AN AUTONOMOUS OBSTACLE AVOIDANCE VEHICLE SYSTEM WITH TINYML FOR LAST MILE DELIVERY

BY

IBRAHIM, Abdulhafiz Adavize

2019/1/75616CP

UGWUNEBO, Oluchi Tiffany

2019/1/76595CP

DEPARTMENT OF COMPUTER ENGINEERING

FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA, NIGER STATE

NOVEMBER, 2025

FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA NIGERIA

DECLARATION OF UNDERGRADUATE PROJECT REPORT COPYRIGHT

Author's full name	:	IBRAHIM ABDULHAFIZ ADAVIZE
Matric Number	:	2019/1/75616CP
Title	:	AN AUTONOMOUS OBSTACLE AVOIDANCE VEHICLE SYSTEM WITH TINYML FOR LAST MILE DELIVERY
Academic Session	:	2024/2025

I hereby declare that this Project Report is classified as:

☐ **OPEN**	I agree that my Project Report may be published as online open access (full text)

I acknowledge that the Department of Computer Engineering and Federal University of Technology Minna reserves the right as follows:

1. The Project Report is a property of the Department of Computer Engineering and the Federal University of Technology Minna.
2. The Library of the Department of Computer Engineering and that of Federal University of Technology Minna had the right to make copies for the purpose of research and academic exchange.
3. The Department of Computer Engineering and the Federal University of Technology, Minna reserves the right to withhold any hardware or product that may have been developed or produced in the course of this research write-up.

Certified by:

	ENGR. DR. EUSTACE DOGO
SIGNATURE OF STUDENT	**NAME & SIGNATURE OF SUPERVISOR**
2019/1/75616CP	
MATRIC NUMBER	

FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA NIGERIA

DECLARATION OF UNDERGRADUATE PROJECT REPORT COPYRIGHT

Author's full name	:	UGWUNEBO OLUCHI TIFFANY
Matric Number	:	2019/1/76595CP
Title	:	AN AUTONOMOUS OBSTACLE AVOIDANCE VEHICLE SYSTEM WITH TINYML FOR LAST MILE DELIVERY
Academic Session	:	2024/2025

I hereby declare that this Project Report is classified as:

☐ **OPEN** — I agree that my Project Report may be published as online open access (full text)

I acknowledge that the Department of Computer Engineering and Federal University of Technology Minna reserves the right as follows:

1. The Project Report is a property of the Department of Computer Engineering and the Federal University of Technology Minna.
2. The Library of the Department of Computer Engineering and that of Federal University of Technology Minna had the right to make copies for the purpose of research and academic exchange.
3. The Department of Computer Engineering and the Federal University of Technology, Minna reserves the right to withhold any hardware or product that may have been developed or produced in the course of this research write-up.

Certified by:

	ENGR. DR. EUSTACE DOGO
SIGNATURE OF STUDENT	**NAME & SIGNATURE OF SUPERVISOR**
2019/1/76595CP	
MATRIC NUMBER	

TITLE PAGE

DEVELOPMENT OF AN AUTONOMOUS OBSTACLE AVOIDANCE VEHICLE SYSTEM WITH TINYML FOR LAST MILE DELIVERY

IBRAHIM, Abdulhafiz Adavize

2019/1/75616CP

UGWUNEBO, Oluchi Tiffany

2019/1/76595CP

A PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)

DEPARTMENT OF COMPUTER ENGINEERING,

SCHOOL OF ELECTRICAL ENGINEERING AND TECHNOLOGY,

FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA, NIGER STATE.

NOVEMBER, 2025

DECLARATION

I hereby declare that this project titled "Development of An Autonomous Obstacle Avoidance Vehicle System with TinyML for Last Mile Delivery" and the content of this write-up are as a result of my research efforts except as cited in the references and that this work has not been or concurrently submitted by anyone for any other qualifications or degree.

Signature: _____

Name: IBRAHIM, ABDULHAFIZ ADAVIZE

Date: NOVEMBER, 2025

DECLARATION

I hereby declare that this project titled "Development of An Autonomous Obstacle Avoidance Vehicle System with TinyML for Last Mile Delivery" and the content of this write-up are as a result of my research efforts except as cited in the references and that this work has not been or concurrently submitted by anyone for any other qualifications or degree.

Signature: _____

Name: UGWUNEBO, OLUCHI TIFFANY

Date: NOVEMBER, 2025

CERTIFICATION

This project, titled *"An Autonomous Obstacle Avoidance Vehicle System with TinyML for Last Mile Delivery"* by Ibrahim, Abdulhafiz Adavize (2019/1/75616CP), meets the regulations governing the award of the degree of Bachelor of Engineering (B.Eng.) of Federal University of Technology, Minna, and it is approved for its contribution of scientific knowledge and literary presentation.

ENGR. DR. EUSTACE DOGO _____

SUPERVISOR Signature & Date

ENGR. DR. EUSTACE DOGO _____

HEAD OF DEPARTMENT Signature & Date

_____ _____

EXTERNAL EXAMINER Signature & Date

CERTIFICATION

This project, titled *"An Autonomous Obstacle Avoidance Vehicle System with TinyML for Last Mile Delivery"* by Ugwunebo, Oluchi Tiffany (2019/1/76595CP), meets the regulations governing the award of the degree of Bachelor of Engineering (B.Eng.) of Federal University of Technology, Minna, and it is approved for its contribution of scientific knowledge and literary presentation.

ENGR. DR. EUSTACE DOGO _____

SUPERVISOR Signature & Date

ENGR. DR. EUSTACE DOGO _____

HEAD OF DEPARTMENT Signature & Date

_____ _____

EXTERNAL EXAMINER Signature & Date

DEDICATION

To my mother, Haj. Habibat Ibrahim; my brothers, Abdulkarim and Muhammed Kabir; my sisters, Hassanat and Rahamat; and my late father, Alh. Hassan Ibrahim, whose vision and guidance carry me to this day.

~Abdulhafiz

To my mother, Ms Ubaku Chijioke Biachi; and my sisters, Odera, Nwamaka and Chimamanda.

~Oluchi

ACKNOWLEDGEMENTS

First and foremost, I would like to express my profound gratitude to God for the wisdom, strength, and granting me the ability to complete this project.

My sincere appreciation goes to my supervisor and HOD, Engr. Dr. Eustace Dogo, for his invaluable guidance, patience, and constructive criticism throughout the duration of this work. I am also grateful to all my lecturers especially Engr. Dr. Lukman Ajao for the knowledge imparted during my coursework, which laid the foundation for this research.

I extend my thanks to the DesciNG community for their partial sponsorship and support, which contributed significantly to the success of this project.

To my project partner, Oluchi, thank you for your collaboration, hard work, and dedication. We made a great team.

I am deeply indebted to my family for their unwavering prayers, financial support, and encouragement.

Finally, a special shout-out to my friends Jamil, Adl, Anwar, Destiny, David, Faith, and all my course mates for being a constant source of motivation and for the moments of relief during stressful times.

~Abdulhafiz

I would like to express my sincere gratitude to everyone who supported me throughout the course of my final year project and my academic journey.

First and foremost, I am deeply grateful to my family for their unwavering financial and emotional support during my time at the university. Their constant encouragement has been the foundation of my achievements.

I would also like to acknowledge my course mate, Destiny Onoja, whose academic support played a meaningful role in my learning experience.

My heartfelt appreciation goes to my friends, Alfred Desire, Dedekimor Onome, Doose Stephanie, Egena Stephanie, and Idris Aishat for making my university journey memorable and enriching.

A special acknowledgement goes to my project partner, Ibrahim Abdulhafiz, whose hard work and dedication were instrumental in bringing this project to a successful completion.

Finally, I extend my sincere thanks to my project supervisor, Engr. Dr. Eustace Dogo, for his invaluable guidance, insight, and mentorship throughout the development of this project.

~Oluchi

ABSTRACT

This project presents the design and implementation of a cost-efficient autonomous obstacle-evading vehicle, which is designed to operate in last-mile delivery with Tiny Machine Learning (TinyML) and Internet of Things (IoT) applications. Resolving the inefficiencies, high costs, and environmental costs of traditional last-mile logistics, the study introduces an embedded system that has the capability to make real-time obstacle detections and autonomous navigation in urban and semi-urban environment. The platform is based on ESP32 microcontroller and it incorporates ultrasonic sensing and Wi-Fi connectivity to enable exchange of data with a remote server. An on-board lightweight FOMO (Faster Objects, More Objects) TinyML model is deployed on board to perform on-board object detection, and, therefore, allows decentralized, low-latency decision making without relying on cloud computation. Navigation and path-planning modules were also put through extreme tests in varied environments to test the stability, responsiveness and delivery success rates. The experimental results prove that the TinyML-based model has been effective in obstacle detection with little computational and energy costs, which justifies its appropriateness to work within resource-scarce settings. The suggested system in turn contributes to the growing body of literature on sustainable, edge-AI-based logistics by providing a scalable and energy-saving prototype of autonomous last-mile delivery in both smart and developing urban settings.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

1.0 Introduction

1.1 Background of Study

For the past hundred years, Autonomous vehicles (AVs) have evolved into one of the most substantial transportation technologies, transforming mobility systems and increasing roadway safety while improving traffic efficiency (Anderson *et al.*, 2014). Research into autonomous vehicles has become extensive because they demonstrate the potential to transform both individual transportation and public transportation systems, as well as logistics operations and several other sectors. Recent scholarly research about autonomous vehicle technology has been compiled to review the technological as well as ethical barriers and practical implementation issues.

The Society of Automotive Engineers (SAE) defines vehicle automation types, which serve as the fundamental principle for understanding autonomous vehicles. SAE has developed a six-level framework that starts with manual operation at Level 0 and progresses through five levels until it reaches Level 5, which represents full autonomous driving systems.

Katare *et al.* (2023) described the 6 levels of autonomy for a driving car based on the SAE. They were of 6 categories. No Automation refers to all driving tasks being carried out by the driver. The Driver Assistance level introduces assist features. However, there is little input from the vehicle sensors. Thus, driving tasks are still mostly carried out by the driver. The Partial Automation level has driving tasks carried out by a computing unit within the car. This computing unit receives sensed input from the surroundings. Features like emergency braking and adaptive cruise control are implemented autonomously here. This level 2 still requires the driver to regularly monitor the vehicle's surroundings. Conditional Automation Level has some tasks like actuation, sensing, and control carried out by the computing unit

and sensors integrated in the vehicle. However, demand and critical situations will still have the driver taking control of the vehicle. The level of High Automation requires vehicular capabilities to cover all driving tasks through communication with other vehicles under certain conditions. The driver always has the option to have the control returned to them.

The highest level of autonomy is described as the execution of driving processes that serve self-driving functionality from a source point to the destination point without any input or control to the vehicle from a human. In this project, the Full Automation level is the case study. In this final level, the vehicle system is perfectly capable of decision-making and navigation under all conditions. The navigation is solely controlled by a remote server, and an obstacle detection and avoidance algorithm is implemented.

The classification system described by Martínez-Díaz and Soriguera (2018) functions to organise a wide range of autonomous vehicle system technologies, thus enabling a better understanding of present capabilities and future automation developments.

Sensor technology remains the fundamental component of autonomous vehicle systems because it integrates cameras in addition to LiDAR and ultrasonic detectors. Vehicle sensors act as foundational elements for generating dependable perception capabilities that assist autonomous vehicles during their complex navigation tasks. Sensor fusion techniques serve as fundamental requirements for autonomous vehicle operation because they merge sensor data to form a single unified depiction of vehicle surroundings. The combination of these techniques creates accurate object recognition and obstacle detection, so autonomous vehicles achieve safer driving operations.

The advancements of deep learning, together with artificial intelligence through recent machine learning developments, have improved the vehicle's understanding of complex environments. The ability of algorithms to process large sensor data inputs enables

autonomous vehicles to make instant decisions based on surrounding detection. The advancement stems from this evolution to create a base for automated systems that minimise human involvement but maximise their response precision (Sahoo & Varadarajan, 2025).

Various obstacles stand in the way of autonomous vehicle practical deployment, even though technological breakthroughs have significantly progressed. The development of laws and regulations remains behind technological growth at a fast pace. A broad spectrum of legal matters surrounds self-driving cars because they require new rules to determine accident accountability and separate licenses and proper handling of data privacy issues. Studies have shown evidence that existing regulations must receive immediate attention because they lack sufficient attention to autonomous vehicle technology features. The matter of responsibility remains difficult to resolve. The determination of blame in regular car crashes depends on human conduct, yet autonomous vehicles utilise programming logic through learning models to function, which makes fault assessment more complicated. Lawmakers require new frameworks to handle the distinctive autonomous vehicle operational characteristics. The manufacturing partnership between technology companies and vehicle producers creates multiple points of responsibility that become difficult to determine.

The adoption of autonomous vehicle technology depends heavily on how well society welcomes this technology. Multiple qualities determine how consumers trust autonomous vehicles, according to research findings, because they base their decisions on perceptions of safety and privacy-related matters and ethical algorithms. Research demonstrates that vehicle users avoid autonomous vehicles because they fear for their safety, especially during emergencies. The speed at which technology adoption occurs depends on how the public perceives it, so manufacturers need to work with policymakers to solve identified issues. The absence of a standardised privacy policy for data protection in automated vehicles creates

extra issues for consumers. The continuous data collection and processing performed by autonomous vehicles for functional improvement creates concerns about personal data privacy protection. The development of autonomous vehicle technology requires users to have both their privacy protected and data secured to build trust with the technology (Gherardini & Cabri, 2024).

Autonomous vehicle technologies present additional moral aspects that complicate their deployment process. The algorithms installed in autonomous vehicles make their critical responses, yet their design processes, together with their inherent moral principles, remain substantial matters of ethical consideration. Decisions involving life-and-death situations force society to establish risk boundaries and ethical decision-making approaches between utilitarianism-based and deontological-based frameworks. Authors agree that algorithms must represent the moral values of society, while developers need to maintain open processes for creating these systems. Ethical dilemmas in autonomous vehicle programming often use the "trolley problem" scenario to demonstrate the critical effects of autonomous vehicle decision-making processes. Does an automated vehicle system need to protect its passenger over all other road users, or must it safeguard everyone, including the passenger? Ethical frameworks must be fundamental to algorithmic design development because they both guide framework code development and gain public acceptance. Ethicists and technologists, along with policymakers and public representatives, should hold progressive dialogues to create acceptable guidelines that harmonise autonomous vehicle decision-making systems with social and ethical standards (Xiang, 2023).

The way people communicate about these ethical concerns shows that we need extensive planning methods that combine different viewpoints and work toward reducing algorithmic prejudice during development. The fast-paced technological advancement requires input from

multiple stakeholders to achieve ethically responsible and morally sound development of autonomous vehicle systems.

Autonomous vehicles cause extensive social consequences when connected to current transportation networks. Research has indicated that autonomous vehicles would expand accessibility opportunities for people who cannot drive regular vehicles, and this includes disabled individuals and elderly people. The integration of autonomous vehicles facilitates more efficient urban planning and supports sustainable transportation goals through decreased congestion. The positive impacts of autonomous vehicles require careful consideration because they will result in job losses in driver-dependent sectors, which require fast transition planning for workforce adjustment.

The environmental effects of this matter demand careful attention. The proper inclusion of autonomous vehicles in green energy programs enables them to provide impressive benefits that improve urban air quality and cut down greenhouse gas emissions. The successful implementation of automated vehicles demands collective guidance between technology creators, urban developers, and government officials to develop autonomous vehicles as sustainability enablers instead of traditional car duplicates.

In conclusion, research into autonomous vehicles produces detailed analyses about this emerging technology, which faces numerous difficulties but presents various promising possibilities. Technical improvements in automated systems and sensor technologies need solution methods for practical barriers to achieve maximum autonomous vehicle functionality, including regulatory standards, social approvals, and ethical challenges. Technology and sociological developments require unified solutions between legal frameworks, ethical evaluations, and technological applications.

The responsible and ethical development of autonomous vehicles depends heavily on sustained academic research and sustained teamwork between technologists, ethicists, policymakers, and the public. Society will be able to unlock autonomous vehicle potential through comprehensive solutions that blend positive contributions for both individual users and social frameworks.

The potential for revolutionary change in transportation through autonomous vehicles requires sustained research, together with discussion, to fully grasp their complex nature. Boundless development of transportation requires both technological innovation and ethical guidelines, together with legislative reforms that reflect current transportation directions. A holistic approach will create the necessary foundation for autonomous vehicle integration within society.

1.2 Problem Statement

Last-mile delivery is often slow, expensive, and logistically challenging, especially in crowded and metropolitan areas. This autonomous vehicle can maximise efficiency and cut down on delivery times because it can travel short distances unmanned, reducing delivery times and optimising efficiency. This is important as last-mile delivery accounts for about half of all shipping expenses and is the most costly and time-consuming aspect of the supply chain (Boysen *et al.*, 2020).

The swift expansion of the e-commerce sector, marked by small package sizes, high volume, and fluctuating delivery frequency, poses significant issues for logistics service providers (Ghajargar *et al.*, 2016).

A delivery personnel conserves more time in urban settings compared to rural settings. These savings stem from a decrease in parking time and the distance walked by the delivery

personnel; this enhanced productivity may lead to a reduction in fleet size and, consequently, the number of automobiles on the road (Reed *et al.*, 2022).

1.3 Aim and Objectives of Study

This research aims to apply IoT and Edge AI for navigation, path planning and obstacle detection in autonomous vehicles. The specific objectives are as follows:

1. Design and implement the vehicle's hardware system, integrating the ESP32 microcontroller with all other components.
2. Implement a web-based communication service to enable real-time data exchange between the vehicle and an external server via Wi-Fi.
3. Train and deploy a TinyML FOMO model for real-time obstacle detection and avoidance.
4. Integrate objectives 1, 2 and 3.
5. Evaluate system performance through field tests assessing navigation accuracy, obstacle detection, and delivery efficiency.

1.4 Justification of Study

There is an ever-growing demand for reliable, energy-efficient, and cost-effective methods in the field of last-mile delivery. This has sped up the research on autonomous vehicles, typically in urban and smart cities. More companies are integrating with robotic delivery systems by the year in an attempt to reduce human labour costs and energy emissions. For this reason, real-time obstacle avoidance is most critical in autonomous vehicles. For such systems to be deployed in resource-constrained environments in terms of both network connectivity and hardware, TinyML becomes essential.

Many autonomous vehicle navigation solutions employ high-compute architecture (Valera *et al.*, 2021; Engesser *et al.*, 2023). They rely on external servers for heavy path planning

algorithmic computations. These designs, though effective in simulation, are reserved for high-end, expensive MCUs (Gómez-Huélamo *et al.*, 2021). Its practicality is not extended to small-scale, battery powered low-cost vehicles or robots that are used in last-mile deliveries in bandwidth-limited towns. The introduction of TinyML in the research field for logistic autonomous vehicles provides ML models optimised for low-power microcontrollers. This allows for the localisation of obstacle detection and decision-making, allowing a less critical dependency on cloud services.

A great amount of literature puts emphasis on high-level algorithm performance, sensor fusion, and multi-agent decision-making (Hamidaoui *et al.*, 2025; Katona *et al.*, 2024). Only a smaller part of the autonomous vehicle research field explores the constraints and impracticalities associated with deployment on embedded devices with limited resources like memory and power. Advanced path planning methods like RRT*, DWA, and RL-based systems, which are commonly used in autonomous vehicle research, require GPUs or edge accelerators (Alverhed *et al.*, 2024; Tijani, 2022). This isn't suitable for resource-deficient situations, which limits the applications of autonomous vehicles in the world.

There is insufficient literature covering how traditional obstacle detection or avoidance methods can be ported to TinyML, which could present a promising frontier for real-time navigation on low-cost platforms. There is an obvious gap in research towards the design of end-to-end autonomous vehicles that detect and avoid obstacles via TinyML for logistics purposes. This study aims to address this niche in research.

There is a common stress in the research on autonomous vehicles for the improvement of localisation, path planning, and decision-making in the case of uncertainty in urban environments, for example, adverse weather, incoming pedestrians, and dynamic obstacles (Hamidaoui *et al.*, 2025; Alverhed *et al.*, 2024; Katona *et al.*, 2024). There is, however, a

drought in solutions that could be deployed on microcontrollers and still maintain robustness in these uncertain conditions. This project aims to directly tackle is gap in research by combining classical and TinyML-based obstacle avoidance for the enhancement of safety and autonomy.

This study aligns with the growing emphasis on energy awareness and sustainability in research on autonomous vehicles (Paksadze, 2025; Valera *et al.*, 2021), especially in the reduction of CO_2 emissions, promoting electric delivery systems. The proposed system in the study contributes to low-emission, scalable urban logistics literature by leveraging TinyML and compact sensors.

In conclusion, there is a literature gap in practical, embedded, and TinyML-powered obstacle avoidance systems for autonomous delivery robots. This study aims to advance the academic conversation and research on collision avoidance for constrained systems, as well as provide a deployable, scalable prototype for last-mile deliveries in smart cities and underdeveloped cities.

1.5 Scope of Study

This study is centred on the design, development, and implementation of a low-cost autonomous vehicle system with the capabilities of avoiding obstacles in real time via a lightweight machine learning model, FOMO (Faster Objects, More Objects) in real time on a typical resource-constrained microcontroller (ESP32). The specific goal of the project is to address last-mile delivery challenges in urban and semi-urban environments where conventional methods prove to be much more costly, slower, and impractical.

CHAPTER 2

2.0 Literature Review

2.1 Preamble

The design of low-cost autonomous delivery vehicles integrates multidisciplinary research spanning autonomous navigation, obstacle detection, embedded microcontroller platforms, cloud and edge AI integration, and last-mile logistics optimisation. This literature review synthesises key contributions in each domain, establishing the theoretical and practical foundations for the proposed ESP32-based delivery vehicle.

2.2 Overview of Autonomous Vehicles in Last-Mile Delivery

The last step in moving products from producer to consumer is known as last-mile delivery, taking them from a central hub to their end location, which is usually a home or business. It is known to be a challenging part of the supply chain because it costs a lot of resources and requires a lot of organisations. It is possible for up to 28% of delivery costs to be due to inefficiencies and operational problems caused by extra stops or changes in delivery conditions (Ranieri *et al.*, 2018).

These issues are being addressed in a new way through autonomous vehicles. Because autonomous vehicles do not need human help, they make the delivery process more consistent and efficient. Some of the main reasons for adopting autonomous delivery are constant operations, better planning of routes with the help of AI, less chance of human errors, and a significant decrease in pollution thanks to greater energy efficiency (Sorooshian *et al.*, 2022; Paiva *et al.*, 2021).

Nowadays, businesses are looking into options that would make it possible for autonomous vehicles to fit smoothly into their current supply chains. Such integration may involve a vehicle that carries goods to a neighbourhood station, with a human delivery person finishing

the job, or it could use small robots that deliver directly to the customer's door (Engesser *et al.*, 2023).

2.2.1 Types of autonomous delivery solutions

Autonomous delivery solutions for last-mile logistics can be sorted into various types. These are Autonomous Delivery Robots (ADRs), Unmanned Aerial Vehicles (UAVs), and Autonomous Delivery Vans. These solutions can be deployed as standalone systems or might be integrated into bigger systems meant to improve every stage of the delivery process.

2.2.1.1 Autonomous delivery robots (ADRs)

Autonomous Delivery Robots are robots designed to be pedestrian-sized, and they deliver items without requiring direct human intervention. There are two main types of ADRs:

1. **Sidewalk Autonomous Delivery Robots (S-ADRs):** S-ADRs are built to be used exclusively on sidewalks or bike lanes. Their designs are optimised for manoeuvrability in places where pedestrians are present. Because they can travel up to 20 km, move at about 8 km/h, and carry loads of 100 kg, these robots focus on delivering goods safely and efficiently in crowded cities (Sorooshian *et al.*, 2022) (Engesser *et al.*, 2023). AS they are not considered motor vehicles by the law, their design helps minimise conflicts with cars and other motorised vehicles (Engesser *et al.*, 2023).
2. **Road Autonomous Delivery Robots (R-ADRs):** In contrast, R-ADRs have been designed to use the same roads as traditional vehicles. They are usually built to resemble electric vans, considering their energy use per customer and the efficiency of their daily operations. They are built to move more parcels and function well in cities, where the busy streets and traffic make it challenging (Engesser *et al.*, 2023; Paiva *et al.*, 2021).

2.2.1.2 Unmanned Aerial Vehicles (UAVs)

Drones, which are another name for UAVs, present a unique option for completing the last part of the delivery. At first, drones were seen as valuable in rural areas with low population, but there are now talks about using them in cities as well. However, airspace control, weight limits and noise must be handled with great attention. UAVs are usually better suited to work with ground vehicles in a two-tiered system for completing deliveries that are not far (Engesser *et al.*, 2023).

2.2.1.3 Autonomous delivery vans

Autonomous delivery vans are a new type of delivery van model evolved from the traditional ones. This technology allows the vehicle to drive through cities on its own, either dropping off packages directly or working as a base for smaller autonomous delivery personnel (human or robotic) who finish the last part of the delivery. One such system is the (depot > van > man > home) model, where a driverless delivery van searches for a good parking spot and a human agent brings the items to each doorstep (Boysen *et al.*, 2020). Using this approach, last-mile deliveries are made more efficient and operationally scalable without adding more staff.

Table 2.1 shows a summary of current research on autonomous solutions and the different autonomous delivery solutions mentioned in the literature. Each of these technologies is tailored to overcome certain challenges that are characteristic of the last-mile delivery sector, such as manoeuvring in pedestrian areas, driving safely on the road alongside other vehicles, or connecting to larger logistical networks.

Table 2.1: Summary of Autonomous Delivery Solutions

TYPE OF AUTONOMOUS SOLUTION	DESCRIPTION	KEY FEATURES	EXAMPLES AND NOTES
Sidewalk Autonomous Delivery Robots (Sorooshian *et al.*, 2022; Engesser *et al.*, 2023)	Pedestrian-sized robots operating on sidewalks or bike lanes.	Range: ~20 km; Speed: ~8 km/h; Payload: up to 100 kg; Designed for pedestrian environments.	Optimised for safe navigation in dense urban areas.
Road Autonomous Delivery Robots (Paiva *et al.*, 2021)	Robots sharing road space with conventional vehicles.	Energy efficiency similar to electric vans; Capable of handling larger parcel volumes.	Requires appropriate regulatory adjustments.
Unmanned Aerial Vehicles (Drones) (Engesser *et al.*, 2023)	Aerial vehicles are primarily designed for quick, lightweight deliveries in less dense areas or as part of integrated systems.	Suitable for rapid, short-distance deliveries; Subject to airspace and noise regulations; May need integration with ground systems.	Often used as a component in two-tiered delivery systems.
Autonomous Delivery Vans (Boysen *et al.*, 2020)	Modern, driverless vans designed for urban delivery that may combine autonomous driving with human-assisted final stages.	Capable of autonomous navigation; Acts as a mobile depot for last-mile delivery agents.	Offers potential for reducing driver workload and optimising routes.

2.2.2 Emerging multi-tiered delivery systems

Beyond standalone autonomous systems, researchers have also pointed to the promise of multi-tiered approaches, in which different autonomous solutions work together to address elements of the last-mile delivery chain. For example, a two-level system could consist of an independent van delivering a bundle of parcels to a local hub, where ADRs or drones make the last leg to individual consumers. Such a hierarchical solution would have the potential to greatly optimise the routing, lower energy consumption, and simplify operations, thus

balancing the advantages and disadvantages of each technology (Sorooshian *et al.*, 2022; Engesser *et al.*, 2023). Figure 2.1 shows a multi-tiered delivery system.

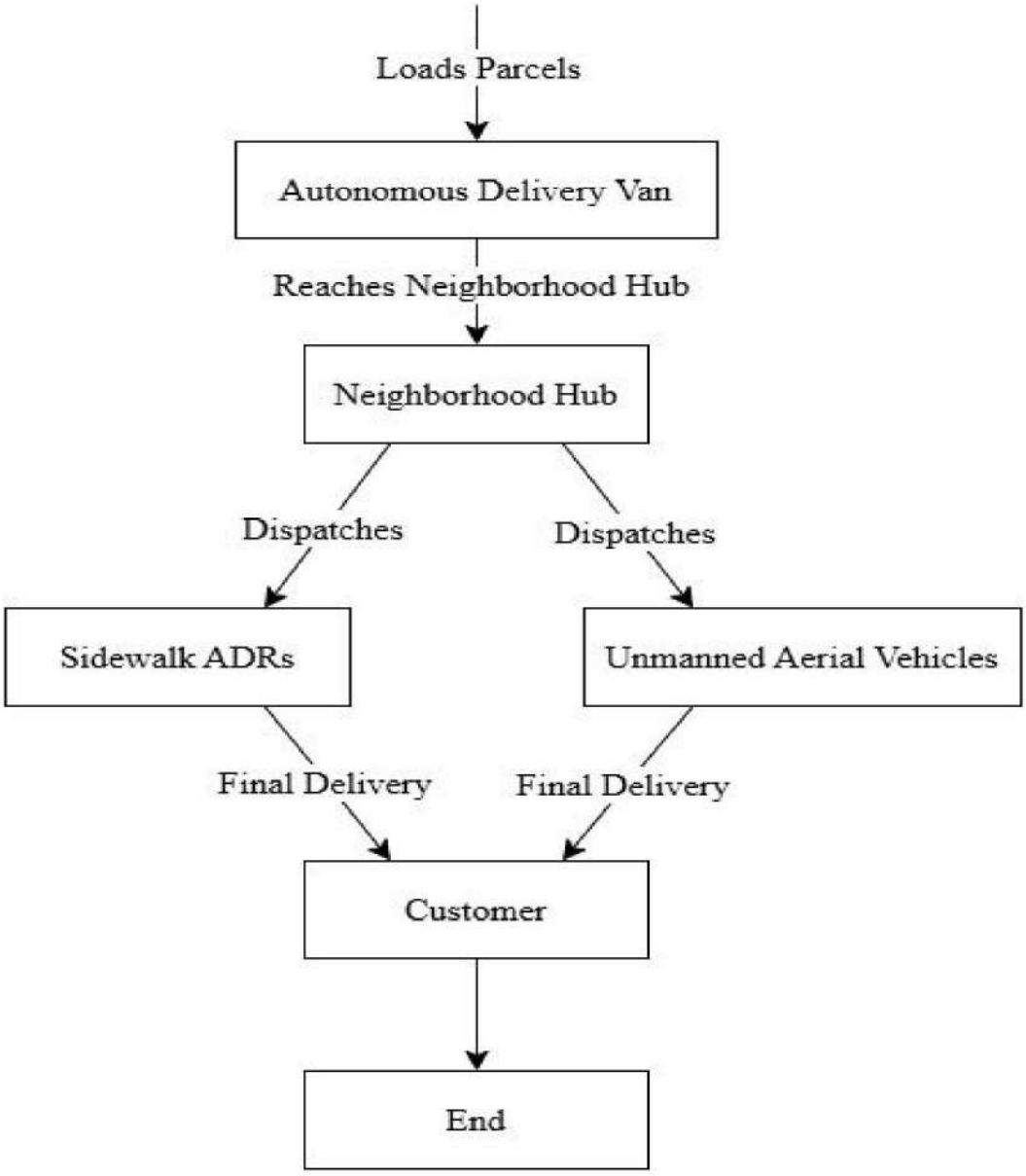

Figure 2.1: Flowchart Illustrating a Two-Tiered Autonomous Delivery System that Integrates Vans, ADRs, and UAVs for Efficient Last-Mile Delivery

2.2.3 Enabling technologies in urban smart mobility

Autonomous delivery solutions do not exist in a technological vacuum. They belong to a larger trend of smarter urban mobility, which uses a combination of enabling technologies to

change urban transport and logistics. The Internet of Things (IoT), Artificial Intelligence (AI), Blockchain, and Big Data analytics are key technologies playing a role in ensuring the successful implementation of autonomous vehicles.

2.2.3.1 Internet of Things (IoT)

IoT technologies enable real-time gathering of data from multiple sensors installed on autonomous vehicles and in city infrastructure. These streams of data allow close tracking of vehicle performance, road conditions and environmental factors, making sure that the delivery operation can adapt dynamically to changing conditions. Such as, intelligent sensors on the delivery robots or vans can be used to optimise routes, as they continuously communicate traffic status and human traffic (Paiva *et al.*, 2021). IoT real-time connectivity plays a major role in the overall efficiency and responsiveness of autonomous delivery systems.

2.2.3.2 Machine learning and artificial intelligence

The operation of autonomous vehicles, in general, and last-mile logistics, in particular, revolve around AI technologies. Advanced analytics and machine learning algorithms allow these systems to consider historical data, identify patterns, and calculate optimal delivery routes on the fly. Sorooshian *et al.* (2022) cited that AI-driven decision support systems are being implemented to optimise operations and cut on delivery inefficiencies, which has been happening in the past few years. AI application in the logistics industry will not only result in more accurate operations but will also spur dramatic changes in customer satisfaction rates due to the quicker and more predictable delivery times.

2.2.3.3 Blockchain technology

Although its main use cases have been in increasing trust and transparency in supply chains, blockchain technology is now starting to enter the realm of autonomous deliveries.

Blockchain can serve as a tamper-proof record of the entire delivery process, including dispatch and final handover, and allow for ensuring that data integrity is maintained throughout the supply chain (Wang *et al.*, 2018). While its direct application in the management of delivery operations is yet to be fully realised, the data security and transaction automation capabilities of blockchain through smart contracts have potential advantages for integrating different autonomous components into a single system.

2.2.3.4 Big data and analytics

Advanced analytics opportunities are presented by the huge volumes of data produced by autonomous vehicles, IoT sensors, and smart city infrastructure. Big Data analytics allow logistics companies to measure the performance of autonomous systems, forecast the maintenance needs, and adjust route optimisation algorithms. The companies can also evaluate the environmental impact of the autonomous delivery operations through the data-driven insights, which can assist them in achieving the sustainability goals, as well as in decreasing the operational costs (Paiva *et al.*, 2021).

The synergy of these enabling technologies is essential to the fact that autonomous vehicles not only work properly, but also learn how to adjust to the complicated and constantly changing environment of the city. These technologies will further synergise as they mature to form a seamless and integrated system that will streamline the last-mile delivery process.

2.2.4 Benefits of autonomous vehicles for last-mile delivery

There are several benefits of autonomous vehicles when it comes to last-mile delivery, which is why they have become an increasingly popular solution among logistics operators. The advantages of autonomous vehicles in urban logistics include operational gains, economic benefits, environmental improvements, and safety, which create a more efficient and sustainable urban mobility ecosystem.

2.2.4.1 Cost reduction and operational efficiency

Significant cost reduction is one of the main benefits of autonomous delivery solutions. The conventional last-mile delivery is as inefficient as it constitutes up to 28% of the overall delivery expenses (Ranieri *et al.*, 2018). Through real-time information and automated routing software, autonomous vehicles have the potential to simplify operational processes, minimise downtime, and improve load and route optimisation. The dual gains experienced are not only on the side of the citizens, who experience shortened delivery times and less urban congestion, but also on the side of the administrations, which can conduct more transparent and efficient supply chain activities (Paiva *et al.*, 2021).

2.2.4.2 Environmental and safety benefits

Autonomous delivery vehicles are usually designed to be electric and extremely efficient. Their deployment can help reduce air pollution and noise levels in cities, which is a major step in minimising transport-related externalities. Electric autonomous vehicles also help to reduce the overall greenhouse gas emissions, which is crucial in the fight against climate change (Ranieri *et al.*, 2018).

Additionally, the precision of autonomous systems decreases the likelihood of accidents resulting from human error, which is a major contributor to road safety (Paiva *et al.*, 2021). Extensive data suggests that with the help of autonomous technology, it is possible to achieve a significant reduction in accident rates and, in addition, facilitate traffic flow in the city.

2.2.4.3 Operation and scalability

Autonomous vehicles have the potential to work 24 hours, unlike traditional delivery systems that are limited to set working hours and human operators. This round-the-clock capability enables it to meet peak work demand, particularly seasonal workload and weekly demand variations, in a more uniform and dependable way (Sorooshian *et al.*, 2022) (Boysen *et al.*,

2020). For instance, the use of autonomous vehicles has been cited to enhance the level of service by guaranteeing that delivery can be executed even at off-peak times or in conditions where the availability of the human workforce is scarce. The scalability of these systems is especially useful in cities where demand can change exponentially over a short period.

2.2.4.4 Decrease dependency on human labour

The logistics sector has faced challenges resulting from an ageing workforce, which has become a major obstacle to the smooth operations of the conventional delivery system (Boysen *et al.*, 2020). Autonomous technologies offer a solution which will minimise the reliance on the human workforce and, therefore, limit the risks of labour shortage or excessive turnover rates. Autonomous systems also liberate human workers, automatically completing some logistics-related duties, including parcel processing, route planning, and direct delivery contact, leaving human workers to concentrate on more advanced processes, including administration and client service (Sorooshian *et al.*, 2022) (Boysen *et al.*, 2020). This will not only enhance operational efficiency but also spur innovation since organisations will free resources to more strategic activities. Table 2.2 offers a summary of operational gains achieved from the use of AVs as well as supporting evidence.

Table 2.2: Benefits and Operational Gains

BENEFIT	DESCRIPTION	SUPPORTING EVIDENCE
Cost Reduction	Automation reduces labour costs and route inefficiencies, cutting up to 28% of total delivery costs.	Observations on last-mile delivery inefficiencies (Ranieri *et al.*, 2018).
Enhanced Environmental Sustainability	Deployment of electric vehicles and optimised routes reduces greenhouse gas emissions and urban congestion.	Emphasis on reducing urban externalities (Ranieri *et al.*, 2018).

Improved Operational Efficiency	Real-time data integration and continuous operation (24/7) streamline the overall logistics process, resulting in faster deliveries and improved customer satisfaction.	Benefits for citizens and administrations (Paiva *et al.*, 2021) (Boysen *et al.*, 2020).
Safety Enhancements	Autonomous systems reduce errors from human drivers, leading to fewer accidents and improved overall road safety.	Reduced reliance on human intervention (Paiva *et al.*, 2021).
Labour Productivity	Reducing dependency on human labour addresses challenges associated with an ageing workforce while reallocating resources to more strategic roles within logistics operations.	Mitigation of labour shortages (Boysen *et al.*, 2020).

Table 2.2 above illustrates how the multifaceted benefits of autonomous delivery vehicles not only lower operational costs but also contribute to broader societal and environmental improvements.

2.2.5 Challenges and barriers in implementation

Even though autonomous vehicles have promising benefits, several challenges impede their widespread adoption for last-mile delivery. Since these issues span technological, legal and societal domains, an integrated solution is needed to address them effectively.

2.2.5.1 Technological and operational challenges

It is complex to develop and deploy autonomous delivery systems in crowded urban environments. Autonomous systems are required to handle unpredictable situations with changing traffic patterns, people on the roads and various kinds of city infrastructures. The following are some of the main technological issues:

1. **System Integration and Real-Time Data Processing:** To use data from IoT sensors, AI and real-time traffic records, it is necessary to have strong communication networks and fast processing capabilities. Any problems with integrating data can

make route optimisation difficult and cause operational inefficiencies (Sorooshian *et al.*, 2022; Paiva *et al.*, 2021).

2. **Reliability and Redundancy Issues:** Autonomous vehicles must be reliable, given that they are expected to work constantly in cities. It is clear from research that while autonomous vehicles can help reduce accidents and make roads safer, any failure in the technology could result in major problems such as traffic congestion or accidents (Sorooshian *et al.*, 2022; Paiva *et al.*, 2021).
3. **Operational Scalability and Flexibility:** Adapting to workload changes in urban environments requires the use of flexible technology. Last-mile delivery systems should be robust enough to handle extra deliveries during busy seasons or in certain regions without affecting their results (Sorooshian *et al.*, 2022) (Boysen *et al.*, 2020).

2.2.5.2 Regulatory and legal barriers

The rules that currently exist in this field are a major barrier to the use of autonomous delivery vehicles. There are many regulatory issues, such as safety rules, insurance concerns and specific guidance on certifications for autonomous systems. Some of the main regulatory issues include:

1. **Lack of Standardised Safety Regulations:** ADRs and other autonomous delivery systems are used in environments that blur the traditional boundaries of vehicle classification. For example, sidewalk robots are not subject to the same safety rules as motorised vehicles, which causes uncertainty as regards their legal deployment (Engesser *et al.*, 2023).
2. **Legal and Liability Uncertainties:** When accidents or system failures occur in autonomous systems, they raise complex legal questions because these systems are

distributed. Organisations should involve legal professionals to design thorough contingency strategies and deal with any legal problems early on (Wang *et al.*, 2018).

3. **Integration with Existing Public Policies:** It is difficult for municipalities to adjust their infrastructure to fit new ways of delivering goods, while keeping in mind the needs of residents, delivery firms and the rules from regulatory bodies. Ensuring new technologies are incorporated into public policies without causing disruptions to current services remains a critical concern (Hoffmann & Prause, 2018).

2.2.5.3 Social and ethical challenges

The social implications of deploying autonomous delivery solutions require thorough consideration. Some key issues in this area include:

1. **User Privacy and Data Security:** Autonomous systems work by sending data continuously, which raises concerns about data security and privacy. If data is not handled properly, it can diminish users' trust in the company and cause psychological distress in some individuals (Sorooshian *et al.*, 2022; Paiva *et al.*, 2021).
2. **Effects on the Labour Market:** While autonomous vehicles may help lower the need for human drivers, they may also cause unemployment. In sectors like last-mile delivery, where the driver is usually the only point of human contact involved, adopting self-driving vehicles could cause many job losses (Sorooshian *et al.*, 2022; Boysen *et al.*, 2020).
3. **Public Acceptance and Trust:** The widespread adoption of autonomous vehicles in cities requires people to trust them. Safety, reliability and transparency must be assured or else the public might not accept these technologies, which could slow down their adoption (Sorooshian *et al.*, 2022; Engesser *et al.*, 2023).

The challenges mentioned above show that all stakeholders involved, including industry stakeholders, policymakers and technology developers, need to work together to tackle these challenges. Some of the recommended actions for bridging this gap include creating pilot projects, establishing regulatory sandboxes and stakeholder engagement initiatives.

2.2.6 Regulatory framework and policy considerations

Since autonomous vehicles are changing last-mile delivery, it is necessary to create a robust set of regulatory frameworks. As ADRs, UAVs and autonomous vans are becoming more important, regulators must focus on keeping the public safe and encouraging new developments.

2.2.6.1 Overview of current regulatory efforts

Several efforts have been made to define the necessary rules and standards for the safe operation of autonomous delivery systems. Even though the rules are not the same everywhere, most regions require following existing safety standards, making special protocols and focusing on data privacy and user privacy (Hoffmann & Prause, 2018; Wang *et al.*, 2018).

There are some instances where the guidelines now consider the unique operational challenges faced by ADRs and UAVs. For instance, autonomous sidewalk robots are not covered by the same rules as motorised vehicles, and this requires making a new set of rules to guarantee their safety around pedestrians and vice versa (Engesser *et al.*, 2023). In addition, those working on autonomous delivery vans are focusing on making sure the vehicles can navigate safely in cities and have fail-safe mechanisms in critical systems (Boysen *et al.*, 2020).

2.2.6.2 Legal and ethical considerations

Laws should address issues of accountability when autonomous vehicles make decisions instead of humans. It has been pointed out in recent studies that the responsibility for accidents or technological failures can be shared among manufacturers, software developers and logistics operators (Wang *et al.*, 2018). For this reason, lawyers, technology experts and policymakers must join forces to create detailed regulations that safeguard everyone.

2.2.6.3 Suggestions for policies in municipalities

The introduction of autonomous delivery systems calls for cities to update both their infrastructure and public policies, and urban administrations are key in this process. Some of the proposed measures include:

1. **Creating Special Lanes or Zones:** It may be useful for municipalities to set up special zones for autonomous delivery vehicles, mainly focusing on sidewalks and roads. With these zones, autonomous vehicles and regular cars can interact safely, and there would be fewer disruptions to pedestrian areas (Engesser *et al.*, 2023).
2. **Infrastructure Modernisation:** By adding sensors to roads and using real-time traffic control, autonomous delivery systems can be made to work better and safer. These upgrades would help other aspects of urban mobility by ensuring that technology and infrastructure cooperate (Paiva *et al.*, 2021).
3. **Stakeholder Engagement and Public Awareness Campaigns:** To foster public trust and ensure hitch-free implementation, municipalities must engage with citizens, industry experts and academics. Clear communication and teamwork allow governments to plan for autonomous vehicles and deal with people's concerns about their privacy, safety and job displacement (Sorooshian *et al.*, 2022; Wang *et al.*, 2018).

Ensuring that new technology is used with updated regulations will help solve the legal challenges that currently hinder the use of autonomous last-mile delivery solutions.

2.3 Object Detection Algorithms

2.3.1 Convolutional Neural Networks (CNNs)

CNNs are a subset of deep learning architecture. They differ from other neural networks in the area of expertise in image, speech, and audio signal processing. Thus, the ConvNet structure is widely used in image analysis and obstacle detection, as it effectively recognises patterns and objects in visual data.

Alzubaidi *et al.* (2021) emphasised that understanding model architecture is a critical factor in improving the performance of different applications. CNN structures generally consist of four distinct layers: The input layer, the convolutional layer, the pooling layer, and the fully connected or dense layer.

1. **Input Layer:** Before an image is passed to this layer, various levels of preprocessing must be applied to it. The input passed to this layer must have equal width and height, with the colour channels either 3 (fully coloured) or 1 (greyscale). Figure 2.2 describes the colour channels of an image.

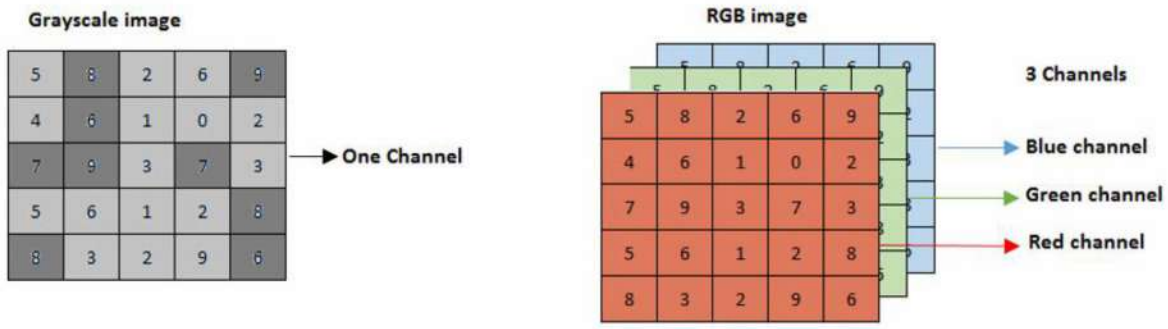

Figure 2.2: CNN Colour Channels of an Input Image (Chauhan, 2023)

2. **Convolutional Layer:** This is where features are extracted from the input stream received from the Input layer. Here, filters (also called kernels) are passed over sets of pixels within the image to extract the most useful information about that section of the image. Information could be about the contours, edges, and pixel intensity. The output of this transformation is passed to an element-wise activation function. The volume of the image (width, height, and colour channels) remains unchanged.
3. **Pooling Layer:** The goal of this layer is to reduce the volume of the image. The bigger the size of an image's volume, the more computation power is required. Downsizing an input stream's volume helps achieve faster processing, reduced memory usage, and a lower chance of overfitting. Figure 2.3 shows the two common types of pooling layers are max pooling and average pooling.

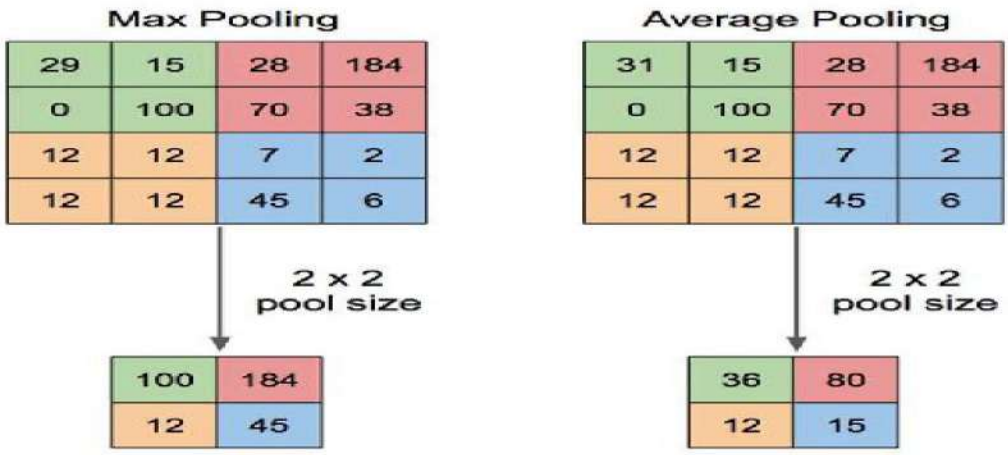

Figure 2.3: Illustration of Max Pooling and Average Pooling (Yani *et al.*, 2019)

4. **Fully Connected Layer:** This layer is aptly self-descriptive. In the pooling and convolutional layers, pixel values of an input image are not directly connected to the output layer. However, in the fully-connected layer, each node in the output layer connects directly to a node in the previous layer. The fully connected layer (FC) is responsible for classification based on features extracted by the previous layers and

their filters. The activation function used in this layer is usually a softmax or sigmoid function, as opposed to the ReLU, which is mostly implemented in the earlier layers. This allows the FC layer to produce a probability between 0 and 1 for classification.

In the context of obstacle detection, multiple images of expected obstacles and non-obstacles would be passed through this network with the expectation that the model would accurately predict the existence of an obstacle in an image.

The shortcoming of this approach for object detection is that the model can only predict the presence of an obstacle. Its size, location, and count are unattainable with this approach. (Pritam *et al.*, 2019) proposed a sliding window method to fix the problem of location. However, this approach is outdated and offers low precision.

2.3.2 Object detection models (YOLO - You Only Look Once)

Chenhao *et al.* (2023) performed an in-depth analysis of 7 different YOLO models for obstacle detection. They found that the YOLOv8 had the best performance, with 68.2% recall and 80% precision. Though not an ideal number for such metrics, it performs reasonably better than a traditional CNN. Thus, it is important to understand the basic architecture of a YOLO model.

Built on top of CNNs, YOLO is a single-pass object detection model. Rather than image classification, YOLO focuses on Image localisation. Image Localisation identifies and locates one or more objects in an image using bounding boxes (rectangular shapes encapsulating the identified object).

Image localisation is the process of identifying the correct location of one or multiple objects using bounding boxes, which correspond to rectangular shapes around the objects. This is achieved through dividing an input image into a set of grids. Each grid is responsible for

predicting the class of the object contained within. As YOLO is a single-shot algorithm (processes an image in one pass), it is more suitable for autonomous systems, as they require quick inference in real time. The YOLO architecture contains 24 convolutional layers, four max-pooling layers, and two fully connected layers. Its primary components are the backbone used for feature extraction, the neck for multiscale detection, and the final head, called the dense prediction layer. A typical YOLO base architecture is shown in Figure 2.4.

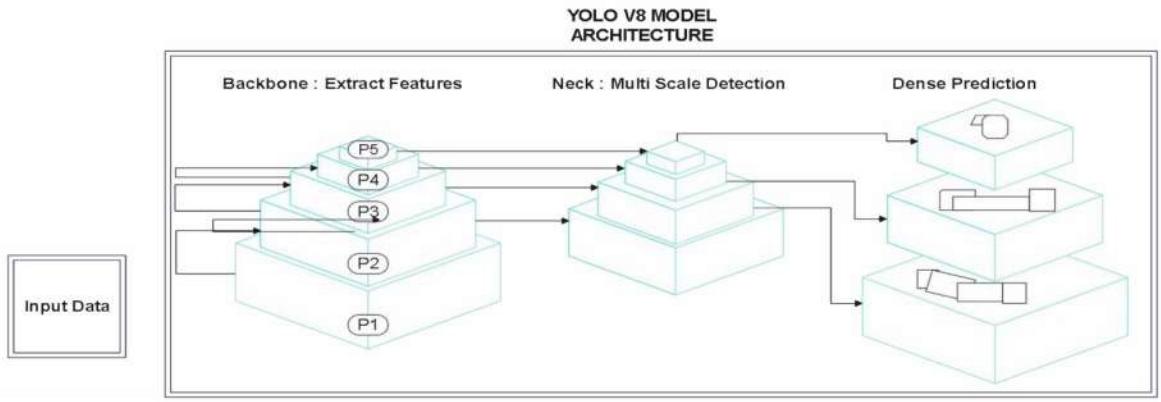

Figure 2.4: YOLOv8 Architecture for Object Detection and Activity Identification (Abraham, 2020)

YOLO is a modification of CNNs where a regression head uses anchor boxes and loss functions. An anchor box is simply a template to ensure accuracy in bounding box predictions for object position and size. The four major sections of the YOLO process are: grid-based predictions, bounding box regression, anchor boxes, and a custom loss function.

2.3.4 Object detection models (SSD - Single Shot MultiBox Detector)

Ashwani *et al.* (2020) claimed that the SSD is the fastest algorithm that uses a single layer of a convolutional network to detect objects in an image. Single Shot MultiBox Detector is an object detection algorithm that balances the speed and accuracy of predictions in time. It is known for its efficiency in classifying objects of varying shapes and sizes. The 'MultiBox' is a technique for bounding box regression designed by C. Szegedy *et al.* (2016).

The VGG-16 architecture is the base network for the SSD architecture. However, the fully connected layers are discarded. The VGG-16 model is known for its impressive performance in the classification of high-quality images, thus a great architecture to build upon. Figure 2.5 shows the SSD architecture with a VGG-16 baseline.

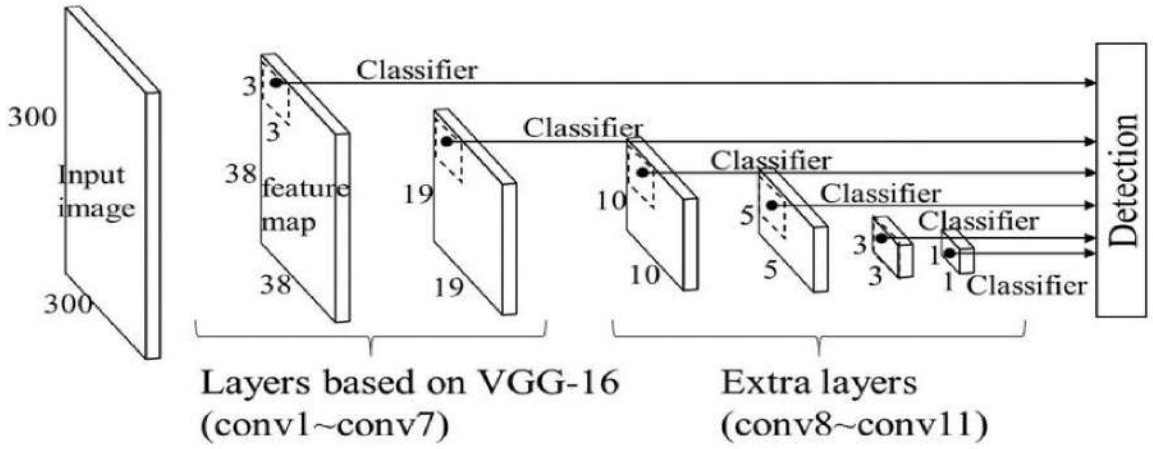

Figure 2.5: Single Shot MultiBox Detector (SSD) Architecture (Koga *et al.*, 2020)

The two loss functions implemented in the MultiBox technique are the confidence loss and the location loss.

1. **Confidence Loss:** This is a measure of how confident the model is in its predictions. Categorical cross-entropy is used to compute this loss. This loss function helps in creating a distinction between the object and the background and also reduces false positives, leading to an overall more accurate predictive model.

2. **Location Loss:** L2-Norm is used here. It is the measure of the accuracy of the predicted boundary box with respect to the 'ground truth' box. This loss function helps in reducing the model's localisation errors, leading to better object size and position estimation.

2.3.5 Image segmentation

Ying *et al.* (2023) explained the Image segmentation process as the division of a single image into multiple regions that possess varied features. The process ends with the extraction of a region of interest within an image (ROI).

Image Segmentation is a computer vision process that divides an image into several regions/partitions. Pixels in these partitions would share similar characteristics. All objects in an image can be categorised as either countable (animals, furniture) or uncountable objects (e.g., sky, grass, sea). There are three ways image segmentation can be implemented:

1. **Semantic Segmentation:** In this subclass, every single pixel in an image belongs to a class. In this method, multiple objects belonging to the same class are not distinguished but are generically considered as a class. A single pixel value is still assigned to the objects in the same class. Semantic Segmentation is best suited for labelling countless objects.
2. **Instance Segmentation:** This is an upgrade from Semantic Segmentation, as different objects from the same class can be distinguished. Different pixel values are assigned to objects in the same class. Instance Segmentation is best for labelling countable objects.
3. **Panoptic Segmentation:** This is an integration of the semantic and instance approaches. Objects in a class are still assigned the same pixel value. However, these objects are still understood to be separate entities and are labelled separately.

The general architecture for an Image Segmentation model can be built on top of a Fully Convolutional Network (FCNs). These differ from the traditional network as the dense layers are replaced with a 1:1 convolutional block. More information about the image can be attained through this convolutional block's up sampling, down sampling, and max pooling.

These extra layers also make FCNs easier to train than traditional CNNs. Figure 2.6 shows a CNN model configuration for semantic segmentation.

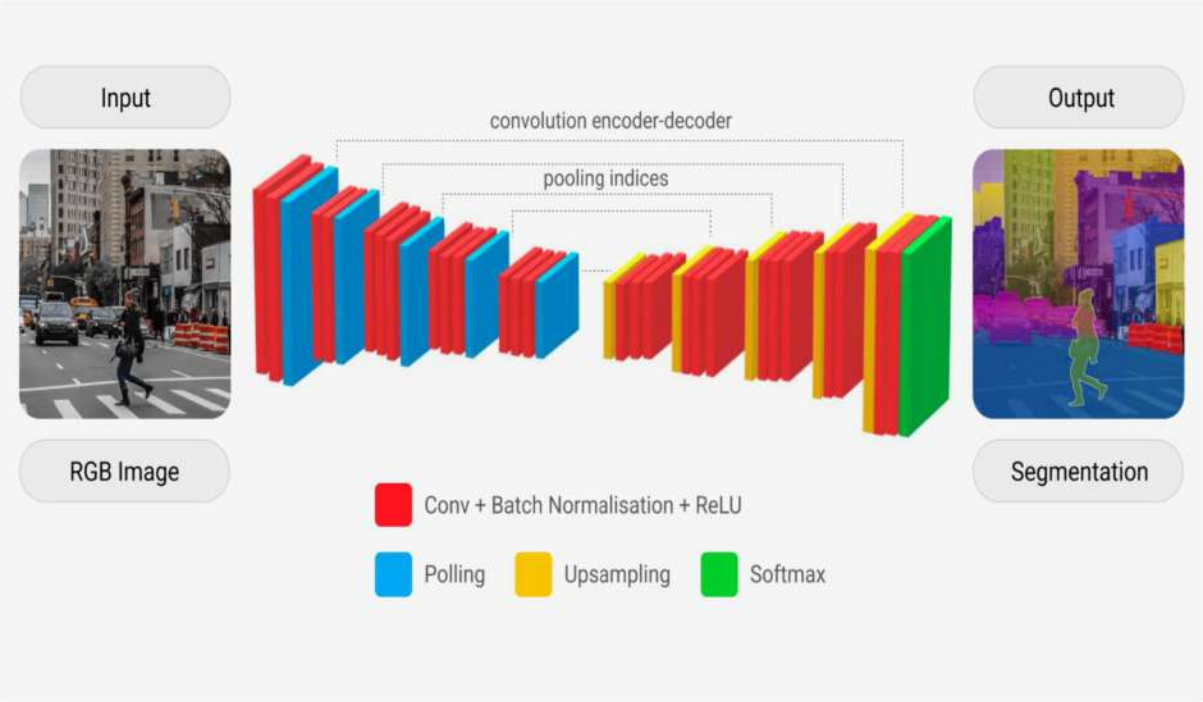

Figure 2.6: A CNN Architecture for Image Segmentation, (Complete Guide to Semantic Segmentation | Mindy Support Outsourcing, n.d.)

2.3.6 Transfer learning

Matteo *et al.* (2024) pointed out the limited size of datasets in the object detection field. They outlined that the best way to address this limitation is through two machine learning strategies. The first would be pretraining on existing general datasets. The second would be creating synthetic datasets tailored to the specific task (in our case, object detection). This is the approach of transfer learning. In Transfer Learning, there are two major domains:

1. **Source Domain:** This is where the base model was trained initially. This domain's dataset possessed a large amount of labelled data for training. Thus, the model's ability to recognise patterns related to the domain is very high.

2. **Target Domain:** Here, the base model is fine-tuned to meet the expectations of the actual objects to be detected. This domain typically has a smaller dataset for training. The model now has to relearn features and patterns related to this target domain.

The YOLO or R-CNN model may be used as the source domain, then fine-tuned using the road traffic labelled dataset (cars, people, signs) for object detection in autonomous vehicles as the target domain. This process is shown in Figure 2.7.

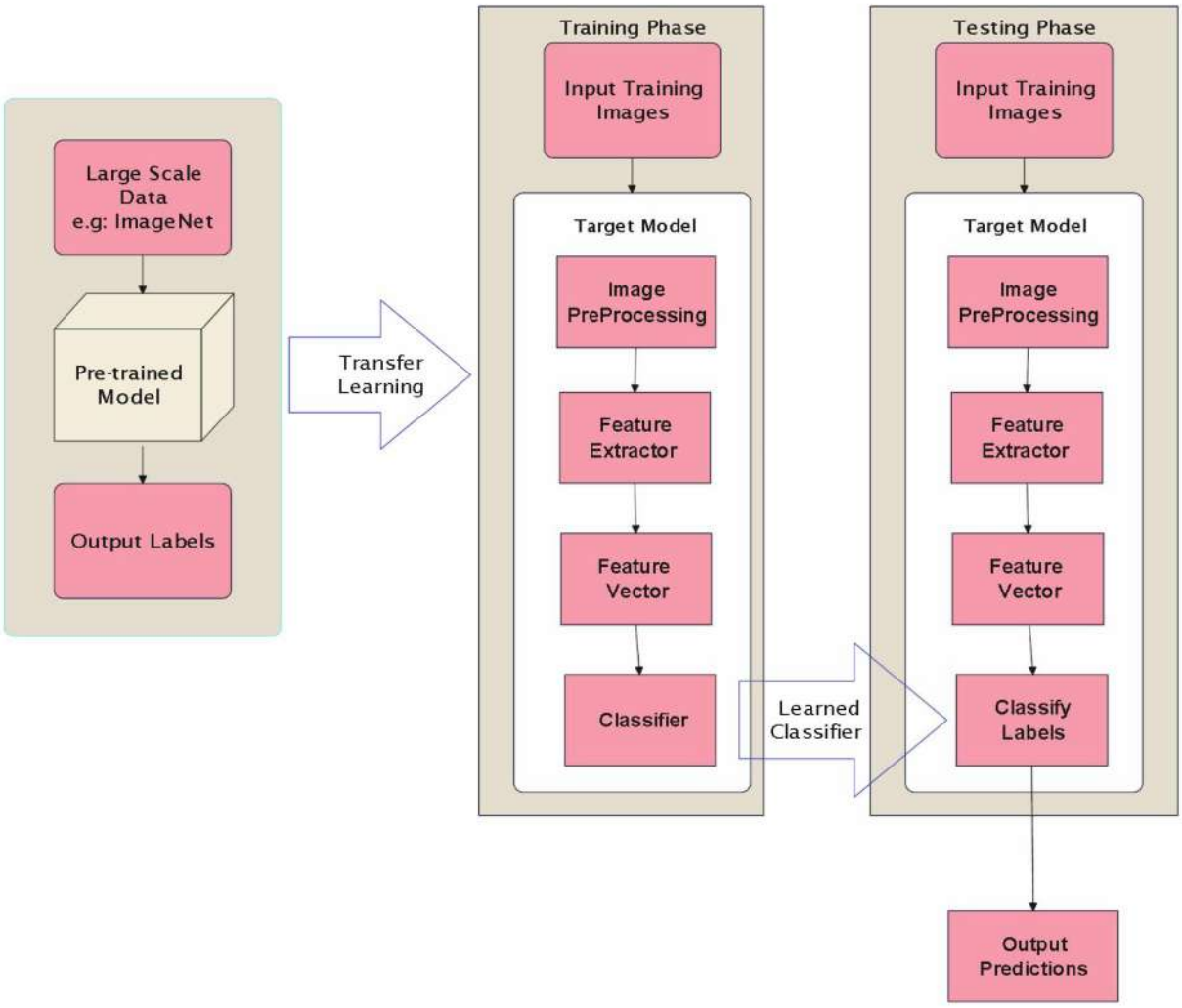

Figure 2.7: The Transfer Learning Process

Below is the general flow for Transfer Learning:

1. A base Model related to the target domain is selected.

2. The Output Layer of the base model is removed or modified. The new output layer must suit the target domain, i.e., have a neuron count equal to the object classes of the target domain.
3. The initial layers of the pre-trained layers are frozen. This is done to avoid a case where the pre-trained weights are updated too quickly. Initial weights are easily transferable across domains, since they capture low-level features.
4. The modified model is trained and evaluated using a validated dataset. In the case where performance metrics are unappealing, the model's hyperparameters are fine-tuned, and the model is retrained.

2.3.7 Faster Objects More Objects (FOMO)

The FOMO model is intended to work in real-time to detect objects on devices with extremely limited memory and computing capabilities, such as microcontrollers (Edge Impulse, 2022). The centroid-based detection (as opposed to bounding boxes) focuses on accuracy, memory, and compute cost, resulting in a balance. Figure 2.8 reveals the basic pipeline of the FOMO model. Each layer of the model is explained in detail below:

1. An Input Layer receives 96x96 3-channel input images, which are then converted to greyscale to save memory without loss of accuracy.
2. Next is a MobileNetV2 ($\alpha = 0.1$) backbone, truncated early in the seventh layer (block6expand_relu). The α represents the multiplier of the width of the backbone that is truncated at ⅛ input resolution. This truncation causes the 96x96 input to become a 12x12 feature map; The output contains 96 feature channels. MobileNetV2, in this case, is used as an efficient yet powerful feature extractor.
3. The FOMO convolutional head, a small convolutional head rather than a fully connected head, is appended. This allows the model to remain fully convolutional and

efficient. It typically contains 3 custom layers sequentially. The last layer produces raw scores (logits per spatial cell), which are mapped to the output classes.

4. The logits are then applied to the output layer through a softmax activation to get the likelihood or probability of the cell belonging to a particular class. The result is a 12x12x3 heatmap grid with each pixel representing one physical location in the feature map.

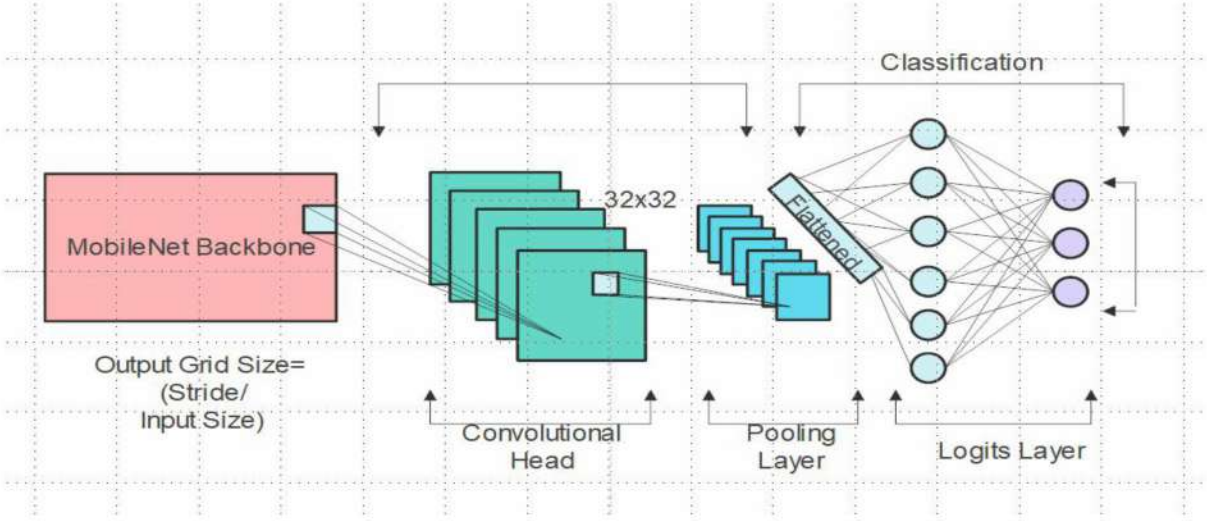

Figure 2.8: The FOMO Model Architecture

In contrast to conventional object detectors, which are based on presuming bounding boxes, FOMO identifies object centres (centroids). Identification is made easier, and the cost of computation is reduced. The positions of objects identified are the heatmap peaks. The main advantages it has over other models are; it is fully convolutional, resulting in saving of a lot of memory and compute resources, it uses a lightweight backbone (MobileNetV2) that can easily run on resource constrained microcontrollers, it can run in under 30 FPS as it is lightweight and it uses a simpler output (heatmaps with centres) allowing real time multi-object detection to be performed on resource constrained microcontrollers.

The FOMO model consists of a lightweight MobileNetV2 trunk as a feature extractor, and a small convolutional classifier head, which will generate spatial heatmaps of class probabilities of object centres (cx, cy) rather than bounding boxes.

Table 2.3 shows a summary of the previously discussed object detection approaches other than the FOMO model, with research by Redmon *et al.* (2018), Liu *et al.* (2016) and He *et al.* (2017) highlighting metrics like accuracy, speed and transferability.

Table 2.3: Comparison between Object Detection Algorithms

METRIC	CNN	YOLO	SSD	TRANSFER LEARNING	IMAGE SEGMENTATION
NMS	Yes (common)	Yes (integral to pipeline)	Yes	Yes (if used for detection tasks)	No (uses post-processing like CRF)
mAP/IoU	High (mAP for detection)	Moderate - High (YOLOv3 ~57.9%)	Moderate (~41.2% mAP on COCO)	Varies based on pre-trained models	High (IoU or mask mAP: ~37.1% on COCO)
Speed (FPS)	Low	Very High (~45-60 FPS)	High (~22-46 FPS)	Varies (based on base network)	Low (10-15 FPS)
Complexity	High (large computation)	Medium (optimised for speed)	Moderate (smaller but complex)	Varies (depends on base network)	High (large model with many parameters)
Localisation Accuracy	High (bounding boxes)	Moderate - High (~85%)	High (~78% SSD512)	Moderate	Very High (pixel-wise accuracy)
Real-Time Suitability	No	Yes (real-time design)	Yes (suitable, slightly slower)	Varies (depends on model complexity)	No (Mask R-CNN is slow for real-time)
Training Time	High	Moderate	Moderate	Low (pre-trained model)	Very High (longer due to pixel-wise tasks)
Model Size	Very Large	Small - Medium	Medium (~100	Varies (smaller	Large (M R-CNN ~250 MB or more)

		(236 MB)	MB)	when pre-trained)	
Transfer-ability	Moderate	Low	Low - Moderate	Very High (core use case)	Moderate (segmentation models are often task-specific).

2.4 Importance of Obstacle Detection in Autonomous Systems

2.4.1 Critical role in safety

Wang *et al.* (2020) studied 128 accidents between the years 2014-2018 and discovered that about 63% of accidents were caused in autonomous modes, with 6% related directly to autonomous vehicles. However, 94% of these accidents were started passively by outside parties, such as other human-driven vehicles and pedestrians. To eliminate this low percentage of accidents caused by autonomous vehicles, obstacle detection and avoidance are greatly required.

In smart cities and urban areas where traffic is at its peak, a minimum of 12 hours a day, smart object detection is greatly favoured. There is only so much that can be detected by the human eye. Coupled with the various distractions like noise, other drivers, and internal conflicts within a vehicle, the driver's accuracy in pinpointing obstacles cannot attain an ideal percentage. This is where obstacle detection using machine learning algorithms and sensors comes in. With enough fine-tuning and re-evaluation, obstacles can be pinpointed for avoidance with nearly ideal accuracy in record time. This will help in preventing road accidents through collisions, ensuring passengers, drivers, road infrastructure, and pedestrian safety. Autonomous systems prioritise human safety and ethical decision-making, thus making this approach better, as human decisions would always completely or partially be made in favour of oneself.

2.4.2 Navigation efficiency

The invention of autonomous systems dates back as far as the 1980s. With the implementation of vehicles that can navigate the road without a physical driver came a lot of problems. The biggest being the ability to effectively evade and reroute due to varying obstacles on the otherwise straight pathway. That isn't to say this issue was non-existent with vehicles controlled by physical drivers. A major cause of accidents on the road is due to a driver's inability to navigate away from an obstacle (usually fast-moving) in record time.

Paul *et al.* (2024) recognised the need for a mapping and mission planning system for autonomous vehicles to ensure the best route is followed while avoiding obstacles on the road.

With the introduction of obstacle detection through artificial intelligence, obstacles moving at varying speeds can easily be detected regardless of frequency in direction change, as these obstacles are identified in real time. This obstacle detection can be combined with GPS navigation for efficient navigation of autonomous vehicles. The goal is to achieve a high accuracy in detection with the highest fps (frames per second), as its latency.

2.4.3 Real-world application in delivery vehicles

Ashraf *et al.* (2024) recognised that the significance of AI in autonomous systems lies in its ability to enhance functionality, efficiency, and safety. They insisted that autonomous vehicles are required to leverage machine learning algorithms and sophisticated sensors for navigation within complex environments to navigate complex environments.

Delivery robots and drones are becoming a frequent go-to in urban environments and smart cities. It is necessary to have these systems follow a safe, unblocked path to protect both pedestrians and the actual mechanism. A great deal of expenses is involved in developing these robots and drones. Due to oversight during programming, a collision between the robots or drones and an obstacle could occur, damaging the equipment, making it unusable. To

avoid this monetary loss, obstacle detection algorithms must be programmed into these robots during their creation. In conclusion, the costs associated with accidents, delays, and system failures are greatly reduced.

2.4.4 Adaptability to dynamic environments

As discussed previously, obstacle detection and avoidance allow vehicles, robots, drones, and other systems to respond to moving objects or obstacles. Moving objects could include other systems, humans, flying objects, stray animals, etc. There are also cases of blind spots in intersected roads. Even in rural areas where roads are still underdeveloped and unstructured, with the necessary traffic signs and lights close to non-existent, obstacle detection is much needed. The road is unpredictable and dynamic. Implementing smart obstacle detection lessens the risks involved in this mode of transportation. Many regions (mainly smart cities) even require obstacle detection systems integrated into autonomous vehicles to be allowed for public deployment.

Ravikumar *et al.* (2022) investigated several path optimisation algorithms of dynamic road agents, like StopNet, MotionCNN, to further assist path planning algorithms. Thus, it creates a safer, more optimal path for autonomous vehicles in dynamic environments.

2.4.5 Enhancement of public trust

Qiyuan *et al.* (2024) deduced that accidents induced by autonomous vehicles heavily reduce public trust and destroy their perception of autonomous vehicles.

With the evolution of dependable obstacle-detecting autonomous systems, there is a rise in public confidence in adopting these methods into their way of life. The more people adopt the use of vehicle autonomy, the fewer people are found physically on busy roads. This reduces the risk of collision and accidents down to merely damage to property and cars, eliminating loss of life.

In the long run, an efficient obstacle-detecting autonomous system will bring about peaceful road navigation, increasing communications between vehicles over the Internet, thereby decreasing common distractions that lead to collisions by physical drivers. In conclusion, advancement in obstacle detection paves the way for more sophisticated and widespread use of autonomous technologies.

2.4.6 Impact on system design

There has been a rise in the development of sensors like cameras, radar, and LiDAR due to the emergence of smart obstacle detection algorithms. This benefits the industries behind the production of these components, as well as serves as a prompt to develop even better versions of these sensors, further improving the reliability and dependability of autonomous systems.

2.5 Review of Related Works

2.5.1 Autonomous navigation systems

Imad *et al.* (2022) proposed a new deep learning enabled local planner that merges semantic segmentation with deep learning-based motion control. A SegNet-based network classifies drivable sidewalk vs. obstacles, supplying the inputs for a nonlinear model predictive control (NMPC) planner. The robot performs collision-free motion by dividing the navigable terrain into partitions and subsequently planning with NMPC. Real-world tests and simulations showed that the system can produce safe velocity commands even in previously unseen environments and outperform a typical Dynamic Window Approach (DWA). Remarkably, the authors attain zero-shot Sim2Real transfer, i.e., they do not retrain on real data, and achieve robust sidewalk navigation. They obtain RGB images methodologically, use binary segmentation to identify sidewalks, and use an NMPC local planner to navigate waypoints and avoid obstacles. Experimental data show better smoothness of motion and collision

avoidance compared to DWA and highlight the usefulness of integrating perception and model-based control.

Katona *et al.* (2024) present an extensive review of path planning and obstacle avoidance algorithms for mobile robots. They survey classic global planners (A*, D*, sampling-based) and reactive obstacle avoidance (Bug algorithms, VFH), as well as contemporary techniques. The comparison of the advantages and disadvantages of these methods and the description of their fields of use are provided in the analysis. This work is useful to practitioners who would like to know how planners and avoidance techniques have been applied in practice, although it is a review. It points to the necessity of hybrid approaches that trade off global optimality with real-time safety and observes that work is underway on machine-learning-based local planners in dynamic settings. The article helps in putting into perspective how modern autonomous vehicles incorporate multiple planners and sensors to enable robust navigation.

Liu *et al.* (2022) addressed the gap in GPS navigation applied to embedded technology. Their motivation was to combine intelligent processing and network information towards realising a faster and accurate positioning technology. The general goal of the article was to accumulate reliable application data, thereby providing a reference for the continuous improvement of the GPS navigation system. The research was carried out on a Zigbee network, which has a Zigbee node as its core. A Zigbee node (an embedded node) consists of a system-on-chip with a core processor, wireless communication module, and peripheral circuits. Several Zigbee networks are connected in what mimics a mesh topology. This allowed a stable cluster network, controlling node parameters. According to the article, the operating efficiency is within the range of 50%-70%. For the hardware design, a processor module, an audio processing module, a video receiver module, a system power module, and a USB module were integrated. As for the software design, they refined the embedded

operating system and the software system architecture to fit their goal. The system's working mechanism has three parts: the priority inversion mechanism, the event handling mechanism, and the system state mechanism. The goal of the priority inversion mechanism was to schedule the received application data in a fixed priority order to ensure the integrity of transmission results. The event handling mechanism was responsible for delivering notifications after an event occurred, as well as subdividing the event into multiple received events according to event complexity to meet the event delivery requirements. The function of the system state mechanism was to monitor system operation parameters and sort and allocate system resources, thereby improving the utilisation of the system operation resources. Two types of interrupt sources were added to the system design: the non-maskable interrupt and the maskable interrupt. For the result analysis, they proposed three tests: The accuracy test, the real-time test, and the adaptability test. For the accuracy test, the evaluation index should be the signal matching rate. This matching rate for the test road needs to exceed 98% to be branded as a ‘good accuracy vehicle. The real-time test operates on the idea that the display position of the experimental vehicle on the electronic map is the same as the true driving position of the vehicle. To have excellent execution efficiency, the update frequency of the position point must be within 1.0s. For the adaptability test, a prediction algorithm should be implemented for calculating new matching points in the case of a signal anomaly problem. The article concluded that optimising the design points of a GPS vehicle navigation system can lead to the realisation of accuracy and real-time map matching. This is positively significant for system practicability improvement.

Nahavandi *et al.* (2022) offered a survey in the rapidly growing field of autonomous mobile robots. The motivation behind this survey was to address the growing influence of deep learning applications in autonomous navigation and also to contribute to the pool of research as the autonomous navigation technology evolves at an alarmingly rapid pace. The strengths

and limitations of the range of sensor combinations and mathematical expressions, and approaches were considered. They covered various elements such as sensor types and fusion, simulation tools, path planning, obstacle avoidance, and SLAM (Simultaneous Localisation and Mapping). The article reviewed popular tools for simulation and introduced publicly available datasets for SLAM development, reducing the need for physical robots.

Mushtaq *et al.* (2021) proposed a two-phase approach for autonomous vehicle traffic management. These were the platooning and collision avoidance strategies with V2V (Vehicle to Vehicle) and V2I (Vehicle to Infrastructure) communications for the purpose of reducing congestion and improving traffic flow. There was a significant improvement in regard to the flow of traffic when using platooning. The travel time was efficiently reduced from 33 minutes to 8 minutes for 600 vehicles in the controlled congestion scenario. Collision avoidance was also effective, with the collision rate dropping from 73% (when only V2I) to 12.5 (when using a V2V/V2I combination). However, energy efficiency and real-time latency by the autonomous vehicles were now considered. For future work, they recommended using C-V2X (long-range cellular-vehicle to everything) and 5D-NR (5G-radio) to implement their strategies, possibly reducing the latency.

Casado *et al.* (2020) presented a simulation framework that could be applied to aid engineering students in the development and testing of autonomous drone navigation systems. They showed how the framework could be used according to the rules of the 'ESII Drone Challenge' student competition and how this framework can be evaluated using multiple test scenarios. The article successfully simplified autonomous drone navigation development through the abstraction of lower-level control details. However, their image processing method lacked analysis on perspective, thereby affecting frame position evaluation accuracy.

The paper also only covered the simulation range, the framework, yet to be validated with real drones in real-life scenarios.

Valera *et al.* (2021) presented an optimal trajectory planning algorithm that allowed the energy-efficient assessment for autonomous light vehicles. The algorithm ensured obstacle avoidance, energy efficiency, and real-time navigation under dynamic conditions using ROS (Robot Operating System) middleware. Both static and dynamic obstacle avoidance were incorporated, and this real-time navigation was tested on an actual car-like vehicle (RBK). The paper prioritised autonomous vehicles' energy consumption as a constraint in the optimisation problem. They considered the minimal time that could be exhausted during the trajectory, taking into account the dynamic behaviour of the vehicle during obstacle avoidance.

2.5.2 Obstacle detection and avoidance algorithms

Clotet & Palacín (2023) introduced SLAMICP, an iterative closest-point (ICP) implementation that reports the output of an ICP's iterative scan "outliers" as obstacles. The library simultaneously estimates robot pose using ICP and labels points as obstacles by intercepting the ICP distinction between matched and unmatched scan points at every iteration. In such a way, both obstacle detection and pose estimation are carried out simultaneously. The results of tests demonstrated that the implementation of obstacle detection within ICP reduced the total processing time by 36.7% in comparison with an additional obstacle-detection scan, thus speeding up the process of obstacle detection in mobile robots as a part of a pre-designed ICP pipeline, which is especially important in the context of real-time navigational strategies.

Tijani (2022) reviewed the stages of autonomous driving, safety, road formation control, path planning, and obstacle detection and avoidance for autonomous vehicles. Equations were

used to address safety models for lane change and collision avoidance. For real-time path planning, an improved APF (Artificial Potential Field) was proposed. The MRF-based obstacle detection and semantic segmentation were discussed in detail. Finally, formation control approaches were covered. This included the leader-follower, virtual structure, and behaviour-based approaches. The paper, however, did not address embedded system constraints or the energy efficiency of these techniques applied to autonomous vehicle technology. They also focused heavily on lane-change scenarios, lacking a deeper evaluation of broader path complexity.

Katona *et al.* (2024) provided an overview of the advantages, limitations, and application areas of obstacle avoidance algorithms like the Bug algorithm, Dijkstra's algorithm, and other newer developments like generic algorithms. The paper analysed both classical and modern (AI/deep learning) methods, providing a structured taxonomy and classification. A table of classical, heuristic, and AI-based algorithms was drafted in the thesis with columns such as calculation and convergence time. They also addressed global and local planning and hybrid systems.

Alaa *et al.* (2024) considered the numerous technological developments that have been made, such as the Internet of Things (IoT) and edge computing, and why it is desirable to include machine learning techniques in embedded devices that have limited resources to achieve dispersed and ubiquitous intelligence. They reasoned that though larger networks are more efficient and versatile than smaller ones, the amount of energy that these networks consume rises in direct proportion to the size of the network. As a consequence of this, the process of expanding neural networks cannot be maintained on a large scale. The paper presented the most important algorithms used in object detection, in addition to the challenges in using low-resource embedded systems that support TinyML technology. Two general methods

were explored: the one-stage method and the two-stage method. For the two-stage method, three machine learning algorithms were studied: Region-Based Convolutional Neural Network (R-CNN), Fast R-CNN, and Faster R-CNN. Faster R-CNN was able to achieve even faster inference times while keeping a high level of accuracy compared to the former two. This is done through the implementation of a Region Proposal Network (RPN) in the model itself.

In the area of the one-stage method, three machine learning algorithms were explored: You Only Look Once (YOLO), Single Shot Detection (SSD), Feature Pyramid Network (FPN), and Faster Objects, More Objects (FOMO). Where the R-CNN does hundreds of evaluations on a single image, the YOLO algorithm just does a single evaluation of an image. This was the basic factor that led to the incredible speed of a YOLO model, more than one thousand times faster than RCNN models and one hundred times faster than Fast R-CNN models, as stated by the author (Lin, Szu-Yin, Hao-Yu Li, 2021).

The SSD's capability to detect objects at an extensive range of dimensions renders it well-suited for an extensive variety of applications, such as robotics, surveillance, and autonomous driving. FPNs are an improvement of the traditional CNNs in terms of feature extraction. The capabilities of classic CNN models can be improved with the help of FPN, which enables these models to generate feature maps that are more useful for subsequent tasks such as object detection or classification analysis.

Finally, for the FOMO model, the methodology is based on the process of dividing the input image into tiles of a predetermined size, followed by the application of a classifier on each grid. At this juncture, it is possible to build a heat map that visually represents the approximate positions of objects, wherein a smaller tile size is indicative of higher levels of accuracy.

The journal admitted that though there were potential heavy gains to be made by the implementation of TinyML in our daily life, there were also several challenges, like hardware limitations, model optimisation, energy efficiency, and security. The journal also provided a table of devices that supported TinyML. They concluded that while there are many challenges to overcome, tiny machine learning offers many advantages, including real-time processing and lower energy consumption. As the technology continues to evolve, it can be expected that more innovative applications of micro-ML in various industries will change the way we live and work.

Hamidaoui *et al.* (2025) provided a comprehensive survey on the primary collision avoidance algorithms regularly used in autonomous vehicles. They covered sensor-based methods for obstacle detection, advanced path-planning algorithms for facilitation of reliable routing and decision-making systems to allow autonomous vehicles to dynamically respond to varied driving scenarios and conditions. They observed the role Machine Learning takes in enhancing the performance of autonomous vehicles when it comes to obstacle avoidance. For path planning, algorithms like A* and RRT were studied, with reinforcement learning as a device for adaptability in dynamic environments. Rule-based and Optimisation methods for decision-making algorithms were considered for lane changes and coordination with other vehicles. The Monte Carlo Tree Search, Reinforcement Learning, and Supervised Learning were reviewed as techniques for object detection, classification, and adaptability in complex traffic conditions. The article concluded with the fact that the future of autonomous vehicle collision avoidance technology greatly depended on overcoming limitations through emerging technologies. Edge computing and federated learning can be adopted for faster, decentralised data processing. Reinforcement learning environments could be improved upon for better decision-making, and Explainable AI can be adopted for more transparent,

trustworthy systems. Finally, much more advanced sensors could be applied for a higher accuracy in environmental perception.

Wu *et al.* (2023) aimed for the improvement of obstacle avoidance and path planning for autonomous vehicles through the integration of attention mechanisms via multimodal information decision-making. This was designed in an end-to-end architecture. Allowing vehicles to perceive their dynamic environments, thus making more intelligent navigation decisions, was the biggest motivation for this study. The system proposed consisted of an attention mechanism for processing multimodal inputs like image data and LiDAR. Then, LSTM (Long Short-Term Memory) networks were implemented for handling sequential decision-making within these dynamic conditions. This attention mechanism and LSTM were combined in an end-to-end architecture, which integrated perception, decision-making and control. Experiments were conducted on real-world datasets such as the Waymo Open, ApolloScape, KITTI, and Cityscapes. The evaluation metrics used were Localisation Error, Object Detection Accuracy, Obstacle Avoidance Success Rate, and Energy Efficiency. This method delivered better localisation (LE=1.02m), greater object detection accuracy (ODA=93.33%), and a high rate of success for obstacle avoidance (OASR=96.97%) for the Waymo dataset. The proposed method also proved to be energy efficient (EE = 0.10 kWh/km) with a faster training time, performing consistently across different environmental conditions.

2.5.3 Embedded microcontroller platforms

Sanchez-Iborra *et al.* (2020) addressed the scarce attention given to edge devices in regard to machine learning due to their inherent computing constraints. They insist that since these entities, Microcontroller Units (MCUs), are embedded in our daily appliances like medical devices, personal gadgets, etc, it is only natural that machine learning, which has been integrated in every niche of life, may also be incorporated within them. The article exposed

the fact that much data captured by sensors is being wasted due to transportation costs, power constraints, and bandwidth limitations. Thus, performing on-device machine learning has become the trend. As quoted, 'Forecasts predict that the global edge computing market will reach 1.12 trillion dollars by 2023, and some strong companies such as Ericsson are already offering TinyML-as-a-Service solutions. The article provides a comprehensive overview of the current challenges and opportunities of TinyML. The specific goals of this research were to provide a comprehensive review of the TinyML ecosystem, its related challenges and opportunities, and the potential services that will be enabled by the development of truly smart frugal objects. They also provided a detailed survey of the available TinyML frameworks for integrating ML algorithms within MCUs. They proposed a Multi-Radio Access Network (RAT) architecture for smart frugal objects. The core element of the device was the MCU, which gathered the processing unit, as well as program and data memories. A range of communication protocols were frequently employed in the MCUs, e.g., Universal Asynchronous Receiver/Transmitter (UART), Serial Peripheral Interface (SPI), or Inter-Integrated Circuit (I2C). Although displays are highly consuming, they were useful to show important events or status messages. The issue of selecting the most adequate communication interface for sending sporadic messages, considering both the status of the device and the characteristics of the data to be sent, was addressed. To this end, several TinyML frameworks were evaluated, and the performances of several ML algorithms embedded in an Arduino Uno board are analysed. The attained results reveal the validity of the TinyML approach, which successfully enabled the integration of techniques such as Neural Networks (NNs), Support Vector Machine (SVM), decision trees, or Random Forest (RF) in frugal objects with constrained hardware resources. The outcomes also show promising results in terms of the algorithm's accuracy and computation performance. The article concluded that the integration of ML within IoT devices will provide flexibility and processing capability never

seen before. The TinyML paradigm proposes to adapt advanced ML techniques to the constraints of MCUs to convert them into real smart objects. As future work, they planned to consider other constrained devices to benchmark additional TinyML frameworks. Table 2.4 provides a comprehensive summary of the ML algorithms and their metrics on embedded platforms.

Table 2.4: Review of ML Algorithms on Embedded Platforms (Sanchez-Iborra *et al.*, 2020)

ALGORITHM	MCT	PARAMETER	PV	METRICS
Support Vector Machine	MicroMLGen	Samples/ Kernel	750	Accuracy = 0.7922 F1 = 0.723 Memory = 25298B Time = 34.1 ± 0.2ms
Multilayer Perceptron	emlearn	Hidden Layers/ Neurons	3	Accuracy = 0.8903 F1 = 0.8431 Memory = 6826B SRAM = 468B Time = 2.9 + 0.01ms
Decision Trees and Random Forest	sklearnporter	RF Estimators	10	Accuracy = 1 F1 = 1 Memory = 13160B Time = 193.6 ± 6.06μs

[1] MCT - Model Conversion Tool PV - Parameter Variation

Krejci *et al.* (2022) introduced a small robotics-embedded system that is ready to be deployed as an Internet-of-Things (IoT) device. They used an ESP32 (Xtensa 240 MHz) module, which includes Wi-Fi connectivity as the core processor, and a set of sensors. The authors also mention the STM32H7-based Nicla Vision board (480 MHz dual-core ARM Cortex-M7), which is more suitable for computationally demanding tasks. The system captures the robot sensor data (vibrations, environmental information) and sends it to the cloud, thus showing how modern 32-bit microcontroller platforms (like the ESP32, STM32) are used in sensing and connectivity while keeping power consumption low and wireless.

[1]

El-Sebah *et al.* (2023) proposed a simple optimum Intelligent PID (SO PID) controller for motor control simplification of autonomous vehicles. The article evaluated various steering control procedures for vehicle performance improvement. They integrated angle, distance calculation, and obstacle detection algorithms for vehicle motion control. An autonomous car prototype was introduced. The prototype was controlled through an Arduino microcontroller board. The transmission and reception of coordinates were performed via a GSPS module. The position of the prototype was specifically controlled by a DC motor armature control model. Armature Resistance, Armature Inductance, EMF Constant, Car and motor Inertia, Car and motor friction coefficient, and sampling time were system parameters adopted for this prototype. Finally, it was noted that the controller performance produces an ideal response during the transient and steady states.

Guzman *et al.* (2024) presented a low-cost solution towards vehicular dynamics control via an ESP32. They developed algorithms such as Active Front Steering (AFS) and Rear Torque Vectoring (RTV), which control the vehicle in real time. The main ESP32 board was used to run custom control algorithms, thus adjusting the torque and steer by real-time data. The second ESP32's function was in driver input mimicry. Analogue signals were used to act out a "steer-by-wire" system. This wiring of components allowed realistic vehicle behavioural simulation. MATLAB and CarSim were used for testing and evaluating the control system. The ESP32 microcontroller, featuring dual-core Xtensa CPUs at 240 MHz, integrated Wi-Fi/Bluetooth, and ample GPIOs, is well-suited for prototyping autonomous vehicles.

Rybczak *et al.* (2024) provide a thorough survey of ML-based control strategies of mobile robots and noting the hardware setup that was used in the prior research. For example, the Arduino Mega 2560 (ATmega2560) microcontroller board coordinates powerful real-time feedback control with sensors and actuators. In some applications, a Raspberry Pi 4B (ARM

Cortex-A72) is used to perform computationally demanding CNN inference, and the Arduino has direct control. Also, the authors cite an SVM-based controller on an ARM9-based microcontroller platform. All these results show that the current state-of-the-art academic prototypes generally combine a low-cost, direct-control microcontroller with a single-board computer, thus offering a low-cost implementation platform in embedded robotics.

Chen (2025) points out a setup whereby a Raspberry Pi 4B (ARM Cortex-A72, quad-core, 1.5 GHz) is used to execute the CNN and SLAM algorithms, but an Arduino Mega 2560 is the low-level controller. Such architecture is typical of delivery robots, where perception and path planning are done by a high-performance ARM CPU, and motor control and I/O are performed by a microcontroller. The mentioned configuration highlights that microcontroller platforms within the last-mile robotics often include

2.5.4 Cloud and edge AI integration

Wang *et al.* (2023) discussed the use of cloud services in combination with onboard planning. They used a DRL-based local planner trained in simulation on obstacle-avoidance scenarios and then applied it to the robot with online adaptation. The research mentions ROS and offboard training, which suggests an edge-cloud workflow. The map server could be updated globally, and this shows how cloud computation (DRL training and global map maintenance) can supplement edge execution.

Krejci *et al.* (2022) used a cloud-centric approach where the data gathered by sensors and health devices were sent to ThingWorx via onboard devices and then monitored and analysed remotely. The onboard ESP32 microcontroller was used as a data gateway that transfers information to the cloud. This work exemplified how cloud connectivity can support diagnostic operations and fleet management in an Internet-of-Robotic-Things environment.

Clotet & Palacin (2023) present SLAMICP, a library that runs SLAM algorithms on the CPU of the robot. Although this implementation is optimised towards resource-limited edge settings, it can be improved by speeding up ICP to allow more frequent communication with a server to update the map. A hybrid approach thus arises, which is a real-time loop-closure onboard and periodic cloud integration.

Katare *et al.* (2023) focused on energy-efficient mechanisms and the techniques for implementing Edge AI and vehicular frameworks towards achieving autonomous driving. They discussed the problems and challenges faced in deploying intelligent services within vehicular edge systems. The definitive goals of this survey were to review and compare connected vehicular applications, vehicular communications, and Edge AI techniques. They wished to provide valuable insights that would be beneficial to the collaborative driving service development on the low-power and memory-constrained systems field of research, while also prioritising energy optimisation for the autonomous vehicles. The survey considered the Edge AI approach and how the algorithms use the generated data from the edge devices to make independent real-time applications without the need for online processing for decision-making. It was stated that the Level 1 to Level 3 autonomous vehicles usually relied on GPU (Graphics Processing Unit) chips for their applications, which alone drain 350Wh of energy per 100km of driving. This is neither energy efficient nor power optimised. Through Edge AI, a fully connected Level 5 autonomous car can be achieved through collaborations between computing systems and edge-server communications. The communications in autonomous vehicles were mainly classified into two parts: Inter-Vehicle Communication and Intra-Vehicle Communication. Intra-vehicle communication is a combination of wired and wireless technologies. These technologies aid in understanding the current state of the vehicle through information and signal exchange between the sensors, actuators, and other electronic devices and components present within the vehicle. In Inter-

Vehicle communication. The vehicles are equipped with radio and communication modules. These modules receive information and signals related to infotainment. The current technology used to achieve this is the HD radio technology. However, for local and long-range communication, researchers have proposed technologies like WiMAX, c-v2x, and DSRC. The main question this survey answered was 'What are the current approaches and trends that can promote the concept of Level 5 self-driving by enabling the Artificial Intelligence at the Edge Devices with an energy-efficient approach?' The survey covered a lot of ground, such as data management and process techniques on the Edge devices, categorisation for autonomous driving use-cases for real-time use-cases, autonomous driving tasks hierarchical categorisation, and the energy implications of them.

Zhu *et al.* (2023) acknowledged that the rapid growth in cloud and edge computing provides more opportunities for autonomous vehicles to offload heavy data processing tasks like GPS navigation to the cloud. This allows transport vehicles greater efficiency. The bottlenecks of unpredictable traffic conditions and cyberattacks were discussed. To address such concerns, they explored solutions like Digital twins (virtual models of real vehicles and systems), better tools for testing safety under unpredictable situations, and methods of protection from attacks by hackers.

Chen (2025) introduces an architecture where path-planning is split between edge and cloud, and onboard computation (CPU-based CNN segmentation, SLAM, APF+DDPG) is used only to perform local tasks. However, the authors mention big sensor datasets, which means that the models are trained in the cloud. Further deployment is based on edge training of the model. The global planning could be updated from a map server, illustrating how cloud computation (for DRL training and global map updates) complements edge execution.

2.5.5 Last-mile delivery applications

Imad *et al.* (2022) tested their framework of sidewalk navigation on a commercial quadruped robot (Barrett Technology). Delivery-like situations were simulated by viewing pedestrians as dynamic obstacles and ensuring that the robot stays in areas where it can drive. Notably, the zero-shot transfer performance of this framework suggests that it is applicable in real-world settings. The applied, practice-oriented character of this test also indicates that the semantic segmentation and NMPC-based methods can become a standard part of the delivery robot systems in the future.

Engesser *et al.* (2023) pointed out that autonomous delivery solutions possess great potential in alleviating workload on e-commerce businesses by overcoming the problem of driver shortages and availability. They discussed multiple autonomous solutions, including Unmanned Aerial Vehicles (UAVs), two- or multi-tiered systems, and the concept of passenger and freight integration, and weighed their benefits and challenges. Conclusively, a research agenda was provided to help researchers, manufacturers, businesses, and governmental institutions prepare for the eventual implementation of autonomous delivery systems.

Levkovych *et al.* (2023) compared autonomous delivery vehicles to human-driven ones. They focused on operational performance and cost-effectiveness in urban environments. Using TCO (total cost of ownership), fuel, maintenance, and especially labour costs, they concluded that though self-driving vehicles are cheaper to operate (about $0.133/km), they deliver fewer parcels compared to traditional vans. They proposed a future innovation of integrated lockers with autonomous vehicles for improved logistics services.

Alverhead *et al.* (2024) performed a study on autonomous delivery robots for last-mile delivery, the value they add to the logistics and transport industry, and the challenges they

face. They explored the major themes involved in public autonomous systems: operations, infrastructure, regulations, and acceptance. Finally, they highlighted how most studies on ADRs have been theoretical, such as models. Thus, there is a great need for real-world case studies and implementations.

Chen (2025) examined the logistical processes in Cainiao Station (China), focusing on dynamic urban delivery. The author highlighted limitations like congested sidewalks and moving pedestrians and used their DynaFusion-Plan approach to produce efficient delivery routes in such complicated environments. This research demonstrates that the very demanding requirements of last-mile operations can be satisfied by an academic algorithm, in this case, by attaining a success rate of obstacle avoidance of nearly 99%. The results, therefore, show that advanced planning methods are possible in real-life delivery logistics.

Paksadze (2025) also explored the solutions brought forward by autonomous systems. They noted that most of the research on autonomous driving systems for logistics purposes has been carried out by private enterprises. Thus, there is a lack of financing and incentives by states towards this domain. This makes it important for companies and universities to provide innovations in autonomous systems in the last-mile delivery domain. Table 2.5 shows the meta-analysis of all the works reviewed above.

Table 2.5: Meta-Analysis Table

S/N	AUTHOR/ YEAR	PURPOSE OF WORK	STRENGTH OF PAPER/WORK	WEAKNESS OF PAPER/WORK
1	Engesser *et al.* (2023)	To identify and compare autonomous delivery solutions suitable for last-mile urban logistics.	The study provided a comprehensive literature review across multiple autonomous delivery technologies and a comparison table of delivery methods (safety, service,	The review relied heavily on the area of simulation and conceptual design. There was limited real-world deployment data.

			energy demand).	
2	Wu *et al.* (2023)	To design and evaluate an end-to-end architecture that integrates attention mechanisms and LSTM models for improved obstacle avoidance and path planning in autonomous driving, using multimodal data.	There was strong experimental validation across four datasets (Waymo, ApolloScape, KITTI, Cityscapes). The design achieved top scores in ODA (Object Detection Accuracy), OASR (Obstacle Avoidance Success Rate), and EE (Energy Efficiency).	This design required high computational resources (used PyTorch, RTX 2080 Ti GPU), which is not ideal for embedded/TinyML platforms. The scalability in unstructured or unpredictable real-world environments was also not addressed.
3	El-Sebah *et al.*, (2023)	To build an Optimum PID controller to simplify and optimise the motion control of an autonomous vehicle.	Successful implementation of a practical prototype for real-world implementation, and included robust testing across different path classes.	While ultrasonic sensors were used for obstacle detection, the paper lacks an in-depth discussion on their range, accuracy, or handling of complex obstacles, which is critical for last-mile delivery scenarios.
4	Liu *et al.* (2022)	To design and optimise a GPS vehicle navigation system using embedded technology for the enhancement of vehicle positioning accuracy and driver convenience in intelligent transportation systems.	The design achieved a high real-time efficiency with position updates within 1.0s and a signal matching rate above 98%, critical for dynamic navigation in delivery vehicles.	No obstacle detection was integrated. The system focused solely on GPS-based navigation. Obstacle detection is a critical requirement for autonomous last-mile delivery vehicles to avoid collisions.
5	Nahavandi *et al.* (2022)	To provide an extensive, updated review of legacy and modern	The review discussed real-world scenarios such as urban, off-road,	Deployment in the area of TinyML and embedded systems was not

		autonomous navigation methods, covering sensors, platforms, path planning, sensor fusion, SLAM, and deep learning approaches for obstacle avoidance.	aerial, and underwater environments. They also provided details of the strengths and limitations of localisation methods (e.g., EKF, PF, MSCKF) and highlighted challenges encountered during the deployment process, like GPS blockage, sensor drift, and computational constraints.	addressed. Some sections, like path following, remained simulation-bound. Real-time deployment for delivery robots was not discussed extensively.
6	Gómez-Huélamo *et al.* (2021)	To validate a ROS-based fully-autonomous driving system using the CARLA simulator by simulating real-world traffic scenarios and testing decision-making via Hierarchical Interpreted Binary Petri Nets (HIBPN).	The open-source architecture is portable and modular. The paper demonstrates effective use cases like Adaptive Cruise Control and Pedestrian Avoidance.	The paper provided simulated testing only. There was no physical road validation in this study. Future work is needed to integrate deep learning for object tracking and richer interactions.
7	Valera *et al.* (2021)	To develop and implement a trajectory planning algorithm for autonomous light vehicles, ensuring obstacle avoidance, energy efficiency, and real-time navigation under dynamic conditions.	Successfully integrated a collision-free trajectory planning algorithm that considered energy constraints.	The study assumed high availability of sensor infrastructure (FARO 3D, VBOX IMU). Real-world scalability (urban traffic, mixed environments) was not addressed.
8	Mushtaq *et al.* (2021)	To propose a two-phase strategy for the improvement of autonomous vehicle traffic flow through	The paper considered multiple realistic scenarios like leader failure, obstruction, and	Some assumptions (like perfect communication) reduce real-world applicability. The

		the combination of platooning for congestion reduction and V2V/V2I-based collision avoidance.	merging platoons. Real-world map (F-8 Markaz Islamabad) was used for scenario testing.	study was also fully simulation-based, as there was no real-world vehicle testing.
9	Sánchez-Iborra *et al.* (2020)	To explore the TinyML ecosystem, outlining its role in embedding machine learning into frugal objects, while reviewing frameworks and presenting a real-world use case involving a multi-RAT smart object on an Arduino UNO.	The paper compared TinyML frameworks like TensorFlow Lite, CMSIS-NN, emlearn and MicroMLGen, offering benchmark results for SVMs, MLPs, decision trees, and RF models on Arduino UNO.	The main focus in this study was on classification tasks (e.g., interface selection). Navigation and obstacle avoidance were not addressed. The use of communication-related, not physical mobility.
10	Casado *et al.* (2020)	To present an educational simulation framework based on Simulink, Stateflow, and Gazebo for the development of autonomous drone navigation systems, aimed at training students in aerial robotics through competitions.	The simplification of complex behaviours like hovering, tracking, and obstacle navigation was achieved via visual tools and MATLAB functions. Realistic simulation (Gazebo) was combined with graphical state machine design (State flow).	The paper focused only on simulated environments. There was no real-world drone deployment or physical hardware interfacing. The simple test setup limited obstacle types and dynamics.
11	Katare *et al.* (2023)	To review state-of-the-art approximate computing and Edge AI frameworks for autonomous vehicles, focusing on energy consumption reduction via model compression, quantisation, and resource-aware AI deployment.	The paper provided detailed compression techniques (pruning, quantisation, knowledge distillation, and low-rank approximation) and discussed TinyML, federated learning, and edge offloading strategies.	The review lacked hardware-specific benchmarking and deployment examples on actual microcontrollers or TinyML platforms. There was a heavy emphasis on cloud/edge ecosystems, not covering localised approaches.

12	He *et al.* (2023)	To evaluate and compare the performance of seven YOLO object detection models for outdoor obstacle detection to assist visually impaired individuals.	The study reviewed a diverse dataset with 15 obstacle classes (e.g., cars, bicycles, poles) from real-world sidewalks and streets, ensuring applicability to practical scenarios like urban delivery routes.	While YOLO models are fast, the study lacked evaluation on edge devices (e.g., microcontrollers). This limits the direct applicability to TinyML-based autonomous vehicles.
13	Zhu *et al.* (2023)	To explore the enhancements in Automated Vehicle performance using cloud and edge computing.	The paper proposed a Mobility Digital Twin (MDT) framework with real-world architecture and case studies.	There was a high dependency on infrastructure like RSUs and cloud APIs, which is not always feasible in low-resource settings.
14	Katona *et al.*, (2024)	To present a comprehensive literature review of classical, heuristic, and AI-based algorithms used for obstacle avoidance and path planning in mobile robot navigation.	An extensive and detailed comparison of over 30 algorithms, including Dijkstra, A*, APF, GA, ANN, MPC, DRL, and PSO, was provided, offering a **summary** table on algorithm properties (convergence, complexity).	The paper was heavy on the theory aspect, and thus could benefit from more comparative performance benchmarking. There was also no discussion on algorithms suited for embedded systems or resource-constrained devices.
15	Xie *et al.* (2024)	The paper suggests a LiDAR-based approach to the automated detection of negative obstacles (ditches) in orchards, which uses tilted LiDAR scans and processing of point clouds.	The system solves the commonly known problem of ground occlusion by intentionally tilting the LiDAR optics, thus removing blind spots. It recorded an obstacle detection accuracy of about	It focuses on recognising ditches and would need adjustment to adapt to other terrains.

			92.7 % at an 8m distance.	
16	Guzman *et al.,* (2024)	To develop and validate a real-time control system for vehicle dynamics using an ESP32 microcontroller, to enhance stability and manoeuvrability.	ESP32 was validated as a viable, energy-efficient alternative to costly automotive systems, aligning with cost constraints for last-mile delivery.	Vehicle dynamics were addressed without the integration of obstacle detection or avoidance algorithms.
17	Alverhed *et al.*, (2024)	To analyse how Autonomous Delivery Robots (ADRs) affect last-mile delivery, explore operational models, infrastructure needs, and legal challenges.	The analysis offered comparisons of truck-based and micro-hub-based delivery systems. Environmental and social considerations (e.g., CO_2, equity, safety) we included. The study highlighted real-world case examples like Starship, Kiwibot, and Scout.	The review heavily relied on theoretical models, real-world data, and limited. There were also a few real-life ADR deployments evaluated academically.
18	Paiano *et al.*, (2024)	To reduce the need for labour-intensive manual labelling by pretraining object detectors on generated images and fine-tuning on a small real dataset.	The paper achieved comparable detection performance (mAP) to models trained on thousands of real images using only hundreds.	The framework used computationally intensive models and required significant resources, making it unsuitable for resource-constrained devices like edge devices and microcontrollers.
19	Alaa *et al.*, (2024)	To review object detection algorithms like R-CNN, YOLO, SSD, FPN, and FOMO, and analyse their feasibility and	The review highlighted trade-offs across YOLO versions, SSD, and FOMO and provided concrete hardware	The paper was purely survey-based. There was no implementation or performance benchmarking from the authors. There

		limitations when deployed on low-resource embedded systems using TinyML.	comparisons (MCUs vs. Cloud vs. Mobile).	was also a lack of in-depth discussion of the training process or dataset handling on-device.
20	Hamidaoui *et al.*, (2025)	To provide a comprehensive survey of collision avoidance algorithms for autonomous vehicles.	Performance comparisons across the 2020 - 2024 literature were presented. Real-world considerations like adverse weather and multi-vehicle coordination were included.	The focus was mainly on software/method-level, with little focus on hardware constraints or scalability to mass-market autonomous vehicle.
21	VImeda Paksadze (2025)	To examine how AI-driven autonomous vehicles are being adopted in last-mile logistics and analyse their benefits in efficiency and sustainability.	Various technologies (drones, AGVs, autonomous ground vehicles) were highlighted. AI and Industry were linked with logistics outcomes (e.g., emissions, cost-efficiency).	Limited cargo for autonomous vehicles. Legal and regulatory barriers have not yet been resolved globally.
22	Imad *et al.* (2022)	It proposes a segmentation-based drivable area estimator to be used as part of a non-linear model predictive control (NMPC) local planner to perform dynamic navigation in sidewalk-like settings.	This work uses semantic segmentation to represent the path accurately and shows smooth and safe execution of trajectories along the paths, in order to overcome the simulator-to-real world gap.	The accuracy of the performance depends on the purity of the segmented labels, which makes the technique susceptible to visual noise and lighting changes.
23	Wang *et al.* (2023)	This work combines classical global planning with DRL-based local navigation between landmarks to reduce training time.	The global planning method enhances deep-learning navigation, leading to better performance in dynamic	No real-world deployments have been reported.

			environments.	
24	Chen (2025)	It introduces DynaFusion-Plan, a combination of computer vision, LiDAR-SLAM, APF, and DDPG for last-mile logistic robot obstacle avoidance and navigation.	It shows a 98.7% success in avoiding obstacles. The architecture is also a hybrid, as it is a combination of classical and learning-based elements, consequently, allowing better performance.	The system is computationally demanding and is associated with a multisensory system, which can restrict its generality.
25	Clotet & Palacin (2023)	It introduces ICP scan-match outliers directly during SLAM to speed up obstacle detection.	The suggested approach speeds up the detection of obstacles since the ICP residuals are analysed directly in the SLAM, which eliminates unnecessary processing and increases the speed of reaction.	The approach has not been tested with dynamic obstacles yet. It assumes the availability of an accurate environment map.

2.5.6 Summary

The meta-analysis in Table 2.5 outlined the significant aspects of research on the topic of autonomous navigation, obstacle detection, and last-mile delivery. More recent systems use simulators, including CARLA (Gómez-Huuelamo et al., 2021) and Gazebo (Casado et al., 2020), to test autonomous vehicle algorithms on a more extensive scale. Deep-learning path planning (Wu et al., 2023; Chen, 2025), control system (El-Sebah et al., 2023) and sensor fusion (Xie et al., 2024) are all notable improvements in the AV sector. The latest evidence on TinyML (Sánchez-Iborra et al, 2020; Katare et al, 2023; Alaa et al, 2024) shows an increased interest in edge computing as a scalable and resource-efficient application for delivery applications.

However, there are still a number of gaps. The available literature has mostly been limited to simulation, which is not a representation of reality. Models like YOLO, LSTMs and other advanced models are prohibitively expensive to use in embedded electronics of low-cost autonomous vehicles. The literature does not provide experiments that incorporate real-time navigation with effective detectors based on vision in a single system, as a real physical experiment instead of a simulated one. Besides, most of the TinyML surveys lack concrete implementation and hardware benchmarking.

In this paper, we will discuss these gaps, providing a low-cost, real-life prototype, which is based on an ESP32 microcontroller with the right navigation sensors. To address the obstacle-avoidance problem, the obstacle-avoidance methodology uses machine learning and consists of the data pipeline that includes dataset generation via FFmpeg, data cleaning and labelling via bounding box and model training and quantisation via Edge Impulse and deployment on ESP32-CAM AI Thinker via Arduino library. The method produces a realistic and scalable embedded autonomous delivery application system design.

CHAPTER 3

3.0 Methodology

3.1 Preamble

This chapter is an outline of the systematic approach that has been used in the design, development, and deployment of an autonomous vehicle that utilises an ESP32 microcontroller for last-mile delivery services. This methodology covers hardware selection, systems architecture, and software development, ensuring reproducibility of the project.

3.2 Research Design

An iterative design-build-test methodology was used in this project, which was composed of five stages:

i. requirements gathering and selection of components,
ii. hardware design and assembly,
iii. creation of navigation, obstacle avoidance and waypoint-tracking algorithms,
iv. system integration,
v. performance optimisation and testing.

This cyclical process allowed for the gradual optimisation of the system and allowed for the quick identification of flaws in its design.

For navigation and waypoint tracking, two key algorithms were implemented:

1. **Distance Calculation (Haversine Formula):** The Haversine formula, shown in equation 3.1, was applied to compute the distance along the great circle between the current GPS position of the vehicle and the next waypoint. This is essential in determining the distance between the vehicle and its next target.

The formula is as follows:

$$d = 2r \sin^{-1}\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \quad (3.1)$$

Where:

d= distance between the two points

r= radius of Earth

ϕ_1,ϕ_2= Latitudes of points 1 and 2 (in radians)

λ_1,λ_2= Longitudes of points 1 and 2 (in radians)

2. **Direction Calculation (Bearing Formula):** Equation 2 is used for the calculation of the direction of movement, the initial bearing between the current position of the vehicle (ϕ_1, λ_1) and the next waypoint (ϕ_2, λ_2). This informs the car in which direction to move.

Equation 3.2 is expressed as follows:

$$\theta = \tan^{-1}2\left(\sin(\lambda_2 - \lambda_1)\cos(\phi_2), \cos(\phi_1)\sin(\phi_2) - \sin(\phi_1)\cos(\phi_2)\cos(\lambda_2 - \lambda_1)\right) \quad (3.2)$$

Where:

θ= initial bearing (in radians)

ϕ_1,ϕ_2= Latitudes of points 1 and 2 (in radians)

λ_1,λ_2= Longitudes of points 1 and 2 (in radians)

The resultant bearing, θ, is then compared with the current heading of the vehicle (read from the MPU6050 IMU) to determine the required steering adjustments to be made by the motors. The gyroscope and GPS data are merged to ensure accurate heading.

3.3 Materials and Tools

3.3.1 Hardware components

3.3.1.1 ESP32

The ESP32, shown in Figure 3.1, is a low-power, highly integrated system-on-chip (SoC) microcontroller designed by Espressif Systems, with built-in dual-core Tensilica Xtensa LX6 processors, and it supports Wi-Fi (IEEE 802.11 b/g/n) and Bluetooth (v4.2 BR/EDR and BLE) communications standards. It has a full range of peripherals, such as serial interfaces (SPI, I^2C, UART), analogue-to-digital converters (ADC), digital-to-analogue converters (DAC), and capacitive touch sensors. It is designed to run on high-performance, energy-efficient, real-time, and the architecture can be used in a variety of applications Internet of Things (IoT) systems, embedded control/automation and wireless sensor systems.

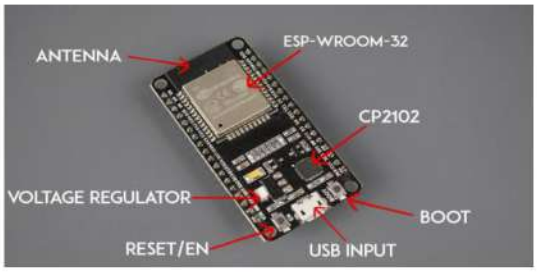

Figure 3.1: ESP32 Microcontroller

3.3.1.2 ESP32-Cam module

The ESP32 Camera module, shown in Figure 3.2, is a two-in-one integrated circuit that combines the ESP32-S SoC with a camera sensor, offering a cost-effective hardware solution for applications that require both wireless connectivity and visual data acquisition simultaneously. Its small size and the module camera make the device especially suitable for Internet-of-Things systems, remote monitoring systems, and other smart-device systems where on-board processing and image capture are essential.

Figure 3.2: ESP32-Cam Module

3.3.1.3 4WD RC car chassis

The 4WD intelligent robot car is a generally simple robotics platform, built on the chassis of a four-wheel, two-layer kit with motors mounted on an acrylic base. The use of M3 screws speeds assembly. The chassis provides optimal stability and facilitates easy expansion, while the absence of additional components enhances overall system integration. The unit can be used in education, that is, with people without prior experience, because it provides direct access to the concepts of Robotics. When extended, the platform can be used in applications like tracing trajectories, avoiding obstacles, distance and speed measurements, and wireless remote control.

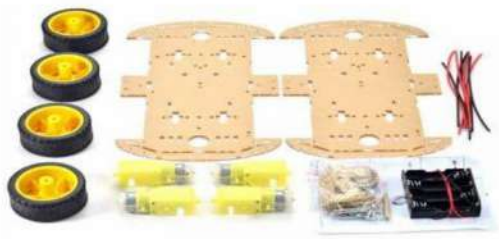

Figure 3.3: 4WD RC Car Chassis

3.3.1.4 L298N motor driver

L298N, shown in Figure 3.4, is a dual H-bridge motor driver integrated circuit produced by STMicroelectronics, which is supposed to drive two DC motors or a single stepper motor out of a single package. Implemented using bipolar Darlington output stages, the device provides

bidirectional control of inductive loads using TTL/logic-level inputs and separate Vs and Vss supply rails. The L298N is usually integrated into low-cost driver modules that incorporate a 5V regulator, flyback protection diodes, and heat-sinking to allow robust motor control.

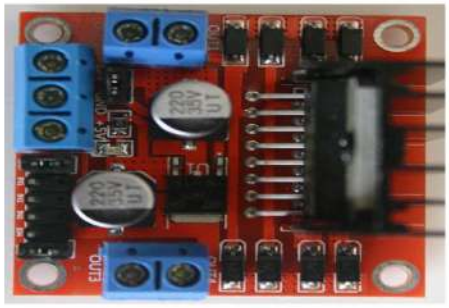

Figure 3.4: L298N Motor Driver

3.3.1.5 Ultrasonic sensor (HC-SR04)

The HC-SR04, shown in Figure 3.5, is a cheap ultrasonic distance sensor which uses a 40kHz acoustic transducer to emit a pulse and senses the echo through a phase-matched receiver. The module produces a duration-coded signal, whose pulse width (1s) is proportional to the round-trip travel time of sound and the range is calculated as $d = \frac{v.t}{2}$ with v representing the speed of sound (which is temperature-dependent). Typical performance has a range of 2cm to ~4m, a nominal resolution of the order of a few millimetres, and an update rate limited by echo timing and necessary averaging. The HC-SR04 is very commonly used in short-range obstacle detection on mobile robots and other low-cost robotics platforms, but its performance is affected by the reflectivity of the surfaces, the angle of incidence, temperature changes and background noise due to acoustics.

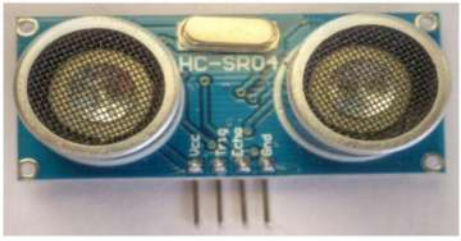

Figure 3.5: HC-SR04

3.3.1.6 SG90 servo motor

The SG90 micro servo, shown in Figure 3.6, is a common, low-cost actuator with a geared DC motor, a position-control feedback potentiometer and internal control electronics which implement a closed-loop angular positioning system. It has standard pulse-width modulated (PWM) control inputs and has a nominal operating range of around 0° to 180°. Due to its small size and relatively low torque, the SG90 servo is commonly used in light-duty actuation applications, in particular within sensor pan/tilt systems on small robots and children's educational systems.

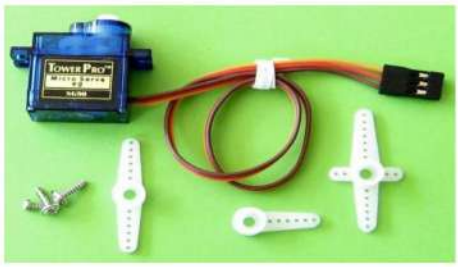

Figure 3.6: SG90

3.3.1.7 GPS module (NEO-7M)

The NEO-7M, shown in Figure 3.7, is a high-performance Global Navigation Satellite System (GNSS) receiver chip manufactured by u-blox. It enables GPS signal acquisition and tracking concurrently with optional QZSS signal reception, providing a tracking sensitivity of -162 dBm. The module combines the u-blox-7 positioning engine, a ceramic patch antenna (or an external antenna through the SMA/IPEX connector), a low-noise amplifier and a TCXO that will ensure stable frequency performance despite changes in temperature. The data is supplied in NMEA 0183 or as the u-blox UBX binary protocol over UART, SPI or I^2C interfaces, making them easy to integrate with microcontrollers. Having a cold-start of about 27s in nominal conditions and a maximum update rate of 5Hz, the NEO-7M will provide a

horizontal position accuracy of better than 2.5m in open sky conditions. Normal operating voltages are 3.3 to 5.0V, and the low power consumption of the module makes it applicable in embedded, mobile, and Internet of Things navigation applications.

Figure 3.7: NEO-7M

3.3.1.8 IMU sensor (MPU6050)

MPU-6050, shown in Figure 3.8, is a six-axis motion-tracking MEMS device manufactured by InvenSense with a built-in three-axis gyroscope and three-axis accelerometer on a single chip. Each of the channels is provided with a 16-bit analogue-to-digital converter (ADC), thus allowing very accurate measurements of angular velocity and linear acceleration. Communication with the microcontroller is through either the I^2C or the SPI interface, with the default I^2C address being 0x68 (or 0x69 when the AD0 pin is switched high). Acceleration measurements to +/-2g to +/-16g and gyroscope measurements to +/-250 degrees/s to +/-2000 degrees/s are available on programmable full-scale ranges. A Digital Motion Processor (DMP) is also included in the device and allows on-chip sensor fusion and alleviates computational load on the host system. Normal operating voltages are 3.3V to 5V, and the low power consumption of the module makes it applicable in robotics, navigation, and gesture recognition applications.

Figure 3.8: MPU6050

3.3.1.9 Li-ion battery

The 18650 lithium-ion (Li-ion) cell, shown in Figure 3.9, is a cylindrically shaped, rechargeable battery with a diameter of 18mm and a length of 65mm, used widely in portable electronics, electric vehicles and energy-storage systems, with a nominal voltage of about 3.6 to 3.7V in normal operation, fully charged voltages of about 4.2V and discharge cutoffs of between 2.5 and 3.0V. Lithium-ion cells have a capacity rating of between 1800 to more than 3500mAh, a range that indicates the variety of designs available. The 18650 Li-ion cell has higher energy density than other cells, longer cycle life and low self-discharge rates when charging is done according to the recommended protocol. This protocol includes constant-current/ constant-voltage (CC/CV) methods, and a battery management system (BMS) is also necessary to operate properly, being able to withstand over-voltage, over-current, and thermal problems to provide safety and a long service life.

Figure 3.9: 3S2P Battery Setup

3.3.1.10 3-series battery management system

A 3S lithium battery management system (BMS), shown in Figure 3.10, operates under the auspices of individual cell voltages in a three-cell series configuration (11.1V nominal, 12.6V full charge) to maintain charge balance, and thus pre-empts overcharging, over-discharging, and excessive current draw. The short-circuit shielding prevents catastrophic cell failure and shields the battery and downstream devices in a concurrent short-circuit. The BMS also offers superior protection of battery integrity, cycle life, and performance by coordinated control of charge and discharge across industries, including portable electronics, robotics, and electric mobility systems.

Figure 3.10: 3S BMS

3.3.1.11 3-series battery level indicator

A 3S lithium-ion battery level indicator, shown in Figure 3.11, is an electronic module specially designed to measure the state of charge (SoC) of lithium-ion battery packs in a 3S configuration (nominal voltage of 11.1V, fully charged at about 12.6V). The device meets this goal by continuously measuring the terminal voltage of the pack and comparing it to a preset voltage-capacity curve, thus allowing an estimate of the remaining charge. The resulting charge information is most often passed to a display (usually a segmented LED or bar display) that can be used to quickly and non-invasively determine the charge status. This ability to monitor the status of the battery forms the foundation of successful energy management in networked portable electronics, robotics, and electric mobility systems, reduces the occurrence of over-discharge, and, ultimately, increases battery life.

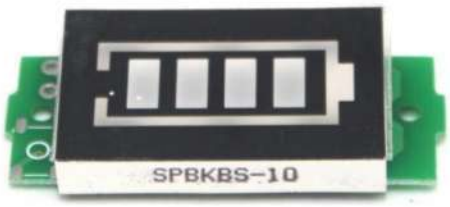

Figure 3.11: 3S Battery Level Indicator

3.3.1.12 DC switch

A Direct Current (DC) switch, shown in Figure 3.12, is an electromechanical component that is used to regulate the flow of direct current in an electrical circuit, either by opening or closing the path between conducting elements. It functions to either open or close conductive contacts by a mechanical switching action and cuts off or makes current flow available. Contrary to Alternating Current (AC) switches, DC switches are rated to carry steady-state unidirectional current, which specification necessitates very close contact spacing, arc suppression, and thermal management because there is no current zero-crossing. They are commonly used in battery-powered applications, renewable energy systems and low-voltage electronic equipment to enable user-controlled isolation of circuits, safe maintenance procedures and to provide device protection against uncontrolled power flowing back to the source.

Figure 3.12: DC Rocket Switch

3.3.1.13 Resistors

A resistor, shown in Figure 3.13, is an electrical component that is used to resist the flow of electric current by providing a definite resistance, which is measured in ohms (Ω). This resistance, which obeys Ohm's law (V = IR), makes the voltage drop across the resistor linearly proportional to the current which flows through it. Resistors are used to control the magnitude of current, divide voltages, bias active devices, and protect sensitive circuits against large currents by dissipating the electrical energy as heat. Commercial resistors have a variety of performance and packaging options, with fixed, variable, and specialised resistance values made using carbon film, metal film, and wire-wound resistor elements, each optimised to specific electrical and thermal properties. Resistors continue to play a fundamental part in almost all electronic systems, including simple circuits and highly complex computing hardware.

Figure 3.13: Resistor

3.3.1.14 4GB TF card

The 4GB TF (TransFlash) card, shown in Figure 3.14, often referred to as a microSD card, is a small, non-volatile, flash-memory device that is meant to be embedded in portable electronic products. It uses NAND flash technology to store digital information and stores data even when there is no continuous power. This storage medium can hold up to 4 gigabytes of data and is therefore appropriate for multimedia files, application datasets and firmware updates. The 4GB TF card has become very common in embedded systems, mobile devices, cameras and Internet-of-things (IoT) applications due to its small form factor, low power consumption, and compatibility with SD card standards by use of suitable adapters.

Figure 3.14: 4GB TF Card

3.3.1.15 Breadboard

The MB-100 solderless breadboard, shown in Figure 3.15, is a special-purpose prototyping platform that is intended to make the construction and testing of electronic circuits simple without the development of a permanent interconnection. The device is defined by 830 tie-points in interconnected rows and columns and by dedicated power rails to distribute voltages. It is made of a plastic substrate with spring-loaded metal clips underneath that firmly hold component leads and jumper wires so that circuit modification can be done quickly. Being reusable and being fully compatible with through-hole components, the MB-100 can be viewed as an essential tool in educational, experimental, and developmental applications in electronics engineering.

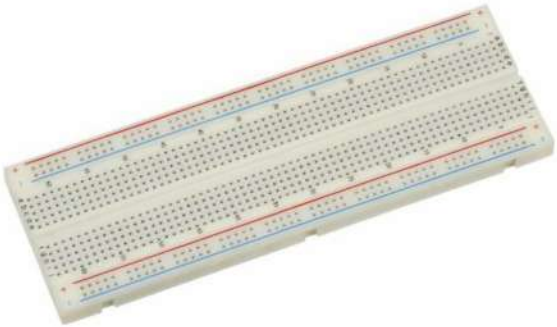

Figure 3.15: MB-100

3.3.1.16 Jumper wires

Jumper wires, shown in Figure 3.16, are short insulated electric conductors used to make temporary connections in a prototyping environment such as breadboards, development

boards and test circuits. Such conductors are generally made of stranded or solid-core copper and terminated with either male or female pin connectors, so that they may be inserted and removed quickly without the need to resort to soldering. They come in a variety of lengths and colour coding schemes and are used to make circuit assembly easier, and are essential to rapid prototyping, educational electronics, and hardware debugging due to the flexibility, reusability and compatibility with standard pin headers and terminals that jumper wires have.

Figure 3.16: Jumper wires

3.3.1.17 Female charging adaptor

A female charging adaptor, shown in Figure 3.17, is an electricity connector that has a recessed connector to hold a matching male connector, to provide a secure and reliable path to conduct power between a charging device and an electronic device or battery system. It is usually deployed in applications of DC power as lithium-ion battery packs, robots, and portable electronics, because it aligns the correct polarity and prevents short-circuiting during charging.

Figure 3.17: Female Charging Adaptor

3.3.1.18 12V charger

A 12V charger, shown in Figure 3.18, is an electrical unit that is used to supply a regulated DC supply having a 12V nominal extra-low voltage. It also includes elements to regulate voltage, control current, overcharging, short circuit, and reverse polarity protection so that the transfer of energy is safe and economical. Typical uses are charging lead-acid, lithium-ion, and lithium polymer battery packs in automobiles, robotics, and handheld electronic appliances or robotics.

Figure 3.18: 12V Charger

3.3.2 Software tools

3.3.2.1 Arduino IDE

The Arduino Integrated Development Environment (IDE), shown in Figure 3.19, is a free-software development environment used to author, compile, and upload code to microcontroller-based devices, the Arduino platform and derivatives, including the ESP32. The IDE provides a simplified application programming interface (API) to program in the C and C++ languages, and has a text editor, a compiler and a serial monitor to perform diagnostic and debugging functions. Pedagogical affordances of the platform, along with a well-developed library ecosystem, have led to its popularity in education, prototyping and embedded systems.

Figure 3.19: Arduino IDE Interface

3.3.2.2 Arduino libraries

Arduino libraries, shown in Figure 3.20, are pre-written collections of code which are meant to extend the functionality of the Arduino development ecosystem. These libraries also help remove the complexity of writing low-level routines by offering developers well-packaged interfaces to integrate peripherals, sensors and software utilities easily without having to write low-level routines themselves. Typically distributed through the Library Manager of the Arduino IDE, Arduino libraries are made up of header files, source code, and extensive documentation, which provide a list of pre-determined functions and classes to accomplish tasks like motor control, sensor data collection, or control of a particular communication protocol. Their use therefore promotes code reuse, improves modularity and accelerates prototyping in the embedded-systems development environments.

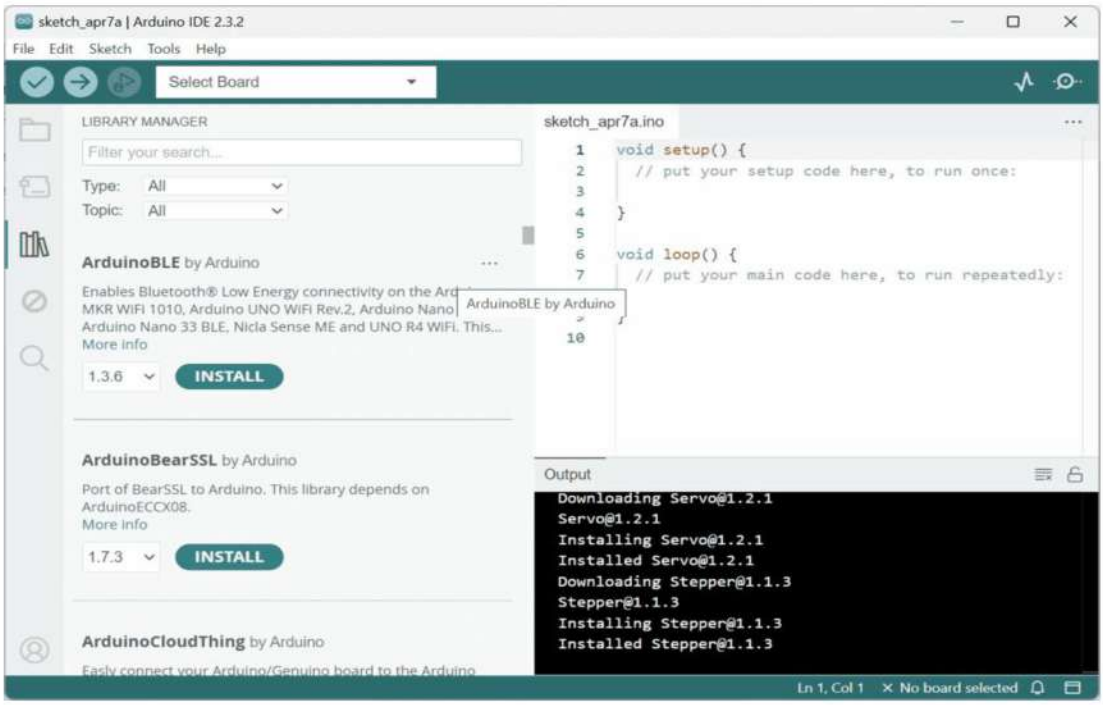

Figure 3.20: Arduino IDE Interface Showing Libraries

3.3.2.3 Web interface

A backend server was created for the sole purpose of route estimation and calculation. It served as a medium for receiving the destination coordinates from the user via JavaScript's geolocation API and the start coordinates from the autonomous car via the GPS module integrated with the ESP32. The two primary parameters were passed through the openrouteservice API for waypoints and coordinates generation.

Express, the framework used in implementing this backend server, is a lightweight and flexible framework built on top of Node.js. As many HTTP utility methods and middleware are available, it is quite easy to build a dependable and robust API in a reasonable timeframe. A general overview of how the Express architecture flows is shown in Figure 3.21

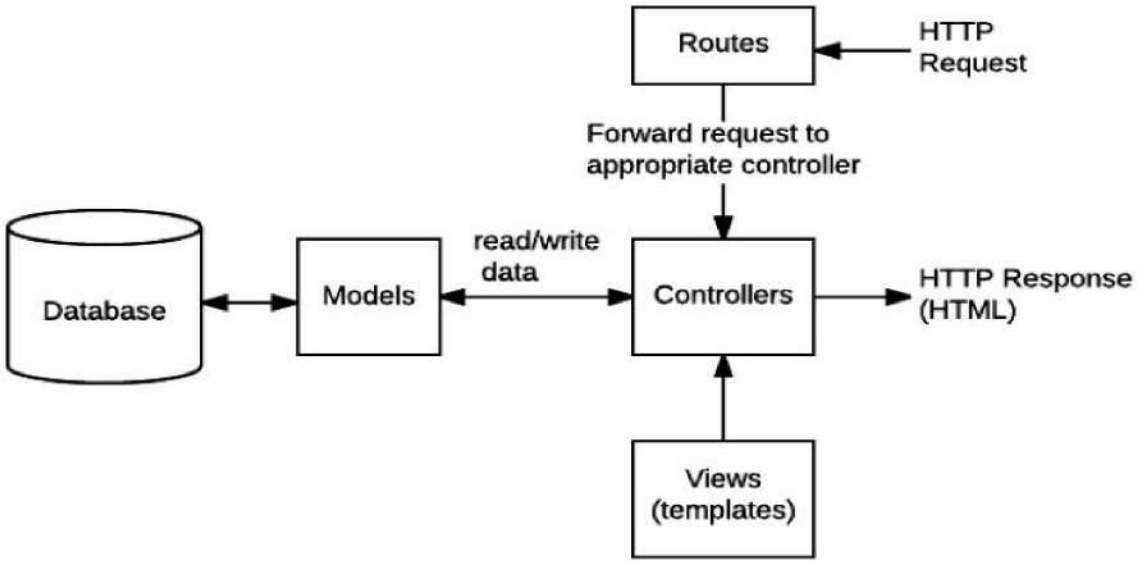

Figure 3.21: The Express Architecture from (Bansod, 2023)

JavaScript's geolocation API was used for collecting user coordinates from the front-end and delivering them to the back-end for further processing.

This API provided a straightforward way to retrieve the user's location without requiring any third-party libraries. To use this API, the user has to permit the browser to read its current location. Various error-handling mechanisms were put in place to handle edge cases where the user rejects permissions, the user's browser version fails to support the API, or even a simple timeout.

3.3.2.4 Cloud services

The 'openroutesservice' API is a free open-source API that consumes user-generated and collaboratively collected geographic data directly from OpenStreetMap. The API offers a variety of services, including directions, time-distance matrices, POIs, isochrones, Pelias geocoding, optimisation and elevation. For this project's use case, the direction service, specifically 'v2/directions/{profile}' request was selected. This request collected only two parameters: the start and end destinations in the form of {latitude, longitude}.

The start coordinates were passed as the current location of our autonomous car, while the end coordinates were passed as the user's location, which was received from the front-end. A GeoJSON response was returned, which consisted of multiple objects and keys describing the path between start and end coordinates. The coordinates array was extracted from the geometry object of the response. The array contained sub-arrays of latitude and longitude pairs for each point along the route to the destination coordinate. This coordinate array was then sent back to the ESP32 for further processing.

Figure 3.22 shows a streamlined process of the server-side communication implementation.

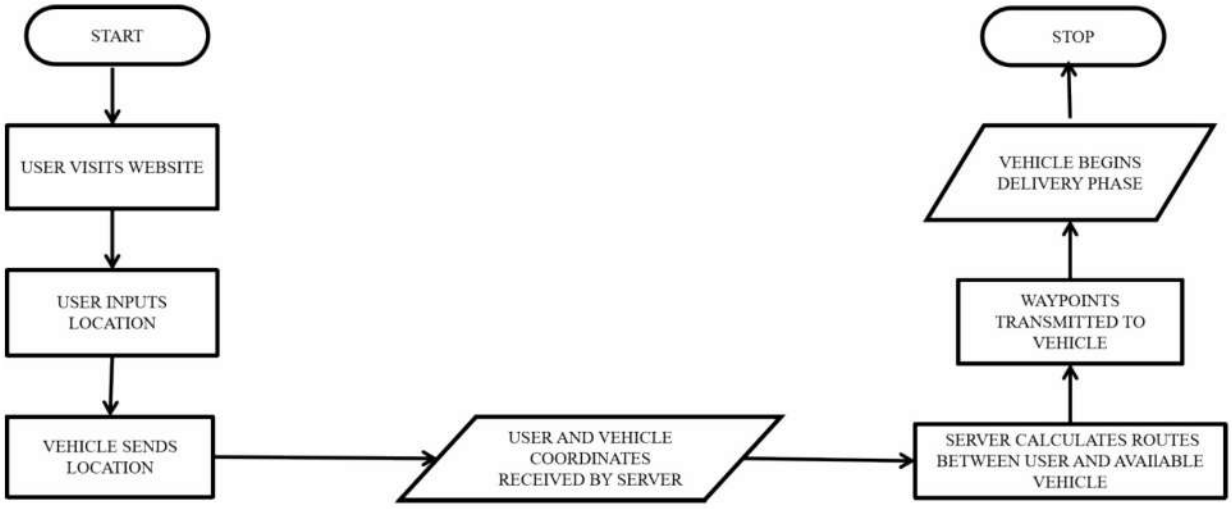

Figure 3.22: Server-Side Communication Flow

3.4 Obstacle Avoidance Methodologies

3.4.1 Obstacle detection via ultrasonic proximity sensing

The primary obstacle sensing system relies on the HC-SR04 ultrasonic sensor for real-time obstacle detection

1. **Sensing:** The sensor continuously measures objects in front of the system and calls a "NavigateAroundObstacle()" function when an obstacle is detected within 30 cm along the vehicle's path.

2. **Scanning and Evasion:** The sensor is mounted on an SG90 servomotor, which pans the sensor left and right. When an obstacle is detected, the system executes a scan to find the clearest path and then turns in the direction before resuming navigation. This provides a foundational level of collision avoidance, ensuring the system's safety during navigation.

3.4.2 Obstacle detection via the FOMO model

3.4.2.1 Dataset collection via FFMEG

Traditionally, data samples are collected manually via pictures through the ESP EYE CAM or a different lens. However, this approach introduced high inefficiency and time consumption, especially when collecting many samples for training. This challenge was addressed in this project by introducing an automated approach. Videos were taken, simulating a moving car avoiding or crashing into obstacles. These videos were then extracted programmatically into images.

The framework responsible for this automation is the FFmpeg. FFmpeg is an open-source multimedia framework, offering decoding, encoding, transcoding and other tools to manipulate media content. Its architecture is made up of four modular libraries.

1. libavcodec: A library used to encode and decode video and audio formats.
2. libavformat: I/O and container format support is provided via this library (e.g., MP4, AVI, MOV).
3. libavfilter: The filtering and processing of video streams are performed by this library.
4. libswscale: This library is for fast scaling of an image and conversion of colour spaces.

Hundreds of labelled images were extracted in a fraction of the time it would have taken to collect these image samples by using FFmpeg. The videos were taken from indoor and outdoor environments and converted to individual frames at a rate of 2 fps (a balance between redundancy and dataset volume). A shell script which executed FFMEG commands was written for the automation of frame extraction (see Appendix B). Figure 3.23 shows how FFMEG works.

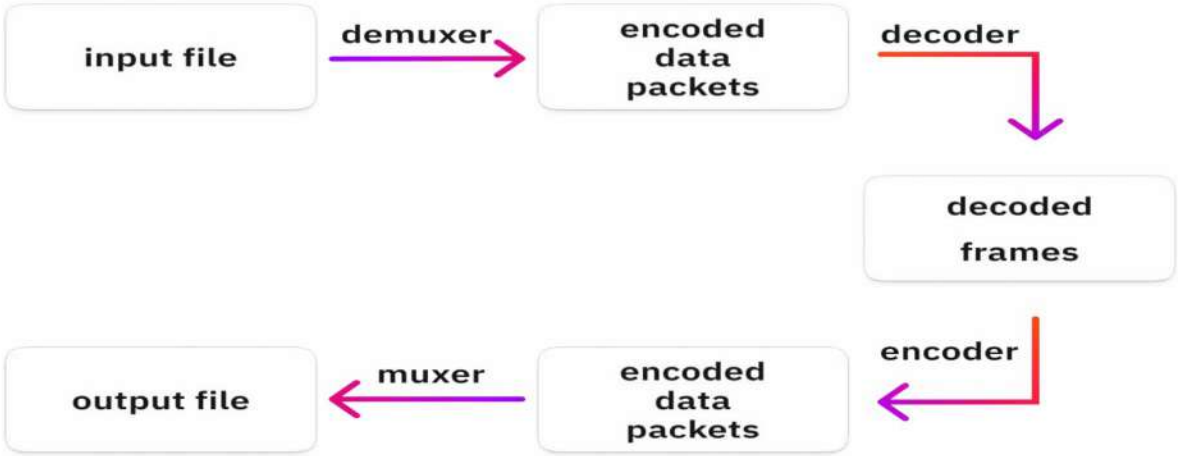

Figure 3.23: How FFMPEG Works from (Kopias, 2023)

There was a significant acceleration of dataset creation through this method. This enabled an efficient source of samples covering varied obstacle scenarios. The automation also ensured consistency in the extraction of frames, hence a more diverse and qualified dataset for the training of the FOMO model. The advantages of this automated extraction process encompass: reduction of manual effort, greater dataset consistency and rapid iteration in model building and training. The dataset collection phase became highly scalable and reproducible by combining shell scripting and FFmpeg's processing. This is important for sufficient TinyML development in autonomous vehicle applications.

The video data was recorded under indoor and outdoor environments. These environments were controlled, with fixed types of obstacles placed before the moving 'car'. The background varied from clear to cluttered, capturing the diversity of a real-world scenario for driving cars. The lighting conditions, camera positions and angles were continuously altered in the process of data collection. The camera was adjusted every few seconds in different orientations, heights and angles. Motion artefacts were intentionally introduced. The video sped up randomly, shaking and turning at intervals. This allowed some of the extracted images to be blurry and imperfect. This mimicked a real moving car and how its point of view remains unpredictable on the busy road under different weather conditions. There is a necessity in recording not only the ideal conditions, as there will always be unpredictable challenges in real-world navigation.

3.4.2.2 Data labelling and cleaning

The Edge Impulse platform was used for data labelling, cleaning, training and deployment. This platform is specifically designed for training large samples of data and exporting the resulting model into a lightweight version that could easily be deployed to microcontrollers and mobile devices.

This project's dataset comprised 1,288 image samples representing two classes: Bin and Chair. The bounding box labelling was the approach used for the annotation of these samples. These helped the model learn the location of objects in a frame through its centroid. This bounding box, approach shown in Figure 3.24, is essentially what differentiates this model from being an object detection model rather than an image classification model.

Figure 3.24: Bounding Boxes of the Two Object Classes

In the area of data cleaning, irrelevant images, such as images in which the objects were out of the focus frame for training, were removed. This ensured a higher F1-score as background images would not be wrongly generalised as 'obstacles. The data was split in the ratio of 8:2 for training and testing. This allowed a balanced distribution of object classes.

3.4.2.3 Data preprocessing

This step is necessary for image enhancement and standardisation. Image resizing, normalisation and augmentation were performed on the labelled samples. For Image resizing, samples were reshaped to 96x96 pixels by the Lanczos3 algorithm with the fit-short resize mode and middle-centre crop anchor. This algorithm ensured the preservation of object proportions whilst size reduction. Data augmentation was applied via random flips, rotations and brightness variations. This was done to further increase the model's robustness and accuracy when predicting in different orientations and lighting situations. In the area of normalisation, pixels were slack to allow faster convergence in the model training process.

The processing blocks in edge impulse, as shown in Figure 3.25, extracted a total of 27,648 features per sample. A balance between feature richness and computational cost was also observed.

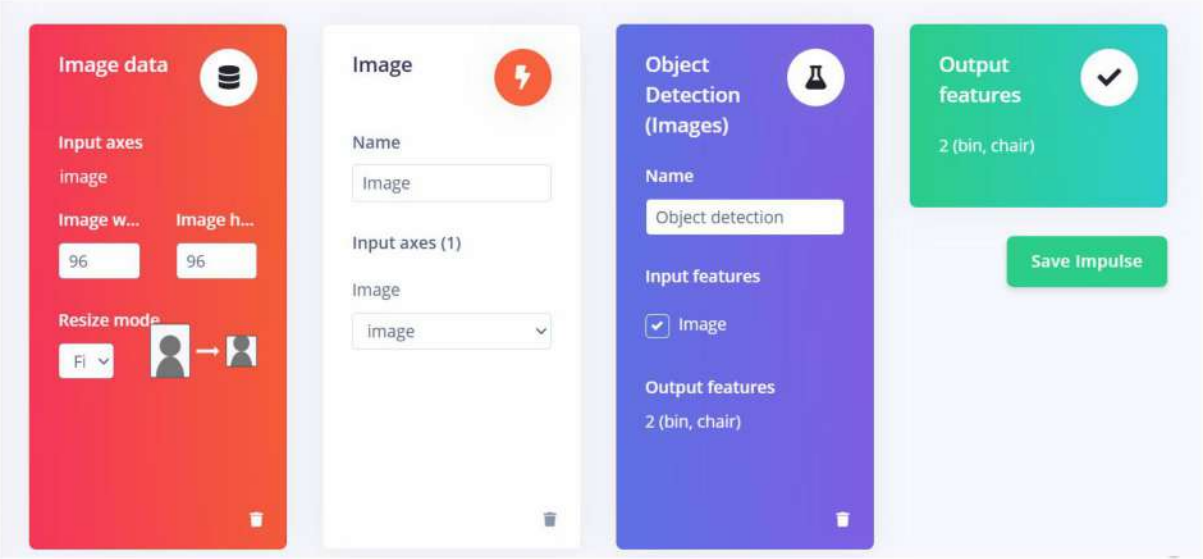

Figure 3.25: Processing Blocks in Edge Impulse

3.4.2.4 Model training

Table 3.1: Model Training Parameters

S/N	PARAMETER	VALUES
1	Epochs	60
2	Learning Rate	0.001
3	Optimiser	Adam
5	Validation Split	20%
6	Detection threshold	0.5
7	Performance Evaluation	Precision, Recall, F1-Score, Accuracy

The behaviour of the model was studied based on the selected performance metrics during the 60-epoch period of the training process. Precision refers to the proportion of the positive predictions of the model that are actually positive.

$$Precision = \frac{TP}{TP + FP} (3.3)$$

Recall, also known sometimes as sensitivity, is the rate of actual positive cases that are correctly predicted.

$$Recall = \frac{TP}{TP + FN} (3.4)$$

The harmonic mean of recall and precision is the F1-score and gives a good balance in the performance when there are unevenly balanced classes.

$$F1 - Score = 2\ x\ \frac{Precision \times Recall}{Precision + Recall} (3.5)$$

Lastly, accuracy measures the level at which the model makes correct predictions as compared to all the predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} (3.6)$$

The final structure of the FOMO model, as seen in Figure 3.26 features a convolutional layer with a ReLU activation, a dropout regularisation technique with a rate of 0.1 and a final convolutional logits layer outputting a 1x1x3, where the raw logits are passed by a softmax activation to output three classes (background, bin and chair)

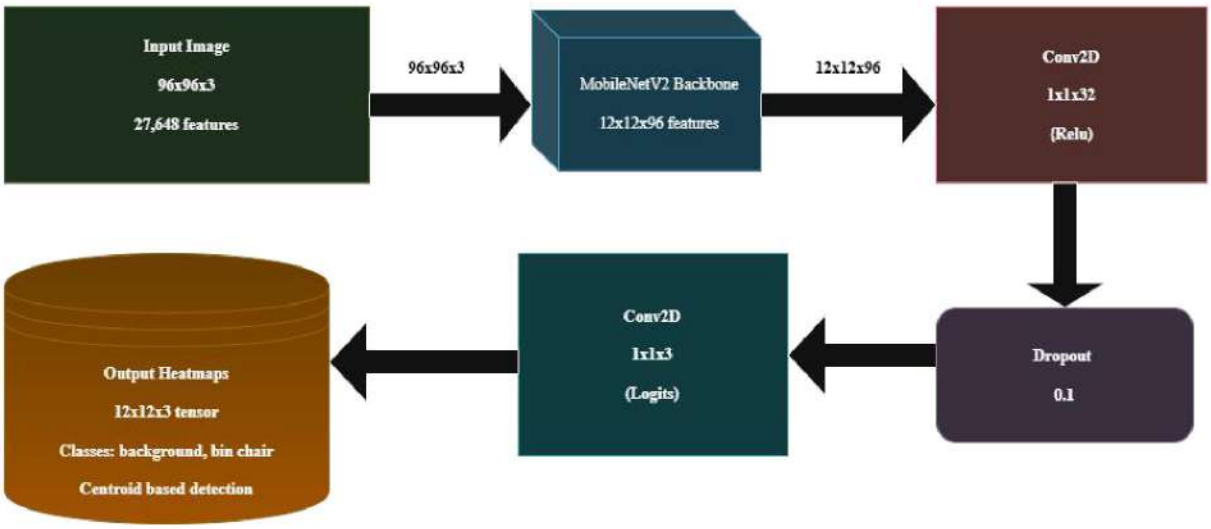

Figure 3.26: Final FOMO Model Architecture Used

It is important to note that both Float32 and Int8 quantised models were trained. This gives room for the engineer to evaluate the trade-offs between each version of the model. The trade-offs are generally resource efficiency and inference speed. All performance history during training and validation was saved to a classification report file on Edge Impulse. Custom Python scripts (see Appendix C) were used to extract relevant metric information from the JSON report.

3.4.2.5 Model deployment

The Arduino Library is the export form chosen for deployment of the trained FOMO model from the Edge Impulse platform. This Arduino library (see in Appendix D) is uploaded to an ESP32-CAM AI Thinker board to ensure real-time obstacle detection. The Float32 version of the model showed a faster inference time but demanded more space (latency 7ms, RAM 399.7KB). The Int8 quantised version of the model provided a compressed memory size, but there was a significant increase in inference time (latency 652ms, RAM 123.9KB). As this project is centred on Autonomous Driving, there is a higher priority on safety and speed of obstacle detection. Thus, the Float32 model was selected for deployment. The entire project from data collection to model testing and deployment is available to the public on the Edge Impulse Platform for cloning and exploration at this link: https://studio.edgeimpulse.com/public/773302/live.

3.5 System Design

3.5.1 Block diagram

The hardware architecture of the system is illustrated below. The ESP32 is the main controlling unit connecting the power system and other sensors (GPS, IMU, Ultrasonic) and actuators (Motor Driver, Servo). The Wi-Fi module is used to provide the connection to the

cloud backend and the local web interface. The ESP32-CAM is a co-processor, which does all the vision-based processing. The diagram is shown in Figure 3.27.

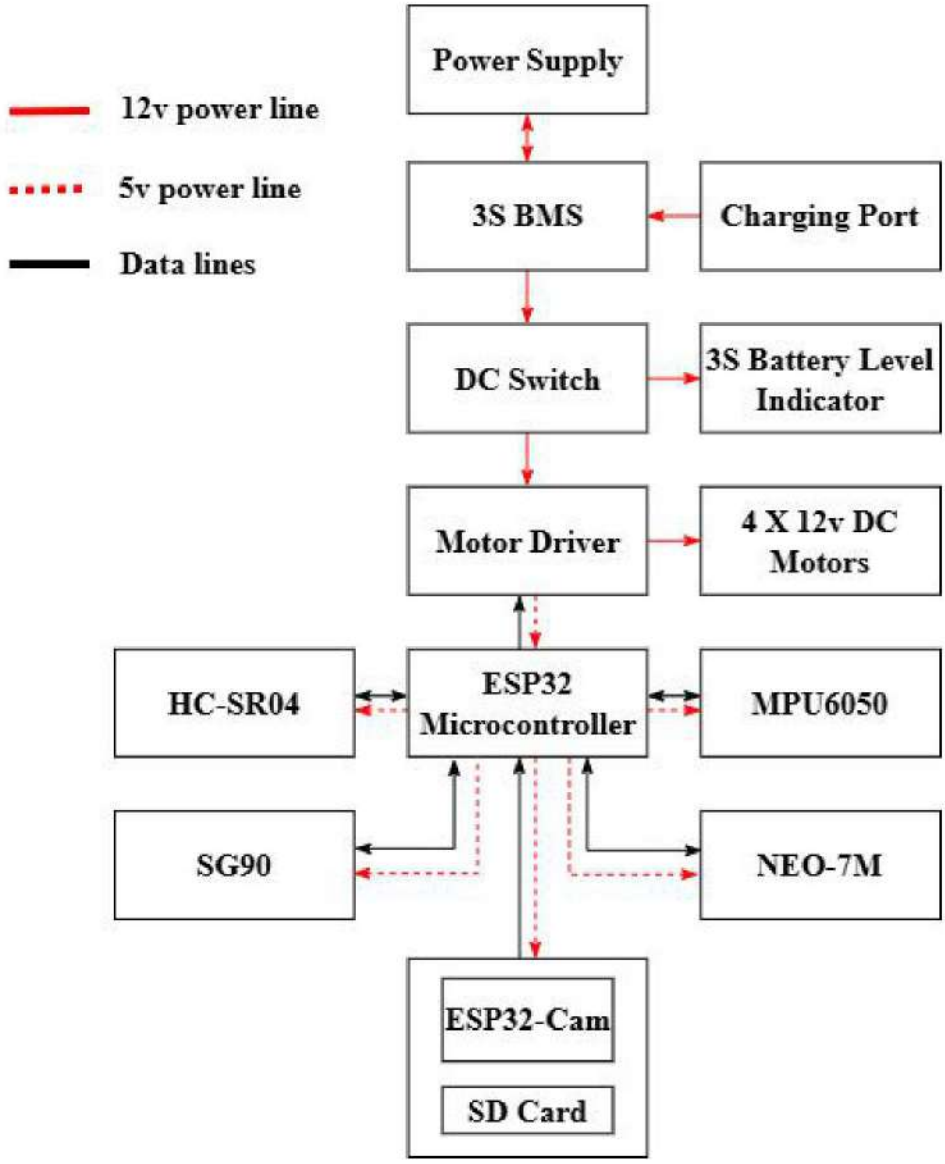

Figure 3.27: System Block Diagram

3.5.2 Circuit diagram

The circuit diagram shown in Figure 3.28 illustrates the electrical connections between all components, detailing the I2C bus, UART connections, PWM outputs, and power distribution from the 3S BMS.

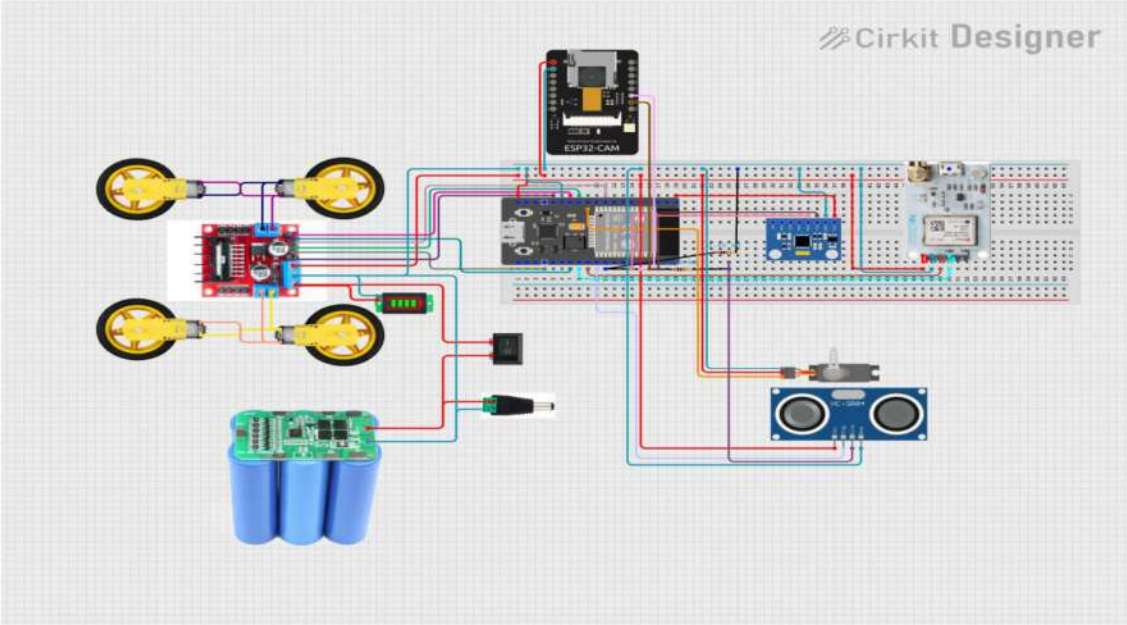

Figure 3.28: System Circuit Diagram

3.5.3 Flowchart of operation

The logic behind the working of the system is shown in Figure 3.29. It starts with the initialisation of all sensors and actuators and the first GPS coordinates and bearing. The system will be triggered to enter the main loop of control when destination coordinates are received. In this loop, it keeps on computing the distance and heading to the target and orders the vehicle to go towards the destination. The system constantly monitors to see if there is an obstacle; in the case of an obstacle being detected, a sub-routine called Navigate around obstacle is called. When the route is clear, the car keeps moving. This process continues until the distance to the target is less than a meter, at which point the vehicle halts.

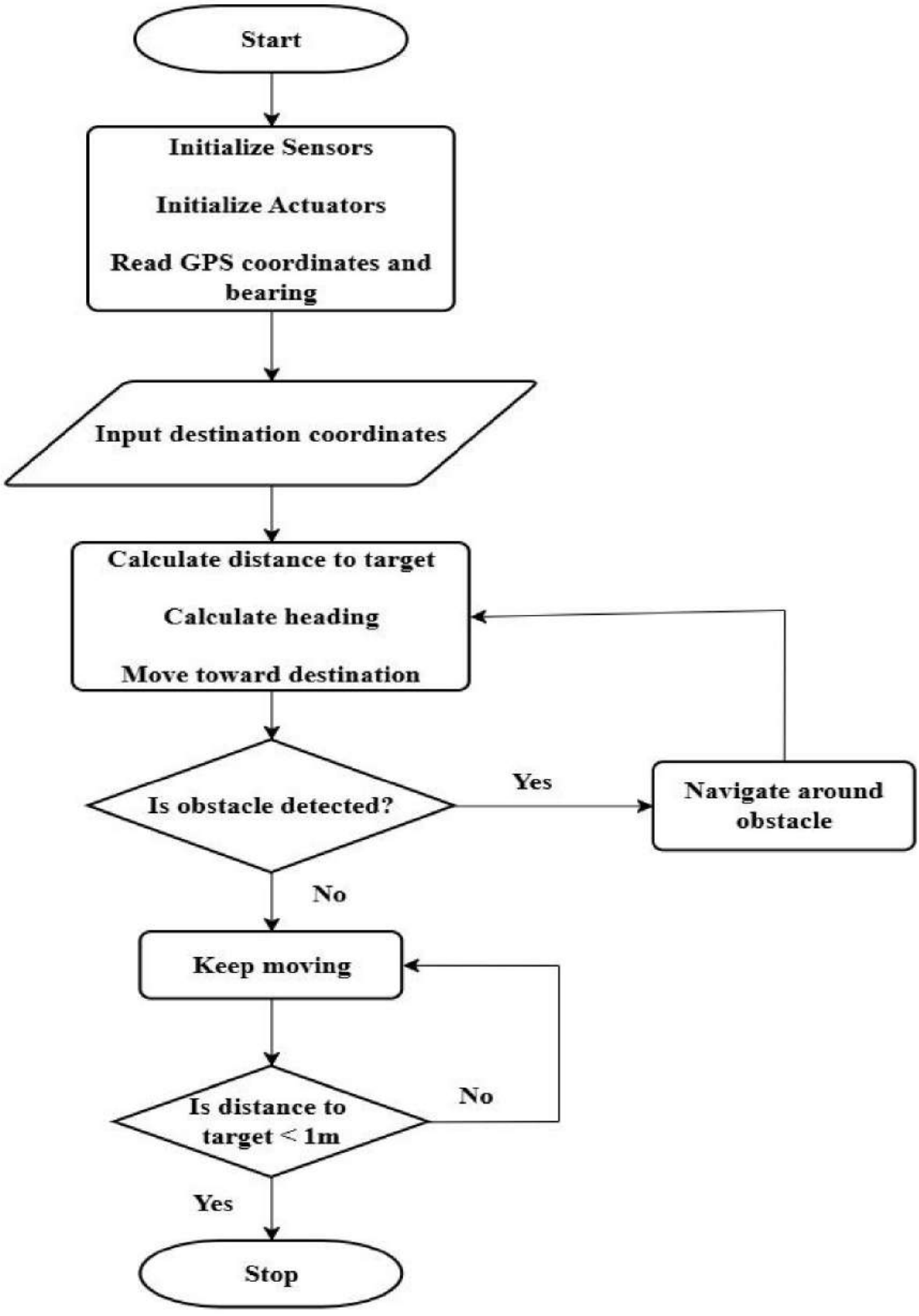

Figure 3.29: System Flow Chart

3.5.4 System Diagram

The hardware prototype assembled is shown in Figure 3.30. This picture shows the physical assembly of the main parts on the 4WD chassis, which are the ESP32 microcontroller and the related modules on the breadboard, the GPS module, and the HC-SR04 ultrasonic sensor on the SG90 servo in the front of the car.

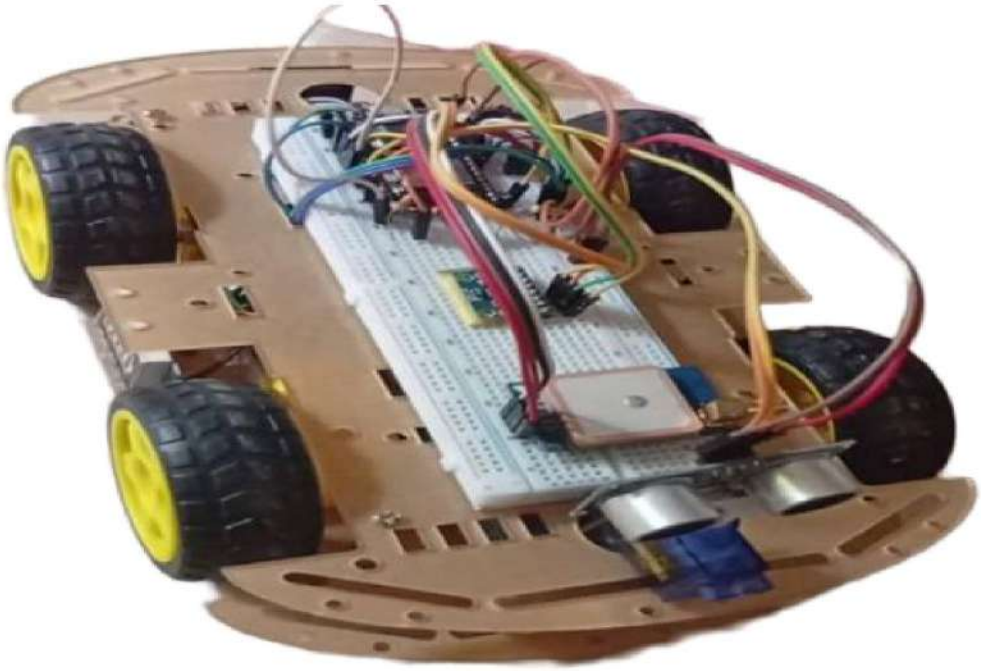

Figure 3.30: Fully Integrated System Hardware

CHAPTER 4

4.0 Results and Discussion

4.1 Object Detection Model Performance Visualizations

Figure 4.1 shows the model training history for the final trial, Trial 6. It can be seen that there is an obvious fluctuation between the 7th and 41st epoch. After that, the peaks of the accuracy line graph start to smoothen. The model's training loss however shows a stagnant level after its initial decline between Epoch 1 and 3.

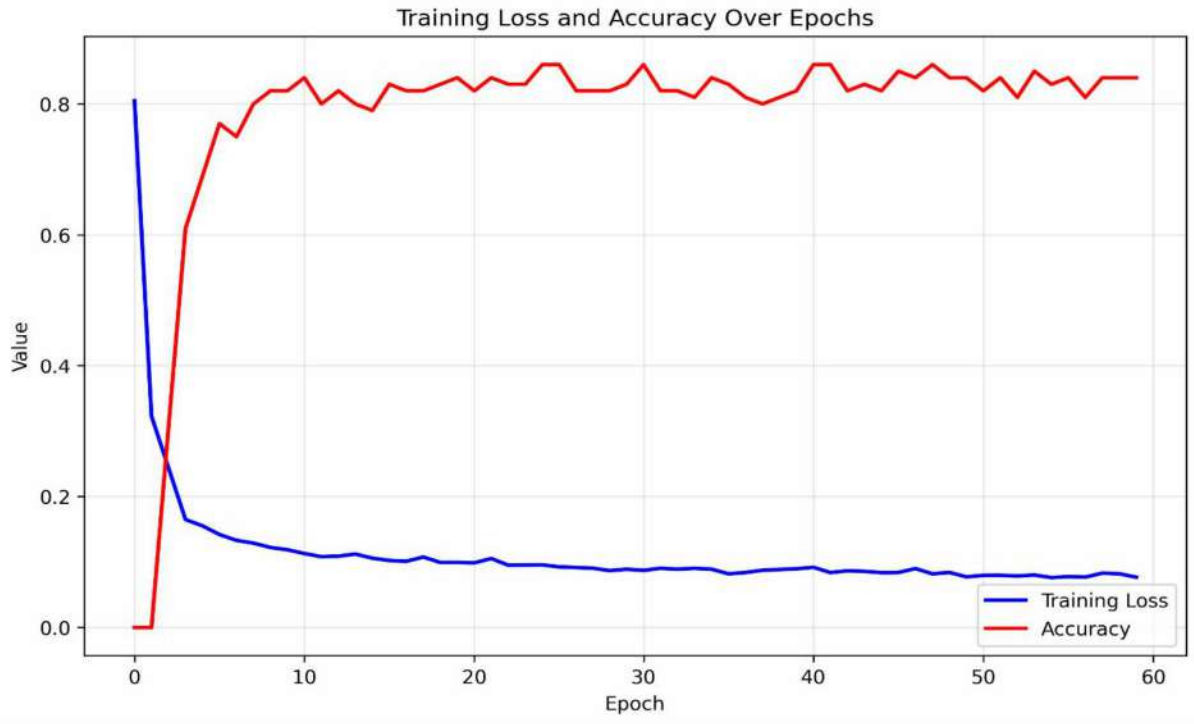

Figure 4.1: Training Loss and Accuracy Over Epochs

Figure 4.2 shows the model validation history for the final trial, Trial 6. Its behaviour is quite similar to the training history. However, the validation loss begins at a lower percentage (40%) than the training loss (80%).

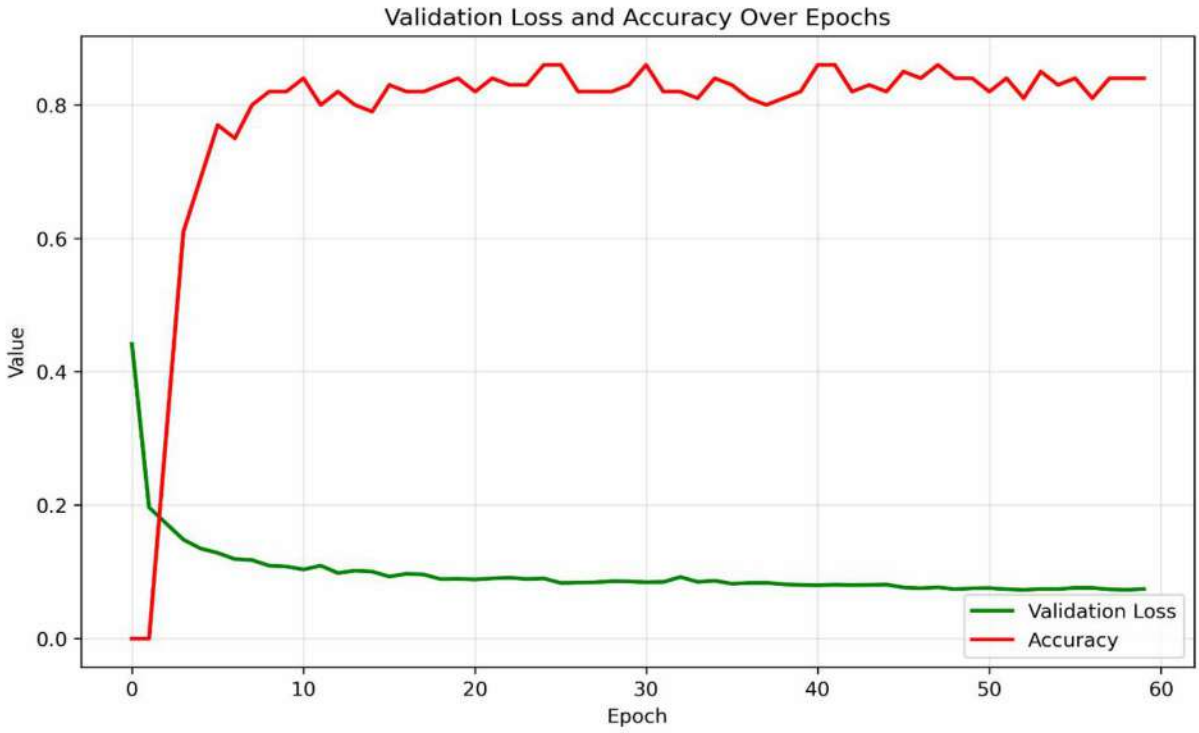

Figure 4.2: Validation Loss and Accuracy Over Epochs

Figure 4.3 reveals the variation in validation metrics across model forms. The float32 version of the model has a better Recall while the int8 version has a higher precision. Though the variation isn't significantly great, there is a tradeoff between performance and resource efficiency of the two model forms. The Float32 model (399.7KB RAM, 7ms latency) can be inferred much faster and has better aptitude to detect objects in real-time, whilst the Int8 model (123.9KB RAM, 652ms latency) can have a smaller memory footprint, but cannot be utilised in time-sensitive applications as the Int8 model has a higher latency.

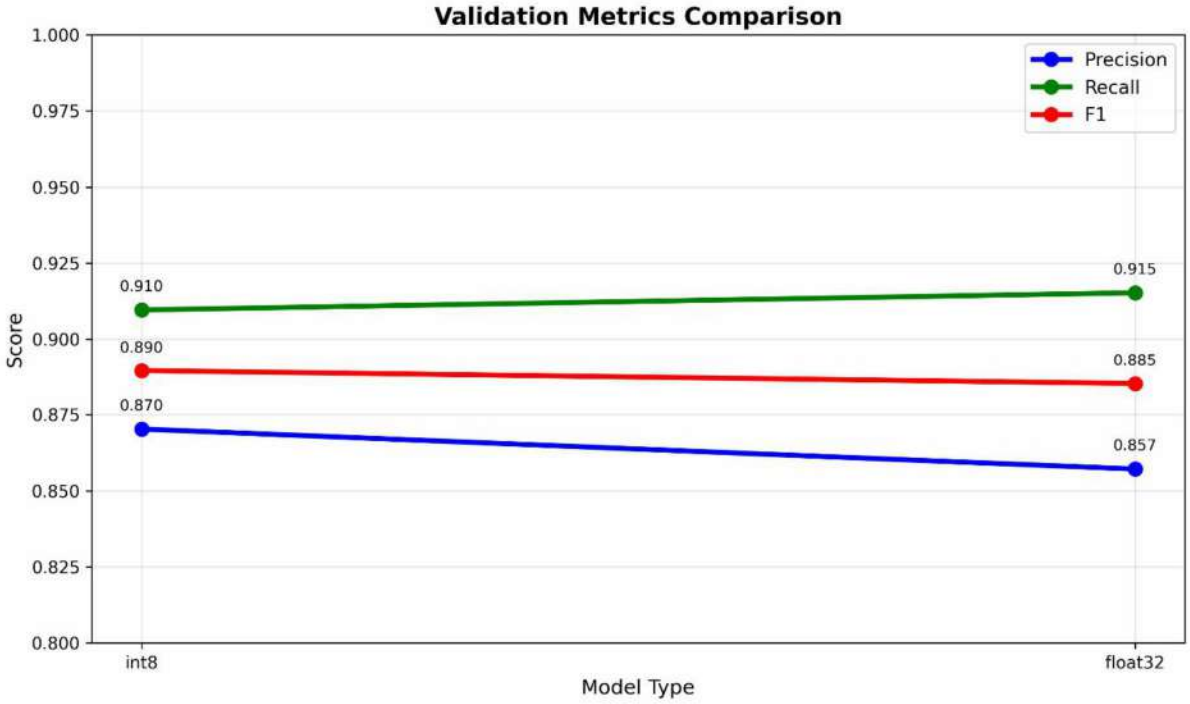

Figure 4.3: Validation Metrics Comparison

Figure 4.4 shows the validation metric performance through the training/validation history of the model. It can be seen that as the metrics approach the Epoch 60 mark, they become smoothened and stagnant. It was observed that after this Epoch 60, the metrics begin to drop, the model underperforming due to overfitting. Thus, the 60 epoch setting was the optimal training time for this usecase.

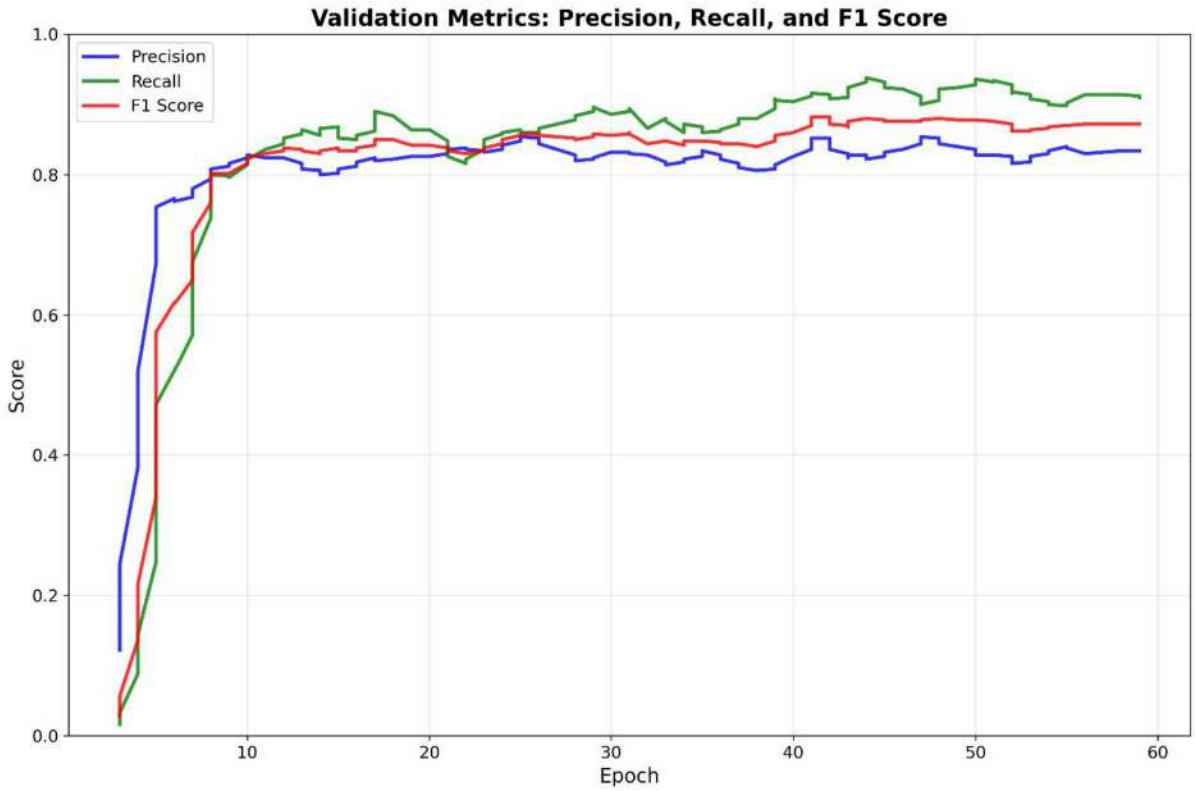

Figure 4.4: Validation Metrics: Precision, Recall and F1-Score

The confusion matrix shown in Figure 4.5 shows the bin class with an extremely high f1-score of 99% while the chair class had a slightly average f1-score of 85%.

The implementation of the object-detection model was done on the basis of the FOMO architecture, which was chosen due to its lightweight nature and the ability to make real-time inference in an embedded system. The convergence and generalisation performance were evaluated based on the epoch-level training and validation losses monitoring.

Figure 4.5 was obtained with the help of the Edge Impulse Results Board shows evidence of the model classification behaviour in relation to the dataset classes. The bin class had the best accuracy (98.1%) and false-prediction rate (1.9%), compared to the chair class, which was also incorrectly classified as background at a rate of 0.12.

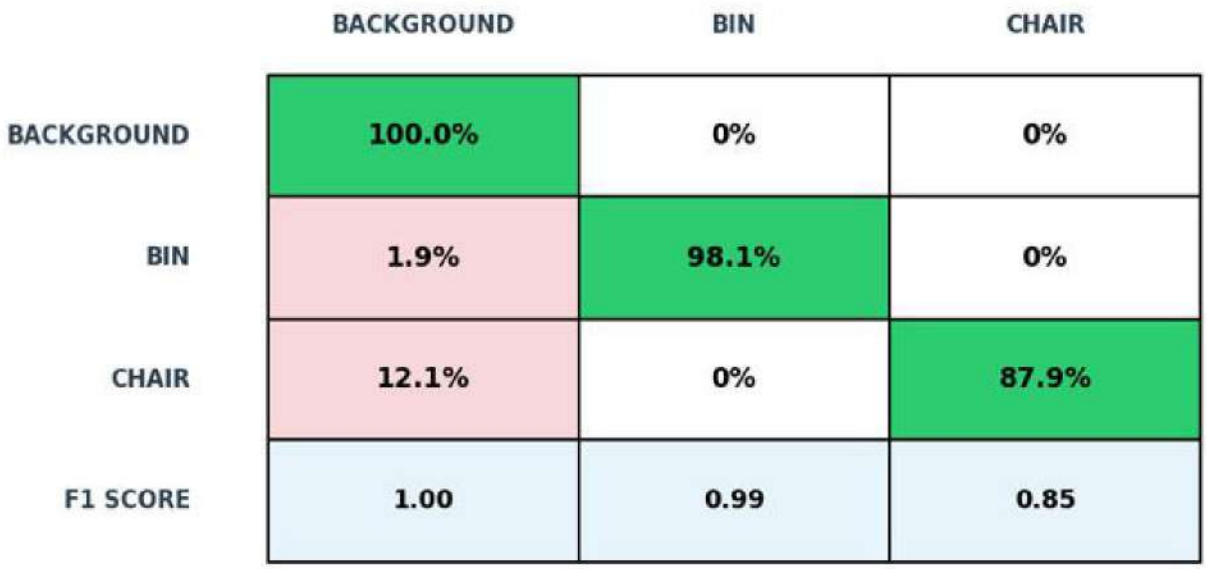

Figure 4.5: Confusion Matrix Extracted from the Edge Impulse Results Board

Figure 4.6 shows a live classification using the trained model on the Edge Impulse Platform. With a 100% precision, recall and f1-score, the test data was correctly predicted to be of obstacle class *bin*. The classification also provided a detailed result of the classification score, width, height x, y, cx and cy coordinates. These details could only be gotten through an object detection model as an Image Classifier is limited to only predicting the presence of the obstacle and not the position and coordinates.

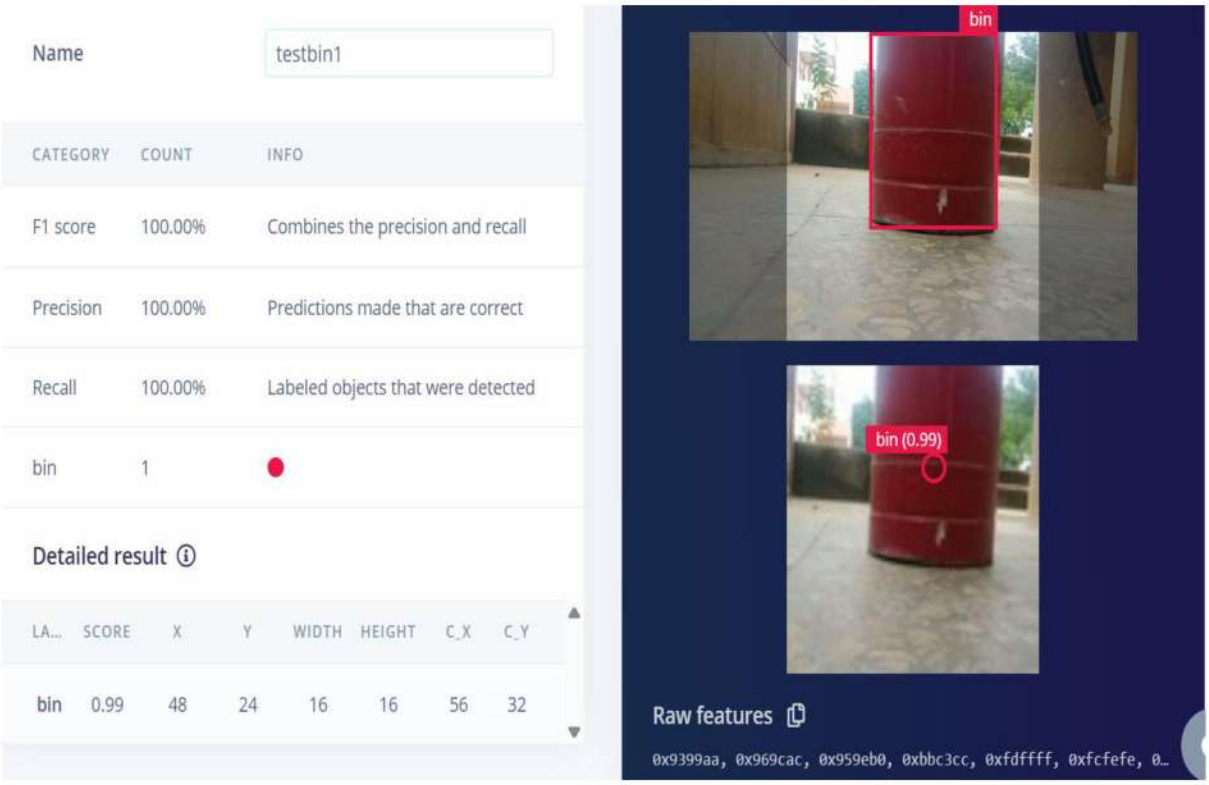

Figure 4.6: Live Classification of Test Data in Edge Impulse Platform

Table 4.1 shows the model training history across all 6 trials. It is observed that an increase in samples is directly proportional to the performance of the model. The model has more sources to learn from, leading to better generalisation and prediction. There is a significant jump in F1-score from Trial 4 as at this point, background images began to be introduced to the training process. This reduced the rate of false positives where classes are being mispredicted as background by the model. The final trial outputs a near 90% in F1-Score overall.

Table 4.1: Model Training History Across 6 Trials Using an 80/20 Split

S/N	TS1	TS2	TS3	VLA	TEA	F1S	PRS	REC	EPC
1	70	56	14	0.44	0.42	0.35	0.50	0.27	30
2	218	175	43	0.62	0.60	0.62	0.90	0.47	60
3	524	420	104	0.70	0.52	0.60	0.74	0.51	60
4	710	568	142	0.72	0.71	0.72	0.77	0.67	60
5	977	782	195	0.86	0.76	0.85	0.81	0.90	60
6	1228	1030	258	0.88	0.85	0.89	0.87	0.91	60

TS1 - Total Samples VLA - Validation Accuracy PRS - Precision

TS2 - Train Samples	TEA - Test Accuracy	REC - Recall
TS3 - Test Samples	F1S - F1-Score	EPC - Epochs

4.1.2 FOMO model performance limitations

The architecture of FOMO was chosen as it is lightweight and can detect in real time on an embedded platform. Such methods as data augmentation were used in the process of training, which led to better recall as the model is more likely to generalise to the changes in the dataset.

Weaknesses of the presented work are the fairly small sample size (around 1,000 samples) and a small number of object classes, which can limit the extrapolation of the model. The dataset needs to be expanded in the future, and more heterogeneous classes of objects need to be considered, as well as more sophisticated quantisation techniques to achieve a more optimal balance between memory efficiency and inference speed.

4.2 Hardware – Software Integration

The software-hardware integration is exhibited in the web-based communication service, which was created to provide real-time data exchange between the vehicle and an external server through Wi-Fi. The front-end interface, as presented in Figure 4.7, is the main user dashboard where monitoring and control are done. It shows the real-time telemetry of the vehicle's microcontroller, including the current location of the vehicle, distance to the target, obstacle status, and yaw, which gives a clear visualisation of the system's operational status.

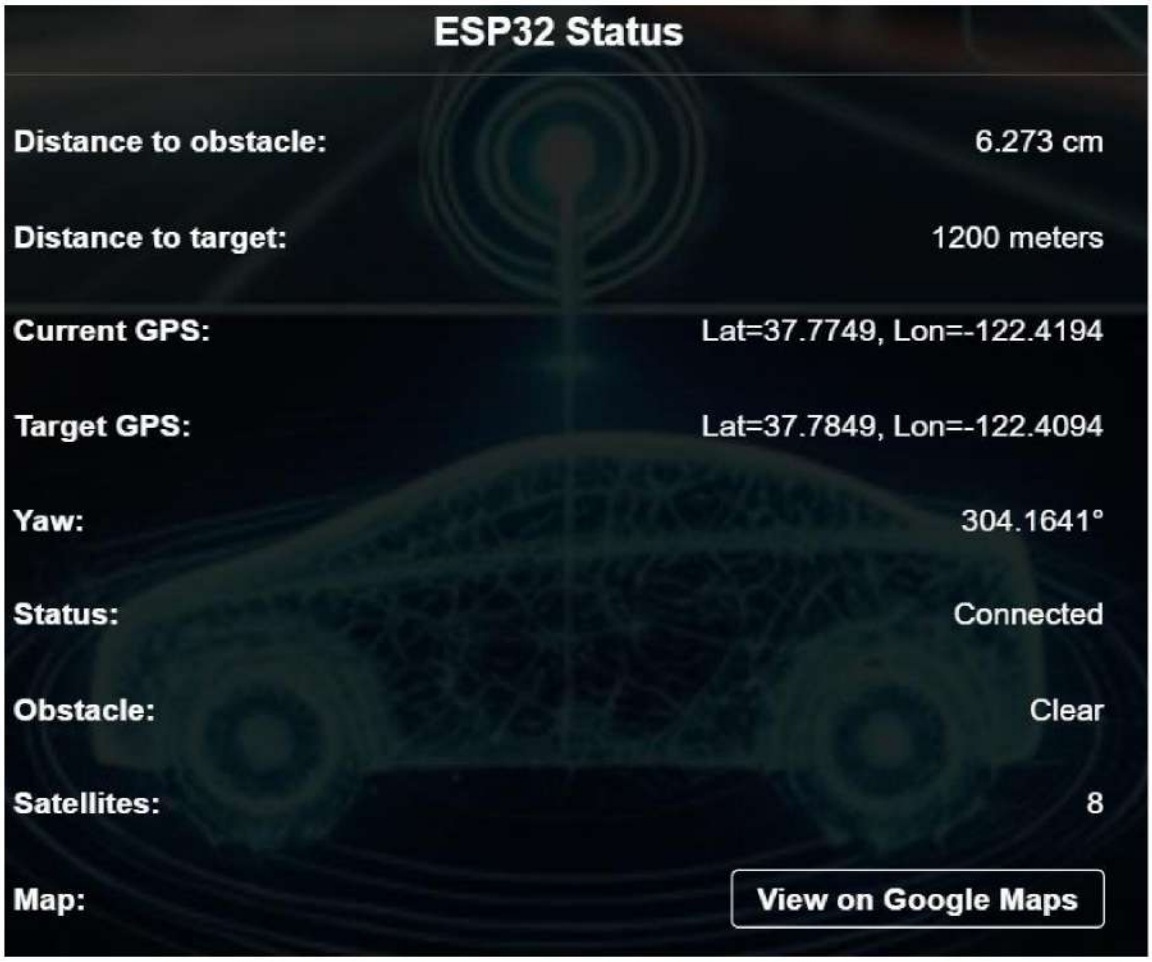

Figure 4.7: Frontend Interface for Vehicle Status Display

CHAPTER 5

5.0 Conclusions and Recommendations.

5.1 Summary

This project aimed to create an autonomous, obstacle-avoiding vehicle using TinyML for last-mile delivery. The introduction identified the growing need for effective and affordable delivery models, particularly in urban areas, and how autonomous vehicles may respond to those issues. The gap as identified in the problem statement was in low-cost, scalable autonomous platforms in low-income economies, with particular attention to balancing onboard processing, latency, and power consumption. The purpose was to combine a hybrid edge-cloud AI pipeline, sensor fusion optimisation and power-efficient sensor fusion control on the ESP32 Rover platform. This study was justified by the importance of autonomous systems in improving safety, efficiency of navigation, and lowering the operational cost of delivery services.

The literature review covered different dimensions of autonomous navigation systems, such as environment perception, localisation, mapping, and path planning, with a specific emphasis on probabilistic models and visual-inertial odometry. It also explored multi-sensor fusion algorithms and the techniques of detecting and avoiding obstacles, dividing them into vision-based (YOLOv3, MobileNet-SSD, Tiny-YOLO) and range-based sensing (ultrasonic sensors) techniques. Microcontroller platforms were discussed, and the ESP32 was selected as the platform to prototype autonomous vehicles due to its dual-core CPUs and built-in Wi-Fi/Bluetooth. Cloud and edge AI integration, and the benefits of moving complex model inference to the cloud, the reduction of latency with edge AI frameworks such as TensorFlow Lite Micro and ESP-DL were also discussed in the review. Lastly, it addressed the last-mile delivery applications, which are very expensive in commercial prototypes, and required low-

cost solutions, especially in areas such as Nigeria, where the RAIN programme is supporting open-source autonomous platforms.

The methodology described the architecture of the project, hardware (ESP32-Cam, 4WD RC Car Chassis, L298N Motor Driver, HC-SR04 ultrasonic sensor, SG90 servo, NEO-7M GPS, MPU6050 IMU, 3S2P battery setup, 3S BMS, battery level display, DC rocket switch, resistors, TF card, MB-100 breadboard, jumper wires, female charging adaptor, etc.), and software technologies. The server-side communication was done with an Express.js backend to estimate routes and calculate them, using the geolocation API of JavaScript to get user coordinates and the OpenRouteService API to generate waypoints. Data collection and preprocessing were described, as well as the training of the model and its implementation on the ESP32, namely, the use of the FOMO model to detect obstacles. A system design included a block diagram, a circuit diagram and a flowchart of operation.

5.2 Limitation

The main weakness of the present study is that the dataset on which the FOMO model was trained is quite small, with around 1,000 samples and a small number of object categories. This limited set of data may be a limiting factor in the generalisation capabilities of the model as it becomes exposed to a broader range of real-world challenges and situations.

5.3 Future Improvements

To make the system more resilient and efficient, the future work should focus on the following aspects:

- **Expansion of Dataset:** Significantly increase the size and variety of the training data set to cover a wider variety of object types. This will increase the capacity of the model to identify and categorise different barriers that are faced in real-life contexts.

- **Advanced Quantisation Techniques:** Research and implement more advanced quantisation techniques. This is aimed at reaching a higher balance between memory performance and inference speed, which is important in embedded systems where resources are limited.
- **Adaptability to Dynamic Environments:** Conduct more research on the flexibility of the vehicle to changeable and unforeseen road situations, especially in underdeveloped regions. This includes the enhancement of its capability to navigate different terrains, changing lighting conditions, and unforeseen challenges, and eventually, its reliability in various real-life scenarios.

REFERENCE

Anderson, J. M., Kalra, N., Stanley, K., & Oluwatola, T. (2014). Autonomous Vehicle Technology: a guide for policymakers. *ResearchGate*. https://www.researchgate.net/publication/296697033_Autonomous_Vehicle_Technology_A_Guide_for_Policymakers

Boysen, N., Fedtke, S., & Schwerdfeger, S. (2020). Last-mile delivery concepts: a survey from an operational research perspective. *OR Spectrum*, *43*(1), 1–58. https://doi.org/10.1007/s00291-020-00607-8

Bansod, A. (2023, May 5). *Guide to building efficient web servers with Express.js* [Webpage]. Retrieved November 3, 2025, from https://devway.hashnode.dev/express-js-simplified

Casado, R., & Bermúdez, A. (2020). A simulation framework for developing autonomous drone navigation systems. *Electronics*, *10*(1), 7. https://doi.org/10.3390/electronics10010007

Chauhan, A. (2023, January 10). Fully understand convolutional neural networks components. *Medium*. https://pub.towardsai.net/fully-understand-convolutional-neural-networks-components-a490e65d307

Chen, Y. (2025). Path planning algorithm for logistics autonomous vehicles at Cainiao stations based on multi-sensor data fusion. *PLoS ONE*, *20*(5), e0321257. https://doi.org/10.1371/journal.pone.0321257

Clotet, E., & Palacín, J. (2023). SLAMICP Library: Accelerating Obstacle Detection in Mobile Robot Navigation via Outlier Monitoring following ICP Localization. *Sensors*, *23*(15), 6841. https://doi.org/10.3390/s23156841

Complete Guide to Semantic Segmentation | Mindy Support Outsourcing. (n.d.). Mindy Support. https://mindy-support.com/news-post/complete-guide-to-semantic-segmentation/

Edge Impulse. (n.d.). *FOMO Object Detection for Constrained Devices.* Edge Impulse Documentation. Retrieved from https://docs.edgeimpulse.com/studio/projects/learning-blocks/blocks/object-detection/fomo

El-Sebah, M. I. A., Syam, F. A., Sweelem, E. A., El-Sotouhy, M. M., & Sotouhy, M. M. E. (2023b). A proposed controller for an autonomous vehicles embedded system. *WSEAS TRANSACTIONS ON CIRCUITS AND SYSTEMS*, *22*, 1–9. https://doi.org/10.37394/23201.2023.22.1

Engesser, V., Rombaut, E., Vanhaverbeke, L., & Lebeau, P. (2023). Autonomous Delivery Solutions for Last-Mile Logistics Operations: A Literature Review and Research Agenda. *Sustainability*, *15*(3), 2774. https://doi.org/10.3390/su15032774

Ghajargar, M., Zenezini, G., & Montanaro, T. (2016). Home delivery services: innovations and emerging needs. *IFAC-PapersOnLine*, *49*(12), 1371–1376. https://doi.org/10.1016/j.ifacol.2016.07.755

Gherardini, L., & Cabri, G. (2024). The impact of autonomous vehicles on safety, economy, society, and environment. *World Electric Vehicle Journal*, *15*(12), 579. https://doi.org/10.3390/wevj15120579

Gómez-Huélamo, C., Del Egido, J., Bergasa, L. M., Barea, R., López-Guillén, E., Arango, F., Araluce, J., & López, J. (2020). Train Here, Drive There: Simulating Real-World Use Cases with Fully-Autonomous Driving Architecture in CARLA Simulator. In *Advances in intelligent systems and computing* (pp. 44–59). https://doi.org/10.1007/978-3-030-62579-5_4

Guzmán, A. N., Lúa, C. A., García-Rodríguez, J. A., Vidrios-Serrano, C., & Meza-Aguilar, M. A. (2024). Real-Time embedded control of vehicle dynamics using ESP32: a

discrete nonlinear approach. *Electronics*, *13*(19), 3967. https://doi.org/10.3390/electronics13193967

He, K., Gkioxari, G., Dollar, P., & Girshick, R. (2018). Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(2), 386–397. https://doi.org/10.1109/tpami.2018.2844175

Hoffmann, T., & Prause, G. (2018). On the Regulatory Framework for Last-Mile Delivery Robots. *Machines*, *6*(3), 33. https://doi.org/10.3390/machines6030033

Imad, M., Doukhi, O., Lee, D. J., Kim, J. C., & Kim, Y. J. (2022). Deep Learning-Based NMPC for Local Motion Planning of Last-Mile Delivery Robot. *Sensors*, *22*(21), 8101. https://doi.org/10.3390/s22218101

Katare, D., Perino, D., Nurmi, J., Warnier, M., Janssen, M., & Ding, A. Y. (2023b). A survey on Approximate Edge AI for Energy Efficient Autonomous Driving services. *IEEE Communications Surveys & Tutorials*, *25*(4), 2714–2754. https://doi.org/10.1109/comst.2023.3302474

Katona, K., Neamah, H. A., & Korondi, P. (2024). Obstacle Avoidance and Path Planning Methods for Autonomous Navigation of Mobile Robot. *Sensors*, *24*(11), 3573. https://doi.org/10.3390/s24113573

Koga, Y., Miyazaki, H., & Shibasaki, R. (2020). A Method for Vehicle Detection in High-Resolution Satellite Images that Uses a Region-Based Object Detector and Unsupervised Domain Adaptation. *Remote Sensing*, *12*(3), 575. https://doi.org/10.3390/rs12030575

Kopias, C. (2022, November 21). *The ultimate guide to FFmpeg* [Webpage]. IMG.LY. Retrieved November 3, 2025, from https://img.ly/blog/ultimate-guide-to-ffmpeg/

Krejčí, J., Babiuch, M., Babjak, J., Suder, J., & Wierbica, R. (2022). Implementation of an Embedded System into the Internet of Robotic Things. *Micromachines*, *14*(1), 113. https://doi.org/10.3390/mi14010113

Liu, C. (2022). Research on GPS vehicle Navigation System based on embedded technology. *Journal of Physics Conference Series*, *2170*(1), 012045. https://doi.org/10.1088/1742-6596/2170/1/012045

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In *Lecture notes in computer science* (pp. 21–37). https://doi.org/10.1007/978-3-319-46448-0_2

Martínez-Díaz, M., & Soriguera, F. (2018). Autonomous vehicles: theoretical and practical challenges. *Transportation Research Procedia*, *33*, 275–282. https://doi.org/10.1016/j.trpro.2018.10.103

Mushtaq, A., Haq, I. U., Nabi, W. U., Khan, A., & Shafiq, O. (2021). Traffic flow management of autonomous vehicles using platooning and collision avoidance strategies. *Electronics*, *10*(10), 1221. https://doi.org/10.3390/electronics10101221

Paiano, M., Martina, S., Giannelli, C., & Caruso, F. (2024). Transfer learning with generative models for object detection on limited datasets. *Machine Learning Science and Technology*, *5*(3), 035041. https://doi.org/10.1088/2632-2153/ad65b5

Paiva, S., Ahad, M., Tripathi, G., Feroz, N., & Casalino, G. (2021). Enabling Technologies for Urban Smart Mobility: Recent Trends, Opportunities and Challenges. *Sensors*, *21*(6), 2143. https://doi.org/10.3390/s21062143

Paksadze, I. (2025). INNOVATIVE SOLUTIONS FOR SUSTAINABLE LOGISTICS: LAST-MILE DELIVERY APPLICATIONS WITH AUTONOMOUS VEHICLES. ResearchGate. https://doi.org/10.5281/zenodo.14846788

Pico, N., Montero, E., Vanegas, M., Ayon, J. M. E. E., Auh, E., Shin, J., Doh, M., Park, S., & Moon, H. (2024). Integrating Radar-Based Obstacle Detection with Deep Reinforcement Learning for Robust Autonomous Navigation. *Applied Sciences*, *15*(1), 295. https://doi.org/10.3390/app15010295

Ranieri, L., Digiesi, S., Silvestri, B., & Roccotelli, M. (2018). A Review of Last Mile Logistics Innovations in an Externalities Cost Reduction Vision. *Sustainability*, *10*(3), 782. https://doi.org/10.3390/su10030782

Redmon, J., & Farhadi, A. (2022). YOLOV3: an incremental improvement. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1804.02767

Reed, S., Campbell, A. M., & Thomas, B. W. (2022). Impact of autonomous vehicle assisted Last-Mile delivery in urban to rural settings. *Transportation Science*, *56*(6), 1530–1548. https://doi.org/10.1287/trsc.2022.1142

Rybczak, M., Popowniak, N., & Lazarowska, A. (2024). A Survey of Machine Learning Approaches for Mobile Robot Control. *Robotics*, *13*(1), 12. https://doi.org/10.3390/robotics13010012

Sahoo, L. K., & Varadarajan, V. (2025). Deep learning for autonomous driving systems: technological innovations, strategic implementations, and business implications - a comprehensive review. *Complex Engineering Systems*, *5*(1). https://doi.org/10.20517/ces.2024.83

Sanchez-Iborra, R., & Skarmeta, A. F. (2020). TinyML-Enabled Frugal Smart Objects: Challenges and opportunities. *IEEE Circuits and Systems Magazine*, *20*(3), 4–18. https://doi.org/10.1109/mcas.2020.3005467

Sorooshian, S., Sharifabad, S. K., Parsaee, M., & Afshari, A. R. (2022). Toward a Modern Last-Mile Delivery: Consequences and Obstacles of Intelligent Technology. *Applied System Innovation*, *5*(4), 82. https://doi.org/10.3390/asi5040082

Valera, Á., Valero, F., Vallés, M., Besa, A., Mata, V., & Llopis-Albert, C. (2021). Navigation of autonomous light vehicles using an optimal trajectory planning algorithm. *Sustainability*, *13*(3), 1233. https://doi.org/10.3390/su13031233

Wang, X., Sun, Y., Xie, Y., Bin, J., & Xiao, J. (2023). Deep reinforcement learning-aided autonomous navigation with landmark generators. *Frontiers in Neurorobotics*, *17*. https://doi.org/10.3389/fnbot.2023.1200214

Wang, Y., Han, J. H., & Beynon-Davies, P. (2018). Understanding blockchain technology for future supply chains: a systematic literature review and research agenda. *Supply Chain Management an International Journal*, *24*(1), 62–84. https://doi.org/10.1108/scm-03-2018-0148

Wu, X., Wang, G., & Shen, N. (2023b). Research on obstacle avoidance optimisation and path planning of autonomous vehicles based on attention mechanism combined with multimodal information decision-making thoughts of robots. *Frontiers in Neurorobotics*, *17*. https://doi.org/10.3389/fnbot.2023.1269447

Xiang, Y. (2023). Autonomous Driving Ethics: Self-driving cars facing trolley problems. *Lecture Notes in Education Psychology and Public Media*, *15*(1), 96–102. https://doi.org/10.54254/2753-7048/15/20231039

Xie, P., Wang, H., Huang, Y., Gao, Q., Bai, Z., Zhang, L., & Ye, Y. (2024). LiDAR-Based Negative Obstacle Detection for Unmanned Ground Vehicles in Orchards. *Sensors*, *24*(24), 7929. https://doi.org/10.3390/s24247929

Zhu, Q., Yu, B., Wang, Z., Tang, J., Chen, Q. A., Li, Z., Liu, X., Luo, Y., & Tu, L. (2023). *Cloud and edge computing for connected and automated vehicles*. https://doi.org/10.1561/9781638283034

APPENDICESS

Appendix A - Technical Specifications of Components

A.1 Processing and Connectivity

- ESP32 Microcontroller (Typical ESP32-WROOM-32 Module
  - Processor: Dual-Core Xtensa LX6, 32-bit, up to 240 MHz
  - Memory: 520 KiB SRAM, 448 KiB ROM
  - Wireless: Wi-Fi 802.11 b/g/n, Bluetooth v4.2 BR/EDR and BLE (Bluetooth Low Energy)
  - GPIO: 34 programmable General-Purpose Input/Output pins
  - Peripherals: 18x 12-bit ADCs, 2x 8-bit DACs, 4x SPI, 2x I^2C, 3x UART, I^2S, Pulse Counter
  - Operating Voltage: 2.2V to 3.6V (Module); Dev boards typically run on 5V via USB/VIN pin

- ESP32-Cam Module (AI-Thinker Model)
  - SoC: ESP32-S
  - Camera: OV2640 (2-megapixel)
  - Image Resolution (max): 1600x1200 (UXGA)
  - Image Formats: JPEG, BMP, Grayscale
  - Storage: Onboard microSD (TF) card slot, supports up to 4GB
  - Flash: Built-in 32Mbit (4MB) PSRAM
  - Operating Voltage: 5V (via pin)
  - Power Consumption: ~6mA (Deep Sleep), 180mA@5V (Flash off), 310mA@5V (Flash on)

A.2 Chassis and Actuators

- L298N Motor Driver Module
    - Driver Chip: L298N
    - Driver Type: Dual H-Bridge
    - Motor Supply Voltage (Vm): 5V to 35V DC
    - Logic Level Voltage (Vss): 5V DC
    - Max Drive Current: 2A per channel (continuous)
    - Max Power Dissipation: 25W
    - Control: Can control two DC motors (direction and PWM speed control) or one stepper motor.

- SG90 Servo Motor
    - Type: Micro Servo
    - Operating Voltage: 4.8V to 6.0V DC
    - Stall Torque: 1.2 kg/cm (at 4.8V), 1.6 kg/cm (at 6.0V)
    - Operating Speed: 0.12 sec/60° (at 4.8V, no load)
    - Rotation Range: 0° to 180° (pulse-width dependent)
    - Control System: PWM (Pulse Width Modulation), ~20ms pulse cycle
    - Gears: Plastic

A.3 Sensors and Navigation

- Ultrasonic Sensor (HC-SR04
    - Operating Voltage: 5V DC
    - Operating Current: 15mA
    - Working Frequency: 40Hz

- Measuring Range: 2cm to 400cm (4 meters)
- Ranging Accuracy: ±3mm
- Measuring Angle: 15°
- Interface: 4 pins (VCC, Trig, Echo, GND)

- GPS Module (NEO-7M)
  - Chipset: u-blox NEO-7M
  - Receiver Type: 56-channel u-blox 7 engine (GPS, QZSS)
  - Horizontal Position Accuracy: 2.5m CEP (Autonomous)
  - Tracking Sensitivity: -161 dBm
  - Acquisition Time (Cold Start): ~27 seconds
  - Update Rate: 1 Hz (default), up to 10 Hz (configurable)
  - Operating Voltage: 3.3V to 5V DC
  - Interface: UART (9600 baud rate default)
- IMU Sensor (MPU6050 Module)
  - Chip: MPU-6050
  - Sensors: 3-axis MEMS Gyroscope + 3-axis MEMS Accelerometer
  - Gyroscope Range (Programmable): ±250, ±500, ±1000, ±2000 °/s
  - Accelerometer Range (Programmable): ±2g, ±4g, ±8g, ±16g
  - ADC: 16-bit on-chip ADCs for both gyro and accelerometer
  - Onboard Processor: Digital Motion Processor (DMP) for sensor fusion
  - Interface: I²C (up to 400kHz)
  - Operating Voltage: 3.0V to 5.0V (Module includes regulator)

A.4 Power System

- Li-ion Battery (Typical 18650 Cell)
    - Type: Cylindrical Lithium-Ion
    - Dimensions: ~18mm (diameter) x 65mm (length)
    - Nominal Voltage: 3.7V
    - Full Charge Voltage: 4.2V
    - Discharge Cut-off Voltage: ~2.75V
    - Capacity: 6800mAh
    - Configuration in Project: 3S (3 cells in series)

- 3-Series Battery Management System (3S BMS)
    - Configuration: 3 cells in series (3S)
    - Nominal Voltage (3S Pack): 11.1V
    - Full Charge Voltage (3S Pack): 12.6V
    - Functions: Overcharge protection (~4.25V/cell), over-discharge protection (~2.5V/cell), short-circuit protection, and cell balancing.
    - Current Rating: 20A continuous discharge

- 3-Series Battery Level Indicator
    - Type: 4-segment LED display module
    - Input: 3S Li-ion battery pack (connects to P+ and P-)
    - Display Logic (Typical):
        - 1 LED: < ~10.5V
        - 2 LEDs: < ~11.1V
        - 3 LEDs: < ~11.7V
        - 4 LEDs: > ~11.7V (up to 12.6V)

- Note: Voltage levels may vary slightly by module design.

A.5 Prototyping and Storage

- 4GB TF Card (microSD)
  - Type: TransFlash / microSD
  - Storage Capacity: 4 Gigabytes
  - Speed Class: Varies (e.g., Class 4, Class 10)
  - File System (Typical): FAT32
  - Operating Voltage: 2.7V to 3.6V

- Breadboard (MB-100 / MB-102)
  - Type: Solderless prototyping board
  - Tie Points (Typical): 830 total
    - 630 component tie-points (in 63 rows of 10)
    - 200 power rail tie-points (4 rails of 50)
  - Pitch: Standard 0.1" (2.54mm)
  - Wire Size: Accepts 20-29 AWG wires

Appendix B - Shell Script for Frame Extraction

```
#!/bin/bash

mkdir -p extracted_frames

for video in /videodataset/*.mp4; do

 filename=$(basename "$video" .mp4)

 mkdir -p "extracted_frames/$filename"
```

ffmpeg -i "$video" -vf fps=1 "extracted_frames/$filename/frame_%04d.jpg"

done

Appendix C - Python Script for Plotting Metrics from Edge Impulse Classification Report

```python
import json
import matplotlib.pyplot as plt
import numpy as np
import os

def parse_json_file(filename):
    """Parse the JSON file and extract validation metrics"""
    with open(filename, 'r') as file:
        data = json.load(file)

    metrics = {}

    # Extract int8 model metrics
    int8_data = data['validation']['int8']
    metrics['int8'] = {
        'precision': int8_data['non_background']['precision'],
        'recall': int8_data['non_background']['recall'],
        'f1': int8_data['non_background']['f1']
    }

    # Extract float32 model metrics
    float32_data = data['validation']['float32']
    metrics['float32'] = {
        'precision': float32_data['non_background']['precision'],
        'recall': float32_data['non_background']['recall'],
        'f1': float32_data['non_background']['f1']
    }

    return metrics
```

```python
def create_validation_plot(metrics, save_path='./validation_plots'):
    """Create and save the validation metrics plot"""

    # Create save directory if it doesn't exist
    if not os.path.exists(save_path):
        os.makedirs(save_path)

    # Prepare data for plotting
    model_types = ['int8', 'float32']
    metric_names = ['precision', 'recall', 'f1']
    colors = ['blue', 'green', 'red']

    # Create the plot
    plt.figure(figsize=(12, 8))

    # Set the width of bars and positions
    bar_width = 0.25
    x_pos = np.arange(len(model_types))

    # Create bars for each metric
    for i, (metric, color) in enumerate(zip(metric_names, colors)):
        values = [metrics[model][metric] for model in model_types]
        plt.bar(x_pos + i * bar_width, values, bar_width,
                label=metric.capitalize(), color=color, alpha=0.8)

    # Customize the plot
    plt.xlabel('Model Type', fontsize=12)
    plt.ylabel('Score', fontsize=12)
    plt.title('Validation Metrics Comparison: Precision, Recall, and F1 Score', fontsize=14, fontweight='bold')
    plt.xticks(x_pos + bar_width, model_types, fontsize=11)
    plt.ylim(0, 1)  # Metrics range from 0 to 1
    plt.legend(fontsize=11)
    plt.grid(True, alpha=0.3, axis='y')

    # Add value labels on bars
    for i, model in enumerate(model_types):
        for j, metric in enumerate(metric_names):
            value = metrics[model][metric]
            plt.text(x_pos[i] + j * bar_width, value + 0.01,
                     f'{value:.3f}', ha='center', va='bottom', fontsize=9)

    # Adjust layout
    plt.tight_layout()

    # Save the plot
    plt.savefig(os.path.join(save_path, 'validation_metrics_comparison.png'), dpi=300, bbox_inches='tight')
    plt.savefig(os.path.join(save_path, 'validation_metrics_comparison.pdf'), bbox_inches='tight')

    # Also create a line plot version
    plt.figure(figsize=(10, 6))

    for i, (metric, color) in enumerate(zip(metric_names, colors)):
        values = [metrics[model][metric] for model in model_types]
        plt.plot(model_types, values, 'o-', linewidth=3, markersize=8,
                 label=metric.capitalize(), color=color)

    plt.xlabel('Model Type', fontsize=12)
    plt.ylabel('Score', fontsize=12)
```

```python
def create_detailed_table(metrics, save_path='./validation_plots'):
    # Create a text file with detailed metrics
    with open(os.path.join(save_path, 'validation_metrics_details.txt'), 'w') as f:
        f.write("VALIDATION METRICS DETAILS\n")
        f.write("=" * 50 + "\n\n")

        for model_type in ['int8', 'float32']:
            f.write(f"{model_type.upper()} MODEL:\n")
            f.write("-" * 20 + "\n")
            f.write(f"Precision: {metrics[model_type]['precision']:.4f}\n")
            f.write(f"Recall:    {metrics[model_type]['recall']:.4f}\n")
            f.write(f"F1 Score:  {metrics[model_type]['f1']:.4f}\n\n")

        # Calculate differences
        f.write("DIFFERENCES (Int8 - Float32):\n")
        f.write("-" * 30 + "\n")
        for metric in ['precision', 'recall', 'f1']:
            diff = metrics['int8'][metric] - metrics['float32'][metric]
            f.write(f"{metric.capitalize()}: {diff:+.4f}\n")

def main():
    # File path - adjust if needed
    json_file = 'ei-fyp-object-detection-model-evaluation-metrics-json-file-model.4 (1).json'

    try:
        # Parse the JSON file
        metrics = parse_json_file(json_file)

        print("Extracted Validation Metrics:")
        print("=" * 40)
        for model_type in ['int8', 'float32']:
            print(f"\n{model_type.upper()} Model:")
            print(f"  Precision: {metrics[model_type]['precision']:.4f}")
            print(f"  Recall:    {metrics[model_type]['recall']:.4f}")
            print(f"  F1 Score:  {metrics[model_type]['f1']:.4f}")

        # Create and save plots
        create_validation_plot(metrics)

        # Create detailed table
        create_detailed_table(metrics)

        # Save numerical data to CSV for reference
        import pandas as pd
        df_data = []
        for model_type in ['int8', 'float32']:
            df_data.append({
                'Model_Type': model_type,
                'Precision': metrics[model_type]['precision'],
                'Recall': metrics[model_type]['recall'],
                'F1_Score': metrics[model_type]['f1']
            })

        df = pd.DataFrame(df_data)
```

Appendix D - C++ Inference Code for using the deployed FOMO model