Smart Steering Wheel Project

Implemented by:

Mahmoud Sami Mouwed Rizk - (LEADER)
Mohamed Samir Ahmed Badawi
Mahmoud Ahmed Ali Ahmed
Abdullah Fathy Mohamed El-Zohiery
Mohamed Osama Hussein Aboubakr
Abdelrahman Morsy Adly Morsy

A report submitted in partial fulfilment of the requirement for the “Egypt Make Electronics: Embedded System Track”.

Information Technology Institute

November 2023

TABLE OF CONTENTS

LIST OF TABLES

TABLE **PAGE**

LIST OF FIGURES

FIGURE **PAGE**

LIST OF ABBREVIATIONS AND SYMBOLS

A	
ADAS	Advanced Driving Assistance System
ARM	Advanced RISC Machine
C	
CAD	Computer-Aided Design
CAN	Controller Area Network
F	
FOTAMOTIVE	Flash Over the Air for Automotive
G	
GUI	Graphical User Interface
N	
NMS	Non-Maximum Suppression
P	
PCB	Printed Circuit Board
PID	Proportional-Integral-Derivative
R	
RISC	Reduced Instruction Set Computer
V	
VSM	Virtual System Modelling
Y	
YOLOv7	You Only Look Once version 7

CHAPTER 1

INTRODUCTION

1.1 GENERAL OVERVIEW

For more than a century, the hydraulic system has dominated vehicle design. It allowed for precise handling and, with adequate maintenance, proved to be durable when paired with manual steering gearboxes. Drivers may have exhibited inefficient driving practices, such as aggressive acceleration, sudden braking, or improper lane changes, without real-time feedback or guidance to encourage safer and more fuel-efficient driving habits. However, thanks for the vehicles' ever-increasing technological complexity. Most of the issues are solved using ADAS.

Advanced Driver Assistance Systems, commonly known as ADAS, refer to a collection of technologies and features designed to assist drivers and enhance safety on the road. ADAS technologies utilize sensors, cameras, radar, and other advanced components as shown in Figure 1.1, to monitor the vehicle's surroundings, detect potential risks or hazards, and provide alerts or take autonomous actions to prevent accidents or mitigate their severity.

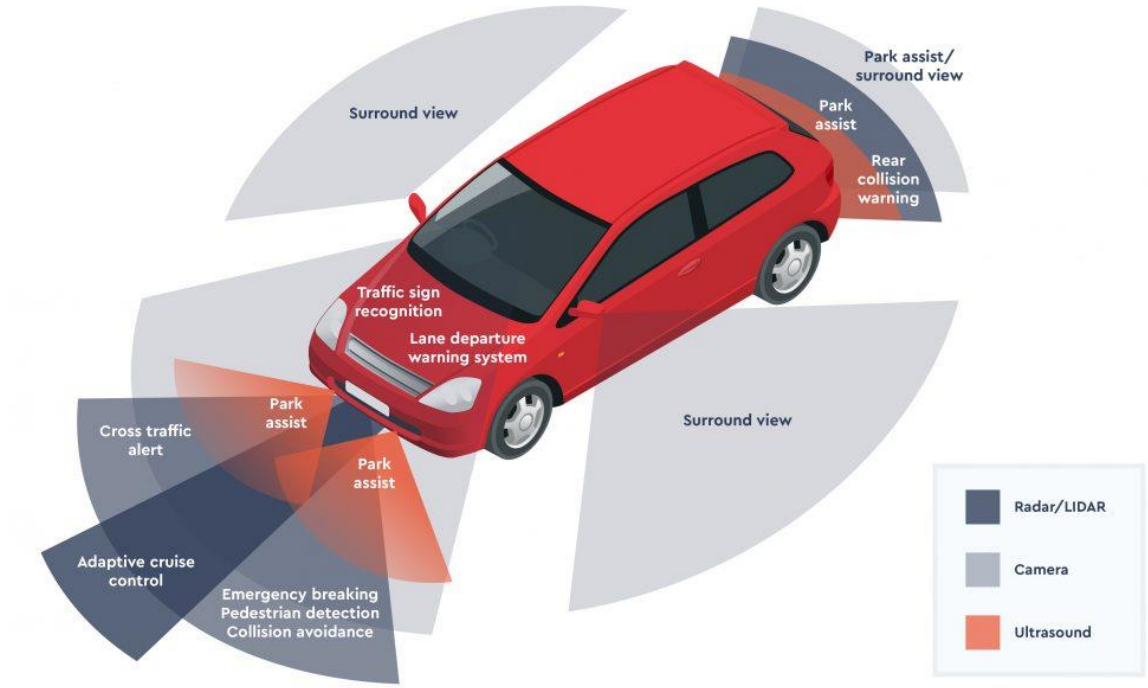

Figure 1.1 Car's Sensors

ADAS is classified into the following levels:

- Level 0 ADAS systems provides no automation, the driver is entirely responsible for managing the vehicle, including steering, braking, accelerating, and slowing down.
- Level 1 ADAS systems provide individual features that assist the driver in specific functions, such as adaptive cruise control or lane keeping assist.
- Level 2 ADAS systems offer a combination of features that can simultaneously control aspects of steering, acceleration, and braking.
- Level 3 ADAS introduces conditional automation, allowing the vehicle to take over the driving tasks under certain conditions.
- Level 4 ADAS systems enable high automation, where the vehicle can operate independently in specific driving conditions or environments.
- Level 5 ADAS represents full automation, where the vehicle can execute all driving tasks and operates without the need for human intervention.

1.2 PROBLEM STATMENT

Before the introduction of ADAS, drivers faced several challenges and limitations that impacted their safety and their overall driving experience. The problem statement encompasses the issues and risks associated with driving in the absence of ADAS technologies. Some key problem areas include:

- Drivers had limited awareness of their surroundings, making it difficult to detect potential hazards or dangers on the road. Without ADAS systems, drivers relied solely on their own visual observation and judgment, which could be prone to errors or oversights.
- Human error, including distractions, fatigue, and inattention, played a significant role in driving accidents. Drivers faced challenges in maintaining consistent attention and focus over extended periods, increasing the risk of accidents due to delayed reactions or poor decision-making.
- Drivers did not have access to real-time assistance or warnings that could help prevent accidents. Without ADAS features such as lane departure warning, forward collision

warning, or blind-spot detection, drivers had to rely solely on their own vigilance and experience to avoid collisions or other incidents.

- Without ADAS, drivers often lacked guidance or feedback on their driving habits or behaviours. This could lead to inefficient practices such as aggressive acceleration, harsh braking, or improper lane changes, negatively impacting fuel efficiency, vehicle wear and tear, and overall traffic flow.
- The absence of ADAS technologies meant there was a reduced safety margin for drivers. In critical situations, drivers had to solely rely on their own reflexes and judgment, which could result in delayed or inappropriate responses, potentially leading to accidents or more severe collisions.
- Certain driving conditions, such as adverse weather, low visibility, or complex traffic situations, posed additional challenges for drivers. Without ADAS systems specifically designed to assist in these conditions, drivers had to navigate through such situations with limited support and increased risk.

1.3 PROJECT OBJECTIVE

The objective of the project is to design, develop, and implement an ADAS level 1 and ADAS level 3, that enhances driving safety, improves situational awareness, and aids drivers in real-time. The project aims to leverage cutting-edge technologies and algorithms to create a comprehensive ADAS solution that addresses various driving challenges and mitigates the risks associated with human error. Specifically, the project objectives include:

- Flash over the air, also known as FOTA, is going used for over-the-air firmware updates, it will have a wireless communication capability to receive firmware updates and potentially other data remotely from a server.
- A private communication protocol channel will be used to exchange data and communicate with other vehicle components.
- Image processing will be used to analyse the surrounding environment in real-time using an input feed from camera modules.
- A system that monitors and adjusts steering input based on feedback from various sensors and the vehicle's dynamics.

CHAPTER 2

CONCEPTUAL DESIGN

2.1 CHAPTER OVERVIEW

In this chapter we will provide a comprehensive overview of all the hardware components purchased and utilized in the project, along with their respective prices. It will also cover the software tools and frameworks employed during the development process. The chapter aims to document and present a detailed account of the project's hardware and software configurations, facilitating transparency and reproducibility.

2.1 MICROCONTROLLER SELECTION

In this section, we will discuss the microcontroller architecture and selection process for our project, as well as the integration of three specific microcontrollers: ARM STM32F103, NVIDIA Jetson, and ESP8266. These microcontrollers serve various purposes within our system, with the ESP8266 acting primarily as a Wi-Fi module.

2.1.1 ARM STM32F103

The ARM STM32F103 microcontroller as shown in Figure 2.1 was chosen for its powerful processing capabilities and wide range of peripherals. It is based on the ARM Cortex-M3 core and offers a high level of performance, making it suitable for demanding applications. The STM32F103 provides ample I/O pins, timers, and communication interfaces, allowing us to interface with various sensors and actuators in our project, for more details check reference section of this documentation (STMicroelectronics, n.d.).

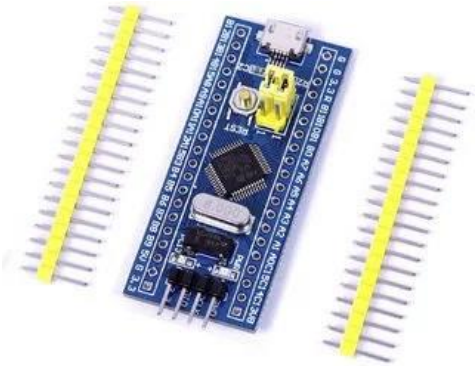

Figure 2.1 ARM STM32F103

2.1.2 NVIDIA Jetson

The NVIDIA Jetson microcontroller platform as shown in Figure 2.2, is specifically designed for high-performance edge computing and AI applications. It features a powerful GPU and CPU combination, enabling accelerated deep learning, computer vision, and parallel processing tasks. The Jetson platform offers excellent computational capabilities, allowing us to perform complex algorithms and real-time processing in our project, for more details check reference section of this documentation (NVIDIA Corporation, n.d.).

Figure 2.2 NVIDIA Jetson

2.1.3 ESP8266

The ESP8266 as shown in Figure 2.3, is a low-cost Wi-Fi module widely used for IoT applications. Although it is primarily a microcontroller, it is often used as a Wi-Fi connectivity solution due to its built-in Wi-Fi capabilities. By integrating the ESP8266 into our system, we can establish wireless connectivity and enable communication with other devices and cloud services over Wi-Fi. Its small form factor, ease of use, and cost-effectiveness make it an ideal choice for IoT projects requiring wireless connectivity, for more details check reference section of this documentation (Espressif Systems, n.d.)

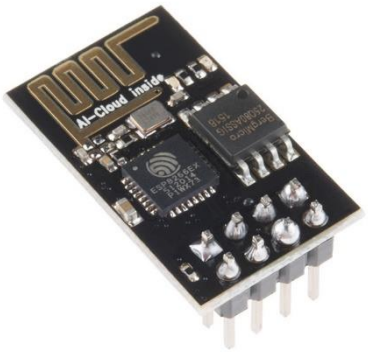

Figure 2.3 ESP8266

2.2 MECHANICAL DESIGN

The design of the project incorporates a steering wheel that is securely attached to a main shaft, ensuring a stable and reliable connection. At the opposite end of the main shaft, a rotary encoder is carefully integrated to accurately measure the rotational position and direction of the steering wheel. This allows for precise tracking of steering inputs.

To enable steering functionality, two DC motors are mounted on the shaft. These motors are responsible for steering the vehicle to the left or right, providing responsive and controlled movement. The DC motors are efficiently controlled using an H-bridge motor driver, which allows for bi-directional control of the motors. This setup ensures smooth and reliable steering operation, allowing the driver to manoeuvre the vehicle with ease.

The integration of the main shaft, rotary encoder, DC motors, and H-bridge motor driver forms a cohesive system that enables effective and precise steering control, as shown in Figure 2.4, Figure 2.5 and Figure 2.6. This design ensures the seamless translation of driver inputs into corresponding steering actions, resulting in a safe and responsive driving experience.

Figure 2.4 Mechanical Design Side View

Figure 2.5 Mechanical Design Back View

Figure 2.6 Mechanical Design Top View

2.3 SOFTWARE REQUIREMENT

In this section, we will outline the software tools and development environments utilized in conjunction with the microcontrollers for our project. These software tools play a crucial role in programming, debugging, and integrating the microcontrollers into our system, and the breakdown of each software used is as follow:

- Eclipse/STMCubeIDE: Eclipse and STMCubeIDE are a popular integrated development environment also known as IDE, used for various programming languages, including C/C++. It provides a comprehensive set of development tools, including a code editor, compiler, debugger, and project management capabilities.
- Notepad++: Notepad is a lightweight text editor and source code editor, it provides a wide range of features, such as syntax highlighting, code folding, auto-completion, and multi-document editing. which assisted in the driver's development.
- SolidWorks: SolidWorks is a powerful 3D computer-aided design also known as CAD, is a software widely used in the engineering and manufacturing industries. It offers a comprehensive set of tools for creating, simulating, and documenting 3D models for mechanical design, product development, and engineering analysis. It used to create 3D models with precise dimensions, simulate physical behaviour, and generate detailed engineering drawings.
- Altium Designer: Altium Designer is software used for designing printed circuit boards also known as PCBs, and electronic systems. It offers a comprehensive set of tools for schematic capture, PCB layout, routing, simulation, and manufacturing outputs.
- Proteus: Proteus is a software suite used for electronic circuit design, simulation. It was used to design and test circuits before they are physically built.
- PyCharm: PyCharm is an IDE, specifically designed for Python programming. It provides a powerful set of tools and features to enhance the development process for the YoloV8 Image Processing.
- Keil: Keil is a Microcontroller Development Kit, is a software development environment for programming microcontrollers, mainly targeting ARM-based microcontrollers. It was used for writing, debugging, and testing embedded applications.

2.4 PROJECT COST ANALYSIS

In this section, we will provide a breakdown of the estimated costs associated with the microcontrollers and their integration into our project. It is essential to consider the costs involved to effectively plan and budget for the development and implementation of the Project.

The Grand Total of our project was brought to 21,387 L.E., and the breakdown of parts bought and used can be viewed in Table 2.1.

Table 2.1 Summary of hardware requirement.

Hardware	Quantity	Price/Unit	Total Price
ARM STM32F103	× 4	175	700
Nvidia Jetson Nano Developer Kit	× 1	12,500	12,500
ESP8266	× 1	125	125
MicroSD 32GB	× 1	200	200
HDMI cable	× 1	25	25
HDMI Video Capture	× 1	250	250
Delivery	× 1	50	50
MCP2515 CAN Bus Module	× 3	150	450
CAN BUS Transceiver Module	× 3	145	435
Motor Driver Dual-channel H bridge Controller	× 1	1,250	1,250
DC Motor 12V 80W 13000rpm	× 2	350	700
360P/R Rotary Encoder	× 1	850	850
12V Output Power Supply	× 1	650	650
5V Output Power Supply	× 2	180	360
Pully 72T	× 1	350	350
Pully 15T	× 2	100	100
Shaft 12mm	× 1	100	100
Belt	× 2	70	140
T-shape screw M5x16	× 18	5	90
V-Slot Aluminium rail profile	× 2	155	310
Aluminium Profile Metal Door Hinge	× 2	72	144
L-Shape 2020 Aluminium Bracket	× 10	26	260
Steel Square Nuts	× 18	3	54
End Cap plastic cover for Aluminium profile	× 8	8	64
Self-align bearing 12mm KP12	× 2	40	80
Momo Steering Wheel	× 1	450	450
Manufacturing	× 1	700	700

SYSTEM ARCHITECTURE

3.1 CHAPTER OVERVIEW

In this chapter, we will discuss the comprehensive division of the SSW project into four distinct sections, with each section being controlled by a dedicated microcontroller. This strategic division allows for a modular and efficient development approach. Each section is responsible for a specific aspect of the SSW system and operates parallel with each other. The microcontrollers act as the central processing units for their respective sections, executing the necessary algorithms, data processing, and control logic. By partitioning the project into these sections, we ensure a streamlined development process, enabling focused implementation, testing, and optimization of each functional area.

3.2 TECHNOLOGIES OVERVIEW

In this section, we will discuss various architectural aspects of the SSW project. We will explore the intricate systems and communication protocols that have been implemented and utilized to bring the SSW to life. The project's architecture framework that integrates multiple components, including sensors, actuators, microcontrollers, and communication modules. Each architecture that was used in the project is dedicated to a specific functionality and is controlled by a designated microcontroller. These architectures collaborate and synergize their operations to create a smooth ADAS level-1 and ADAS level-3 experience.

3.2.1 Bootloader

A bootloader is a fundamental software component responsible for initializing a computer system. It primarily loads the operating system (OS) or other firmware into the system's memory during the startup process. The bootloader plays a crucial role in the system's bootstrapping process, ensuring a smooth transition from the hardware initialization phase to the execution of the main software components by fixing bugs and continuously updating main system with new features.

A bootloader must fulfil several key requirements to ensure the reliable and secure booting of a system. Some of the essential requirements are as follows:

- Reset and Initialization: Upon system power-up or reset, the microcontroller or processor executes the bootloader code located in a predetermined memory location, commonly referred to as the reset vector. The bootloader starts by initializing the hardware and setting up the initial execution environment.
- Flash Driver: The bootloader interacts with the flash memory to read and write data. A flash driver is responsible for low-level operations, such as erasing and programming the flash memory sectors. It provides an interface between the bootloader code and the flash hardware, enabling the bootloader to access and modify the firmware stored in the flash memory.
- Hex Parser: The bootloader typically reads the firmware image from a storage medium, such as a flash memory or external storage device. The firmware image is often represented in a hexadecimal format (hex file). The hex parser component within the bootloader reads and interprets the hex file, converting it into binary data that can be written to the flash memory.
- Firmware Verification and Authentication: Before loading the firmware into memory, the bootloader may perform integrity checks and authentication procedures to ensure the firmware's validity and protect against unauthorized modifications. This can involve cryptographic algorithms, digital signatures, or checksum verification techniques.
- Firmware Loading: Once the firmware passes the verification process, the bootloader copies the firmware from the storage medium (e.g., flash memory) into the appropriate memory location. It sets up any necessary parameters or data structures required by the firmware and transfers control to the loaded software, typically the operating system.

3.2.2 PID Controller

Proportional-Integral-Derivative also known as PID controller, is a widely used feedback control mechanism employed in various control systems. It is designed to regulate a process variable by continuously adjusting an actuator's output based on the error between the desired setpoint and the actual system output. The PID controller calculates and applies control signals to maintain stability and achieve desired system performance. The PID controller utilizes three main components to compute the control signal:

- Proportional (P) Term: The P-term generates an output proportional to the current error. It is responsible for providing an immediate response to the error, where the control

signal is directly proportional to the error magnitude. The P-term helps reduce steady-state error but may result in overshoot and oscillations if used alone.

- Integral (I) Term: The I-term integrates the error over time, compensating for any sustained error that persists over a long period. It helps eliminate steady-state errors and brings the system to the desired setpoint. The I-term is particularly useful when dealing with system biases or disturbances.
- Derivative (D) Term: The D-term accounts for the rate of change of the error. It provides a damping effect, reducing overshoot and improving system stability. The D-term anticipates the future trend of the error and helps prevent sudden changes in the control signal.

3.2.3 CAN Protocol

Controller Area Network also known as CAN protocol is a robust and widely used communication standard in the automotive and industrial sectors. It was originally developed by Robert Bosch GmbH in the 1980s to facilitate reliable communication between electronic control units (ECUs) in vehicles. The CAN protocol offers a flexible and efficient means of transmitting data and commands among networked devices within a controlled area.

The CAN protocol provides a set of rules and procedures for message transmission and reception within a CAN network. Its key features include:

- Message-Based Communication: The CAN protocol employs a message-based communication approach, where devices transmit packets of data known as CAN frames. Each CAN frame contains an identifier, payload, and other control information.
- Deterministic and Event-Driven: CAN offers deterministic communication, meaning that messages are prioritized based on their identifiers. Higher-priority messages are transmitted before lower-priority ones. Additionally, CAN supports event-driven communication, allowing devices to transmit messages as soon as they have data to send.
- Arbitration and Collision Avoidance: CAN utilizes a non-destructive bitwise arbitration mechanism to resolve conflicts when multiple devices attempt to transmit simultaneously. This approach ensures fair access to the communication medium and minimizes the risk of data collisions.

- Error Detection and Fault Tolerance: CAN employs a robust error detection and fault tolerance mechanism. It uses a cyclic redundancy check (CRC) to verify the integrity of transmitted data. Additionally, CAN supports error detection and error confinement mechanisms to handle faulty nodes and improve network reliability.

3.2.4 YOLOv7

You Only Look Once version 7, also known as YOLOv7, is an object detection algorithm that belongs to the YOLO family of models. It is designed to detect objects efficiently and accurately in images or video frames. YOLOv7 builds upon its predecessors, incorporating improvements to achieve better performance in terms of accuracy and speed.

YOLOv7 typically employs a deep neural network architecture, such as Darknet-53, as its backbone. Darknet-53 consists of multiple convolutional layers, residual connections, and down sampling operations to extract features at different scales. It requires a large, labelled dataset for training. The network is trained using a combination of image samples and corresponding bounding box annotations. During the training process, the network learns to detect objects by minimizing a loss function that penalizes errors in bounding box predictions and class probabilities.

Once trained, YOLOv7 can perform object detection on new images or video frames. The algorithm takes an input image, passes it through the detection network, and generates bounding box predictions along with class probabilities. These predictions can be further refined using post-processing techniques like NMS.YOLOv7 can be integrated into various applications and systems. It can be utilized as a standalone object detection algorithm or incorporated into larger systems for tasks such as autonomous driving, surveillance, object tracking, and more.

The YOLOv7 algorithm performs object detection by dividing an input image into a grid and predicting bounding boxes and class probabilities for each grid cell. It operates in two stages:

- Detection Network: The detection network processes the input image and generates a set of bounding box predictions along with the associated class probabilities. It utilizes a convolutional neural network (CNN) architecture, often based on Darknet, to extract features and perform object detection in a single pass.

- Post-processing: The post-processing stage applies filtering and non-maximum suppression also known as NMS to refine the bounding box predictions. NMS removes duplicate or overlapping bounding boxes, keeping only the most confident ones based on their class probabilities.

Therefore, we could conclude that some of the advantages and limitations that YOLOv8 offers, which can be shown in Table 3.1:

Table 3.1 Advantages and Limitations of YOLOv8

Advantage	Limitations
Real-Time Performance	Problems in detecting smaller objects due to the coarser grid scale
Good Accuracy and Generalization	Difficulty in detecting objects with extreme aspect ratios
Performs object detection in a single pass	Trade-off Between Speed and Accuracy

3.3 SYSTEM IMPLIMENTAION

The SSW project is divided into four different segments as shown in Figure 3.1. Each segment is powered by four STM32F103 microcontrollers. All four segments are interconnected through a CAN bus, enabling seamless communication between them. Each segment is linked to a CAN transceiver using the SPI protocol. The CAN-H and CAN-L lines from each transceiver are connected to establish the bus.

By connecting the CAN-H and CAN-L lines from each segment's transceiver together to form a bus, all segments can exchange data and commands seamlessly. Each segment can transmit and receive messages over the CAN bus, allowing for real-time communication and coordination between the different functionalities and components of SSW project.

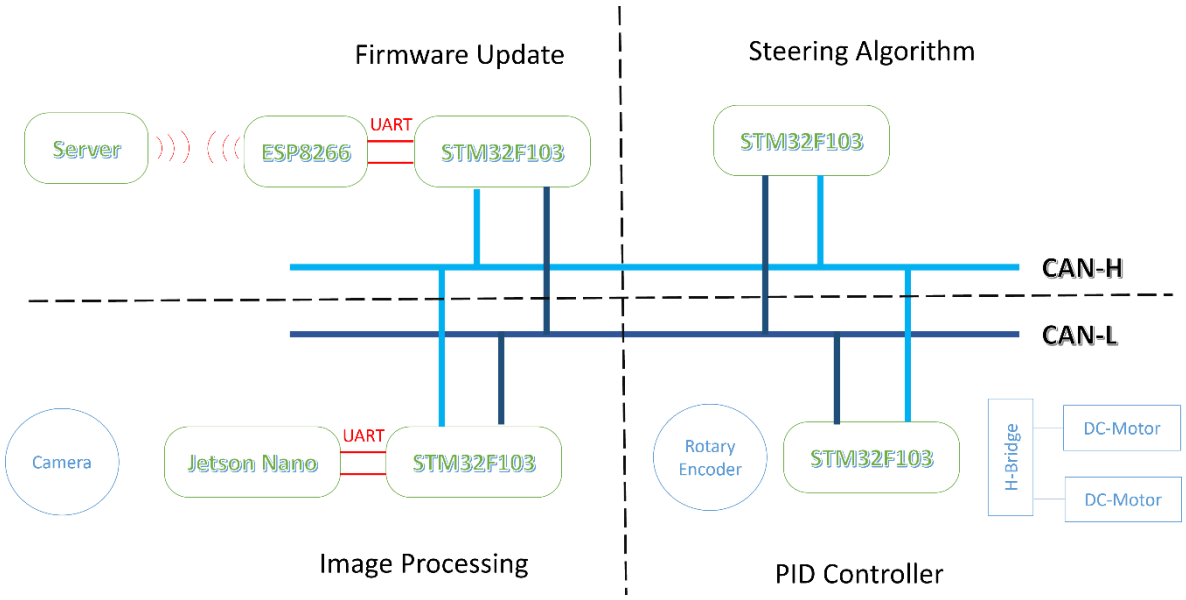

Figure 3.1 Work Flow Diagram

3.3.1 Segment 1: PID Controller

Segment 1 incorporates an STM32 microcontroller responsible for receiving input from a rotary encoder that detects the angle of the steering wheel. The microcontroller utilizes a PID controller algorithm to adjust the steering angle based on the received feedback, which was printed on a PCB board as shown in Figure 3.2. To control the steering mechanism, an H-bridge motor driver is employed, which controls the 2 DC motors. The microcontroller

exchanges data with the other segments through the CAN bus, allowing seamless communication.

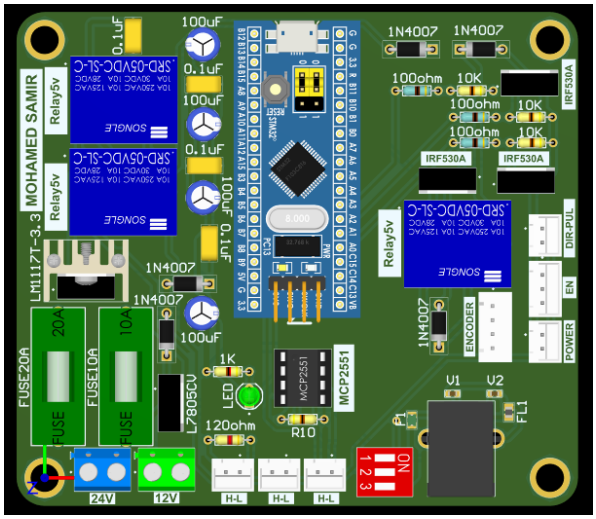

Figure 3.2 PID Controller PCB layout

3.3.2 Segment 2: Image Processing

Segment 2 incorporates an NVIDIA Jetson Nano microcontroller, responsible for image processing tasks. This segment captures objects through a camera and performs image processing to detect potential threats. Each object is assigned a Y-axis length, and if it exceeds a certain threshold, it indicates close proximity and flag it as a threat. The Jetson Nano communicates the threat data to Segment 3 via the CAN bus, enabling appropriate action to avoid potential collisions.

3.3.3 Segment 3: Steering Algorithm

Segment 3 consists of another STM32 microcontroller that receives feedback data from Segment 1 and Segment 2 via the CAN bus. This feedback includes information about the position and direction of the steering wheel and if obstacle is detected. Using a steering algorithm, this microcontroller calculates a new angle for the steering wheel and transmits the updated command back to Segment 1 over the CAN bus. The algorithm ensures precise and efficient steering control.

3.3.4 Segment 4: Firmware Update

Segment 4 comprises an STM32 microcontroller and an ESP8266 microcontroller functioning as a Wi-Fi module. This segment FOTAMOTIVE firmware updates for any microcontroller on the CAN bus. The ESP8266 establishes a Wi-Fi connection with a server hosting the firmware files. The STM32 microcontroller coordinates with the server, downloads the relevant firmware in the form of a hex file, and initiates the flashing process on the target microcontroller via the CAN bus.

CHAPTER 4

RESULT AND DISCUSSION

4.1 CHAPTER OVERVIEW

In this chapter we will discuss result of our project's application and provides a comprehensive analysis and evaluation of the outcomes achieved during our work. This section aims to present the findings, observations, and insights gained from implementing the project design, conducting experiments, and performing tests. It offers a platform to discuss the implications, limitations, and potential future directions of the project, fostering a deeper understanding of its significance and impact.

4.2 APPLICATION ACHIEVED

The main application of our project involves capturing potential obstacles using a camera, which is then processed through image processing techniques on a Nano Jetson platform using YOLOv8 software. The camera serves as the input source, providing a real-time feed of the surrounding environment. Through the application of advanced image processing algorithms, the software analyses the video frames to detect and track moving objects accurately.

To determine the relevance and potential danger of the detected obstacles, the system measures their position on the y-axis as shown in Figure 4.1. If the y-axis value increases, indicating that the detected object is moving closer or is positioned at a higher level, it is flagged as an obstacle that must be avoided. This information is then utilized to trigger appropriate actions or alerts, ensuring that the vehicle or the driver takes necessary precautions to navigate safely and avoid potential collisions.

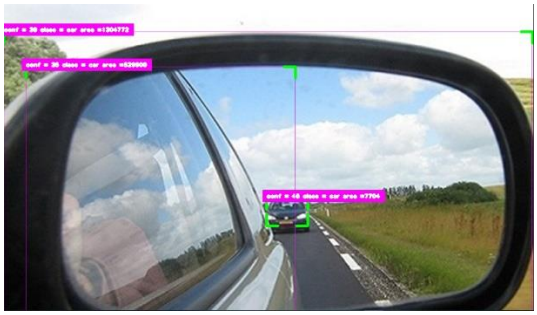

Figure 4.1 Obstacle detection

By combining camera-based obstacle detection, image processing, and YOLOv8, the Smart Steering Wheel system is able to achieve two distinct applications that enhance driver safety and provide advanced driver assistance, which will be discussed in section 4.2.1 and section 4.2.2

4.2.1 Side Collision Warning (ADAS Level 1)

The first application of the SSW system is to provide a side collision warning functionality, which operates at ADAS Level 1. By utilizing the image processing inputs results, the system detects potential side collisions with objects or vehicles in the vehicle's blind spots. When a potential side collision is detected, the SSW system generates a haptic feedback response, such as steering wheel vibration or a buzzer, to alert the driver about the impending danger. This timely warning allows the driver to take appropriate action, such as adjusting their position or changing lanes, to avoid the collision and maintain a safe driving environment.

4.2.2 Driver Assistance System (ADAS Level 3)

The second application of the SSW system involves implementing a driver assistance system that operates at ADAS Level 3. This advanced system goes beyond providing warnings and actively takes control of the steering wheel to assist the driver in potentially dangerous situations. When the SSW system detects a potential object or obstacle that poses an immediate threat, it engages a takeover mechanism and quickly steers the vehicle to the left or right. This response is designed to be faster than the driver's reaction time, ensuring a swift and decisive manoeuvre to avoid a collision. By providing active steering assistance, the SSW system enhances the driver's ability to avoid accidents and enhances overall driving safety.

CHAPTER 5

CONCLUSION

5.1 CONCLUSION

In conclusion, our project has successfully developed and implemented a steering wheel (SSW) system with two crucial applications: side collision warning and driver assistance. Through the integration of sensor inputs, image processing techniques, and advanced control algorithms, the SSW system enhances driver safety by providing timely alerts and active steering assistance in potentially dangerous situations.

Overall, our project has laid the foundation for an efficient and reliable steering wheel system that improves safety and provides valuable driver assistance. With continued dedication and innovation, the project can serve as a stepping stone for further advancements in autonomous driving and contribute to the realization of safer and more intelligent vehicles on our roads.

5.2 FUTURE RECOMMENDATION

For those who wish to continue and upgrade our project, we offer the following recommendations:

- Enhance Object Detection by improve the accuracy and reliability of object detection algorithms by exploring advanced computer vision techniques or deep learning models. This will ensure more precise detection and classification of potential obstacles.
- Integrate Additional Sensors, such as LiDAR or ultrasonic sensors, to complement the existing sensor inputs. This multi-sensor fusion can provide a more comprehensive perception of the surrounding environment, further enhancing the system's ability to detect and respond to potential hazards.
- Refine Steering Control Algorithms by continuously optimize the steering control algorithms for smoother and more natural steering responses. This can involve fine-tuning the control parameters, incorporating adaptive control techniques, or exploring advanced control methodologies to enhance the system's performance.

- Expand ADAS Capabilities by explore the implementation of additional ADAS features, such as lane keeping assistance, adaptive cruise control, or automated parking. This will broaden the capabilities of the SSW system, providing a more comprehensive suite of driver assistance functions.
- Introduce GUI: Invest in improving the graphical user interface also known as GUI, and human-machine interaction aspects of the SSW system. This can involve developing intuitive graphical interfaces, voice commands, or integrating with existing vehicle infotainment systems to provide a seamless and user-friendly experience for the driver.

By incorporating these recommendations, future iterations of the project can further enhance the SSW system's performance, expand its capabilities, and contribute to the development of advanced driver assistance technologies.

REFERENCES

Espressif Systems. (n.d.). Retrieved from ESP8266: https://www.espressif.com/en/products/socs/esp8266

NVIDIA Corporation. (n.d.). Retrieved from Jetson Nano Developer Kit: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

STMicroelectronics. (n.d.). Retrieved from Microcontrollers& Microprocessors: https://www.st.com/en/microcontrollers-microprocessors/stm32f103c8.html