed-Wing / Glider / Helicopter
RF Range	2.408–2.475 GHz
RF Power	< 20 dBm
RF Channels	135
Bandwidth	500 kHz
2.4 GHz System	AFHDS 2A / AFHDS
Modulation Type	GFSK

Table 6.4: FS-i6X Transmitter Specifications[12]

Figure 6.3: FS-i6X Transmitter[12]

FS-iA10B Receiver Specifications

Specification	**Value**
Item	FS-iA10B Receiver
Number of Channels	10
Suitable Models	Airplane / Glider / Helicopter
Frequency Range	2.4–2.48 GHz
Number of Newspapers	160
Transmitting Power	≤ 20 dBm
Receiver Sensitivity	-105 dBm
2.4G Modes	Automatic Frequency, 2nd Gen Digital System
Encoding	GFSK
Antenna Length	26 mm × 2 (Dual Antenna)
Weight	19.3 g
Input Power	4.0–6.5 V DC
Dimensions	47 × 33.1 × 14.7 mm
Color	Black
Certification	CE0678, FCC
i-BUS Interface	Yes
Data Acquisition Interface	Yes

Table 6.5: FS-iA10B Receiver Specifications[12]

Figure 6.4: FS-iA10B Receiver[12]

6.5 INTEGRATION OF ELECTRIC POWER STEERING (EPS)

In the development of an autonomous off-road navigation vehicle, it was initially planned to use a 180 kg-rated servo motor to operate the steering mechanism of a 110cc quad bike. However, during early field testing, particularly on rugged and uneven terrain, it became evident that the steering system experienced higher-than-anticipated resistance. The high friction between the tires and irregular ground surfaces required significant steering torque, which the servo motor alone could not sustainably provide.

As a result, the servo motor underwent continuous over-torque stress, leading to overheating and eventual failure. This limitation in the original design prompted a redesign of the steering system. The team decided to integrate an Electric Power Steering (EPS) unit, sourced from a Hyundai vehicle, to enhance the steering capabilities. The selected EPS unit functions independently when powered by a 12V battery and does not rely on vehicle RPM or ECU inputs. This autonomy made it well-suited for adaptation to the modified quad bike platform.The EPS is capable of delivering torque assistance in the range of 4–5 N·m, which substantially reduces the mechanical effort required to steer, particularly at low speeds and during tight turns, conditions typical in off-road environments.

This integration improves the responsiveness and reliability of the steering system while minimizing wear on mechanical components. It also enables effective implementation of autonomous path planning and control strategies. The use of EPS was inspired by prior research conducted at the University of North Carolina, where similar issues with servo motor overheating were addressed through the adoption of

Figure 6.5: Power Steering[1]

powered steering enhancements[4].

The EPS specifications are summarized in Table 6.6.

Table 6.6: Specifications of Electric Power Steering (EPS)[1]

Parameter	**Value**	**Condition/Notes**
Power source voltage (PIG supply)	10 to 16 V	Power steering in operation
ECU power source voltage	10 to 16 V	ECU power on
Motor terminal voltage	10 to 16 V	Steering wheel turned left
Torque sensor 1 (TRQ1)	2.3 to 2.7 V (centered)	No load
Torque sensor 2 (TRQ2)	2.3 to 2.7 V (centered)	No load
Torque sensor voltage (TRQV)	7.5 to 8.5 V	Signal input to ECU
Torque sensor signal range	0.3 to 4.7 V	Turning from center
Motor current (actual)	-128 A to 127 A	Current to motor
Motor current (command)	-128 A to 127 A	Requested by ECU
ECU temperature range	-50°C to 205°C	Measured at ECU
Steering effort (reference)	5.5 N·m (49 in·lbf)	At 90° wheel turn
Steering wheel free play	30 mm max	Without load

6.6 INTEGRATION OF ARDUINO NANO FOR STEERING CONTROL

The Arduino Nano microcontroller has been integrated into the electronic power steering (EPS) control system of the autonomous off-road vehicle. Based on the

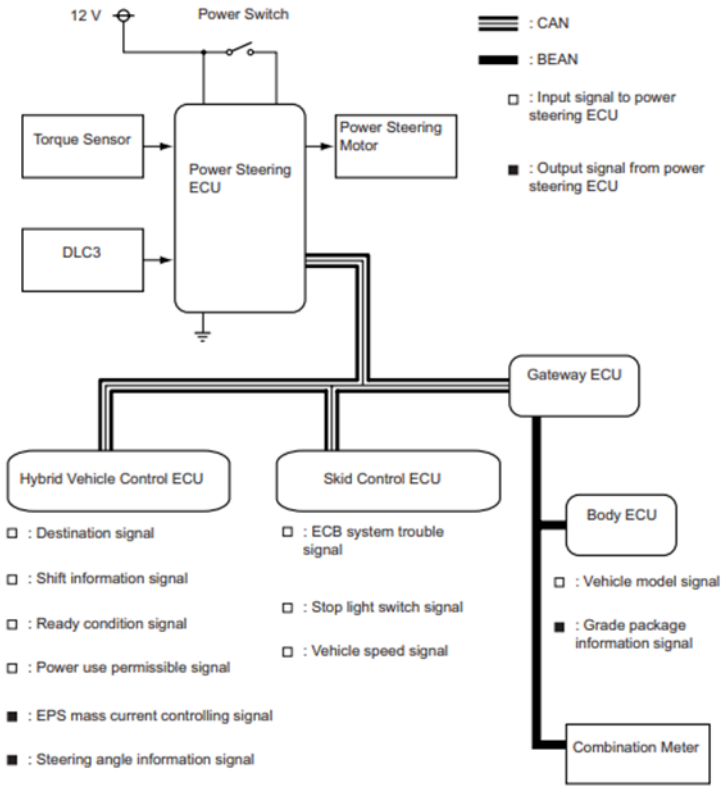

Figure 6.6: Power Steering Wiring[1]

ATmega328P chip, the Arduino Nano is a compact and flexible board widely used in embedded systems and robotic applications due to its small form factor, reliable performance, and extensive I/O capabilities. It supports digital and analog inputs/outputs, pulse-width modulation (PWM), and serial communication interfaces, making it highly suitable for real-time control systems.

The primary reason for incorporating the Arduino Nano was to address a specific limitation of the Pixhawk flight controller. Pixhawk generates PWM signals within the standard range of 1000 to 2000 microseconds, which suffices for conventional servo systems. However, the steering mechanism employed in the project requires an extended PWM range, approximately from 500 to 2500 microseconds, to achieve a full 270-degree rotation necessary for optimal steering performance [**?**].

To overcome this mismatch, the Arduino Nano is configured to receive control inputs and output the required extended-range PWM signals. Its precision in PWM signal generation and its adaptability make it an effective intermediary between the Pixhawk controller and the EPS unit. This integration ensures accurate and smooth steering control, particularly under demanding terrain conditions.

Table 6.7 summarizes the technical specifications of the Arduino Nano used in the system.

Table 6.7: Specifications of Arduino Nano

Parameter	**Value**
Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7–12V
Input Voltage (limits)	6–20V
Digital I/O Pins	22 (14 Digital, 8 Analog)
PWM Channels	6
Analog Input Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	32 KB (2 KB used by bootloader)
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
Dimensions	18 x 45 mm
Weight	7 g
USB Connector Type	Mini USB

6.7 BATTERY SELECTION AND SIZING

To meet the distinct power demands of the system—including the electronic steering, control components (Pixhawk, Arduino Nano, and acceleration and brake control servos), and engine ignition—two separate power sources are utilized.

The first power source is a 12V, 10A Volta dry cell battery[25], dedicated to powering the Electric Power Steering (EPS) system and providing a self-start function for the quad bike's 110cc petrol engine. This dry cell battery was selected for its ability to deliver steady current within the EPS's operating range of 10 to 16 volts. The EPS typically draws 5 to 8 amps, with peak demands reaching 10 amps during high-torque conditions such as tight turns or rough terrain. With a maximum power output of 100 watts, this battery comfortably supports both the steering system and ignition without risking voltage drops or overheating.

In addition, the system uses two 3S LiPo batteries [27](Lithium Polymer) connected in series to supply high voltage to the vehicle's high torque steering servo motor and the control electronics on board. Each 3S battery delivers a nominal volt-

age of 11.1V, resulting in a combined output of 22.2V to 25.2V. This voltage range aligns with the servo's optimal operating point of 24V. LiPo batteries were chosen for their high energy density, lightweight construction, and ability to deliver high discharge currents, making them ideal for off-road robotic applications.

To ensure compatibility with low-voltage components such as the Arduino Nano, a buck converter is used to step down the 24V supply to a stable 5V output. This ensures safe and efficient operation of the microcontroller and its associated sensors or PWM control lines. The dual-battery configuration not only separates high-power and low-power circuits for improved system safety but also enhances reliability and performance under demanding terrain conditions.

Current Requirements of System Components

Table 6.8: Component Current Ratings

Component	**Current Rating**
Pixhawk + GPS	280 mA
Arduino Nano	40 mA
160 kg Steering Servo	4.2 A @ 24V
Power Module	500 mA
Brake Servo	4 mA
Accelerator Servo	4 mA
Total Current Drawn	**5028 mA**

Battery Capacity Calculation

$$\text{Battery Capacity} = \frac{\text{Time (min)} \times \text{Average Current (mA)}}{60}$$

$$\text{Battery Capacity} = \frac{60 \times 5028}{60}$$

$$\text{Battery Capacity} = 5028 \text{ mAh}$$

This calculation ensures that the selected battery can sustain the system's operation for a minimum of 60 minutes under average load conditions.

(a) 3S LiPo Battery[27] (b) Volta Dry Cell Battery[25]

Figure 6.7: Battery Types Used in the System

As shown in Figure 6.8, the complete control architecture includes the Pixhawk 5X flight controller, Arduino Nano, electronic power steering (EPS) motor and controller, brake and throttle servos, dual 3S LiPo batteries, a Volta dry cell battery for ignition and EPS, and a buck converter for voltage regulation. Color-coded lines represent live (red), ground (black), and signal (yellow) connections, helping visualize the power and signal flow across components.

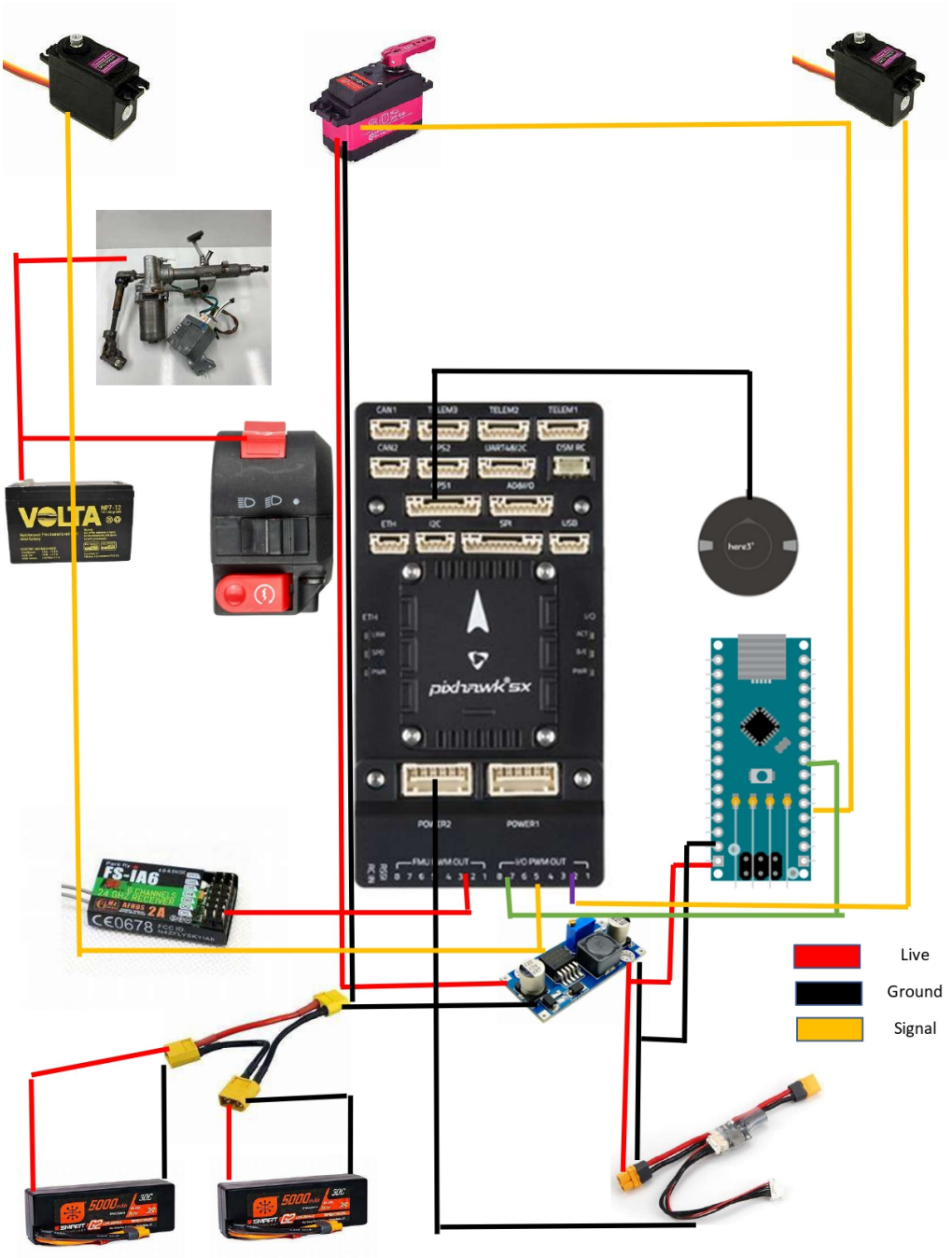

Figure 6.8: System Overview Diagram .

6.8 BILL OF MATERIALS

Table 6.9 lists the major hardware components required for the autonomous ATV prototype. High-end perception sensors such as Lidar and RGBD cameras are ex-

cluded due to cost constraints and are reserved for future development phases.

Table 6.9: Bill of Materials (Excluding Vision Sensors)

Item	**Description**	**Qty**	**Unit Cost (Rs)**	**Total Cost (Rs)**
Battery	3S 5000mAh LiPo	2	22,000	44,000
Pixhawk 6C	Low-level controller	1	200,000	200,000
Raspberry Pi	Edge computer	1	50,000	50,000
ATV (Chinese 120cc)	Vehicle chassis	1	100,000	100,000
WiFi Router	5G wireless module	1	20,000	20,000
GPS	Here 3 module	1	100,000	100,000
Transmitter	Remote control unit	1	50,000	50,000
Telemetry	Radio telemetry system	1	20,000	20,000
Servos	Steering, Braking servos.	3	30,000	90,000
Miscellaneous	Wiring, PCBs, etc.	1	20,000	20,000
Total				**694,000**

Chapter 7

IMPLEMENTATION AND TESTING

The testing phase of the system involved both simulation-based validation and real-world hardware implementation. Initial testing was conducted in a Linux-based environment using PX4 and ROS 2, deployed on the Ubuntu 22.04 operating system. The PX4 simulation was initiated by executing the appropriate startup script, followed by building the Gazebo simulation workspace. To enable communication between PX4 and ROS 2, a DDS (Data Distribution Service) communication script was run, establishing a reliable data link between the two systems. Subsequently, the ROS 2 navigation script was executed to perform point-to-point navigation, during which the vehicle's simulated path and waypoint tracking behavior were observed and evaluated.

Upon successful simulation validation, the system proceeded to hardware testing. All servo connections—including those for the accelerator, brake, and steering—were manually verified to ensure correctness and absence of short circuits. The connections to the Pixhawk flight controller and the radio receiver were also checked, along with the wiring associated with the steering servo controlled via an Arduino interface.

Following connection validation, Li-Po and dry batteries were connected to the power distribution system, and the servos were tested individually using a transmitter. The vehicle's engine was then powered on using a dry battery, and the activation of the power steering system was confirmed by the characteristic motor sound. Subsequent transmitter commands were used to verify the operation of the accelerator and brake servos. The steering system was also tested to confirm proper angular response to control inputs.

A short test drive was conducted to assess dynamic performance parameters including balance, vibration levels, and speed. Special attention was given to checking the thermal insulation of wiring and ensuring that system heat did not compromise cable insulation integrity.

For real-time implementation, a Raspberry Pi was mounted onboard the vehicle to act as an onboard companion computer. Pixhawk calibration was performed, followed by validation of sensor functionality and servo responsiveness. The starting waypoint was set using QGroundControl, the ground control station (GCS), and navigation commands were transmitted to the vehicle over a shared network via telemetry. The mission, consisting of predefined GPS waypoints, was uploaded to the Pixhawk flight controller. The operating mode was then switched from MANUAL to OFFBOARD to enable autonomous point-to-point navigation.

During operation, the system's telemetry link was monitored, and the PX4 SITL (Software-In-The-Loop) heartbeat was tracked to ensure continuous communication without error. The vehicle was observed to successfully reach each navigation waypoint, thereby validating the navigation logic and the system's integration.

Chapter 8

RESULTS AND DISCUSSION

The project concluded by yielding significant outcomes in both simulation and physical hardware testing. Using ROS 2, Gazebo, and PX4 created a high-quality simulation platform allowing for robot testing of the point-to-point navigation capabilities of the UGV. The results are divided into three categories:

8.1 SIMULATION VALIDATION

Using Gazebo as the base, a high-quality simulation environment integrated with a physics engine was created to simulate the UGV in various terrains. Furthermore, integration with ROS 2 and PX4 introduced flexibility and extended control capabilities for testing waypoint-based navigation. Results include:

- Stable development of the vehicle in the Gazebo world
- Successful integration with ROS 2 and PX4
- The vehicle demonstrated stable path following capabilities using the pure pursuit algorithm
- Successful integration of sensor data such as GPS with ROS 2 nodes and simulation in Gazebo

Fig 8.1 shows the original mission plan on QGC for point-to-point navigation on the UGV. In the Figure 8.1, the left section shows the simulated vehicle, while the right section shows the 4 way points that the UGV must pass or approach through.

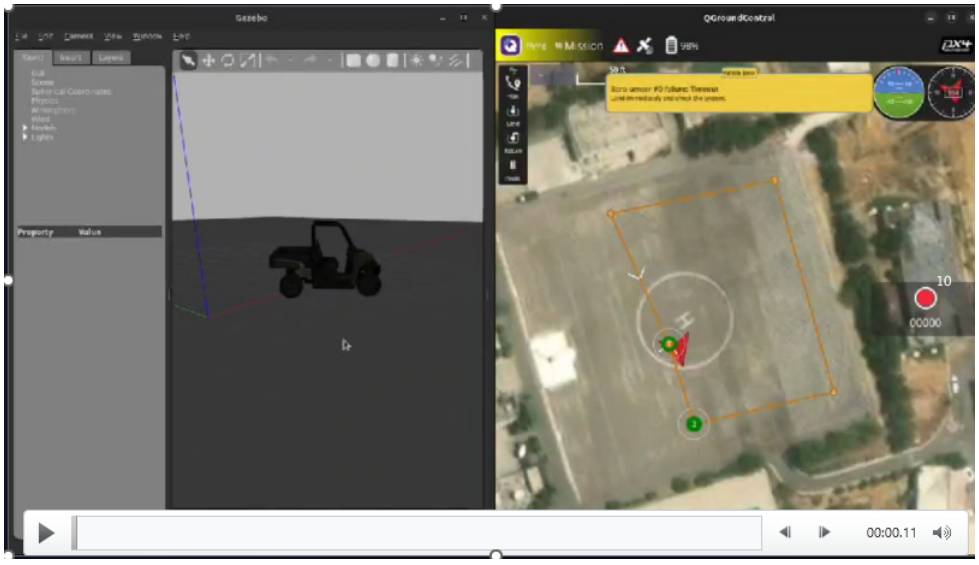

Figure 8.1: Gazebo simulation with way points

In the Fig 8.2 shows in Red the path that the UGV actually followed. The in built PID controller in Pixhwak compensates the deviation of the UGV from the waypoint and navigates accordingly as discussed in detail in section 5.1.1.

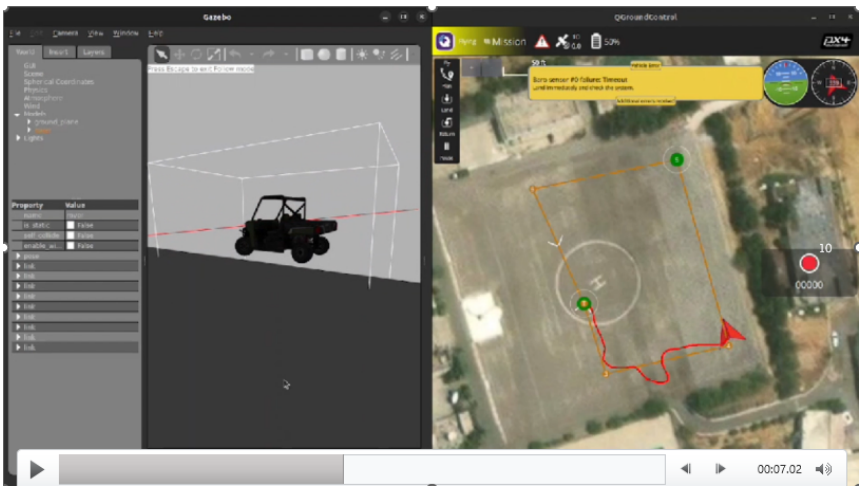

Figure 8.2: Gazebo simulation with the path followed by the UGV in Red

8.2 MECHANICAL AND AUTOMATION RESULTS

Mechanically, the body was designed and developed (as discussed in Section **??**) to provide a robust exterior allowing for extended capacity for electronics and battery, as well as providing space for future integrations with sensors, a companion

computer, or other types of payloads. Key results include:

- Successful development of the mechanical body and integration with the chassis
- Successful automation of the vehicle, including:
  - Conversion of accelerator and brake from manual to servo control
  - Integration of servo with gears and power steering to implement steering control

The results of the final vehicle are shown in Figure 8.3. The vehicle can be seen driving in a sandy terrain, with the blue light emitting device being the GPS.

Figure 8.3: Final ATV driving in sandy terrain

8.3 POINT-TO-POINT NAVIGATION RESULTS

This involved testing of the point-to-point navigation script, which was validated in the simulation environment. A transmitter and GPS module were connected to the framework in order to implement full vehicle control. Key results include:

- Successful operation of the vehicle using the developed framework, including

power-up, compatibility checks, and control of the vehicle using the transmitter

Figure 8.4 shows the operation of the vehicle driving using point to point navigation command sent through QGC.

Figure 8.4: Autonomous Driving of the vehicle

8.4 PURE PURSUIT SIMULATION FOR PATH TRACKING

8.4.1 Overview

To validate the performance of the Pure Pursuit path-following algorithm on an Ackermann-steered vehicle, a simulation environment was developed in MATLAB. The simulation mimics the dynamics of a 110cc internal combustion engine (ICE)-based ATV, incorporating throttle nonlinearity and response delay. The control approach and modeling are designed to closely represent the actual behavior of the vehicle prior to hardware deployment using PX4 autopilot firmware.

8.4.2 Model Description

The vehicle is modeled using the kinematic bicycle model which approximates Ackermann steering. The state variables are vehicle position (x, y) and orientation

θ, while the control input is the steering angle δ derived from the Pure Pursuit control law. The vehicle velocity v is computed from a nonlinear throttle response model.

The kinematic equations of motion are described in the follwoing equations 5.3, 5.4, and 5.5 where L is the wheelbase of the ATV (measured as 0.98 m).

8.4.3 Pure Pursuit Controller

The Pure Pursuit algorithm computes the curvature to a point on the reference trajectory located at a fixed lookahead distance L_d. The steering angle δ required to follow this arc is computed using the equation 5.12.

To reflect realistic ATV behavior, a nonlinear throttle map is used to relate throttle percentage to velocity. A first-order lag system is applied to simulate engine delay:

$$\tau \dot{T}_{\text{applied}} + T_{\text{applied}} = T_{\text{desired}} \tag{8.1}$$

Velocity is derived using interpolation over a throttle-to-speed lookup table:

Throttle (%)	Speed (m/s)
0.0	0.0
0.2	1.0
0.4	2.5
0.6	4.5
0.8	6.5
1.0	7.0

8.4.4 Simulation Setup

An infinity loop path consisting of multiple waypoints spaced evenly were defined. The simulation executes the following logic:

- The current and previous waypoints are updated as the vehicle reaches each target waypoint.
- Pure Pursuit computes a target steering bearing toward the lookahead point.

- Steering and throttle dynamics are simulated using vehicle constraints.

8.4.5 Results

Figure 8.5 shows the trajectory of the vehicle following a figure-eight path using the Pure Pursuit controller. The controller performs well when using a lookahead distance of 6 m, closely tracking the desired path. While figure 8.6 shows rover path deviation from the actual path becuase of large lookahead distance.

Figure 8.7 illustrates the cross-track error over time for a lookahead of 6 m. The vehicle maintains minimal error throughout the trajectory. However, when the lookahead is increased to 7 m, as shown in Figure 8.8, the tracking performance degrades, particularly in sharp turns. This reflects the trade-off between stability and responsiveness in Pure Pursuit.

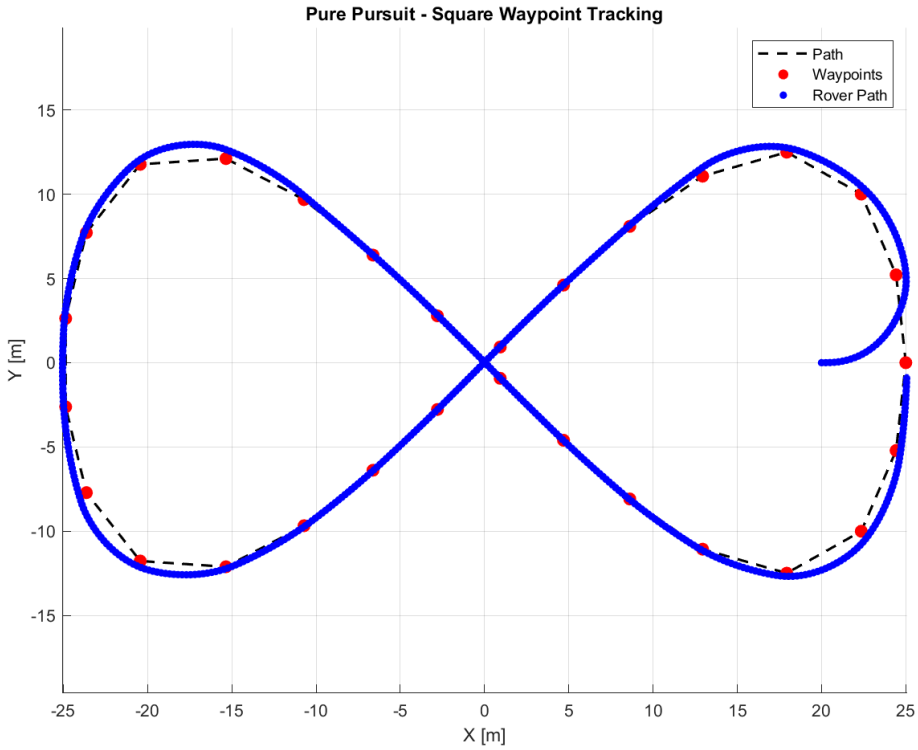

Figure 8.5: Trajectory Tracking Lookahead = 6 m

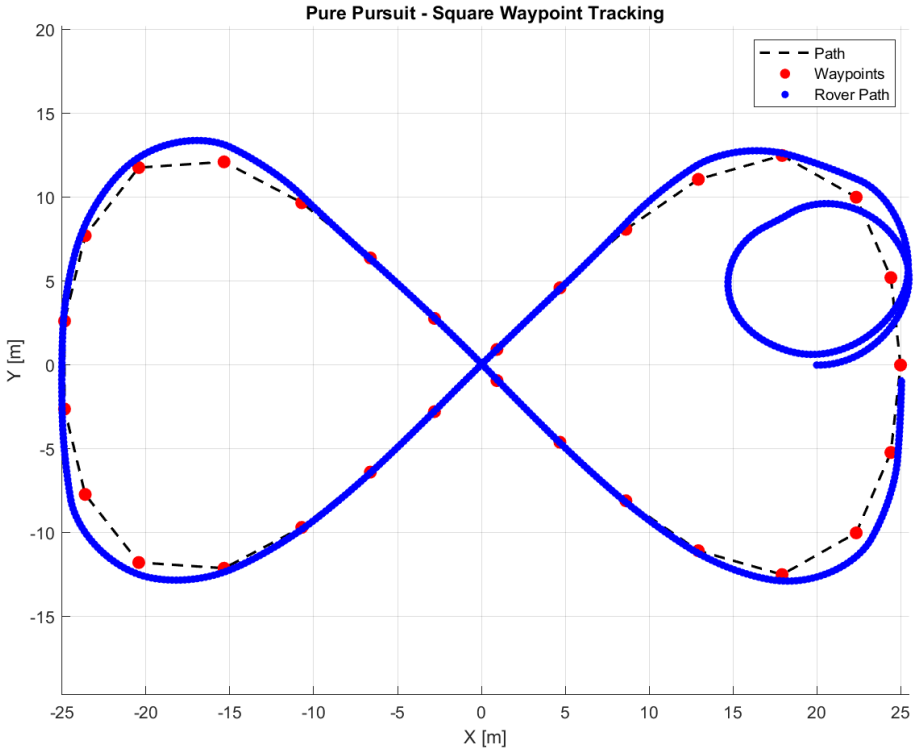

Figure 8.6: Trajectory Tracking Lookahead = 7 m

Figure 8.7: Cross-track Error with Lookahead Distance of 6 meters

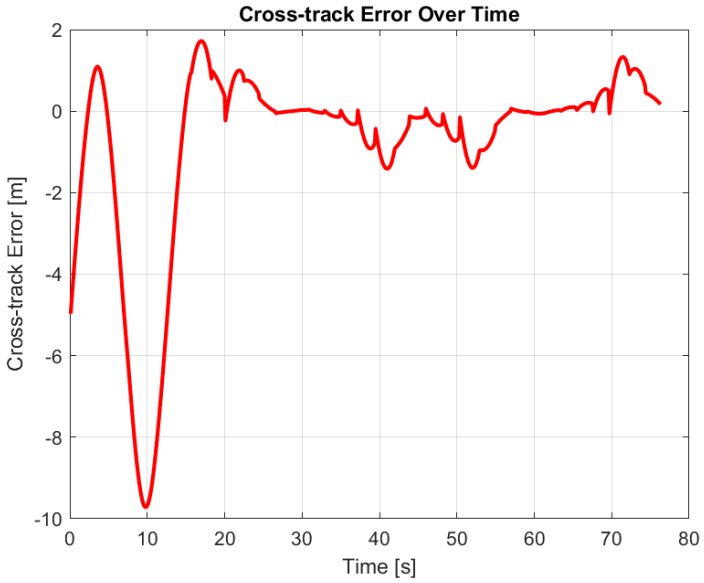

Figure 8.8: Cross-track Error with Lookahead Distance of 7 meters

8.4.6 Conclusion

The simulation successfully demonstrates that Pure Pursuit is capable of following complex paths with high accuracy under realistic vehicle dynamics. Smaller lookahead distances improve precision but may reduce stability. This simulation provides a valuable testbed for tuning the controller before deployment on the actual ATV using PX4 autopilot.

Chapter 9

CONCLUSION AND FUTURE WORK

In conclusion, this thesis project lays down the foundation for autonomous navigation by placing stress on implementation on a full-scaled ATV and considering the mechanical, electrical, and software complexities that are often overlooked in a smaller-scaled prototype. The simulation framework, integrated with ROS2 and PX4 on Gazebo, provides a great interface to test and plan autonomous missions beforehand. The hardware based on ROS2 and PX4 lays the foundation for future integration with sensors such as LIDAR, Camera, GPS etc. extending the UGV's capabilities for different mission based on the requirements.

For future work, different sensors such as LIDAR, Camera, GPS and a companion computer etc can be integrated to extend the UGV's capabilities for more complex missions such as surveillance, detection, object avoidance etc. Furthermore, through the companion computer, such as Raspberry pi or Jetson Nano, perception and deep learning capabilities can be added allowing for fully autonomous missions using Simultaneous Localization and Mapping (SLAM) and Computer Vision. In addition, ATV as the base for the UGV allows heavy payloads such as robotic arm, UAV launch pads to be easily mounted and integrated with the software framework allowing for operations in more complex terrains and weight intensive applications.

Bibliography

[1] Ps - power steering. Toyota Prius EPS Technical Manual. Accessed: 2025-04-29.

[2] X. Chen, Y. Li, and Z. Wang. Enhanced pure pursuit path tracking algorithm for mobile robots optimized by nsga-ii with high-precision gnss navigation. *Sensors*, 24(3):745, 2024.

[3] A. Cortner, J. M. Conrad, and N. A. BouSaba. Autonomous all-terrain vehicle steering. In *Proceedings of the IEEE SoutheastCon*, 2012.

[4] A. Cortner, J. M. Conrad, and N. A. BouSaba. Autonomous all-terrain vehicle steering. In *2012 Proceedings of IEEE Southeastcon*, pages 1–5. IEEE, 2012.

[5] R. C. Coulter. Implementation of the pure pursuit path tracking algorithm. Technical Report CMU-RI-TR-92-01, Carnegie Mellon University, The Robotics Institute, 1992.

[6] R. Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical Report CMU-RI-TR-92-01, Pittsburgh, PA, January 1992.

[7] CubePilot Team. Here 3 gnss module manual. `https://docs.cubepilot.org/user-guides/here-3/here-3-manual`, 2024. Accessed: 2025-05-02.

[8] PX4 Autopilot Developers. Px4 autopilot official website, 2024. Accessed: 27-Apr-2025.

[9] Inc. Dronecode Project. Qgroundcontrol - drone control, 2024.

[10] T. Fan, P. Long, W. Liu, and J. Pan. Multi-sample long range path planning under sensing uncertainty for off-road autonomous driving. In *Robotics: Science and Systems (RSS)*, 2018.

[11] Thomas Fermi. Kinematic bicycle model for vehicle control. https://thomasfermi.github.io/Algorithms-for-Automated-Driving/Control/BicycleModel.html, 2020. https://thomasfermi.github.io/Algorithms-for-Automated-Driving/Control/BicycleModel.html.

[12] Flysky. FS-i6X Transmitter Specifications, 2025. Accessed: 2025-05-02.

[13] H. Gonzalez and K. Iagnemma. Model predictive control for aggressive driving over uneven terrain. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[14] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, 2004.

[15] S. Lee, Y. Kim, H. Lee, and J. Lee. Autonomous driving system architecture with integrated ros2 and adaptive autosar. *Electronics*, 13(7):1303, 2024.

[16] T. Lin and S. Shen et al. Wroom: An autonomous driving approach for off-road navigation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[17] S. Macenski, F. González, and M. Blanc. Regulated pure pursuit for robot path tracking. *arXiv preprint*, 2023.

[18] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.

[19] PX4 Autopilot Team. Ros 2 user guide. https://docs.px4.io/main/en/ros2/user_guide.html, 2025. Accessed: 2025-05-04.

[20] PX4 Development Team. Holybro pm02d power module — px4 autopilot user guide. https://docs.px4.io/main/en/power_module/holybro_pm02.html, 2024. Accessed: 2025-05-02.

[21] PX4 Development Team. Pixhawk 5x — px4 autopilot user guide. https://docs.px4.io/main/en/flight_controller/pixhawk5x.html, 2024. Accessed: 2025-05-02.

[22] Niranjan Ravi and Mohamed El-Sharkawy. Real-time embedded implementation of improved object detector for resource-constrained devices. *Journal of Low Power Electronics and Applications*, 12(2), 2022.

[23] Cosmin Rus, Monica Leba, Nicoleta Negru, Răzvan Marcuş, and Alin Costandoiu. Autonomous control system for an electric atv. *MATEC Web of Conferences*, 343:06003, 08 2021.

[24] Jarrod M. Snider. Automatic steering methods for autonomous automobile path tracking. Technical Report CMU-RI-TR-09-08, Carnegie Mellon University, Pittsburgh, PA, February 2009.

[25] Volta Batteries. Volta np7-12 12v 7ah dry cell battery, 2025. Accessed: 2025-05-02.

[26] J. Wang, M. T. H. Fader, and J. A. Marshall. Learning-based model predictive control for improved mobile robot path following using gaussian processes and feedback linearization. *Journal of Field Robotics*, 40(5):1014–1033, 2023.

[27] ZOP Power. Zop power 5000mah 60c 11.1v 3s lipo battery xt60 plug for rc quadcopter car airplane, 2025. Accessed: 2025-05-02.