Design and Implementation of an Autonomous Driving System with a Deep Learning Approach on a Scaled Vehicle Platform

by

Jesudara Omidokun

A thesis in partial fulfillment
of the requirements for the degree of
Master of Science in Engineering
(Electrical Engineering)
in the University of Michigan-Dearborn
2022

Master's Committee

Assistant Professor Jaerock Kwon, Chair
Associate Professor Bochen Jia
Associate Professor Alireza Mohammadi

Jesudara Omidokun

jomidoku@umich.edu

ORCID iD: 0000-0002-0304-7379

Dedicated to my Mom, Dad, and Uncle for their sacrifices and devotion towards my success.

Acknowledgements

This thesis would not have been possible without the knowledge, support, and guidance of my thesis advisor Professor Jearock Kwon. I owe you a debt of gratitude for providing me with the necessary resources and showing me how to be a good researcher and student.

I express my deepest gratitude to my thesis committee, Bochen Jia and Aliraza Mohammadi. They have served as great scholarly examples to me and have taken the time to lead me throughout my time as a graduate student at the University of Michigan-Dearborn. To Professor Jia for his clear direction and support throughout my time as a graduate student. Words can not fully articulate my gratitude. Lastly, I would like to acknowledge the Bimi team for the friendship that developed throughout our time together. Aws, thanks for your assistance and oversight when necessary

Table of Contents

List of Figures

List of Tables

List of Appendices

Abstract

The thesis project is developed to create a platform for autonomous driving research and education purposes. The main goal of this study is to present a complete and adequate way for building a platform with a drive-by-wire system and sensor packages that are both comprehensive and appropriate for testing and deploying machine-learning-based algorithms on a scaled vehicle. The vehicle platform addresses reproducibility issues, onboard processing capability restrictions, vehicle scale, and system cost with the existing related vehicle platforms. Our vehicle platform system integrates both drives-by-wire and an autonomous system enabled with sensor packages in an easy-to-implement format. The platform is a generic and multilevel system, with flight controller programs combined with a GPS module handling the mid to high-level motor controls and a laptop powered by a graphics processing unit (GPU) capable of handling the advanced and more complex algorithms. The vehicle platform is validated by employing it in a deep-learning-based behavioral cloning study. The platform's affordability and adaptability would benefit broader research and the education community.

Chapter 1: Introduction

1.1 Background

The autonomous driving system has enormous importance to the future of transportation systems. Thus, as the year progresses, different aspects of the research have been conducted to achieve a fully autonomous system in the driving sector. With the advancement in technology and the increase in time spent in automobiles, this time must be as enjoyable as possible. In addition, the rise in technology has called for more avenues where autonomous vehicles have been more critical than ever before. For instance, most delivery services, such as Doordash, Amazon, and FedEx, have contributed hugely to the development of fully autonomous vehicles. Most importantly, driving is a mechanical activity that requires attention and focus, a change in the mood while using drugs, medications, or other substances that can impair the perception of the environment. Therefore, automating the driving system will eliminate human mistakes and significantly decrease traffic congestion.

As the autonomous vehicle community increases, one major issue is that there are not enough people and platforms for research. As a result of the low turnout of people and research platforms, it's taking longer to bring a fully autonomous vehicle to market. Over the last decade, there have been numerous platforms to aid autonomous driving research and education. However, most existing platforms are challenging to replicate, leading to decreased interest in autonomous driving system platforms.

The proposed solution is to create a vehicle system platform that will aid research and education purposes in the autonomous driving system, which will improve the existing

autonomous vehicle platforms in design simplicity, modification necessity, and cost-effectiveness.

1.2 Motivation

The F1TENTH is a good platform for autonomous driving systems, but reproducing the system was challenging. Although there are instructions on replicating the system, the problem is that it is designed with different embedded components, and getting those components are challenging to readily unavailable.

Over the last decade, there have been numerous platforms for the drive-by-wire system for research and education purposes in automated driving systems. However, most platforms need to improve in simplification and long time useability. The complication with designing those platform systems is that they are challenging to reproduce and have onboard processing capability restrictions. First, materials are always problems because the system is designed with multiple embedded components, which might be out of the store or out of production. In addition to the system's reproducibility, the platforms are scaled small and need to be more cost-effective.

Regarding the onboard processing capability restrictions, most existing platforms are powered by the NVIDIA Jeston series of embedded computers. However, with constant improvement, previously used versions do not have the update for enabling the evolving capacity of the deep learning and computer vision API(Application Programming Interface). Although the NVIDIA Jeston family has shown reasonable performance with the small existing vehicle platforms, there are more viable solutions with a full-scale vehicle platform.

The research aims to find a solution to the problem discussed in the previous paragraphs, creating a simplified, cost-effective, long-time useability and better-scaled vehicle platform for an automated driving system.

1.2.1 Research Objectives

1. Design a vehicle that can accommodate a human driver to control the speed, throttle, and steering information.
    a. The vehicle platform will have fewer components and be easy to replicate.
    b. The vehicle platform will have a sizeable open-source community to assist with modifications.
2. The vehicle platform will be cost-effective and would be appropriately scaled to enable indoor and outdoor applications
3. The vehicle platform will be cable of implementing other advanced driving system algorithms.

1.3 Organization

The background information, motivation, and research objective of the thesis are presented in Chapter 1. Chapter 2 presents the related work, and Chapter 3 will discuss the history and present state of the art of autonomous systems regarding self-driving vehicles. Chapter 4 introduces the system overview and divides the system into two main areas: drive by wire system and an autonomous driving system. Chapter 5 dives into the system drive by wire system, explaining the hardware design and components of the system. In Chapter 6, the thesis examines the autonomous system and elaborates on the systems architecture. Chapter 7 discusses neural networks and their various implementation in the system. In Chapter 8, the result of the different experiments will be discussed. Chapter 9 will be the discussion and future work section. Finally, Chapter 10 summarizes the conclusions of the entire thesis.

Chapter 2: Related Work

The MIT racecar [1], also known as the Rapid Autonomous Complex-Environment Competing Ackermann-steering Robot, is an open-source instructional platform that includes an autonomous model vehicle and a course on basic robotics. In 2014, Sertac Karaman, Mike Boulet, Owen Guldner, Michael Lin, and the MIT racecar were founded. The first mobile robot created for education, the RACECAR, included a potent Graphics Processing Unit (GPU). The NVIDIA Jetson Tegra K1 computer was used in the first batch of RACECARs. The NVIDIA Jetson Tegra X and, eventually, the NVIDIA Jetson Xavier computers were used in later iterations. Since then, the RACECAR platform has been used in numerous MIT courses and research projects and has encouraged several related projects around the nation. Add-ons, like an IMU and a Lidar camera, allow for customization. The vehicles also have a VESC wheel odometer, an open-source Electronic Speed Control component that controls speed. Despite having a GitHub repository and being an open platform, it necessitates substantial programming on the developer's part to carry out each task.

The Turtlebot 3 [2] is a compact, differentially driven mobile platform with a relatively low price that runs on ROS. It is a multi-partner open-source collaborative project assembled from premium modular parts [3]. It is based on an extendable chassis that has been 3D printed and is managed by a powerful controller and Single Board Computer. The Turtlebot's primary sensor is a low-cost LIDAR that can conduct SLAM and navigational tasks. Other sensors, such as RGB and RGBD cameras supported by ROS software modules, can also be added. By

connecting a manipulator module, it can function as a mobile manipulator. In graduate teaching and research, the Turtlebot has been employed successfully.

The Leo Rover [4] is a reliable open-source platform for autonomous research in outside settings. It can be modified by adding extras like a manipulator, GPS module, camera, or IMU. Four separate DC gear motors propel the robot, and its suspension system is driven by a Core 2 ROS driver board and Raspberry Pi. Despite having a GitHub repository and being an open platform, it necessitates substantial programming on the developer's part to carry out each task. Thus, its scope is different than that of educational boards.

Robotnik's Summit-XL [5] platform is a robust, adaptable framework built on a four-wheel skid-steering configuration with a high load capacity. Using Mecanum wheels, it is simple to convert it to an omnidirectional configuration. It has an IMU and can connect to a laser scanner and a camera. It is appropriate for research and surveillance and has a built-in radio transmitter for remote operation. It is programmed using open ROS architecture and is controlled by a PC. The Summit-XL is a midrange model of the industrial-grade robots that Robotnik makes.

F1TENTH [6] is an international community of researchers, engineers, and autonomous systems enthusiasts. It was first established in 2016 at the University of Pennsylvania, but it has now expanded to numerous other institutions around the world. F1TENTH car is an open-source project developed for a community of researchers and students. The NVIDIA Jetson TX2 and Traxxas Ford Focus chassis serve as the foundation for the hardware arrangement. In addition, it has a Hokuyo 10LX LIDAR sensor and a wheel odometer that operates on VESC 6, which aids in speed control.

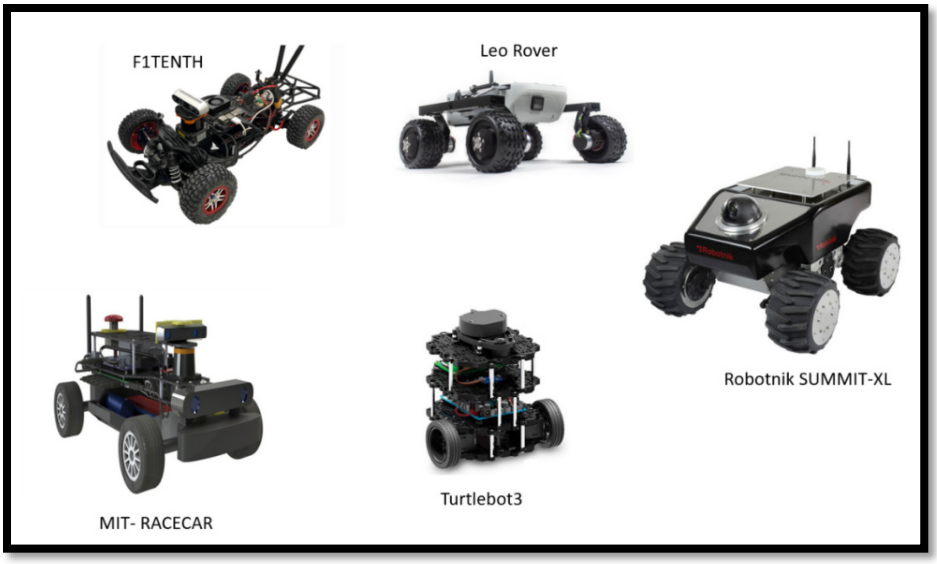


Figure 2-1: Images of related robot platforms.

Table 2-1: Related work: target group, cost, scale, and technology.

Name	Platform	Appro-ximate Cost ($)	Open Source	Sensors	ROS	Controller/ CPU	Programming Tools	Car Scale
Turtlebot3 [2]	Education	800	YES	Camera, LIDAR, acc, gyro, magn	YES	Raspberry Pi + OpenCR	Block-based/	1/12th
Leo Rover [4]	Research/ industry	2500	YES	Fish-eye camera, wheel encoders	YES	Raspberry Pi + Core2 ROS (low level)	ROS Programming	1/10th
Summit-XL [5]	Industry	15000	YES	3D camera, Laser scanner + optional sensors	YES	Intel processor/PC	ROS Programming	1/8th
MIT-racecar [1]	Research/ Education	2600	YES	3D camera, Laser scanner + optional sensors	YES	NVIDIA Jetson Nano	ROS Programming	1/10th
F1TENTH [6]	Research/ Education	3400	YES	Laser scanner + optional sensors	YES	NVIDIA Jetson NX	ROS Programming	1/10th

Chapter 3: State of the Art

3.1 Introduction

In the 1860s, Robert Whitehead invented the self-propelled torpedo, the first known guidance system [7]. The self-propelled system is designed to propel towards a target and interlock to the target location. The self-propelled torpedo guidance system raises the idea of a self-propelling system [7]. The first autonomous car was dated to an exhibition at the 1939 New York world's fair. In the exhibit, Norman Bel Geddes, General Motors designer, created the first self-driving car, an electric vehicle guided by radio-controlled electromagnetic fields generated with magnetized metal spikes embedded in the roadway. General Motors was able to make this concept a reality in 1959. [8] The car was designed with a sensor attached to the front. The sensor is called a pick-up coil, which detects the current flowing through a wire embedded in the road. The sensor allows the vehicle to move along the metal rod; the current could be manipulated to steer the vehicle wheel left or right.

The design from 1959 was improved in 1977 by a Japanese company [9]. They improved the idea with a camera system that relayed data to a computer to process images of the road. However, this vehicle could only travel at speeds below 20 mph. An improvement to the system came from the Germans a decade later in the form of the VaMoRs, a vehicle outfitted with cameras that could drive itself safely at 56 mph [9]. As technology improved, so did self-driving vehicles' ability to detect and react to their environment.

In 2005, the second edition of the DARPA (Defense Advanced Research Project Agency) Grand Challenge was a massive milestone for the autonomous driving industry after the failure of the first edition of the challenge in 2004 [10]. This competition consisted of a long-distance race with a two million dollar prize for the winner, in which its participants could only be autonomous vehicles. The competition's main goal was to encourage the collaboration of experts in many diverse technological fields from private companies and academic institutions to get them to improve the technology by sharing information between them [10]. After the failure of the first edition of DARPA, DARPA's second edition laid the groundwork for the future of the self-driving vehicle. There were 195 vehicles enrolled, with 23 of the vehicles classified for the main event. Five of the cars made it to the final round, which Volkswagen Touareg from the University of Stanford named “Stanley” emerged as the winner. Stanley covered 212km (132 miles) in 6 hours and 53 minutes. The DARPA 2005 made a difference by providing incentives and motivation for various industries to research, test, and implement self-driving cars [10].

3.2 Artificial Intelligence

The study of artificial intelligence aims to comprehend and create intelligent beings. Even though Warren McCulloch and Walter Pitts produced the first work, now acknowledged as Artificial Intelligence in 1943, it is one of the newest and most well-liked topics in science and engineering [11].

In 1950, the article "Computing Machinery and Intelligence" introduced the Turing Test, machine learning, genetic algorithms, and reinforcement learning. The article was one of the first to present the concept of Artificial intelligence [12]. Two years after the article, Marvin Minsky and Dean Edmonds, Havard students, built the first neural network computer called the SNARC.

SNARC used 3000 vacuum tubes and an automatic pilot mechanism from a B-24 bomber to simulate a network of 40 neurons [12].

In 1956, the field of artificial intelligence was born at Dartmouth College by John McCarthy, Marvin Minsky, Claude Shannon, and Nathaniel Rochester. They organized a two-month workshop to fully introduce the concept of AI in which six more people participated [11].

The introduction of the internet helped expands the field of artificial intelligence, which became a web-based application, a tool used by all the tech giants today. For example, the system creates a recommendation system with very different approaches; however, it uses the recommendation approach based on previous search data and likes regarding genres. For example, Netflix uses the system to recommend movies, TV shows, etc. On the other hand, Spotify and Amazon base their suggestions only on similar items the user has already listed.

Over the years, the AI community has evolved from algorithms improvement to more of a data-driven system because data availability increases every second. James Hays and Alexei A. Efros explained an example of the effect of the data-driven approach. They developed an algorithm to complete images with holes by finding similar images in the database. The algorithms do not require labeling the images previously [13]. Instead, the algorithm used the GIST scene descriptor to group semantically similar scenes (city, sea, buildings, forests, etc.) and for place recognition. Although, when the dataset uses ten thousand images, the result is poor compared to when the image dataset was increased to two million images, the results improved drastically.

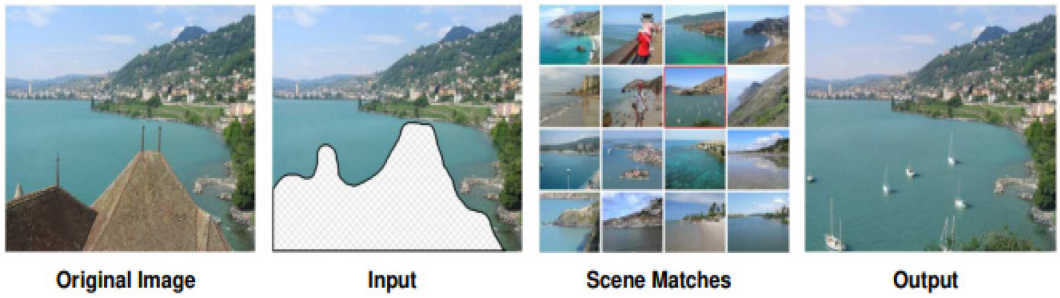


Figure 3-1: Scene completion algorithm [13].

3.3 Computer Perception and Relative Problem

Human has been able to identify object shape, patterns of light, and object formation without the influence of the background in the image. For example, humans can differentiate the number of people that appear in a picture, their genders, names, guess the age, or even their mood by their facial expressions or gestures. The process of identifying and differentiating between patterns in images is called perception. Humans perceive the world in three dimensions without any difficulty. The process of teaching intelligence to machines and making them understand the things humans do is called Computer Vision.

Computer Vision has been an essential part of perception in self-driving vehicles in the past year. Image is a 2D matrix of pixel intensities of shape (row, column). There have been various techniques to recover the tridimensional shape and appearance of the images developed by computer vision investigators during the latest years. One of those techniques is overlapping hundreds of images from a different perspectives. From the overlapping model, object movement can be identified by the computer. However, even with these technological advances, it is still difficult for the computer to interpret an image as humans do.

Over the last three decay, machine learning and deep learning techniques have improved the ability of computers to understand images and extract information from the visualization of

the data. One first problem is image classification. A classification problem involves classifying images into two or more classes. For example, the binary classification problem involves two classes for which the neural network's last layer will contain one neuron with a sigmoid activation function.

In the 1950s, the image classification problem was tackled with the Havard Mark 1 perceptron machine for image classification. However, the algorithm was effective in data-structured but not in classifying different geometric shapes. Decades later, a supervised learning model that analyzes data for classification and regression analysis was developed in bell laboratories called SVM algorithms. The SVM algorithm was able to tackle high-dimensional data with a minimum amount of samples, such as small image datasets.

In the 1990s, the first convolutional neural networks were inspired by Neocognitron. The introduction of the CNNs model revolutionized computer vision and even significantly improved after the first GPU implementation of a CNN described in 2006 by K. Chellapilla [14] . The result was four times faster than the equivalent implementation on the CPU.

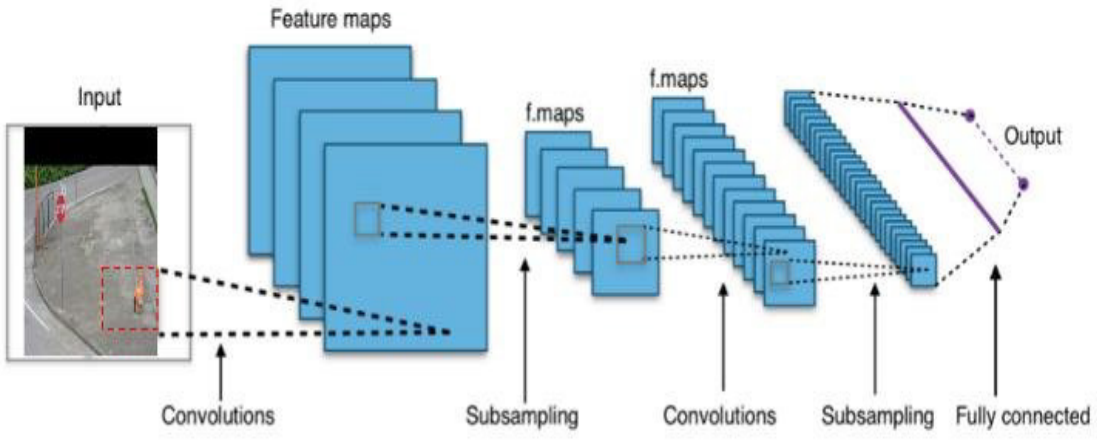


Figure 3-2: Operation of the CNN Model [15].

3.4 Autonomous Vehicle

According to the society of automotive engineers (SAE International), the terms autonomous vehicles or driverless cars are often used erroneously in the industrial perception of the terms, and people's understanding of the term is wrong. Therefore, the term autonomous vehicle or driverless cars is substituted for “driving automation” and “driving automation system.” However, driving automation can be confused with vehicles that are operated by human drivers and automated driving systems (ADS). With the help of the National Highway Traffic Safety Administration (NHTSA), the automated driving system is divided into five achievable levels. The automated driving system level is often referred to as Driving Automation levels in SAE J 3016-2021. Figure 3-3 summarizes the SAE J3016.

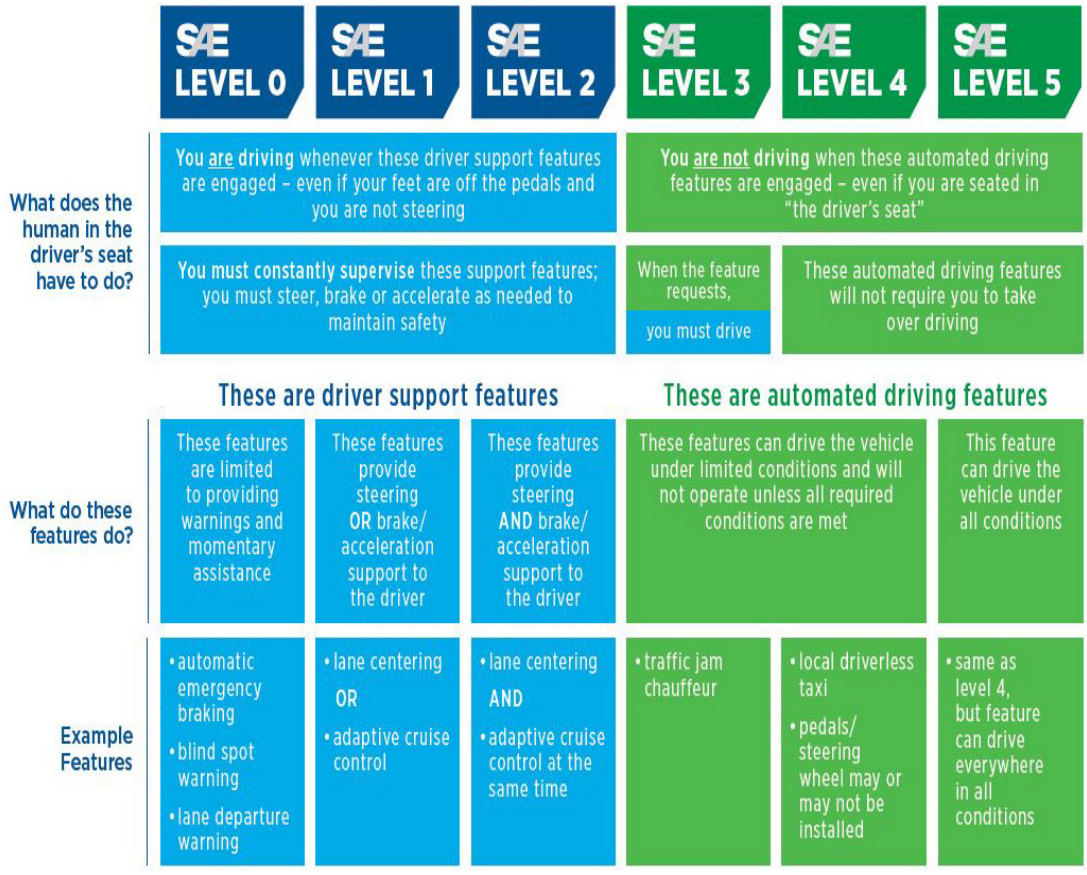


Figure 3-3: SAE J3016 summary table [16].

3.4.1 Level 0 - No Driving Automation

The majority of the vehicles available today are in this category. Level 0 has no driving automation system implies that the human driver performs the entire driving tasks. Regardless of the notion of no automation system, the vehicle in this category may have a system designed to help the driver make the best judgment during driving. An example of such a system is the emergency braking system. The emergency braking system in the vehicle is based on the proximity of objects with the sensors. Therefore allows the vehicles to apply brakes when needed. However, the emergency braking system does not drive the vehicle. Consequently, it does not qualify as an automation level.

3.4.2 Level 1 - Driving Assistance

The first level of automation is level 1. This level requires a fully engaged driver when the vehicle is in operation. However, the system provides continuous assistance with either acceleration or steering. An example of this system is the adaptive cruise control, which allows the vehicle to keep a safe distance behind the next car while maintaining a constant acceleration.

3.4.3 Level 2 - Partial Driving Automation

The term for level 2 is commonly referred to as the advanced driver assistance system(ADAS). Similar to Level 1 but characterized by both simultaneous vehicle motion control. The vehicle can control both accelerating/decelerating and steering. The vehicles in this category fall short of a self-driving system because it requires the driver remains fully attentive to take over when needed. For example, the tesla autopilot and Cadillac from general motors with the supercruise systems qualify as Level 2. Vehicles with highway pilots qualify as level 2.

3.4.4 Level 3 – Conditional Automation

From a human perspective, the difference between level 2 to level 3 is subtle, but from the technology perspective, it is regarded as a great leap forward in the self-driving industry. Vehicles are categorized as level 3 and include environmental detection capabilities that allow them to make an informed decision, such as accelerating past a slow-moving vehicle [17]. However, the human override feature is included. The systems perform the majority of the driving tasks while the human driver remains fully engaged for takeover. The 2019 Audi A8L arrives with a driving system called the Traffic Jam Pilot, which combines a lidar scanner with advanced sensor fusion and processing power with built-in component fail redundancies [17].

The US mandate for autonomous vehicles is to keep the driver engaged. Thus, the Audi A8L is classified as a level 2 automation in the United States, which preclude the key features for level 3 functionality. However, the Audi A8L vehicles in Europe are classified as level 3 automation.

3.4.5 Level 4 – High Automation

The major difference between Level 3 and Level 4 automation is that in the system in level 4, the vehicle can operate without expecting a user to respond to a request to intervene. In level 4, the system is fully responsible for driving tasks within limited service areas while occupants act only as passengers. However, a human still has the option to operate the vehicle manually. Examples of level 4 vehicles are NAVYA shuttles, cabs, and Waymo taxis.

3.4.6 Level 5 – Full Automation

In Level 5, the system is fully responsible for driving tasks while occupants act only as passengers [18]. Vehicles in this do not require any steering wheels or acceleration/braking pedals. They will be free from geofencing, able to go anywhere and do anything that an

experienced human driver can do. Fully autonomous cars are undergoing testing in several pockets of the world [17].

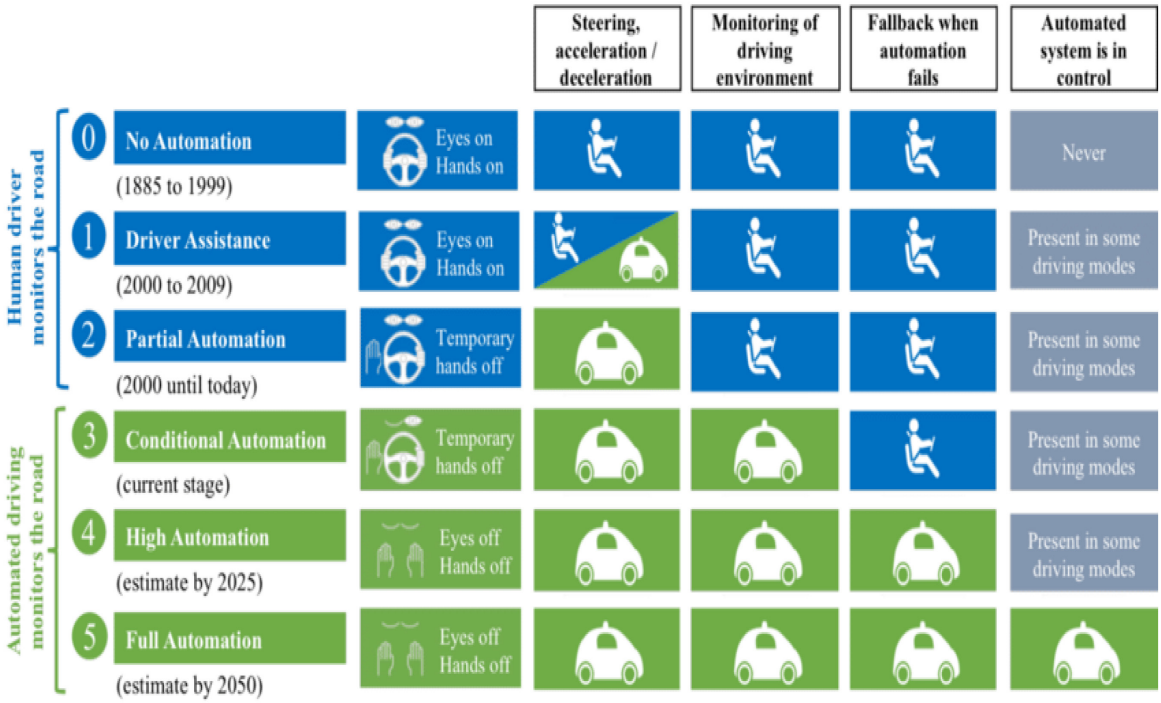


Figure 3-4: SAE J3016 summary and year projection [16].

Chapter 4: System Overview

The system is named OSCAR_PX4 (Open-Source robotic Car Architecture for Research and Education based on Pixhawk 4). The system design is a drive-by-wire system. In this case, a drive-by-wire system is a system that can control the steering, throttle, and brake programmatically without using any extra mechanical actuators. The Pixhawk 4 flight controller eliminated multiple embedded components previously used on related projects. The Pixhawk 4 flight controller is integrated with MAVLink for message protocol communication. The MAVLink communication protocol is developed with ROS (Robotic Operating System) nodes to communicate with ROS packages. The package that aids the communication between MAVLink and ROS is called MAVROS. The Pixhawk 4, also known as PX4, is suitable for this project because it has MAVROS software integrated with the board, allowing ROS topics, nodes, and packages to be used.

The EFI V Twin High-Speed Ride-On Car was chosen for a low budget with indoor and outdoor capability. The vehicle has four motors, and the power rate of each motor is 12 volts and 45 watts. A 12-volt battery powers the entire vehicle system. The Sabertooth 2x32 motor driver is a powerful motor driver for the motors inside the EFI V Twin High-Speed Ride-On Car. Sabertooth motor driver is a dual channel motor driver capable of supplying 32 amps to two motors, with peak currents of up to 64 amps per motor [19].

The process of combining, configuring, and communicating the Pixhawk 4 with software components and the Sabertooth motor drive is called the OSCAR_PX4 system. The combination and configuration setup instructions can be found in Appendix B:. The system has two working

modes. The first mode is used when a person wants to control the car remotely, collect driving data and assist the driving system. This mode is called the Manuel mode. The driver has complete functional control of the vehicle through the joystick F710 attached to the interface computer. The interface computer sends control data via the joystick to the Pixhawk 4; the Pixhawk 4 converts the control signal to a PWM signal with proper output to motor drive, allowing complete control of the car's braking, throttle, and steering. See Figure 4-1.

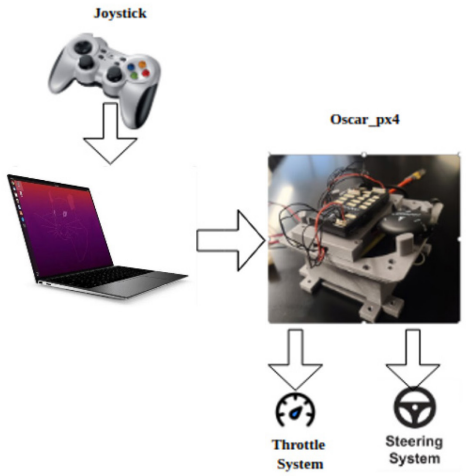


Figure 4-1: Manual mode setup.

The second mode is the autonomous mode. The vehicle can follow a pre-determent path and avoid collision without human intervention. This system mode integrates the sensor packages into the drive-by-wire system. A camera placed at the vehicle's bumper sends a direct feed to a companion computer integrated with ROS and MAVROS. The image from the environment with the driving data is sent to the neural network to process the information. It predicts the driving data based on the input image, sends predicted driving data to the PX4 via MAVROS, and converts it to the PWM signal for the motor drive, as shown in Figure 4-2.

The human driver drives the vehicle to collect training or driving data during the manual mode. The driving data will include the throttle, steering, and brake, images from the front

camera, and the odometry data from the vehicle's position. The driving data are stored for the training and testing phase of the neural network, as shown in Figure 4-3. In Figure 4-4, the Oscar_PX4 system plugin with the vehicle to implement the modes described for the system.

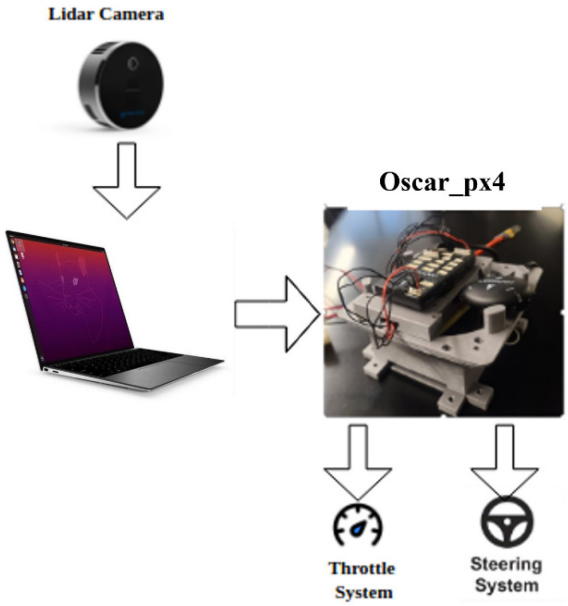


Figure 4-2: Autonomous mode setup.

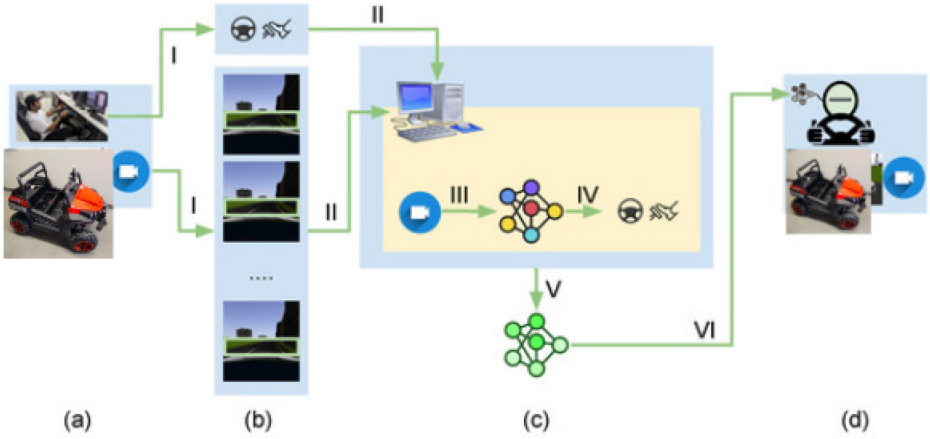


Figure 4-3: End-to-end driving. (a) A human driver drives a vehicle as we collect driving data. (b) The driving data, including the front camera images with synchronized control signals, are saved in storage. The collected data must have all the necessary features that can be expected in a testing phase of the neural network. (c) The training station is where a neural network is trained with the collected data to associate input with output. (d) The trained neural network is deployed to the AI chauffeur, who drives the vehicle using inferred steering angles, throttle, and brakes [20].

Figure 4-4: The OSCAR_PX4 system plugin with EFI V Twin High-Speed Ride-On Car.

4.1 Bill of Materials

The bill of materials (BOM) is the list of hardware used and the price at the time of the execution, as shown in Table 4-1. The huge problem in an autonomous vehicle system is road testing because of its economic implications. Road testing is costly. Therefore, designing a less expensive system is a useful solution to accelerate the production of the autonomous system. Furthermore, the Oscar_px4 vehicle platform is a flexible system, thus allowing different sensors fusion to be easy to integrate.

Table 4-1: Bill of Materials.

Item	Description	Q't	Price	Total
Camera	Intel Realsense L515	1	USD 550	USD 550
Pixhawk 4	Pixhawk flight controllers with 2MB flash memory and 512KB RAM	1	USD 150	USD 150
Pixhawk 4 GPS Module	UBLOX M8N module, IST8310 compass, tri-colored LED indicator, and a safety switch	1	USD 40	USD 40
Holybro Power Board PM07	Power distribution board with a PCB current 120 A output and 7~15 v input voltage	1	USD 18	USD 18
Sabertooth dual 32A motor driver	32A continuous, 64A peak per channel 6-30V nominal, 33.6V absolute maximum	1	USD 130	USD 130
HC-020K encode	Double Speed Measure Module with Photoelectric Encoder	1	USD 8	USD 8
Logitech F10 Wireles	2.4GHz Wireless Controller	1	USD 40	USD 40
Electric Ride-On Car	4 Wheel Motor drive with 2*12 v battery	1	USD 320	USD 320
Total of core items			USD 1256.00	USD 1256.00
Optional Items				
Laptop	i7 i770HQ 2.8GHZ 16GB RAM with GTX 1060 TI 6 GB GDDR	1	USD 1250	USD 1250
Total of optional Items			USD 1250.00	USD 1250.00

Chapter 5: Hardware

This section will elaborate on the hardware used in the project, explain their implementation requirements, and how they were implemented.

5.1 Pixhawk 4

Pixhawk 4 is an advanced autopilot designed and made in collaboration with Holybro and the Auterion team [21]. It is an inexpensive flight controller suitable for academic and commercial developers. It features the most advanced processor technology from STMicroelectronics, sensor technology from Bosch, InvenSense, and a Nuttreal- time operating system, delivering incredible performance, flexibility, and reliability for controlling any autonomous vehicle. In addition, the Pixhawk 4's microcontroller now has a 2MB flash memory and 512KB RAM with the power and RAM resources, allowing it to implement complex algorithms and models.

High-performance, low-noise IMUs on board, is designed for stabilization applications. Data-ready signals from all sensors are routed to separate interrupt and timer capture pins on the autopilot, permitting precise timestamping of sensor data. The added vibration isolations enable more accurate readings, allowing vehicles to reach better flight performance.

The two external SPI buses and six associated chip select lines allow additional sensors and SPI-interfaced payload. In addition, four I2C buses are dedicated for external use, and two are grouped with serial ports for GPS/Compass modules.

Pixhawk 4 is designed with easy-to-plug connectors making it easy to interact with various sensors; Figure 5-1 shows the function of each connector. Table 5-1 explains the detailed specifications of the Pixhawk 4.

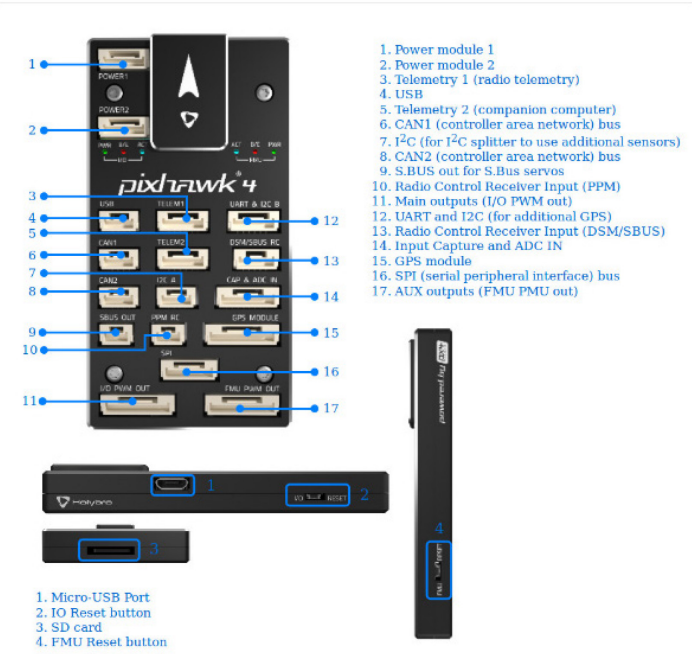


Figure 5-1: Pixhawk 4 Connectors [21].

Table 5-1: Pixhawk 4 Specifications [21].

Specification	**Electrical Data**	
	Voltage Ratings	**Mechanical Data**
Main FMU Processor: STM32F76 (32 bit ARM® Cortex® M7, 216 MHz Processor running NuttX RTOS)	Power module output: 4.9~5.5V	Dimensions: 44x84x12mm
Abundant connectivity options for additional peripherals (UART, I2C, CAN)	Max input voltage: 6V	Weight(plastic case): 33.3g
Redundant power supply inputs and automatic failover	Max current sensing: 120A	Weight(aluminum case): 49g
External safety button for easy motor activation	USB Power Input: 4.75~5.25V	
Sensors with higher temperature stability	Servo Rail Input: 0~36V	
Multicolor LED indicator		
Integrated vibrations isolation		
High-power, multi-tone piezo audio indicator		
microSD card for long-time high-rate logging		
16 PWM outputs		

5.1.1 Pixhawk 4 Configuration for OSCAR_PX4 System

For the OSCAR_PX4 design, there are some steps to set up the Pixhawk 4:

The first is to connect to the power module 1 connector (see Figure 5-1). Connecting to the power module is to interchange power from the power management board (PM07) with the Pixhawk 4. The connector provides 5v power output from the Pixhawk 4.

The second is to connect to the Main output connector (I/O PWM) (see Figure 5-1, labeled 11). The purpose of the connection is to provide the power management board (PM07) with I/O PWM outputs. In this case, the importance of the PWM outputs is to access the 8 PWM output from the 16 PWM outputs provided by the Pixhawk 4. In addition, PWM (pulse width modulation) is needed as a pulse signal to drive the motor controller.

Next is the connection to the GPS module (see Figure 5-1, labeled 15). The purpose of the connection is to provide the vehicle with a GPS sensor (Figure 5-2). The complete wiring of the system is in Appendix B:

Figure 5-2: GPS sensor [22].

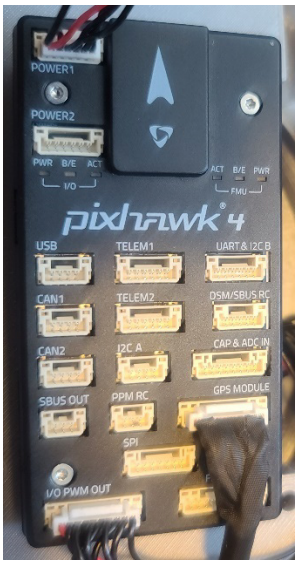


Figure 5-3: Pixhawk 4 wiring.

5.2 Power Management Board PM07

The Power Management Board (PM07 Board) serves the purpose of a Power Module and a Power Distribution Board [23]. In addition to providing regulated power to Pixhawk 4 and the ESCs, it sends information to the px4 about the battery's voltage and current supplied to the flight controller and the motors. As discussed in the previous section, Pixhawk 4 and PM07 board works together. The board's primary purpose is to expand and manage the power function of the Pixhawk 4.

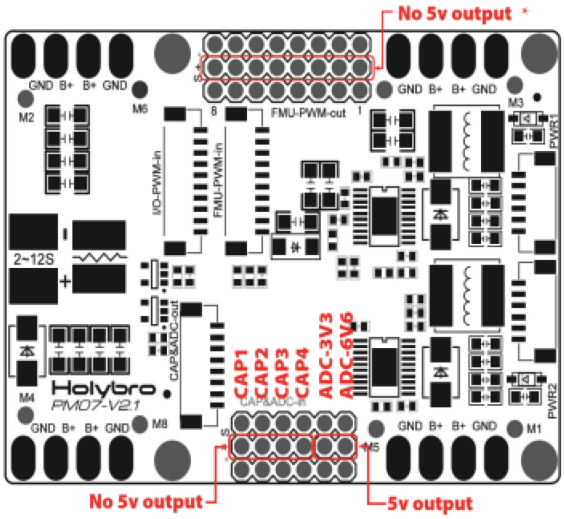


Figure 5-4: Power management board with Pixhawk 4 output schematics [24].

Table 5-2: PM07 with Pixhawk4 pins and connections [24].

PIN & Connector	Function
I/O PWM-IN	See table below for connection to Pixhawk 4
M1	I/O PWM OUT 1: connect signal wire to ESC of motor 1 here
M2	I/O PWM OUT 2: connect signal wire to ESC of motor 2 here
M3	I/O PWM OUT 3: connect signal wire to ESC of motor 3 here
M4	I/O PWM OUT 4: connect signal wire to ESC of motor 4 here
M5	I/O PWM OUT 5: connect signal wire to ESC of motor 5 here
M6	I/O PWM OUT 6: connect signal wire to ESC of motor 6 here
M7	I/O PWM OUT 7: connect signal wire to ESC of motor 7 here
M8	I/O PWM OUT 8: connect signal wire to ESC of motor 8 here
FMU PWM-IN	See table below for connection to Pixhawk 4
FMU PWM-OUT	If FMU PWM-IN is connected to Pixhawk 4, connect signal wires to ESC or signal, +, - wires to servos here
CAP&ADC-OUT	Connect to CAP & ADC IN port of Pixhawk 4
CAP&ADC-IN	CAP & ADC input: See back of the board for pinouts
B+	Connect to ESC B+ to power the ESC
GND	Connect to ESC Ground
PWR1	5v output 3A, connect to Pixhawk 4 POWER 1
PWR2	5v output 3A, connect to Pixhawk 4 POWER 2
2-12S	Power Input, connect to 2-12S LiPo Battery

5.2.1 Materials for Pixhawk 4 and PM07 Board

Several materials are needed to integrate Pixhawk 4 and PM07 to make the process easy. The materials used for OSCAR_PX4 Pixhawk and PM07 connection are:

1. Soldering iron
2. Solder
3. Soldering iron tips
4. Soldering iron holder and cleaning sponge
5. Wire stripper
6. Clips to hold your work
7. Exhaust fan
8. Magnifying Glasses
9. Male-to-male jumper wires
10. Male-to-female jumper wires
11. AWG Hook up Primary Wire(for the motors and motor drives)

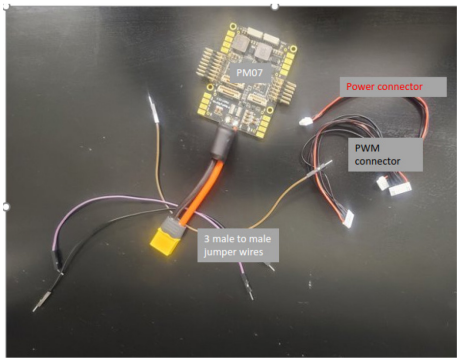


Figure 5-5: Material Pixhawk 4 and PM07 connections.

5.3 Sabertooth 2x32 [19]

Sabertooth 2x32 is a dual-channel motor driver capable of supplying 32 amps to two motors, with peak currents of up to 64 amps per motor. It can be operated from radio control, analog, TTL serial, or USB inputs. It uses regenerative drive and braking for efficient operation [19]. In addition, the sabertooth has various operating modes which allow custom operation, such as switching between radio control, computer-driven inputs, and emergency stop or panel control overrides.

The sabertooth can be used for a closed-loop position or speed control with an encoder or analog feedback. In addition, the driver can be monitored in real-time, making debugging more accessible and faster.

Table 5-3: Saberbooth Specifications [19].

	Mechanical specifications		
Dimensions	2.75 x 3.5 x 1.0 inches (70mm x 90mm x 26mm)		
Weight	4.5 ounces (125 grams)		
	Minimum	Typical	Maximum
Wire size, battery	16 gauge	10 gauge	10 gauge
Wire size, motors	16 gauge	12 gauge	10 gauge
Wire size, signal	28 gauge	24 gauge	18 gauge
Operating temperature	0F (-20C)	70F (25C)	160F (70C)[1]
	Electrical Characteristics		
	Minimum	Typical	Maximum
Input voltage, B+ and B-	6.0 Volts	12 or 24 Volts	33.6 Volts
Continuous output current, M1 and M2	-	-	32 amps[1]
Peak output current, M1 and M2	-	-	64 amps[2]
Output voltage, M1 and M2	-95% of input voltage (average)	-	95% of input voltage (average)
Voltage, P1 and P2	0V	-	Input voltage +.3V
Output current, P1 and P2			8 amps, sink only
Output voltage, 5V	4.85	5.0	5.15
Output Current, 5V	-	-	1A
Input voltage, S1 and S2	-.3V	0V to 5V	12V[3]
Input voltage, A1 and A2	-.3V	0V to 5V	12V[3]
Output Voltage, S2 and A2	0V		3.5V

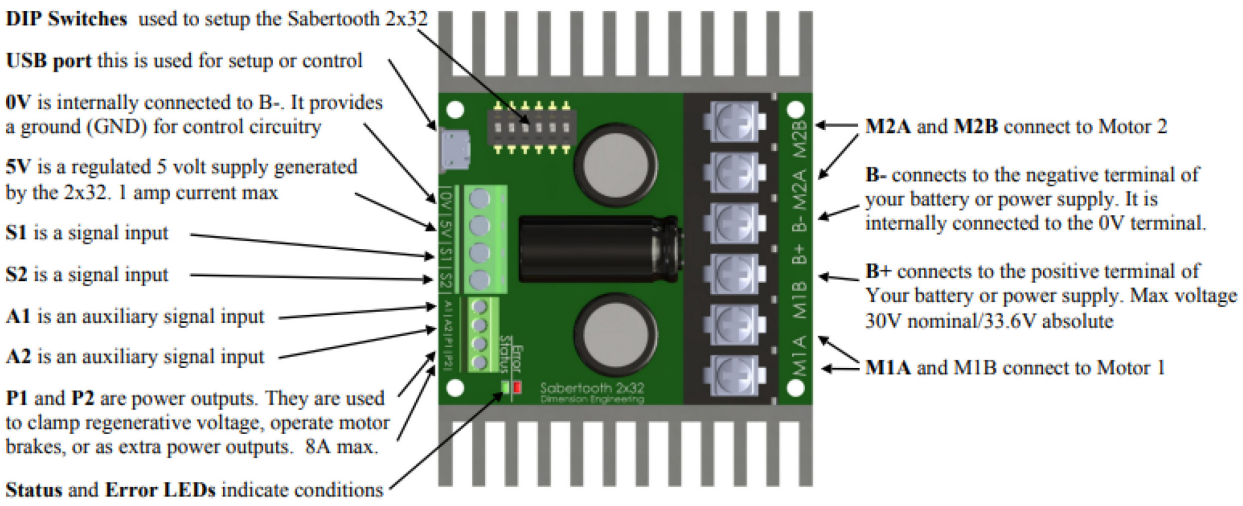


Figure 5-6: Sabertooth pinout functions [19].

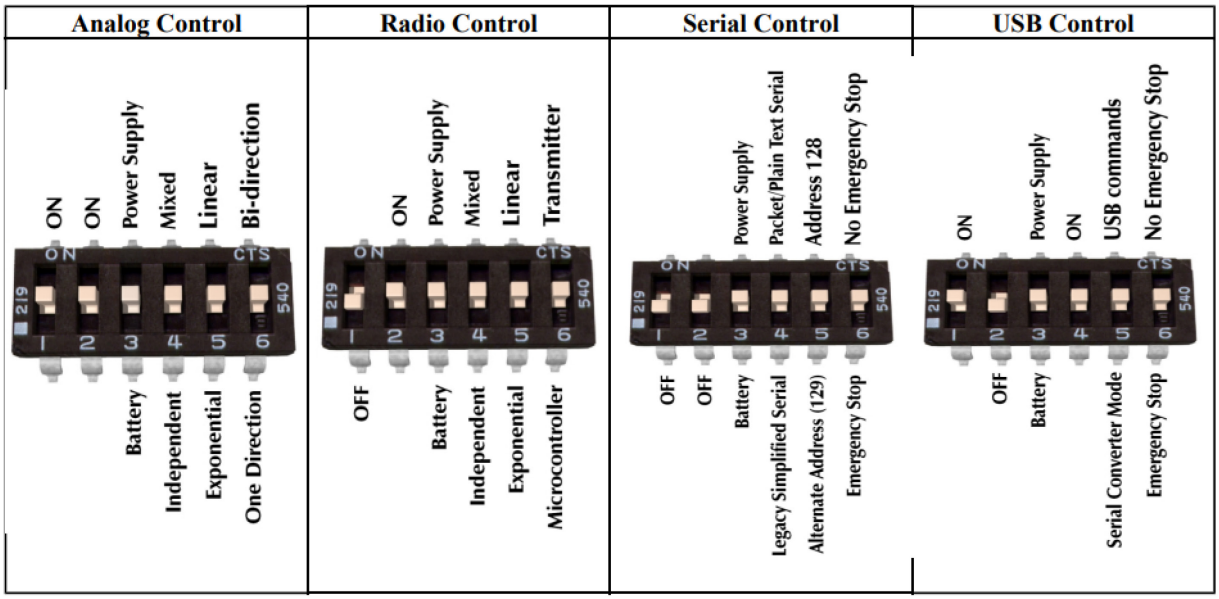


Figure 5-7: DIP Switches configurations [19].

For the Oscar_px4 design, there are some steps for wiring between the sabertooth and PM07 board:

The first is to connect s1 on the sabertooth to the M2 in the power management board. This is the PWM output signal that controls the throttle signal.

The second is to connect s2 on the sabertooth M4 in the power management board. This is the PWM output signal that controls the steering signal.

The third is to connect 0v on the sabertooth to the ground on the power management board. The ground connection is vital to create a reference point for the output signal. Without the ground, the s1 and s2 would not send signals to the motor drive (sabertooth).

On Sabertooth, M1A and M1B are connected to the throttle motor. On Sabertooth, M2A and M2B are connected to the steering motor. On Sabertooth, B+ and B- are connected to the 12V battery (to drive the motors). On Sabertooth, DIP switches 1 and 2 are in the OFF and ON positions, respectively. These disable the analog control mode and enable the radio control mode.

On Sabertooth, DIP switch 3 is the OFF mode. These indicate that power s supplied with a battery, protecting the board against the excess flow of current. On Sabertooth, DIP switch 4 is the OFF mode. These indicate the independent mode, which allows the signal from s1 and s2 (PWM signal) to be controlled independently. On Sabertooth, DIP switch 5 is in the ON mode. These indicate the linear mode, which allows the signal from s1 and s2 (PWM signal) to be controlled with a linear function. Finally, on Sabertooth, DIP switch 6 is in the ON mode. These indicate the Bi-directional mode, which allows the signal from s1 and s2 (PWM signal) to be controlled with positive and negative signs.

5.4 System Housing

Designing a secure system to house the component is essential. There are two 3d models intended for the oscar_px4; the first is to house the Pixhawk and power management board; the second provides housing for the motor controller. The current version of the model can be downloaded here.

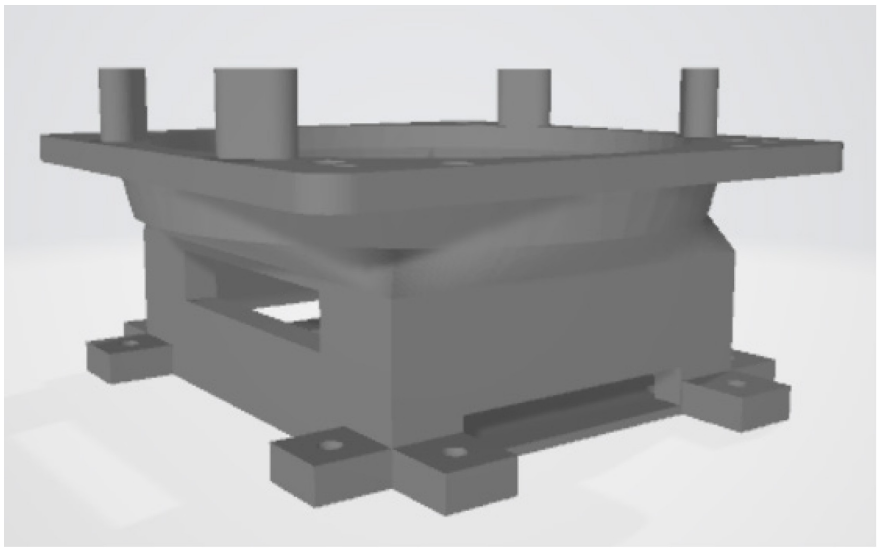


Figure 5-8: Sabertooth case.

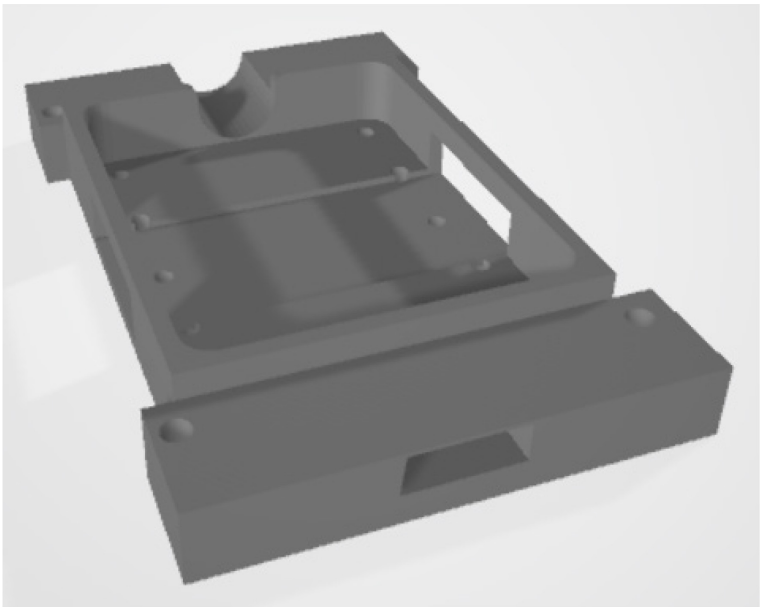


Figure 5-9: Pixhawk 4

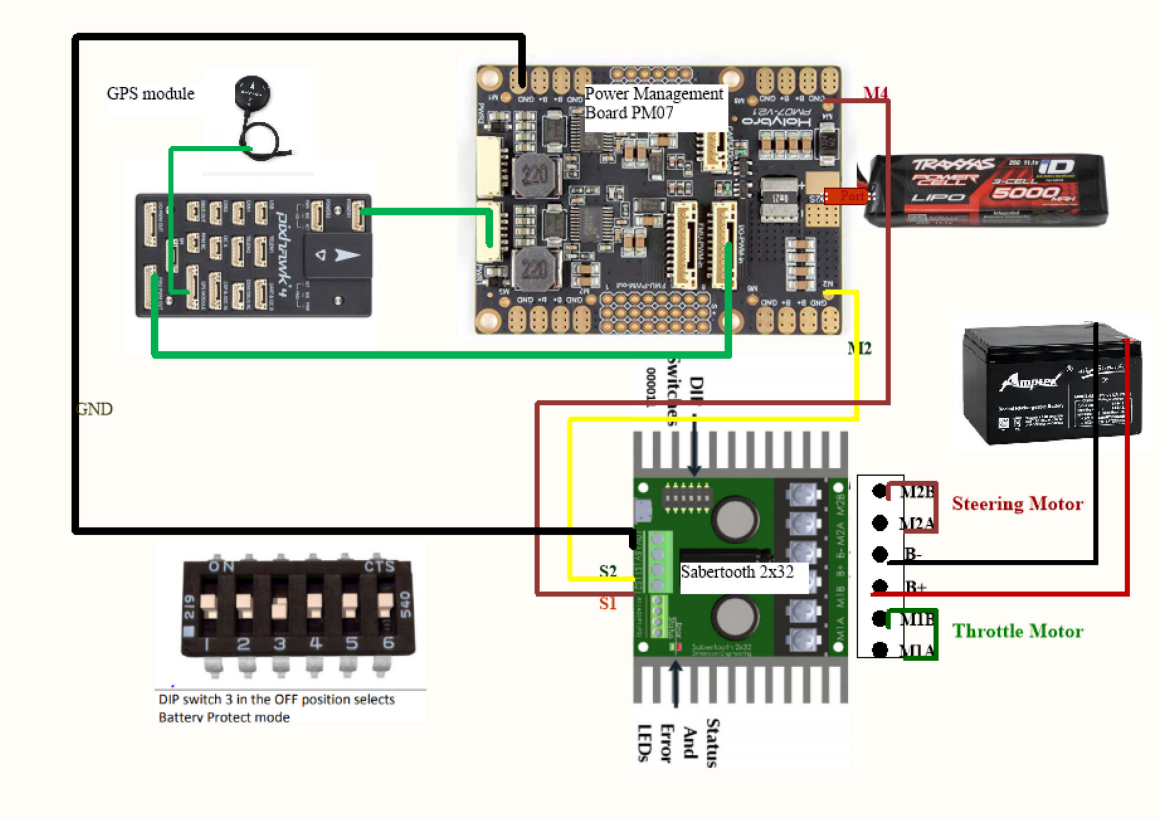


Figure 5-10: Complete drive-by-wire system.

5.5 HC-020k Encoder for Steering Control

The HC-020k encoder [25] is a photoelectric speed measurement module; it utilizes a spoked sensor disk and IR transmitter/receiver to measure the motor's rotation speed. It's useful to sense the motor's position and velocity. The HC-020k encoder takes an input of 5V. The connection mode is red 5V, and the black to GND with the third pin has the output voltage. The output voltage is digital output which ranges between 0V and 5V. 0V is when the beam is unblocked, allowing light to pass through it, and 5V is when the beam is closed. The flight controller can read the pulse train to determine the rotation of the motor. H2-020k encoder has a 3-pin header connector (with pull-up equipped input) that can be connected to different probe types. The sensor/probe hardware needs a pulse signal. The signal input accepts +5V TTL logic or open collector outputs. The maximum pulse frequency is 20 kHz with a 50 percent duty cycle.

The probe connector provides a +5V power supply from the I²C bus, the maximum power which could be used is limited by the RC filter.

HC-020k encoder is helpful in determining the steering angle or position. The steering control configurations are PWM signals sent from px4 to the motor drive needed to control the steering angle. Knowing the steering angle information is critical to measure the amount of power the motor drive needs to turn and its turning directions.

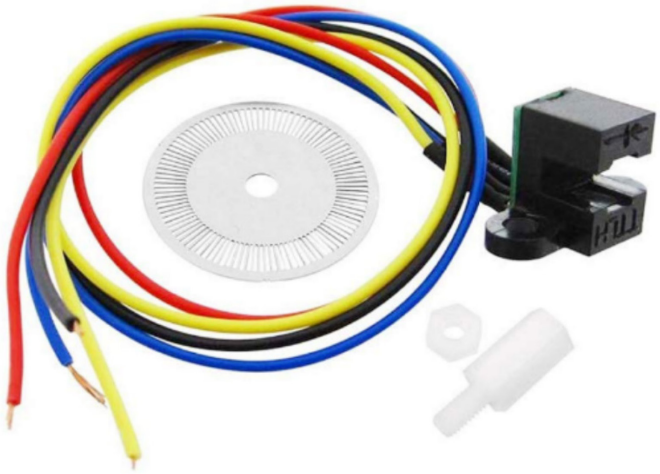


Figure 5-11: HC-020K encoder [25].

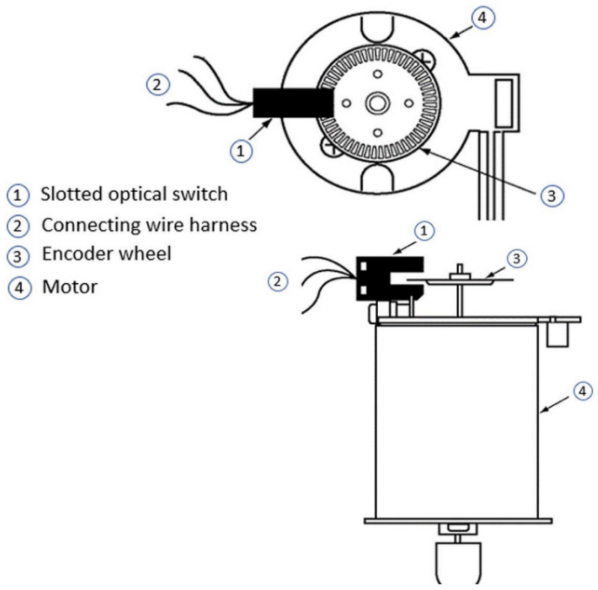


Figure 5-12: Schematic for HC-020k encoder setup [25].

The first step is knowing the max steering angle of the vehicle column in the left and right directions. For oscar_px4 (electrical toy) maximum steering angle to the right and left of our vehicle is 22.5 deg. The second is to connect the three pins from the encoder to the AUX output from PM07. Next, attach the wheel disk sensor to the steering column.

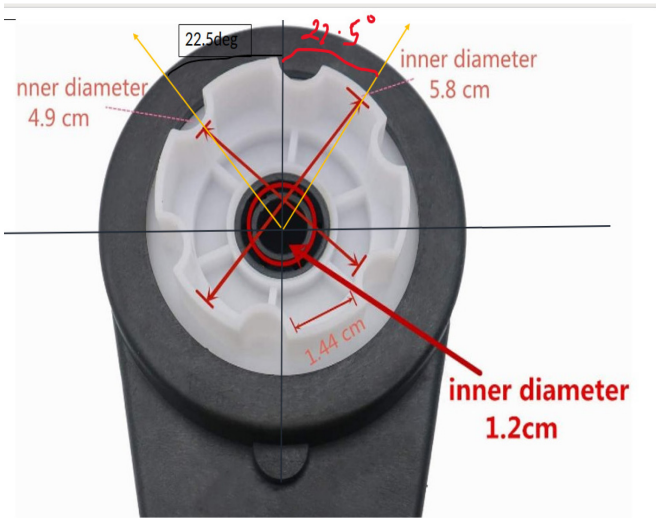


Figure 5-13: Maximum angle measurement from the vehicle motor and steering column.

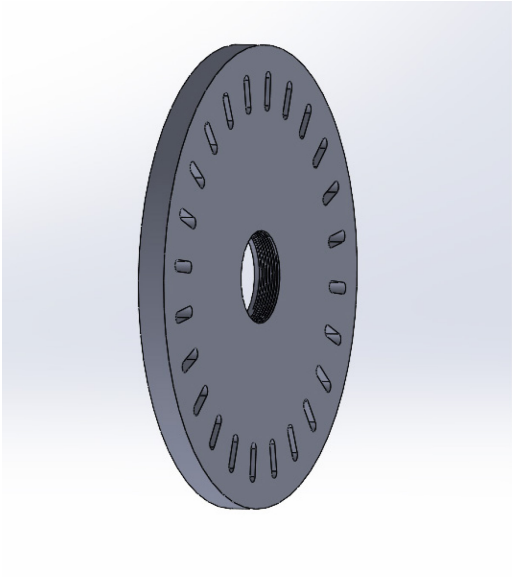


Figure 5-14: Spoked sensor disk for the wheel encoder.

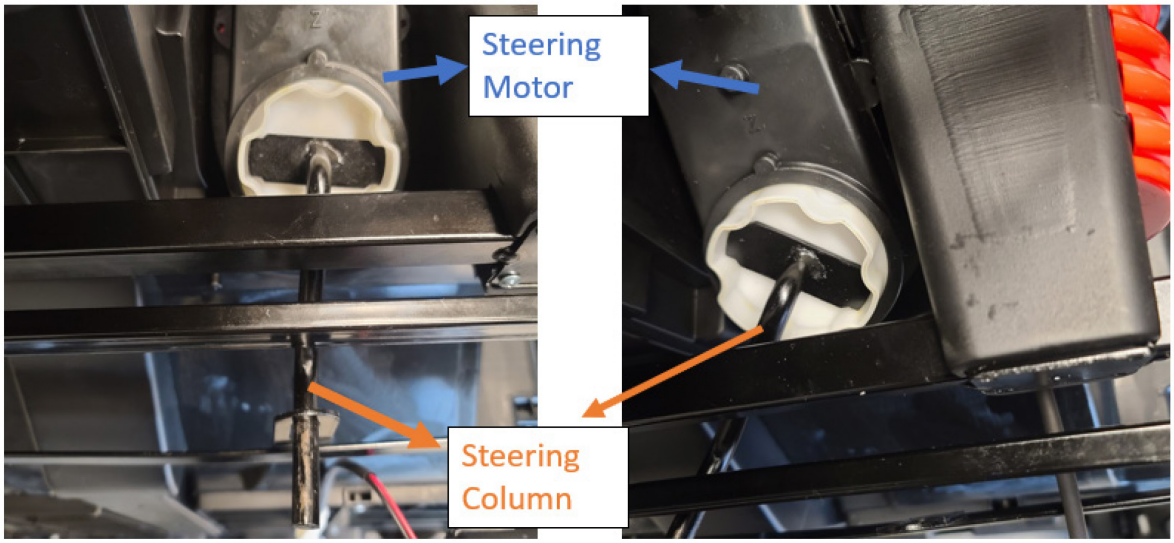


Figure 5-15: The ground view of the vehicle where the steering column is located.

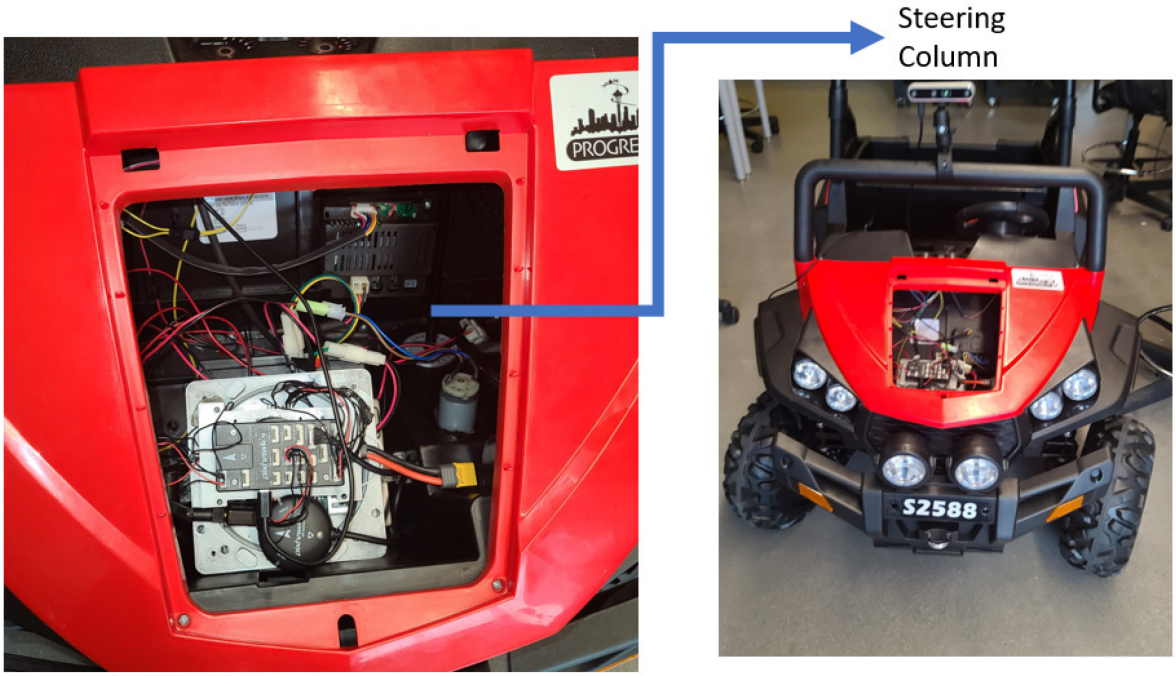


Figure 5-16: Image of the vehicle and the steering column location.

5.6 EFI V Twin High-Speed Ride-On Car

The EFI V Twin High-Speed Ride-On Car is the body of the vehicle which is approximately 1/4th scale electric ride-on car. The vehicle uses an Ackermann steering geometry arrangement, which allows the wheel to turn inner and outer at appropriate angles. Figure 5-17 and Table 5-4 show the vehicle body and its specifications.

Table 5-4: Vehicle Specifications [26].

Dimensions: 130 cm x 94 cm x 83 cm	Speed: 3-7 km/h.
Seat width: 58 cm	Drive: 4 x Motor 12V 45W
Sprung axles	Battery: 2x12 V
Soft EVA wheels	Charging time: 5-8 hours
Weight: 38,00 kg. Max load: up to 35 kg, tested load capacity: 55 kg	The driving time: 2 to 3 hours, depending on the weight and the terrain

Figure 5-17: EFI V Twin High-Speed Ride-On Car [26].

Chapter 6: Software

This section will be about the software used in the project, explaining their implementation requirements and how they were implemented.

6.1 ROS

ROS (Robot Operating System) is a set of software libraries and tools to help software developers create robot applications [27]. It provides services from a typical operating system, such as hardware abstraction, device drivers and controls, libraries, visualizers, message-passing between processes, implementation from commonly used functionality, and package management. In addition, ROS is an open-source platform that makes sharing and collaboration of robotic applications easy.

There are various goals for developing the ROS framework: First, a platform for sharing and collaboration for robotic application platform; a framework designed to be abstract as possible to make code written for ROS transferable to other robotic software frameworks; to make the ROS framework is easy to implement in any modern programming language; it is designed with a built-in test framework for diagnosis and visualization, and it's designed for large runtime systems and significant development processes.

The ROS platform currently runs entirely on Unix-based platforms. Software for ROS is primarily tested on Ubuntu and Mac OS X systems, though the ROS community has been contributing support for Fedora, Gentoo, Arch Linux, and other Linux platforms [27].

6.2 MAVLINK

MAVLink is a lightweight messaging protocol for communicating with drones and between onboard drone components [28]. It's a binary telemetry protocol designed for resource-constrained systems and bandwidth-constrained links. The MAVLink follows a modern hybrid publish-to-subscribe and point-to-point design pattern: The data streams are published as topics, while configuration sub-protocols are point-to-point with retransmission.

MAVLink has code generator software libraries for specific programming languages, created from XML message definitions, and can be used by drones, ground control stations, and other MAVLink systems to communicate [29].

The key features of using MAVLink are as follows: MavLink is very efficient because it doesn't require any additional framing. It is very well suited for applications with limited communication bandwidth. MAVLink is very reliable because it has proven to be resilient since 2009. It has been able to communicate between different vehicles, ground stations, nodes, and challenging communication channels with high latency and noise. MAVLink runs on various microcontrollers/operating systems with multiple programming languages. It supports up to 255 concurrent systems on the network.

6.3 MAVROS

MAVROS is an extendable communication node between MAVLink and ROS with the UDP proxy for Ground Control Station [30]. The communication features include communication with autopilot via serial port, UDP proxy for Ground Control Station, mavlink_ros compatible ROS topics (Mavlink.msg), plugin system for ROS-MAVLink translation, parameter manipulation tool, and waypoint manipulation tool.

6.4 QGroundControl

QGroundControl provides flight control and mission planning for any MAVLink-enabled drone. Its primary goal is ease of use for professional users and developers. The px4 platform designed has an open source provides integrating the system with the different airframes, such as rover, in this case, making it easy to implement and change the airframe parameters. The process of configuring the vehicle is loading firmware, setting the Airframe, sensors configuration, joystick, and parameter settings.

Chapter 7: Implementation

This section will elaborate on the components of implementation after designing the Oscar_px4 communication modes and discuss various neural networks and algorithms used in the experimental sections of the project.

7.1 Connection Interface

The connection interface is the stage after the configuration of the oscar_px4 in Appendix B:. The connection involves the Laptop, Remote control, Vehicle, and oscar_px4.

7.1.1 Laptop Computer

Complex perception and control algorithms based on deep learning libraries are executed on a GPU-powered computer. Therefore, running the algorithms on a laptop with good speed and GPU memory capacity is essential. The experiment was tested on a laptop Acer Predator Helios 300 powered by GTX 1060 6GB GDDR5; therefore, it is recommended to use GTX 1060 6Gb GDDR5 or above.

7.1.2 Remote Control

A Logitech F710 2.4 GHz wireless controller is used to send joystick commands that are translated to the control signals.

7.2 Communication

This subsection will explain connecting and communicating with the entire system.

- The first is to connect the nano receiver from the joystick to the computer. The connection will register to the computer as \dev\input\js*. The start "*" symbol indicates the number the device registered. However, if only one joystick device

is connected to the computer, it registers as \dev\input\js0. The "ls \dev\input\js*" command helps to check where the device is registered.

- The second is to connect the USB cable from the intelRealsense camera to the USB 3.0 port on the computer. **Note: if the camera is not connected to a USB 3 and above, the camera can not function**.
- In the oscar_px4 hardware system, connect the micro-USB from the pixhawk4 to the computer. The connection will register as "/dev/ttyACM0".
- In the oscar_px4 hardware system, connect the labeled throttle motor to the electric vehicle motor and connect the labeled steering motor to the electric vehicle motor connection. Then, connect the power from the motor drive to the 12-v battery attached to the vehicle.
- Then start the QgroundControl for the GUI interface.

7.3 Validation

The validation subsection elaborates on how a deep-learning-based research algorithm works with the OSCAR_PX4 system. Front-facing images must be captured and stored along with steering angles and throttle values information in the deep-learning-based behavior cloning system described on the Oscar page. The camera sensor provides the environment's perception. The manual control system enables the vehicle to be remotely controlled by a human driver's steering and throttle actions. The neural network model's actions after training were tested in the ELB's hallways.

7.3.1 System Architecture

The deep-learning neural network system architecture utilized to achieve end-to-end learning for vehicle control is shown in Figure 7-1. A human driver physically drove the vehicle

through the ELB hallway, collecting image sensor data with the encoder readings from the steering wheel during the data-gathering stage. The data collected was then fed into the neural network model for training. The results from training the neural network deploy to the vehicle's control system. Then, the OSCAR_PX4 system autonomously drove the vehicle through different paths.

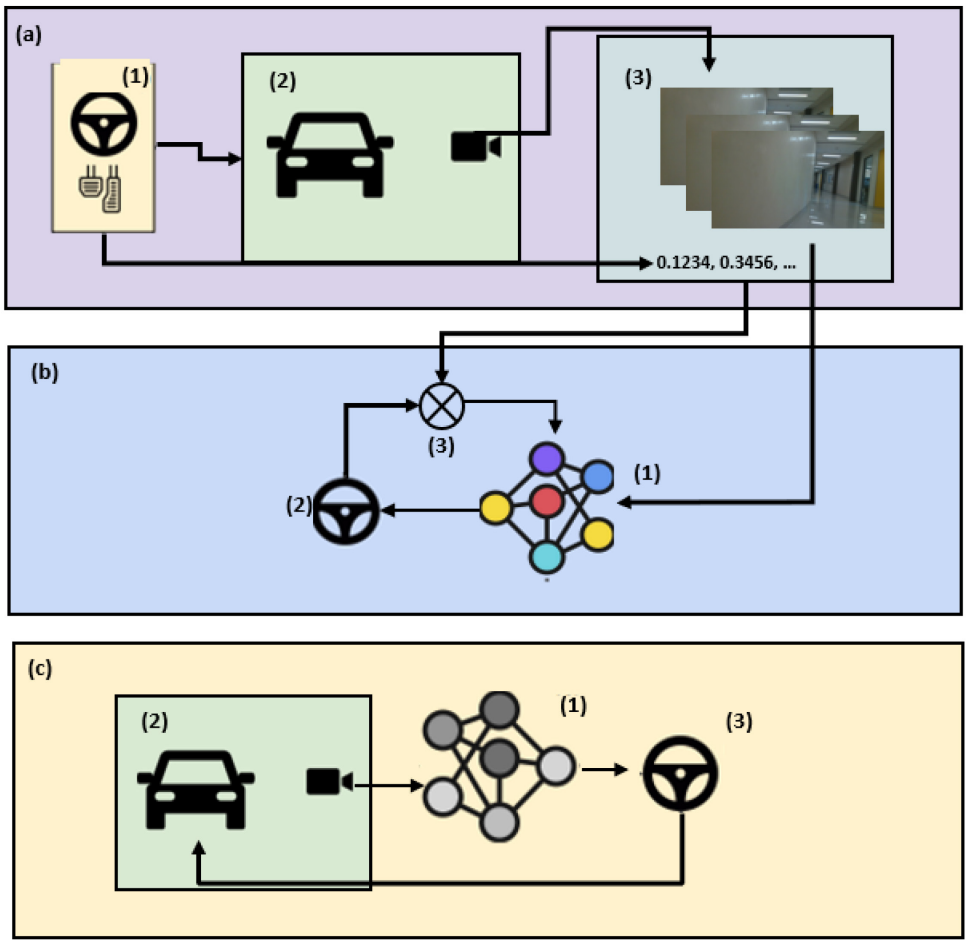


Figure 7-1: The system architecture of end-to-end behavioral cloning. (a) Data acquisition system: (a-1) human driver. (a-2) vehicle platform with the camera sensor. (a-3) collected data (images, steering, and throttle). (b) Neural network training: (b-1) input data. (. (b-2) steering angle prediction. (b-3) errors are fed to the neural network. (c) Testing the trained neural network: (c-1) trained neural network deployed. (c-2) vehicle platform. (c-3) steering angle predictions from the neural network are fed to the vehicle platform [20].

7.3.2 Data Acquisition

The primary task of the data acquisition process was to collect data for training the models. In order to collect images while measuring the steering and throttle actions, the Oscar px4 vehicle was driven along the ELB hallways. Six rounds were conducted to gather pictures and as much data as possible to get good results because the halls have a curved path with a few straight stretches. Therefore, the model's output depends on the data's quality. The vehicle is driven close to the center of the hallway during the data collection. Figure 7-2 shows a sample of RGB images collected with the Intelrealsense camera.

Figure 7-2: Examples of images collected from the camera.

7.3.3 Training Neural Network

The neural network for camera input was built using Keras. It is crucial to pre-process the data following the data acquisition process. The quality of the trained network depends on the quality of the data. Therefore, even with a small amount of poor data, the weights in the network may not be updated to optimized values. The results will lead to a significant discrepancy between projected and actual values, increasing the value of the cost function. The network model's cost function was calculated using mean squared error (MSE). The output of the trained model depends on the quality of the data. Therefore, it is necessary to filter out the bad-quality data collected. For example, the vehicle occasionally slammed into the wall due to the

narrowness of some of the halls it passed through. Consequently, at the pre-processing step, the data where the vehicle crash occurred was eliminated from the training data.

The steering readings range from −1 to 1, and the throttle value range from 0 to 1, where 0 means no throttle control. The network used mean squared error as a cost function. 70% of the data was used for training and 30% for validation.

Table 7-1 shows the architecture of the network for the camera model. The same architecture of Sharma [38] was used for the model, which only runs with RGB information. The number of images used for training this network was 180,000. The first layer is the batch normalization layer, followed by the 2D convolution layer and fully connected layers.

Table 7-1. Network architecture for the camera model [20].

Layer (Type)	Output Shape	Parameters
Lambda_1	(None, 70, 160, 3)	0
Conv2D_1	(None, 70, 160, 24)	1824
Maxpooling2D_1	(None, 69, 159, 24)	0
Conv2D_2	(None, 69, 159, 36)	21,636
Maxpooling2D_2	(None, 34, 79, 36)	0
Conv2D_3	(None, 34, 79, 48)	43,248
Maxpooling2D_3	(None, 17, 39, 48)	0
Conv2D_4	(None, 17, 39, 64)	76,864
Maxpooling2D_4	(None, 8, 19, 64)	0
Conv2D_5	(None, 8, 19, 64)	102,464
Maxpooling2D_5	(None, 4, 9, 64)	0
Flatten_1	(None, 2304)	0
Dropout_1	(None, 2304)	0
Dense_1	(None, 256)	590,080
Dropout_2	(None, 256)	0
Dense_2	(None, 128)	32,896
Dropout_3	(None, 128)	0
Dense_3	(None, 64)	8256

Chapter 8: Results

8.1 Manuel Mode

Experiments were designed to show successful operations of the vehicle using a remote controller to collect images from a camera. Figure 8-1 illustrates how the OSCAR_PX4 Vehicle platform accepts orders from the joystick while presenting sensor data using the 3D visualization tool (RViz).

8.2 Neural Network

The control system was divided into three sections based on environment information for validation: left turn, right turn, and straight pathways. The autonomous driving capabilities of the car were compared with those of the manual mode. The vehicle's performance in each of the three portions of the section was evaluated using the two control modes (Offboard mode and Manual mode).

The vehicle's driving performance was depicted in Figure 8-4 for each control mode executing a left turn, right turn, and straight drive, respectively. The performance data's statistical average and standard deviation are based on five rounds of driving test data. In this study, error bars are utilized to show the uncertainty or error corresponding to the variation of the result from the average value for each model’s driving data. Figure 7-3 shows the OSCAR_PX4 vehicle driving autonomously, heading to a right-turn path.

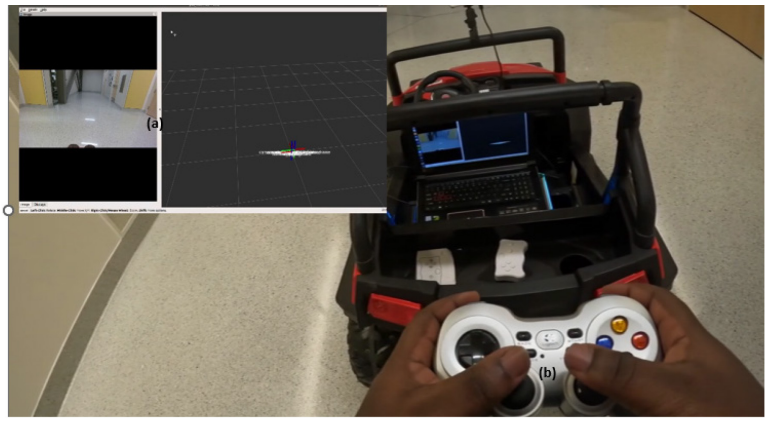


Figure 8-1: Rviz screen with multiple ROS nodes to use sensor packages. (Left): Remote control. (a) A screenshot of the ROS Visualization and our remote control ROS node. (b) A remote controller in action. Video: https://www.youtube.com/watch?v=DxVYBfYuDPo.

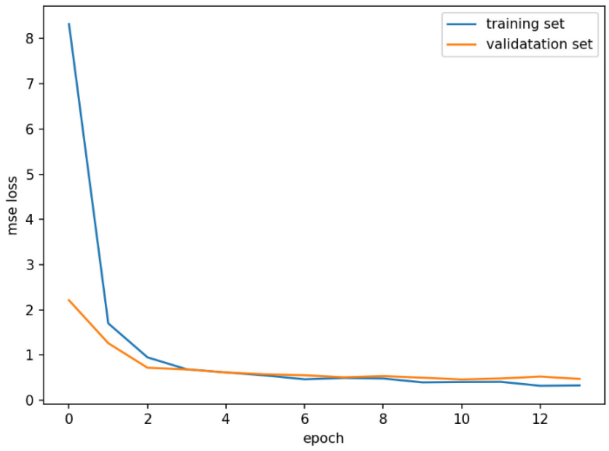


Figure 8-2: Comparing the validation set to the training set based on mean squared error.

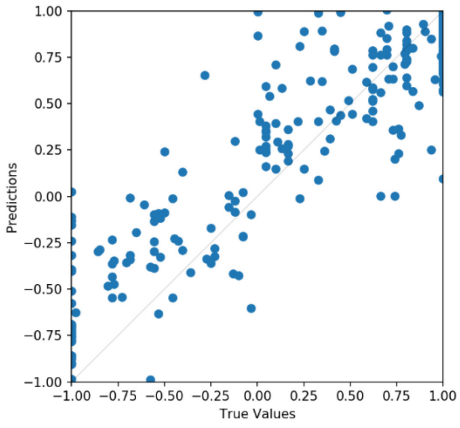
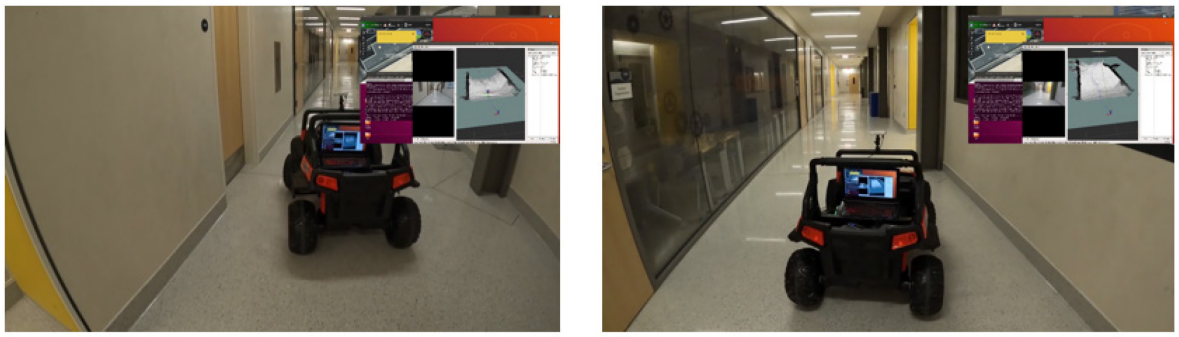


Figure 8-3: True value vs. Prediction result from testing the model.

Figure 8-4: The vehicle performance during neural network testing. (a) A screenshot of the Visualization interface turning right for actual and predicted data. (b) A screenshot of the Visualization interface turning left for actual data and predicted data. (c) A screenshot of the Visualization interface moving straight for actual and predicted data.



(a) **(b)**

Figure 8-5: OSCAR_PX4 steering right and light while driving autonomously. **(a)** Right turn to avoid the wall. **(b)** Left turn to avoid the wall. Video: https://www.youtube.com/watch?v=iN4jbnmJ2Cc.

Chapter 9: Discussion and Future Work

The research objectives section discussed the benefits and motivation of developing the OSCAR_PX4 platform. In summation, due to the high cost and safety concerns, employing a full-scale vehicle platform for research and educational purpose in autonomous driving systems is not a viable solution. Therefore, suitable alternatives, such as scaled vehicles and simulation environments, are required. Constraints like sensor package restrictions and computation problems are hard to eliminate in embedded systems for scaled vehicles. Although simulation is important, hardware-in-the-loop (HIL) testing of perception and control algorithms is essential. The Oscar_px4 platform overcomes all of these concerns while providing a viable alternative to employing a full-scale vehicle, as shown in Table 9-2.

The experiment results indicate that driving the OSCAR_PX4 vehicles using a remote controller to collect data from the camera sensor was a successful operation. These tests would not have been possible if the platform had not been well-designed. The Oscar px4 platform was considered appropriate for research applications requiring high computational capacity from the results of the end-end neural network algorithms.

Further work, the system can include a second neural network based on LiDAR data. In addition, add a sensor fusion to a third neural network model to fuse the data between the camera and the LiDAR. Similarly, test and create diverse experiments for indoor and outdoor applications (e.g., path planning). Furthermore, using a GPS sensor fused to the OSCAR_PX4 for waypoint navigation.

Table 9-1: Comparing related vehicle platforms with OSCAR_PX4

Name	Platform	Appro-ximate Cost ($)	Open Source	Region	ROS	Controller/CPU	Programming Tools	Car Scale
Turtlebot3 [2]	Education	800	YES	Indoor	YES	Raspberry Pi + OpenCR	Block-based/	1/12th
Leo Rover [4]	Research/ industry	2500	YES	Indoor/ Outdoor	YES	Raspberry Pi + Core2 ROS (low level)	ROS Programming	1/10th
Summit-XL [5]	Industry	15000	YES	Indoor/ Outdoor	YES	Intel processor/PC	ROS Programming	1/8th
MIT-racecar [1]	Research/ Education	2600	YES	Indoor	YES	NVIDIA Jetson Nano	ROS Programming	1/10th
F1TENTH [6]	Research/ Education	3400	YES	Indoor	YES	NVIDIA Jetson NX	ROS Programming	1/10th
OSCAR_PX4	Research/ Education	1200	YES	Indoor/ Outdoor	YES	PC+Pixhawk 4	ROS Programming	1/4th

Table 9-2: Comparing different options for drive-by-wire system development.

	Full-Scale Vehicle	RC-Based Car	Simulation Environment	OSCAR_PX4
Cost	High	Low	Low	Low
Saftey Concerns	High	Low	Low	Low
HIL	Yes	Yes	No	Yes
Onboard Computer	Yes	No	N/A	Yes
Deep-Learning Capabilities	Yes	No	Yes	Yes

Chapter 10: Conclusion

This paper presents the design and transformation of a mid-low level car platform equipped with various sensors with an affordable budget for testing autonomous vehicle algorithms and ADAS systems. The platform designed here is scalable and economically viable (approximate cost is shown in Table 4-1). The present sensors can be upgraded and substituted according to the user's requirements. Other workable options or a new version can be used to replace the lower control section made possible by Pixhawk 4. The Pixhawk 4, however, is a cable system that can be used for inside and outdoor experiments. In addition, better-performance laptops can take the role of the higher computation part. The study emphasizes the usage of ROS and Mavros, which allows to visualization of messages in numerical and visual formats with the help of a visualization tool, in addition to passing messages and controls across processes on different computing platforms.

Finally, the validation of the OSCAR_PX4 was conducted using deep-learning-based algorithms. The OSCAR_PX4 platform will be utilized for research and education, enabling the entire user community to contribute to the ADAS/AD study fields by testing and validating numerous unique algorithms.

Appendices

Appendix A: Installations

A.1: ROS

The OSCAR_PX4 was tested on the melodic as the ROS DISTRO. Other versions of the ROS can be located in the ROS website

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
sudo apt install curl
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
sudo apt update
sudo apt install ros-melodic-desktop-full
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
source ~/.bashrc
sudo apt install python-rosdep python-rosinstall python-rosinstall-generator python-wstool build-essential
sudo apt install python-rosdep
sudo rosdep init
rosdep update
```

A.2: MAVROS

The MAVROS and MAVLink can be installed using the following commands. The commands work for all ROS 1 DISTRO.

```
sudo apt-get install ros-$ROS_DISTRO-mavros ros-$ROS_DISTRO-mavros-extras
wget https://raw.githubusercontent.com/mavlink/mavros/master/mavros/scripts/install_geographiclib_datasets.sh
chmod a+x install_geograph
sudo ./install_geographiclib_datasets.sh iclib_datasets.sh
```

A.3: Intel RealSense

The camera used in Oscar_px4 is the Intel RealSense L515 camera. The following commands will help install Intel RealSense SDK packages and ROS packages.

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-key F6E65AC044F831AC80A06380C8B3A55A6F3EFCDE || sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-key F6E65AC044F831AC80A06380C8B3A55A6F3EFCDE

sudo add-apt-repository "deb https://librealsense.intel.com/Debian/apt-repo $(lsb_release -c

sudo apt-get install librealsense2-dkms

sudo apt-get install librealsense2-utils

sudo apt-get install ros-$ROS_DISTRO-realsense2-camera

sudo apt-get install ros-$ROS_DISTRO-realsense2-description
```

To test if the packages are properly installed, run the following commands.

```
realsense-viewer

roslaunch realsense2_camera rs_camera.launch

rostopic list
```

A.4: Clone the Oscar repository

The oscar page explains the installation process and instructions for the project process.

A.4.1: Install anaconda

In the oscar page, the conda environment is used to install the libraries and python packages needed for the project. Therefore, installing anaconda must be done before cloning the repository.

```
sudo apt-get update

sudo apt-get install curl

sudo apt-get install git

cd /tmp

curl {O https://repo.anaconda.com/archive/Anaconda3-2020.02-Linux-x86_64.sh

sha256sum Anaconda3-2020.02-Linux-x86_64.sh
```

```
bash Anaconda3-2020.02-Linux-x86_64.sh
```

A.4.2: Clone oscar

```
sudo apt-get update
 sudo apt-get install curl
sudo apt-get install git
git clone -b devel_mrover https://github.com/jrkwon/oscar.git --recursive
sudo apt install ros-$ROS_DISTRO-fake-localization
sudo apt install ros-$ROS_DISTRO-joy
cd oscar
conda env create --file config/conda/environment.yaml
conda activate oscar
source ./setup.bash
```

A.5: QgroundControl

For the oscar_px4, QgroundControl was the ground station interface used to visualize the communication between MAVLink, MAVROS, and sensors connected to Pixhawk 4.

A.5.1: Before installation

```
sudo usermod -a -G dialout $USER sudo apt-get remove modemmanager -y
sudo apt install gstreamer1.0-plugins-bad gstreamer1.0-libav gstreamer1.0-gl -y sudo apt install libqt5gui5 -y
```

A.5.2: Restart the computer

```
sudo reboot now
```

A.5.3: Download QGroundControl.AppImage.

A.5.4: Installation

```
chmod +x ./QGroundControl.AppImage
./QGroundControl.AppImage
```

Appendix B: Configuration and Setup

The QGroundControl requires some configuration and setup for the initial setup.

B.1: QgroundControl Setup

B.1.1: Loading Firmware

For the firmware, choose PX4 Pro Stable Release v1.12, and follow the prompt. Note that any version above v1.12 would work.

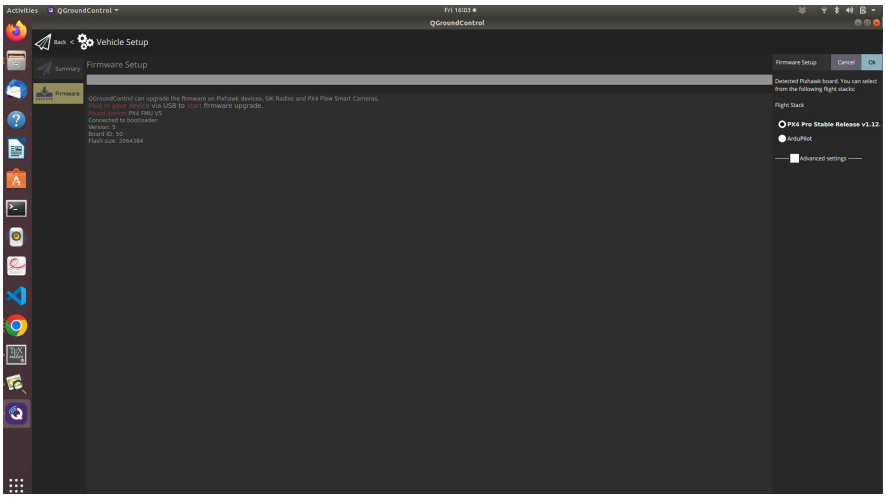


Figure B-1: Loading Firmware.

B.1.2: Airframe.

Select the Airframe in PX4: After connecting the QGroundControl with the vehicle. Select the QGroundControl icon and vehicle setup and click the Airframe as shown in Figure B-2. Then, select the generic ground vehicle, as shown in Figure B-3. Click Apply in the following prompt to save the settings and restart the vehicle.

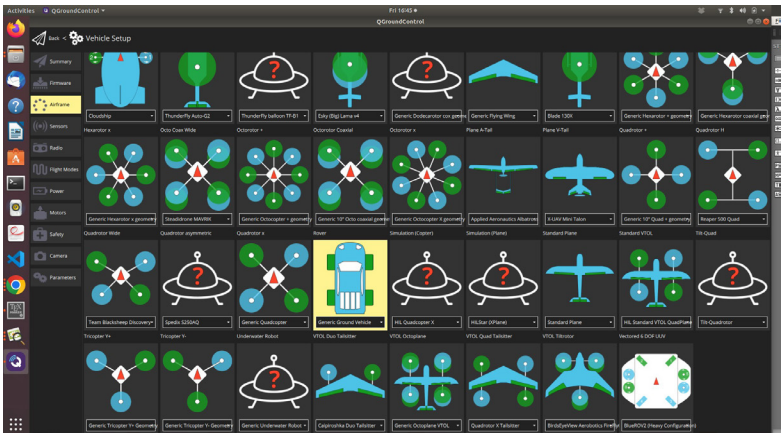


Figure B-2: Vehicle Airframe.

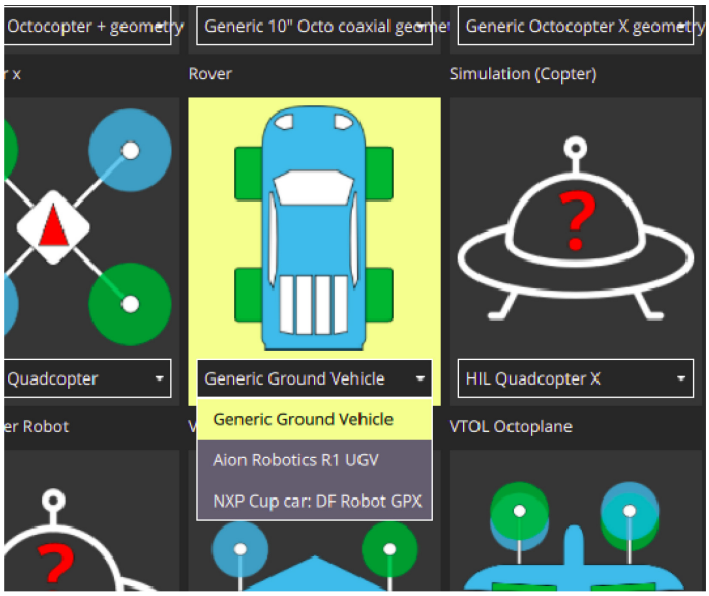


Figure B-3: Generic Ground Vehicle.

B.1.3: Sensors Configuration

The Sensor Setup section allows the configuration and calibration of the vehicle's compass, gyroscope, and accelerometer. The available sensors are displayed as a list of buttons beside the sidebar, as shown in Figure B-4. Sensors marked with green are already calibrated, while sensors marked with red require calibration before arming the vehicle. Sensors with no light are simple settings with default values that you may choose not to calibrate

1. Compass: The process guides the vehicle's position in several sets of orientations and rotates the vehicle about the specified axis.
2. The calibration process is to click on the button for each sensor to start and follow the prompt.
3. Accelerometer: The accelerometer calibration process is to place and hold the vehicle in several orientations and follow the prompt.

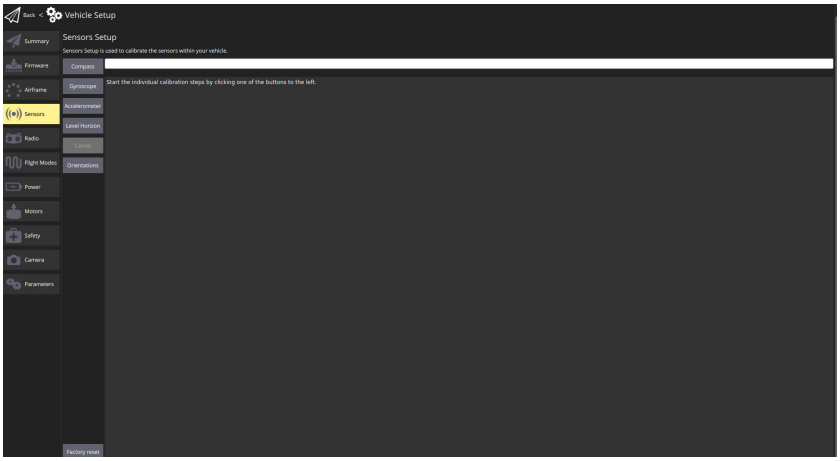


Figure B-4: Sensors Configuration

B.1.4: Joystick.

In the absence of an RC transmitter or a Radio channel controller, the joystick is a good substitute. For the Joystick support in PX4, the default parameter must change from COM RC IN MODE to 1 -*Joystick*/No RC Checks. This enables the joystick and disables the RC parameter check for arming the vehicle.

The joystick needs to be calibrated and configured for ground rover settings. On QGroundControl, after connecting to a vehicle(px4) and joystick using the USB port o the computer. In the vehicle setup on QGroundControl, the Gear icon indicating the joystick should be selected as shown in Figure B-5 and follow settings as shown in Figure B-5 and Figure B-6.

Some additional options are available in the Advanced tab. These options are useful for specific, unusual setups, increasing sensibility, and handling noisy joysticks. Center stick is zero throttles: Centered or lowered stick sends 0 in MANUAL CONTROL z, raised stick sends 1000 as shown in figure 8

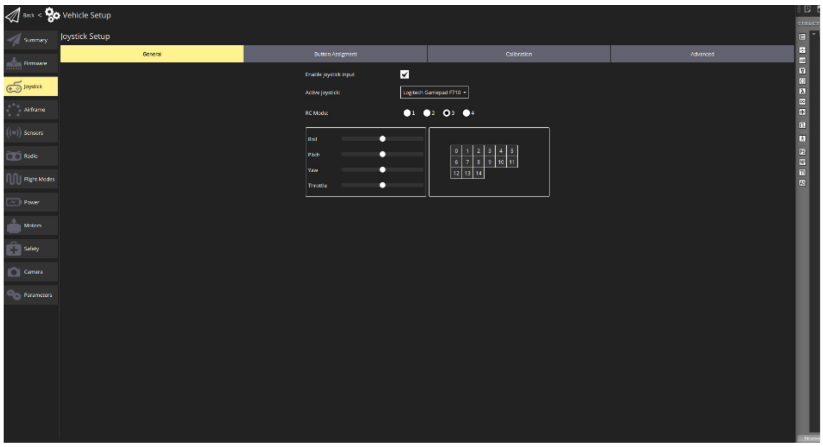


Figure B-5: Joystick Setup

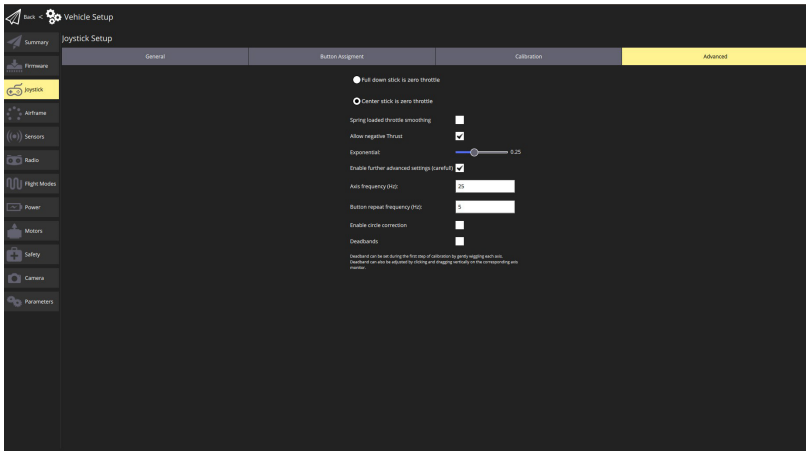


Figure B-6: Joystick Advanced Setup.

B.1.5: Parameters

The Parameters screen allows finding and modifying any of the parameters associated with the vehicle. To change the value of a parameter, click on the parameter row in a group or search list, as shown in Figure B-7. These will open a side dialog, which allows updating the parameters (the dialog provides additional detailed information about the parameter)

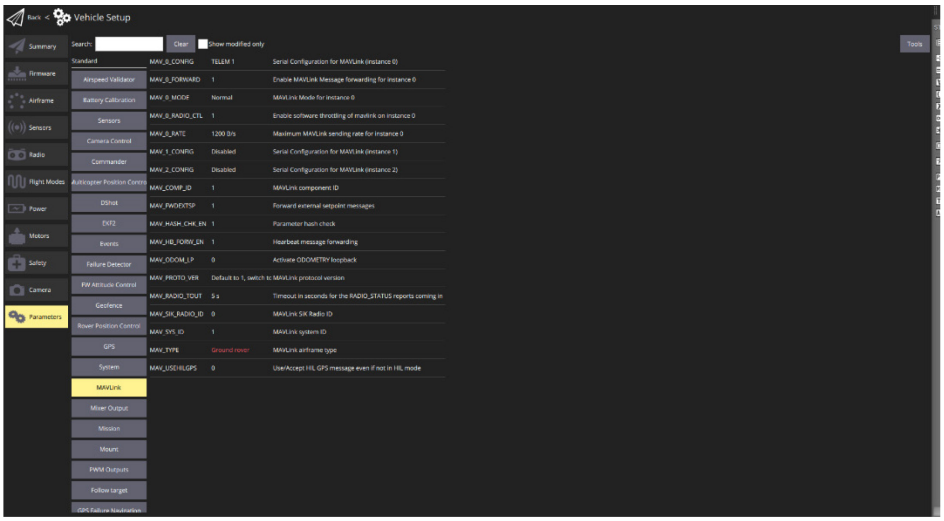


Figure B-7: Parameters

References

[1] S. Karaman, A. Anders and J. Vivilecchia, "Project-based, Collaborative, Algorithmic Robotics for high school students: Programming self-driving race cars at MIT," *2017 IEEE Integrated STEM Education Conference (ISEC),* 2017.

[2] "ROBOTIS.US," ROBOTIS, INC., 2016. [Online]. Available: https://www.robotis.us/turtlebot/.

[3] ChrisBogdon, "ros.org: TurtleBot," 01 2020. [Online]. Available: https://wiki.ros.org/Robots/TurtleBot.

[4] "Leo Rover: Robot developer kit: Open-source: Ros and for outdoor use.," [Online]. Available: https://www.leorover.tech/#learn-more.

[5] Robotnik, "Summit-XL Mobile Robot - Indoor & Outdoor: Robotnik®," 10 2022. [Online]. Available: https://robotnik.eu/products/mobile-robots/summit-xl-en-2/. [Accessed 11 2022].

[6] "F1TENTH," 2020. [Online]. Available: https://f1tenth.org/. [Accessed 10 2022].

[7] "SciHi Blog," 03 01 2022. [Online]. Available: http://scihi.org/robert-whitehead-torpedo/. [Accessed 1 10 2022].

[8] HowStuffWorks, "1959 Cadillac cyclone," HowStuffWorks, 10 2007. [Online]. Available: https://auto.howstuffworks.com/cadillac-cyclone.htm. [Accessed 10 2022].

[9] A. Reiser, "TOMORROW'S WORLD TODAY®," 24 06 2022. [Online]. Available: https://www.tomorrowsworldtoday.com/2021/08/09/history-of-autonomous-cars/. [Accessed 5 10 2022].

[10] DARPA, "The DARPA Grand Challenge: Ten Years Later," 13 03 2014. [Online]. Available: https://www.darpa.mil/news-events/2014-03-13. [Accessed 20 01 2022].

[11] S. Russel and P. Norvig, "Intelligence Agents," in *Artificial Intelligence: A Modern Approach*, New Jersey, Pearson, 2009, p. 34.

[12] D. M. West and J. R. Allen, "Brookings," 069 03 2022. [Online]. Available: https://www.brookings.edu/research/how-artificial-intelligence-is-transforming-the-

world/. [Accessed 2 10 2022].

[13] J. Hays and A. A. Efros, ""Scene Completion Using Millions of Photographs," *SIGGRAPH,* 2007.

[14] K. Chellapilla, S. Puri and P. Simard, "High Performance Convolutional Neural Networks for Document Processing," 10 2006.

[15] K. Muhammad, J. Ahmad, I. Mehmood, S. Rho and S. Baik, "Convolutional Neural Networks Based Fire Detection in Surveillance Videos," *IEEE Access,* vol. PP, pp. 1-1, 03 2018.

[16] "J3016A: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," *SAE International,* p. 14, 08 2016.

[17] S. Automotive, "The 6 levels of vehicle autonomy explained.," 22 03 2022. [Online]. Available: https://www.synopsys.com/automotive/autonomous-driving-levels.html . [Accessed 20 09 2022].

[18] NHTS, "Automated vehicles for safety," 05 2018. [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety.

[19] "Sabertooth Product description," June 2019. [Online]. Available: https://www.dimensionengineering.com/products/sabertooth2x32. [Accessed 15 June 2022].

[20] A. Khalil, . A. Abdelhaed,, . G. Tewolde and . J. Kwon, "Ridon Vehicle Drive-by-Wire System for Scaled Vehicle Platform and Its Application on Behavior Cloning," *Energies,* vol. 8039, p. 14, 2021, doc 10.3390/en14238039.

[21] "Pixhawk 4," 24 08 2021. [Online]. Available: https://docs.px4.io/main/en/flight_controller/pixhawk4.html. [Accessed 10 09 2022].

[22] Holybro, "Holybro M8N GPS," 08 2019. [Online]. Available: http://www.holybro.com/product/pixhawk-4-gps-module/. [Accessed 23 06 2022].

[23] "Pixhawk 4 power module PM07," 01 12 2019. [Online]. Available: http://www.holybro.com/product/pixhawk-4-power-module-pm07/. [Accessed 25 08 2022].

[24] P. Q. S. G. -. Holybro, 05 10 2019. [Online]. Available: http://www.holybro.com/manual/PM07-Quick-Start-Guide.pdf. [Accessed 15 09 2022].

[25] H.-0. M. S. S. Module, "ProtoSupplies," ProtoSupplies, 08 06 2022. [Online]. Available: https://protosupplies.com/product/hc-020k-motor-speed-sensor-module-2-pack/. [Accessed 1 10 2022].

[26] "BeneoShop.com," BeneoShop.com, 10 2015. [Online]. Available: https://www.beneoshop.com/electric-ride-on-toy-car-rsx-red-2-4ghz-buggy-24v-4-x-motor-remote-control-two-seats-in-leather-soft-eva-wheels-fm-radio-bluetooth.html. [Accessed 11 2022].

[27] A. Dattalo, "ros.org," Open Robotic , 08 08 2018. [Online]. Available: http://wiki.ros.org/ROS/Introduction. [Accessed 2022 10 05].

[28] H. Willee, "Introduction · ham_mavdevguide," [Online]. Available: https://hamishwillee.gitbooks.io/ham_mavdevguide/content/en/.

[29] H. Willee, "Overview · ham_mavdevguide," [Online]. Available: https://hamishwillee.gitbooks.io/ham_mavdevguide/content/en/about/overview.html. [Accessed 20 9 2022].

[30] "MAVROS (MAVLink on ROS) · PX4 Developer Guide," [Online]. Available: https://dev.px4.io/v1.10_noredirect/en/ros/mavros_installation.html. [Accessed 07 2022].

[31] Vooon, "ros.org," 03 03 2018. [Online]. Available: http://wiki.ros.org/mavros. [Accessed 05 10 2022].

[32] L. Meier, "Introduction · MAVLink Developer Guide," 05 08 2015. [Online]. Available: https://mavlink.io/en/. [Accessed 05 10 2022].