Software Frameworks for Rural AI Robotics

A thesis

submitted in partial fulfilment

of the requirements for the degree

of

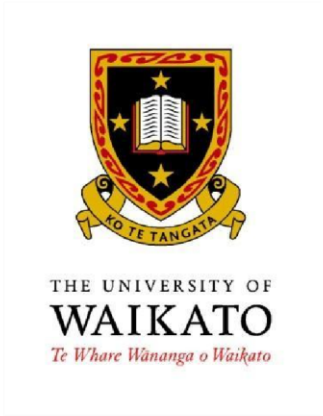
Master of Engineering(Software Engineering)

at

The University of Waikato

by

Faizan Bashir



2026

Abstract

Robotic Autonomous robotic systems intended for agricultural environments must operate under conditions of unreliable connectivity, limited infrastructure, and high environmental variability. These constraints present significant challenges for system architecture, communication design, and human–robot interaction. This thesis investigates these challenges through the design, analysis, and evaluation of Nicobot, a mobile agricultural research platform developed to explore software frameworks for rural AI robotics.

The research focuses on how system architecture, user interface design, and communication mechanisms influence the feasibility of autonomous operation in rural and low-connectivity settings.

Architectural analysis reveals that, despite the presence of edge computing components and multiple communication technologies, decision-making and fault recovery remain heavily dependent on human operators. The system therefore operates under a supervisory autonomy model, where the robot supports controlled behaviour but does not independently interpret sensor data or adapt to connectivity degradation. Evaluation of the user interface shows that technical data exposure, limited feedback, and reliance on user interpretation reduce usability for non-technical agricultural users. Communication analysis further demonstrates that multiple channels alone do not guarantee robustness unless they are integrated with adaptive system behaviour.

The findings highlight that autonomy in rural agricultural robotics cannot be achieved through the addition of technological components alone. Instead, autonomy emerges from deliberate architectural allocation of responsibility, clear separation of control roles, and interface designs that support transparency and reduced cognitive load. This thesis contributes design-oriented insights for future rural robotic systems, emphasising contextual independence, supervisory interaction, architectural coherence, and simplicity as guiding principles for practical agricultural autonomy.

Dedication and acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Judy Bowen, for her guidance, constructive feedback, and continued support throughout this research. Her expertise and critical insight have been instrumental in shaping the direction and quality of this thesis. I am also thankful to my co-supervisor, Professor Melanie Ooi, for her valuable advice and encouragement during the course of this work.

I gratefully acknowledge the Department of Computer Science and Software Engineering at the University of Waikato for providing the academic environment, technical resources, and access to the Nicobot research platform that made this project possible.

I would also like to thank my family for their unwavering support, patience, and encouragement.
Their belief in me has been a constant source of motivation throughout my postgraduate studies.

Faizan Bashir

January 2026

Author’s declaration

I, Faizan Bashir, declare that this thesis entitled “Software Frameworks for Rural AI Robotics” is my own work and has not been submitted, either wholly or in part, for any other academic qualification at this or any other institution.

This research was conducted in partial fulfilment of the requirements for the degree of Master of Engineering with endorsement in Software Engineering at the University of Waikato, New Zealand. Where the work of others has been used, it has been appropriately acknowledged and cited in accordance with academic conventions.

This thesis is a design-centred research study based on the Nicobot research platform. It draws on published literature, system documentation, and original software development undertaken by the author. No human or animal ethics approval was required for this research. I take full responsibility for the content of this thesis, including any errors or omissions.

To support the preparation of this document, I used Grammarly to assist with grammar correction, English language refinement, and limited paraphrasing of cited material. Diagrams and figures were created using Base44 and Lucidchart. I also used QuillBot, a generative AI–based tool, to assist with sentence restructuring and language clarity where required. OpenAI tools were used to support understanding of thesis structure and academic writing style.

For software development, I consulted a range of publicly available technical resources and documentation. All software developed for the Nicobot system was written by me, including code in C++, HTML, JavaScript, and Python. I assume full responsibility for any errors or omissions in this document.

SIGNED: .....Faizan Bashir..... DATE: .... January 2026.....

Table of Contents

List of Table

List of Figures

Chapter 1: Background

1.1 Introduction

The agriculture sector has been experiencing tremendous change in recent years due to the adoption of new technology, especially robotics and automation technology. Modern agriculture is gradually becoming intelligent and efficient in response to issues such as labour shortages, increased global food demand as well as the quest to adopt sustainable farming methods. One of the key factors in this change is the use of agricultural robots, also known as agribots, to perform repetitive, labour-intensive, or dangerous duties with considerable precision and consistency. Such activities are cropping surveillance, spraying, seeding, harvesting, and environmental sensing. Robotic systems enhance productivity, lower the operational costs and aid precision agriculture methods which make better use of resources by automating such functions.

The Nicobot, a semi-autonomous mobile robot developed to assist in small to mediumscale farming processes, is one of the developments in this area. The idea for Nicobot was developed, and an initial prototype was created, as part of an applied research project at the University of Waikato in 2021. The project addressed challenges in agricultural robotics, with particular focus on usability, autonomous reliability, and communication resilience in rural environments. The robot has a modular hardware and software design, which incorporates the low-power embedded systems, multiple communication protocols (Wi-Fi, LoRa, MQTT) and a dual-mode control interface that allows both manual and autonomous operation.

Nicobot was developed as a proof-of-concept agricultural robotic platform intended to explore how mobile robots could support tasks such as crop inspection and environmental monitoring. The system is not designed as a deployed field robot; instead, it is used to investigate usability, autonomous behaviour, and communication resilience in rural and low-connectivity environments. It has multiple sensors on-board and a frontfacing camera system to provide autopilot navigation and interact with the environment autonomously. Notably, Nicobot is designed with usability in mind, with end-users, including farmers and field technicians, not necessarily having high-level technical skills, being the target audience of the product. Nicobot seeks to popularise the idea of agricultural robotics by providing a user-friendly interface and high-quality remotecontrol features, which will enable people to apply agricultural robots in practice.

1.2 Background and Context

Advances in robotics and automation have led to significant developments across a wide range of domains, including logistics, manufacturing, healthcare, and defence. In recent years, the integration of artificial intelligence, machine learning, and Internet of Things technologies has expanded the capabilities of autonomous systems, enabling them to operate with greater efficiency, adaptability, and precision. As a result, autonomous mobile robots have become an active area of research and deployment in both structured environments, such as factories and warehouses, and unstructured environments, such as outdoor and remote settings.

Autonomous mobile robots are designed to navigate and operate with limited or no human intervention. Core research challenges in this field include reliable autonomous navigation,

obstacle avoidance, environmental perception, and decision-making under uncertainty. To address these challenges, such robots typically rely on a combination of sensing technologies, including cameras, LiDAR, and ultrasonic sensors, which provide real-time information about the surrounding environment. This sensor data supports localisation, mapping, path planning, and safety-related decisions, making perception a fundamental component of autonomous robotic systems.

Within this broader field, agricultural robotics has emerged as a particularly important application area. Agricultural environments introduce unique challenges due to variable terrain, changing weather conditions, limited infrastructure, and unreliable network connectivity. Robots operating in these settings must be capable of navigating uneven ground, detecting crops, obstacles, and equipment, and operating safely in proximity to humans and animals. These requirements place additional demands on sensing accuracy, system robustness, and fault tolerance.

Agricultural robots are increasingly investigated for tasks such as crop inspection, environmental monitoring, and data collection, with the aim of improving productivity and supporting precision agriculture practices. However, many existing systems remain experimental, and further research is required to examine usability, communication resilience, and the practical limitations of autonomous operation in real agricultural conditions. Consequently, agricultural robotics continues to be an active research area focused on the development and evaluation of prototype platforms that inform the design of future fieldready systems.

1.3 Research Motivation and Objectives

The motivation for this research is to develop a clearer understanding of the types of hardware configurations and system architectures required to support autonomous agricultural robots operating in rural environments. Agricultural settings differ significantly from controlled or industrial environments due to uneven terrain, variable environmental conditions, limited infrastructure, and unreliable network connectivity. These factors introduce design constraints that must be addressed at the architectural level to support reliable and usable robotic systems.

Many existing robotic systems are designed primarily for structured environments and do not explicitly account for the architectural requirements of agricultural operation. There is therefore a need to examine how sensing, computation, communication, and user interaction can be organised within a single system architecture to better suit the demands of agricultural robotics. This research is motivated by the need to identify and describe such architectural considerations in a systematic manner.

Nicobot is used in this study as a reference system through which these architectural considerations are examined. The system is not presented as a deployable agricultural robot, nor is it used for experimental validation. Instead, it provides a concrete basis for analysing existing design choices and for reasoning about how hardware and architectural arrangements influence system suitability for agricultural use. Through this analysis, the study identifies

architectural characteristics that would be required to support a transition towards a more robust agricultural robotic system.

The primary objective of this study is to design and describe a hardware and software architecture suitable for autonomous agricultural robotics operating in rural and lowconnectivity environments. This includes defining the roles and interactions of processing units, sensing components, communication mechanisms, and user interfaces within a coherent architectural framework.

A secondary objective is to identify architectural limitations present in the current Nicobot design, such as manual system initialisation, dependence on network availability, and constraints within the user interface. These limitations inform proposed architectural refinements aimed at improving scalability, reliability, and ease of use, without making claims regarding performance testing or deployment readiness.

The contribution of this dissertation lies in the architectural design and analysis of an autonomous agricultural robotic system. By using Nicobot as a reference system, the work provides structured design guidance for others seeking to develop similar robotic platforms intended for operation in rural and disconnected environments.

In this research, autonomy is not treated as the complete removal of human involvement, but as a supervisory model in which the system assumes greater responsibility while retaining transparency and the ability for manual intervention. This framing reflects the safety-critical and context-dependent nature of agricultural environments.

1.4 Research Questions

This research examines the design challenges associated with autonomous robotic systems intended for use in rural and agricultural environments. The study is structured around three research questions that guide the architectural and interface-focused analysis presented in this dissertation.

RQ1: How does the system architecture of Nicobot support autonomous operation in remote agricultural settings?

Chapter 3 examines the hardware configuration, system architecture, and operational workflow of Nicobot. The chapter analyses how sensing components, processing units, communication mechanisms, and control interfaces are organised within the existing system and identifies architectural limitations relevant to operation in rural and lowconnectivity environments. A proposed architectural specification is then presented to address these limitations.

RQ2: How does the design of the Nicobot user interface support usability for nontechnical users in agricultural contexts?

Chapter 4 analyses the existing Nicobot web-based control interface and documents its structure, interaction mechanisms, and presentation of system information. The chapter then presents a redesigned interface based on usability principles, focusing on clarity, control, safety visibility, and suitability for field-based use by non-technical operators.

RQ3: How are communication mechanisms incorporated within the Nicobot architecture to support operation in low-connectivity environments?

Chapter 3 examines the communication components of Nicobot as part of the overall system architecture, including Wi-Fi, LoRa, and message-based communication mechanisms. The chapter analyses how these communication methods are integrated with the control architecture and how they influence system operation in environments with limited or unreliable network infrastructure.

Together, these research questions define the scope of the dissertation and structure the analysis of Nicobot as a reference system.

1.5 Thesis Overview

Chapter 1 introduces the background and motivation for the study, outlining the growing role of robotics in agriculture and the unique challenges posed by rural environments. It defines the research problem, establishes the objectives of the study, and presents the three research questions that guide the investigation. The chapter also clarifies the scope of the research and positions Nicobot as a research platform used to reason about architectural and interaction design choices rather than as a field-deployed agricultural product.

Chapter 2 provides a comprehensive literature review examining prior work in agricultural robotics, rural artificial intelligence, and autonomous systems. Particular attention is given to architectural approaches that address limited connectivity, including edge computing, hybrid edge–cloud models, and decentralised control. The chapter also reviews literature on communication resilience and user-centred design in robotic systems. These studies establish the theoretical foundation for the architectural and interface analysis undertaken in later chapters.

Chapter 3 describes the existing hardware configuration and system architecture of Nicobot. The chapter documents the physical components, layered architectural organisation, communication mechanisms, and operational workflow of the system. Through architectural analysis, it examines how sensing, control, communication, and user interfaces are currently structured and identifies limitations that affect autonomous operation in remote agricultural settings. This chapter directly addresses Research Question 1 and Research Question 3 by analysing architectural support for autonomy and communication under low-connectivity conditions.

Chapter 4 focuses on the evaluation of the existing Nicobot user interface. The chapter analyses the structure, layout, interaction mechanisms, and feedback provided by the webbased control dashboard. A heuristic usability evaluation is conducted using Nielsen's usability principles to assess the interface from the perspective of nontechnical agricultural users. The chapter highlights usability limitations related to feedback, cognitive load, and interpretability, addressing Research Question 2.

Chapter 5 presents a structured discussion of the findings in relation to the research questions. Drawing on the architectural analysis and interface evaluation, the chapter reflects on how Nicobot currently supports autonomy, usability, and communication in rural contexts. The

discussion emphasises the system's reliance on manual control and human supervision, framing Nicobot as a supervisory autonomy platform rather than a fully autonomous system. The chapter also situates these findings within the broader literature on agricultural robotics and rural AI.

Chapter 6 concludes the thesis by summarising the key insights gained from the study and reflecting on the implications of the design-oriented findings. It discusses autonomy as an emergent property of system organisation, highlights the role of human oversight in agricultural robotics, and considers ethical, environmental, and socio-technical implications.

Chapter 2: Literature Review

2.1 Edge Computing of Rural AI Robotics

Autonomous robotic systems operating in rural and agricultural environments are commonly designed under conditions of limited, intermittent, or unavailable network connectivity. In response to these constraints, researchers describe architectural approaches that minimise reliance on continuous communication with centralised cloud services. One such approach involves organising data processing locally, either onboard the robot or on nearby computing devices, to enable operation in environments where stable internet access cannot be assumed (Sathya et al., 2024).

In descriptions of agricultural and mobile robotic platforms, local processing is frequently assigned responsibility for tasks that require timely system responses, including navigation, obstacle detection, and interpretation of sensor data. These architectures typically place the processing of camera imagery, distance measurements, and environmental sensor readings close to the data source, rather than routing this information to remote servers for immediate handling. Authors frame this organisational choice as a way of limiting dependence on persistent network connectivity, which is often unavailable in rural deployment contexts (Xie et al., 2022).

A further theme emerging from the literature concerns operational independence from continuous network connectivity. Edge-oriented designs are described as enabling robotic platforms to continue basic operation even when communication with cloud services is unavailable. In contrast to cloud-dependent architectures, which rely on persistent data transmission to remote servers, edge-based systems are structured to limit reliance on external communication for core functionality. This characteristic is repeatedly highlighted as relevant to agricultural robots deployed in remote or rural regions where connectivity may be intermittent or absent (Kalyani and Collier, 2021).

Privacy and data handling are also discussed in relation to edge computing within rural robotic systems. Several authors note that processing data locally reduces the need to transmit sensitive information, such as environmental measurements or location data, beyond the robotic platform. By retaining raw data at the edge, system architectures limit exposure associated with remote data transfer, an issue that is particularly relevant in rural deployments where secure communication infrastructure may be limited (Singh et al., 2024).

Edge computing is additionally framed as a means of managing computational load within distributed agricultural systems. Rather than transferring all collected data to central servers, many designs described in the literature allocate preliminary processing tasks locally and reserve cloud resources for non-time-critical functions. This division of responsibilities is presented as a way of limiting bandwidth usage and avoiding unnecessary strain on centralised infrastructure as the number of deployed robots or sensors increases (Aldossary and Alharbi, 2021).

Across agricultural contexts, examples cited in the literature include robotic platforms, autonomous vehicles, and distributed sensor systems that generate operational data within the field. In these systems, edge computing units are commonly positioned as intermediaries

that process and filter data close to the source, while cloud platforms are used for storage, aggregation, or later analysis. This separation supports responsive local operation while maintaining the capacity for broader system coordination through higherlevel services (Szekely et al., 2024).

2.2 Cloud Integration for Scalable Rural Automation

Cloud integration is also very critical in the creation of scalable and effective autonomous systems, specifically in rural automation. Although edge computing has the advantage of processing data in real-time on the robot, cloud integration has a number of benefits regarding scalability in the long term, data aggregations, remote monitoring, and data analytics. Cloud integration enables rural automation to operate efficiently over a broad geographical region, communicate with each other and other systems, and access powerful cloud-based services that would otherwise not be achievable in an entirely local environment (Kalyani and Collier, 2021).

Figure 2.1 demonstrates a system flow based on which the different layers of the adaptive architecture are interacting with in a rural AI robotics framework. On a tractor level, various forms of data are gathered, such as geolocation (latitude and longitude), energy usage (fuel or power usage), and model weights depending on the tasks that the tractor performs. This information is then pooled and sent to a worldwide server where model differentiation, cluster formation and selection of driver nodes are conducted. The network availability (3G, 4G or 5G) is tested during the system operation to identify the possibility of transmitting updated models to the tractor back to undergo different training and be checked at the check point. This is a closed-loop process that can update the operations of the tractor in an adaptive and context-sensitive manner depending on the network conditions, which are usually prevalent in the rural setting.

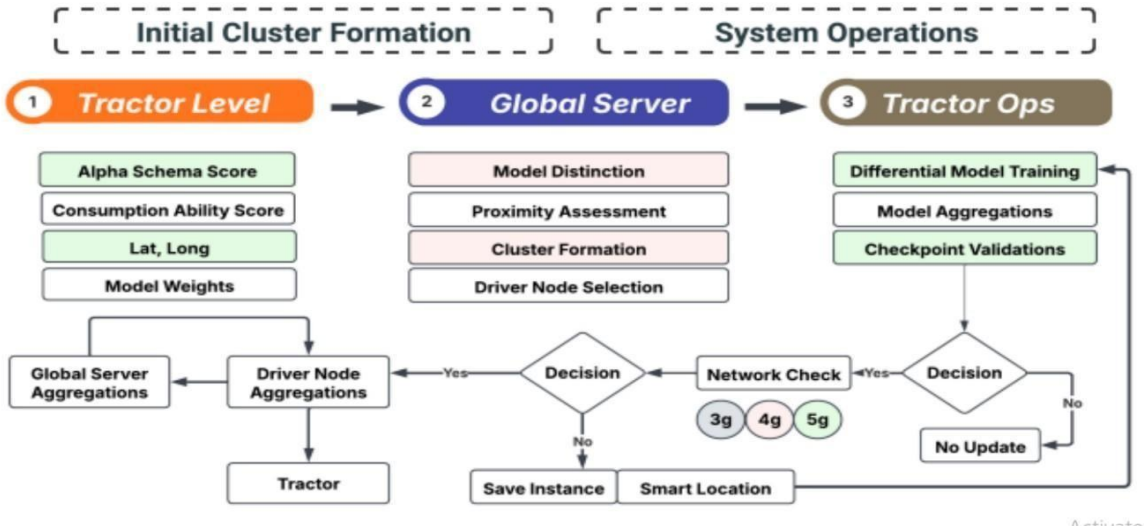


Figure 2.1: Adaptive architecture for rural AI robotics, depicting the clustering, server processing, and decision-making flow in the system. (Source: Puppala and Sinha, 2025) Cloud integration enables distributed agricultural systems to share data and coordinate decisionmaking through centralised platforms. In farming contexts, cloud-based architectures allow information collected from multiple field devices to be aggregated and processed to support automated irrigation and resource management. By centralising data analysis and

control logic, such systems reduce the need for continuous human supervision while optimising resource use across the deployment (Phasinam et al., 2022).

Remote monitoring and control constitute an important advantage of cloud integration in smart agricultural and robotic systems. In rural environments, operators are not required to be physically present in the field to oversee system operation. Cloud-enabled platforms allow real-time data collected from sensors and automated devices to be transmitted to remote servers, where it can be accessed through web-based dashboards or mobile applications. This enables operators to continuously monitor system performance, track operational progress, and remotely issue control commands to actuators when necessary. In addition, cloud platforms can automatically generate alerts related to abnormal conditions, equipment faults, or maintenance requirements, thereby improving reliability and reducing response time. Such remote supervision capabilities are particularly important for large-scale agricultural deployments, where coordinating multiple automated systems across geographically dispersed areas would otherwise be logistically complex and labour-intensive (Thilakarathne et al., 2023).

In addition to real-time control and monitoring, cloud integration can be used to perform powerful data analytics and machine learning (ML) functions. Large-scale data storage and computing resources accessible on the cloud can help robots recognize trends in their operations and keep on enhancing their operation. As an illustration, the ML algorithms can utilise the past data on the conditions of crops or soil wellbeing to predict tendencies, schedule optimally, and suggest certain measures that would result in more efficient and sustainable farming methods (Benos et al., 2021).

The internet connectivity can be considered one of the most crucial issues in the rural automation. The limitation can be addressed with the help of hybrid cloud-based architectures where the most important data are handled in the robot by edge computing, and the data that are not so time-sensitive are syncretized with the cloud once the connection is established. This strategy will guarantee the robots can be functional and autonomous even when the network connection is on the way and occasionally unreachable (Kuchuk & Malokhvii, 2024).

Cloud integration can also support the aggregation and organisation of operational data generated by agricultural robotic systems. By enabling sensor readings, energy-related metrics, and task status information to be transmitted to remote computing resources, cloudbased architectures allow system data to be processed, analysed, and accessed beyond the physical deployment site. This capability supports timely observation of system behaviour and facilitates remote assessment of operational conditions. In agricultural contexts, such architectures contribute to more efficient field management by allowing computationally intensive analysis and system optimisation to be performed off-board, thereby improving responsiveness to operational issues and supporting informed decision-making at a system level (Ahmad et al., 2025).

Cloud integration is essential to scalable rural automation, which is needed not only to make autonomous robots more efficient and cooperative but also to provide credible monitoring and remote control. In solutions like Nicobot, edge computing and cloud assets provide a tradeoff, allowing robots to work independently in the remote regions and still utilize the

strong data aggregation, analysis and control of the cloud. This combination of both increases the overall functionality of the robot with the ability to remain scalable, flexible, and adaptive to the requirements of the system of rural and large-scale automation (Ayranci and Erkmen, 2024).

2.3 Federated Learning

Data privacy and security are important factors to take into consideration when considering rural AI robotics and involving any agricultural data, i.e., soil moisture levels, crop health, and environmental conditions, which can often be sensitive by nature. Federated learning provides a decentralised option to conventional machine learning (ML) systems, as compared to which data has to be centralised to train a model, making it impossible to use sensitive data. This is particularly applicable in the rural automation systems where challenges involved in privacy, connection and data sovereignty may become a major challenge (Dembani et al., 2025).

Federated learning is a distributed machine learning method which enables two or more devices or systems to jointly educate a design without communicating unprocessed information. As opposed to sending the data to a central server, every single device (in this instance, every autonomous robot or sensor) locally calculates the model updates and merely sends them to the central server. The server consequently combines the updates so as to narrow down the global model. This way, data will be stored on the local devices, preserving privacy and minimising the security issues involved in centralised data storage to an enormous level (Shen, 2020).

This decentralised learning structure is particularly suitable for soil moisture prediction in rural agricultural environments, where sensor data are spatially distributed and highly sensitive at the farm level. In federated learning-based soil moisture modelling, each sensing node or robotic platform trains a local prediction model using its own measurements, while only model parameters are exchanged with the central aggregation server. The aggregated updates are then used to refine a global model capable of estimating soil moisture conditions across the entire network. By avoiding the transmission of raw environmental data, this approach preserves farmer privacy and reduces security risks associated with centralised storage. In addition, it accommodates unreliable rural connectivity by limiting communication to lightweight model updates rather than continuous data streaming. Recent studies demonstrate that federated learning can achieve prediction accuracy comparable to centralised approaches while maintaining strong privacy guarantees and robustness in distributed agricultural IoT deployments (Zakzouk & Said, 2025).

Within the Nicobot system, this approach enables real time soil moisture forecasting without reliance on persistent cloud connectivity. When deployed in remote agricultural locations, Nicobot can make irrigation related decisions locally while still benefiting from collaborative model improvement across multiple robots and sensing nodes. Each robot adapts its prediction model to local soil and environmental conditions through on device training, while contributing to the refinement of the global model through parameter sharing. This combination of local autonomy and system wide cooperation supports both operational resilience and privacy preservation, making federated learning a suitable architectural choice for intelligent and scalable rural robotic deployments (Žalik and Žalik, 2023).

In addition, federated learning enhances scalability and flexibility of the systems in automation of moisture content of soil in rural areas. The system can support many robots and sensors widely spread without any difficulty because raw data are not transmitted to a central training node. This renders the federated learning especially appropriate to a large-scale agricultural environment where a substantial number of robots gather soil data in various positions and can all contribute to improving the global model (Gupta et al., 2025).

The agricultural sector faces substantial data privacy challenges, as farmers are often reluctant to share operational information due to concerns about unauthorised access, lack of transparency, and potential misuse by technology providers or third parties. Limited control over farm data, unclear legal agreements, and fragmented governance frameworks have been shown to reduce farmer trust in digital and automated agricultural technologies, thereby slowing their adoption. In response to these concerns, federated learning provides a technically appropriate approach by enabling predictive model development while keeping raw farm data locally stored on each robotic platform or sensing device. By restricting data exchange to model parameters only, federated learning supports accurate analytics while preserving confidentiality and aligning with the privacy and governance requirements emphasised within the agricultural data protection. This approach can therefore help strengthen user confidence and support wider adoption of automated agricultural systems (Kaur et al., 2022).

Other recent works have also revealed that federated learning can contribute a considerable amount to the accuracy of soil moisture forecasting, and the level of privacy protection is also high. Precision agricultural studies have already proven that the technique can produce quality models that can predict harvests, keep track of soil moisture, as well as identify plant infections without losing the sensitive information about the farm (Mohammed, 2025).

Figure 2.2 depicts a federated learning architecture described by (Almurshed et al. 2022), in which model training is distributed across multiple edge devices while raw sensor and image data remain local to each device. In this arrangement, individual nodes train local models using their own data and share only model parameters or updates with higherlevel aggregation nodes, which are then combined into an intermediate and global model. The figure demonstrates a hierarchical aggregation structure intended to reduce the need for centralised data collection in distributed systems.

(Almurshed et al. 2022) further describe a self-adaptive task coordination mechanism based on a Greedy Nominator Heuristic (GNH). Within this framework, the heuristic is used to select which edge devices participate in a given training round based on their current availability and communication state. Devices that are unable to contribute updates due to connectivity or resource constraints are temporarily excluded, while available nodes are prioritised. The authors use this strategy to explain how federated learning processes may be organised in environments where network conditions are variable, such as geographically distributed or rural deployments.

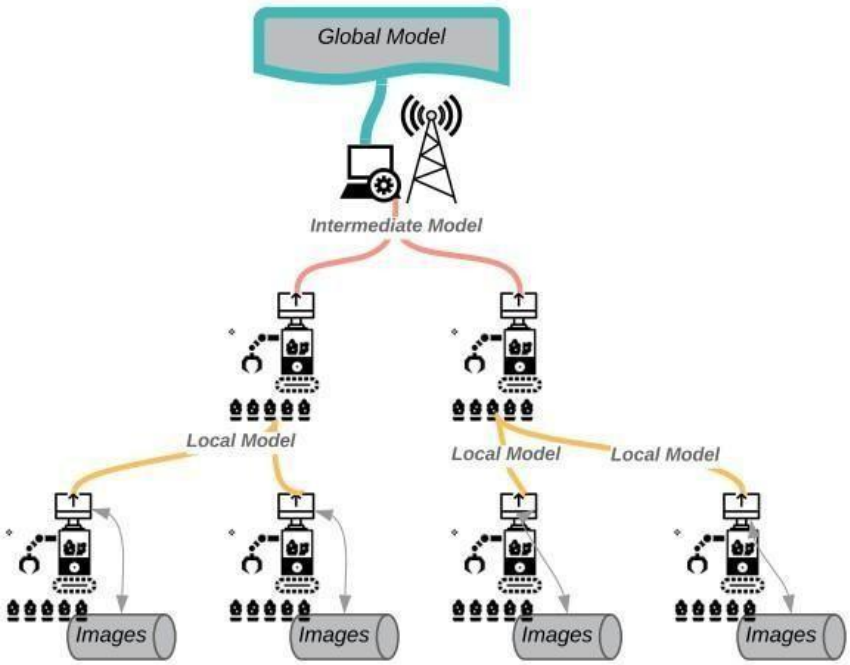


Figure 2.2: Aggregating learned models across robots using Federated Learning (Source: Almurshed et al., 2022)

The centralised data aggregation risks are minimised through distributed optimisation. The privacy and scalability of hierarchical model aggregation is ensured by aggregating local models into intermediate models before transforming them into a global one thus ensuring that raw data is not leaked. Studies on fault-tolerant scheduling indicate that tasks are not assigned to faulty and compromised nodes, and this guarantees that the privacy of data is maintained even when the system fails.

2.4 Serverless Functions of Scalable Rural Robotics

As Serverless computing is described in the literature as an architectural approach in which application logic is deployed as discrete cloud-hosted functions that are executed in response to specific events. In this approach, responsibility for provisioning, scaling, and maintaining the underlying server infrastructure is assigned to the cloud service provider, rather than to system developers or operators. This execution structure is discussed as being suitable for distributed systems where computational demand is variable and difficult to predict over time (Patros et al., 2022).

Within rural robotics research, serverless computing is framed as a means of supporting processing tasks that do not require continuous execution. Instead of relying on permanently active servers, computational tasks are organised as individual functions that are invoked only when required, such as when data becomes available or a systemlevel request is issued. Authors describe this event-driven structure as aligning with deployment contexts where activity levels fluctuate and where maintaining persistent cloud infrastructure may be impractical (Zangana et al., 2024).

Scalability is addressed through the ability of serverless platforms to allocate computational resources dynamically. Authors describe how multiple instances of a function may be executed

concurrently when demand increases, and reduced when demand subsides, without requiring manual intervention. This behaviour is discussed as supporting distributed robotic systems that generate uneven workloads across time and location, particularly in geographically dispersed rural environments (Calavaro et al., 2025).

Several studies position serverless computing as a complementary component within edge–cloud architectures that must operate under intermittent connectivity. In such systems, serverless functions can be invoked when communication becomes available, enabling deferred processing and periodic synchronisation in environments such as rural deployments where continuous connectivity cannot be assumed (Gangwani, 2024).

Overall, the literature presents serverless computing as an architectural mechanism that supports event-driven, on-demand processing within scalable rural robotics systems. Rather than replacing edge-based computation, serverless functions are positioned as a cloud-side complement that enables flexible resource allocation and deferred processing in environments characterised by variable workloads and constrained infrastructure (Farahani et al., 2024).

2.5 Human–Robot Interaction and Usability in Agricultural Robotics

Human-robots interaction (HRI) is an important part of creation and application of autonomous systems, especially in the agricultural robotics, where robots must work with human operators in the real-world conditions. Such systems cannot be effective solely because of their technical and mechanical performance but also because of their interactivity with human operators. This involves the familiarity of the user with the robot, how they can control the robot and the amount of trust they have on the system. The physiological functionality of the robotic system will also be critical in the successful implementation and the sustainability of robots in agricultural settings where the education level of farmers, technicians, and field workers might be low (Adamides and Edan, 2023).

The effectiveness and the user acceptance of the system is directly related to the design of human-robot interface and the comfort of the operator working with the robot. Usability is the nature of the intuitiveness, ease of use and simplicity of a system to use and it includes layout of the interface of the system, system feedback, amount of control and training needed. Agricultural robots are expected to operate under limited supervision and respond to changing environmental factors because they are often sent to spend time in unpredictable environments (Schraick et al., 2025).

Agricultural robots are often used in the conditions when time-related decisions need to be taken. An example is that a robot might be required to change irrigation systems dynamically according to sensor data. Such decisions, in case the human-robot interface is not clear and easy to use, can be postponed or have an error, which may influence productivity. Consequently, the interface should allow operators to make decisions, which are informed and not rushed, and make them, even at the moment of pressure (Yerebakan and Hu, 2024).

Successful human-robot interaction during agricultural application is based on proper training and onboarding. Researchers emphasize that a key to the successful implementation of autonomous systems in agriculture is the availability of easy access to the training materials suitable even to users who do not have a technical background (Halder, 2023). Easy and clear

materials, including illustrative manuals, visual guides, and instructional videos, have been demonstrated to dramatically lower the learning curve, therefore, allowing farmers and technicians to incorporate robotics in their daily work.

Human–Robot Interaction also focuses on trust and reliability. To achieve long-term acceptability, robots should not be expected to act autonomously only but also have predictable behavior and deliver obvious feedback to a user. To alert users to system mistakes or errors, the systems should have simple communication tools, including alerts or visual indicators. According to (Gackstetter et al. 2023), user trust and interaction are boosted by interfaces that provide easy-to-understand troubleshooting advice and discuss the nature of an issue when things cannot be connected or mechanical errors occur.

2.6 Summary and Research Gaps

The main technological pillars of autonomous rural robotics, such as edge computing, cloud integration, federated learning, serverless computing, and human-robot interaction (HRI) have been covered in this chapter. The literature review reveals the personal and combined advantages of these technologies, i.e. edge computing can make fast decisions at the data collection point, cloud can be used for scalability and remote control, and federated learning can offer privacy-preserving data analysis at a distributed mode. Moreover, the design of HRI and usability has been identified as important to achieve acceptance and later adoption in the non-technical users in the agricultural setting.

Nevertheless, there are several existing research gaps despite these advances. The use of edge computing has been extensively experimented in controlled setups but its capability in unpredictable rural conditions has not been fully tested. On the same note, although the cloud-based systems are scalable, little is known about the performance of the hybrid edgecloud architecture in cases where the connectivity is weak or intermittent. However, despite being highly promising with respect to privacy and distributed intelligence, federated learning still needs further investigation with respect to preserving the model accuracy on more diverse, decentralised datasets, as is common in agricultural fields.

Applications to rural robotics with serverless computing are also not well developed, especially where there is a need to operate offline or have nearly real-time responsiveness. Also, a significant part of the currently available HRI studies has been done in controlled or laboratorylike conditions, and little study of actual user experience and interaction issues in active agricultural contexts.

These gaps motivate the use of the Nicobot system as a reference platform through which architectural and design considerations for rural agricultural robots can be discussed. By considering the use of edge computing, federated learning, and serverless processing within a hybrid architectural framework, this research aims to clarify how such technologies may be structured to address commonly identified challenges related to autonomy, scalability, and operation under constrained connectivity. In addition, usability is considered from a design perspective through a heuristic evaluation of the control interface, rather than through direct end-user testing. The discussion arising from this work is intended to inform understanding of the requirements and design trade-offs involved in developing agricultural robotic systems for

rural deployment, contributing to broader discussions on scalable and user-centred system design in agricultural robotics.

Chapter 3: Evaluation and Redesign of Architecture

3.1 Existing Hardware Configuration of Nicobot

The current hardware configuration of Nicobot extends considerably beyond the early prototype. The system now incorporates multiple sensing devices, dual computing units, structured communication modules and a redesigned power-distribution network. These elements are mounted on an upgraded chassis that supports the spatial arrangement, thermal requirements and wiring constraints associated with a multi-component robotic platform.

Nicobot's hardware is organised around four primary functional subsystems: the processing and control units, the sensing system, the communication hardware and the actuation and power-supply subsystems. These subsystems operate together to enable manual control, remote operation, data acquisition and experimental development of autonomous behaviours. The physical layout is arranged so that high-current wiring is separated from lowvoltage signalling, sensor fields of view are unobstructed, and wireless antennas remain clear from metallic interference.

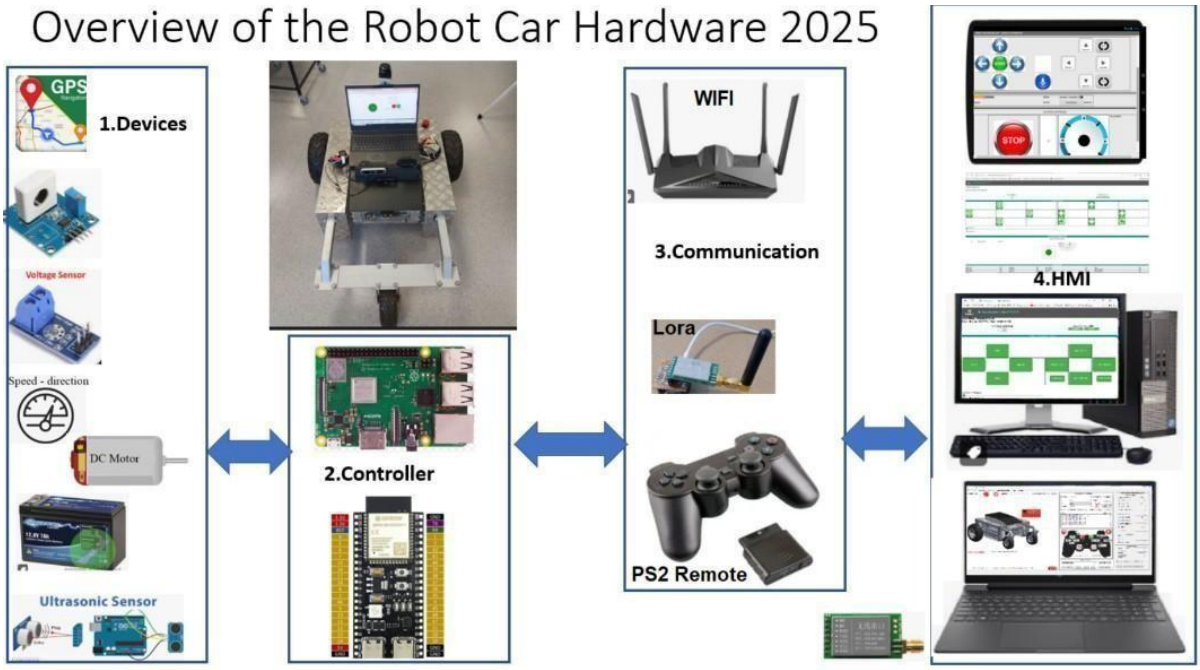


Figure 3.1: Overview of the hardware used in Nicobot. (Source: Pham X. K., 2025)

3.1.1 Processing and Control Hardware

The processing and control functions of Nicobot are divided between an ESP32 microcontroller and a Raspberry Pi 3B+. These devices operate as independent but interlinked controllers.

The ESP32 serves as the real-time controller for all low-level operations. Its responsibilities include generating PWM signals for the BTS7960 motor driver, acquiring ultrasonic readings, measuring voltage and current levels, receiving positional data from the GPS module and managing MQTT and LoRa client communication when required. Time-critical tasks such as distance measurement and motor actuation are routed through the ESP32 to ensure minimal latency and consistent timing.

The Raspberry Pi 3B+ functions as the supervisory controller. It hosts the Mosquitto MQTT broker, manages the Apache web server used for browser-based control and handles the USB video stream from the Logitech webcam. All high-level operator commands are routed through the Raspberry Pi before being transmitted to the ESP32. This configuration positions the Raspberry Pi as the central network gateway for the system.

The placement of these controllers reflects engineering considerations related to wiring length, electromagnetic interference and cooling. The ESP32 is mounted near the motor driver to reduce inductive effects on PWM lines, while the Raspberry Pi is positioned to ensure reliable Wi-Fi reception and to avoid proximity to high-current cabling.

Figure 3.2: Physical assembly of Nicobot showing the chassis, mounted sensors and control hardware. (Source: Pham X. K., 2025)

3.1.2 Sensing System

Nicobot incorporates a heterogeneous sensing suite designed to capture depth, distance, visual and electrical information from the environment and internal subsystems. The Intel RealSense D455 depth camera provides aligned RGB and depth imagery and is mounted at the front of the chassis to obtain an unobstructed view of the robot's direction of travel. Its USB interface connects directly to the Raspberry Pi, which processes the incoming camera frames.

A 360-degree RPLIDAR C1 laser scanner is installed in an elevated position to ensure a clear scanning plane. It provides planar range information suitable for obstacle detection and mapping tasks. Two ultrasonic sensors are positioned at the front and rear of the robot to supply short-range proximity readings. These sensors are connected directly to the ESP32 for timing accuracy.

Electrical performance is monitored through voltage sensors that measure supply levels in both the motor and control battery groups, and through current sensors that measure the load

drawn by the motors. A GPS NEO-8M module supplies geolocation estimates and timing synchronisation when satellite visibility is available.

Each sensor communicates through a designated protocol determined by device type and required sampling rate. Devices requiring high-frequency acquisition, such as the ultrasonic modules, communicate with the ESP32, while higher-bandwidth sensors such as the depth camera interface with the Raspberry Pi.

3.1.3 Camera System

Nicobot includes two cameras that support perception and remote monitoring. The RealSense D455 provides depth and RGB frames suitable for object detection and environmental analysis. The Logitech webcam delivers live video through the Raspberry Pi's web interface. Both cameras are rigidly mounted to preserve calibration and are angled forward for consistent observation of environmental features.

3.1.4 Communication Hardware

The communication subsystem supports multiple modes of interaction between Nicobot and external clients. The ESP32 and Raspberry Pi each include Wi-Fi functionality that enables MQTT messaging, web-dashboard communication and WebSocket traffic. A LoRa SX1278 transceiver provides long-range low-bandwidth communication for environments where WiFi connectivity is unavailable or unstable. A PS2 wireless controller and paired receiver provide a direct manual-control channel that bypasses network dependencies. The placement of communication modules ensures antenna exposure and reduces interference from motor wiring and metal components.

3.1.5 Power-Supply System

Nicobot operates using two electrically isolated battery groups. The propulsion system is supplied by ten 7 Ah, 12 V batteries wired to support the current demands of the two DC motors. The control system, including the ESP32, Raspberry Pi and sensors, is powered by three 7 Ah, 12 V batteries. This isolation prevents voltage drops caused by motor loads from affecting sensitive components. Voltage and current sensors are positioned in the distribution network for monitoring and diagnostics.

3.1.6 Motor-Driver and Actuation Hardware

The two 12 V DC geared motors provide propulsion for the robot. These motors are controlled through a BTS7960 H-bridge driver, which receives PWM and direction signals from the ESP32. The driver regulates the current delivered to the motors and protects against electrical faults. Wiring between the driver and motors is kept as short as possible to minimise resistive losses and noise.

3.2 Specifications of the Hardware Design

Table 3.1 documents all hardware components incorporated in Nicobot, together with their functional roles and quantities. The system uses one ESP32 microcontroller that conducts realtime motor control, sensor interfacing, PWM generation and MQTT and LoRa client functions. A single Raspberry Pi 3B+ provides high-level network coordination, including running the MQTT broker, the Apache web server and videostream handling. The sensing components include one Intel RealSense D455 depth camera for RGB and depth measurement, one RPLIDAR C1 sensor for 360-degree planar scanning, two ultrasonic sensors positioned at the front and rear for short-range distance measurement, two current sensors for assessing motor load and a GPS NEO-8M receiver for positional data.

The camera system includes one Logitech USB camera mounted forward for live video streaming. Communication modules consist of two Wi-Fi interfaces, one each on the ESP32 and Raspberry Pi, enabling MQTT and WebSocket communication. One LoRa SX1278 module is installed to provide long-range low-bandwidth communication, and a single PS2 wireless controller with receiver is retained for manual and emergency control.

Actuation is achieved through one BTS7960 motor driver that governs the electrical flow to two 12 V DC geared motors. The motors serve as the primary propulsion units for the robot. Power for the propulsion system is provided by ten 7 Ah, 12 V batteries, while three 7 Ah, 12 V batteries supply the control electronics and sensors.

Component Category	**Component Name**	**Quantity**	**Purpose**
Processing Units	ESP32 Microcontroller	1	Real-time motor control, PWM generation, sensor interfacing, MQTT client and LoRa handling
	Raspberry Pi 3B+	1	MQTT broker, Apache web server and wireless message coordination
Sensing Components	Intel RealSense D455	1	Depth measurement and RGB imaging

	RPLIDAR C1	1	360-degree obstacle scanning
	Ultrasonic Sensors (front and rear)	2	Short-range distance measurement
	Current Sensors	2	Measurement of motor current draw
	GPS NEO-8M	1	Acquisition of geolocation and positional data
Camera System	Logitech USB Camera	1	Live video streaming
Communication Modules	Wi-Fi Interfaces (ESP32 and Raspberry Pi)	2	MQTT and WebSocket communication
	LoRa SX1278	1	Long-range lowbandwidth communication
	PS2 Wireless Controller and Receiver	1	Manual and emergency control
Actuation Hardware	BTS7960 Motor Driver	1	H-bridge motor control
	12 V DC Geared Motors	2	Propulsion
Power Supply	7 Ah, 12 V Motor Batteries	10	Dedicated power for the motor driving system
	7 Ah, 12 V Control Batteries	3	Power for the ESP32, Raspberry Pi and sensors

Table 3.1: Hardware components and quantities used in Nicobot

3.3 Existing System Architecture of Nicobot

Figure 3.3 presents the consolidated architecture of Nicobot, showing the arrangement of software services, communication mechanisms, controller units and hardware components across four structured layers. The diagram provides a high-level representation of the current

system and illustrates how user interfaces, network services and physical devices operate together.

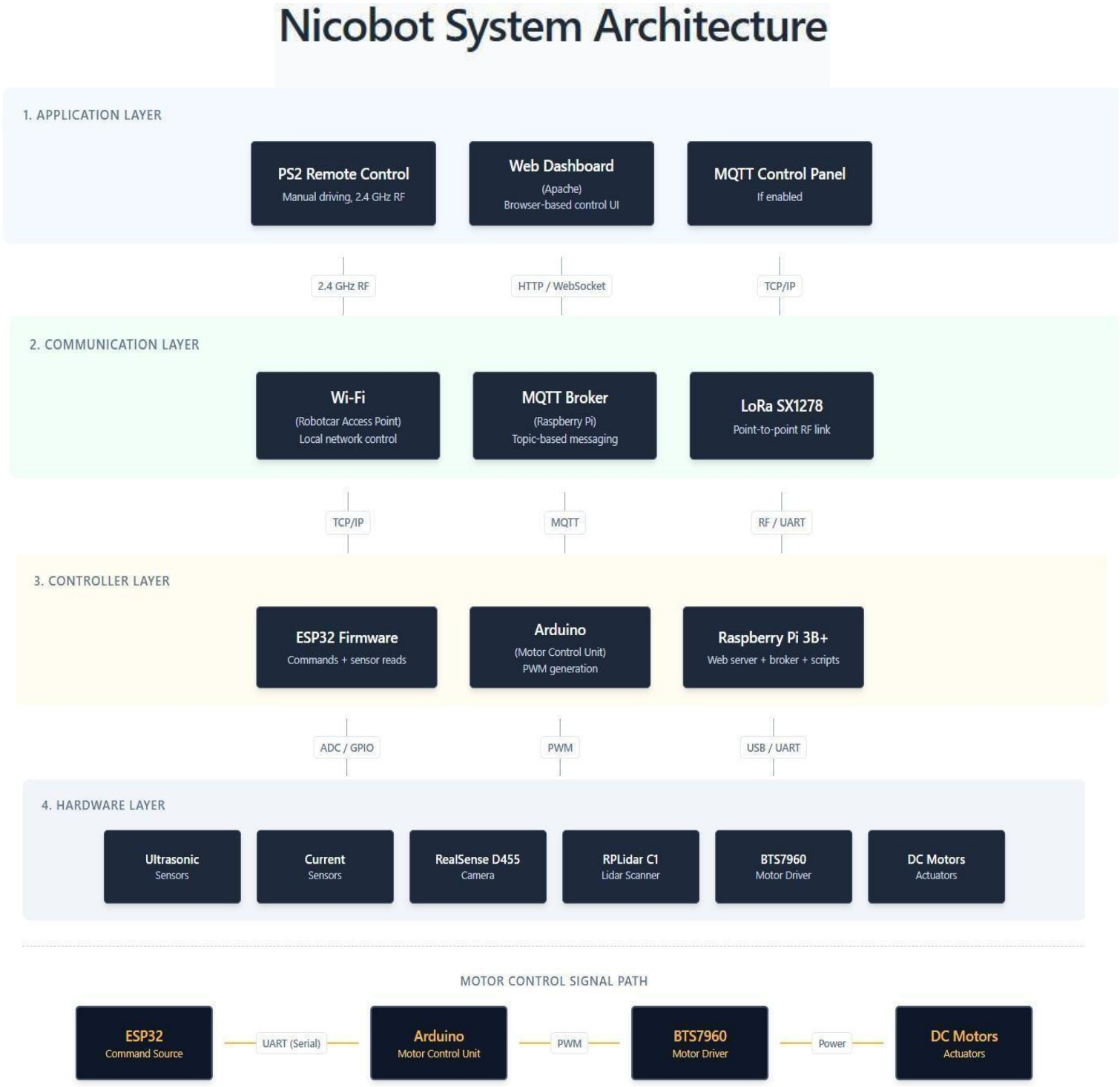


Figure 3.3: Layered Architecture of Nicobot

3.3.1 Application Layer

The application layer contains the user interfaces through which operational commands are issued to Nicobot. Three independent interfaces are available. The first interface is the PS2 remote-control unit, which uses a 2.4 GHz RF link to transmit manual driving inputs. This

interface provides a direct method for issuing movement commands without requiring network connectivity. The second interface is the web dashboard, delivered through the Apache server running on the Raspberry Pi. It is accessed through a standard browser and communicates using HTTP or WebSocket protocols. This interface allows the operator to issue movement commands, view system information and, when enabled, access camera feeds. The third interface is the MQTT control panel, which becomes operational when MQTT services are active. Commands are published by the operator to predefined MQTT topics and are retrieved by the relevant system components for execution. These interfaces operate independently, and the operator selects the desired interface depending on the operational context.

3.3.2 Communication Layer

The communication layer contains the technologies that support data exchange between user interfaces and controller devices. Three communication mechanisms form this layer. The first mechanism is Wi-Fi, provided by the Raspberry Pi configured as a local access point. This supports communication using TCP/IP and enables browser-based interaction through the web dashboard. The second mechanism is the MQTT broker, also hosted on the Raspberry Pi. This service allows structured, topic-based messaging between the controllers, laptops or mobile applications when MQTT is enabled. The third mechanism is LoRa communication, implemented through the SX1278 transceiver. This provides a long-range, low-bandwidth radio link suitable for situations where Wi-Fi connectivity is limited or unavailable.

3.3.3 Controller Layer

The controller layer contains the processing units that interpret commands, execute logic and generate control signals for the robot. Three controllers form this layer. The ESP32 acts as the real-time control processor. It collects sensor readings through ADC and GPIO interfaces, interprets incoming command data and prepares control outputs for the actuation subsystem. The Arduino board performs the role of a motor-control unit in earlier configurations. It receives serial input from the ESP32 and generates PWM signals required by the motor driver. Although this component is no longer the primary controller in later revisions, it remains part of the documented architecture for completeness. The Raspberry Pi 3B+ provides high-level service execution. It hosts the Apache web server, the MQTT broker (when enabled) and various operational scripts that facilitate system communication and user interaction. It does not perform real-time control but provides the supporting infrastructure required for networkbased operation. These controllers operate as separate processing units with specific responsibilities, reflecting the absence of a unified control framework in the current system.

3.3.4 Hardware Layer

The hardware layer contains the sensing, actuation and power components that form the physical foundation of Nicobot. The sensing devices include the ultrasonic modules, current sensors, voltage sensors, the Intel RealSense D455 depth camera and the RPLIDAR C1 scanning unit. These devices provide distance measurements, electrical parameters and environmental perception data to the controller units. The actuation subsystem consists of the BTS7960 highcurrent motor driver and the 12 V DC geared motors. Motor drive signals generated at the

controller layer are converted into regulated current output for the motors, enabling forward and reverse motion.The hardware layer also incorporates the power-supply system, which consists of isolated battery groups dedicated to the motors and the low-voltage electronics. This separation maintains electrical stability across the sensing and controller units.

3.4 Operational Workflow

The operational workflow of Nicobot begins when power is applied to the system and proceeds through a sequence of checks performed by the ESP32 and the Raspberry Pi. These checks determine the point at which the robot becomes ready for manual or network-based control. The workflow progresses through two stages: the initialisation stage and the controlready stage.

3.4.1 Stage 1: Initialisation and System Verification

When the system is powered on, the ESP32 and the Raspberry Pi begin their boot processes. The ESP32 activates its Wi-Fi interface and scans for nearby networks, while the Raspberry Pi initialises its access point and web server. The workflow then checks whether the ESP32 detects an available Wi-Fi network. If no network is found, the ESP32 indicates this through its LED status and must be reset manually. If the indicators remain inactive after reset, the Raspberry Pi is restarted to re-establish its services.

Once a network is detected, the ESP32 checks for the availability of the MQTT broker at the configured address. If the broker is not found, the ESP32 displays a corresponding LED pattern and prompts the operator to open its configuration page through the local web server. The operator can review and adjust the broker settings and then restart the ESP32 to repeat the detection cycle. If the settings are correct but the broker remains unavailable, the Raspberry Pi is restarted to restore the service. When the ESP32 successfully detects the MQTT broker, it enters a periodic scanning routine that checks broker availability at fixed intervals based on the system configuration. During this stage, the system is not fully ready for network-based control. The only active control method is the PS2 remote until stable synchronisation between the ESP32 and the Raspberry Pi is confirmed.

3.4.2 Stage 2: Control-Ready State

When the initialisation stages are completed and the ESP32 has established reliable communication with the Raspberry Pi, the system enters the control-ready state. The robot can then be controlled through the interfaces supported by the current configuration. These include the PS2 remote operating over a 2.4 GHz RF connection, the ESP32-hosted web server for direct command input, the Raspberry Pi web server using its Apache dashboard accessed through the local Wi-Fi network and a locally hosted control page running on a personal computer using Flask or Apache on the same network.

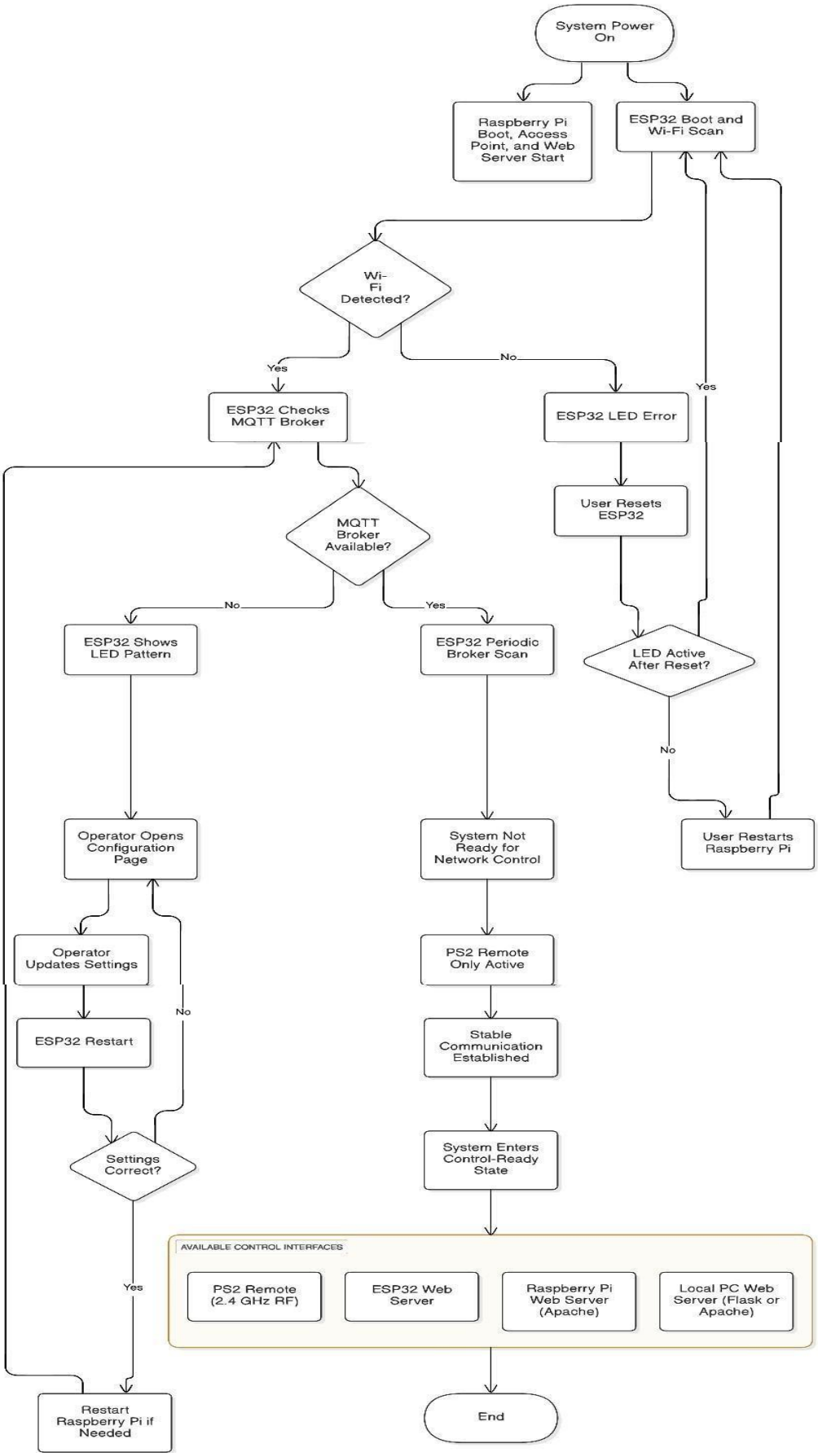


Figure3.4: *Operational Workflow of Nicobot*

3.5 Evaluation of the Current Architecture

The evaluation of Nicobot's current architecture shows a system that is functional for manual and supervised operation but constrained by structural fragmentation across the hardware, controller, communication and application layers. These constraints limit the robot's reliability, responsiveness and capacity for autonomous behaviour.

The hardware subsystems operate correctly in isolation, but they do not form an integrated sensing-and-actuation workflow. Sensors deliver raw, unprocessed measurements, and these data streams do not feed into a unified perception or decisionmaking process. The ESP32 handles time-critical inputs such as ultrasonic readings and voltage measurements, while the Raspberry Pi manages high-level services, yet no mechanism exists for synchronising or combining information across the controllers. This separation prevents the formation of closedloop control or adaptive behaviours.

The communication layer operates through independent channels—Wifi, MQTT and LoRa—that function correctly when individually enabled but do not share a coordinated communication strategy. There is no arbitration between channels, and message handling depends on manual activation of specific services. As a result, timing behaviour varies significantly, and the architecture cannot guarantee consistent message propagation when multiple services are active.

The application layer also exhibits several limitations. Each control interface—the PS2 remote, the ESP32-hosted web page, the Raspberry Pi dashboard and external Flaskbased control pages—operates independently without a supervisory mechanism to manage command precedence or prevent conflicts. The Raspberry Pi provides HTTP, WebSocket and MQTT services, but its processing capacity is limited when handling simultaneous tasks such as camera streaming, dashboard communication and broker operations. This contributes to delays, reduced responsiveness and inconsistent control behaviour.

The operational workflow reinforces these architectural limitations. Initialisation requires sequential verification steps by the ESP32 and Raspberry Pi, and successful progression depends on both devices being synchronised. The system cannot recover automatically from failures such as Wi-Fi unavailability or MQTT broker inactivity; instead, the workflow relies on manual resets or restarts. During the broker-detection stage, the architecture provides only partial control through the PS2 interface, indicating limited fault-tolerance and minimal support for degraded operational modes.

Across all layers, the absence of a unified control framework prevents Nicobot from performing data fusion, autonomous navigation or coordinated decision-making. The architecture supports experimental development and supervised operation, but it does not provide the structural features required for scalable sensing, real-time communication control or autonomous behaviour.

This architectural arrangement positions Nicobot within a supervisory autonomy model, where the human operator remains responsible for interpretation and intervention while the system supports controlled, semi-autonomous operation.

3.6 Proposed System Architecture

This section describes a proposed system configuration for Nicobot. The proposed configuration was developed through a structured review of the existing hardware configuration, system architecture and operational workflow described in Sections 3.1– 3.5. This review examined the current arrangement of sensing devices, processing units, communication modules, actuation hardware and power-supply components, together with the integration and layout constraints identified during system operation.

Based on this review, a design process was followed in which the existing hardware components were reorganised into a clearer physical layout, and additional supporting components were specified where required to address identified limitations in sensing placement, wiring separation, power distribution and environmental protection. The resulting configuration represents a design-based specification that documents intended component selection, physical positioning and electrical interconnection, rather than experimentally validated performance outcomes.

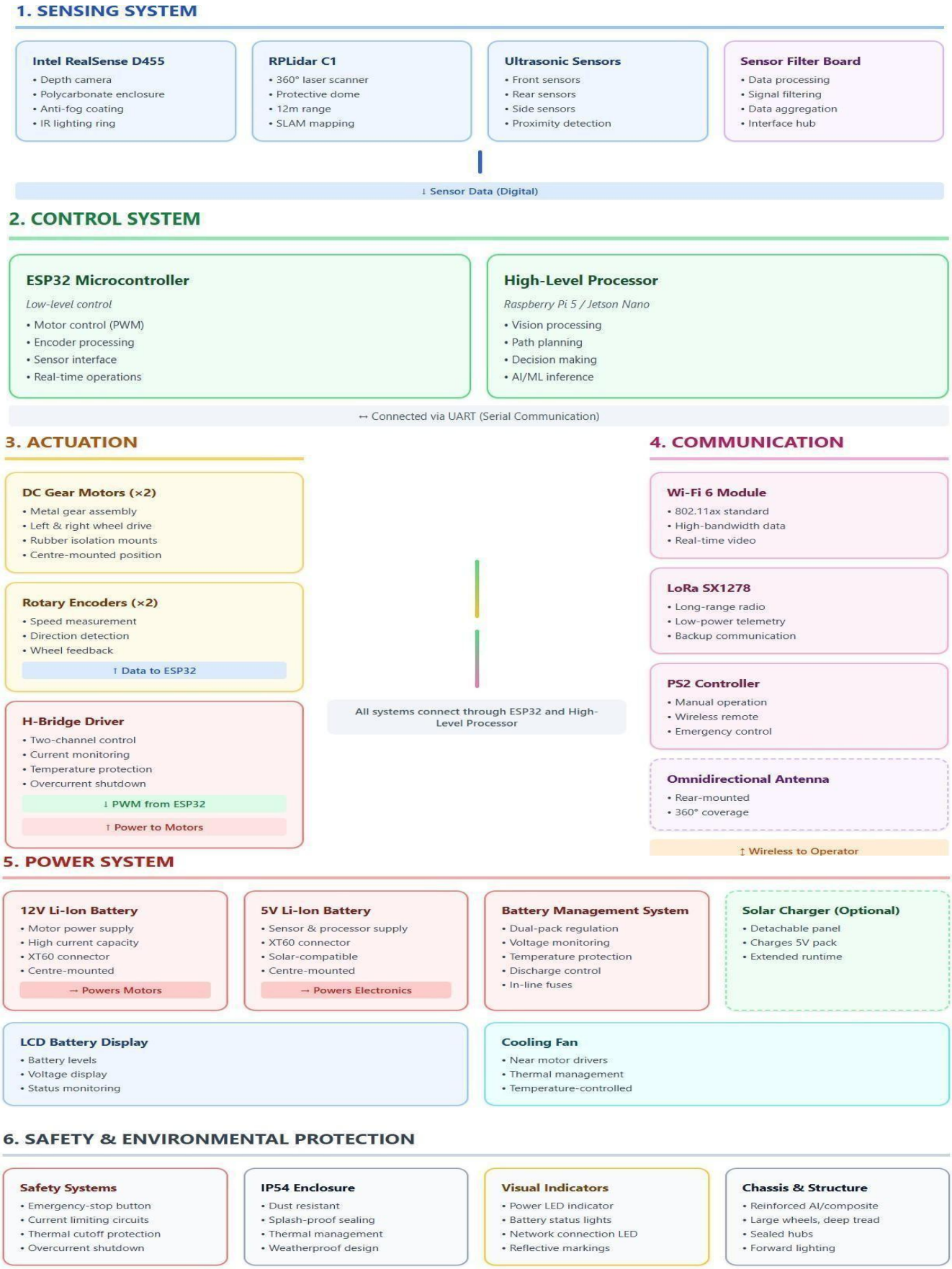


Figure 3.5: Block Diagram of the Proposed system Configuration of Nicobot

Figure 3.5 presents the proposed system configuration of Nicobot, organised into the sensing system, control system, actuation components, communication modules, power system and safety-and-environmental-protection features. The diagram sets out the relationships between the Intel RealSense D455, RPLidar C1, ultrasonic sensors, sensor-filter board, the ESP32 microcontroller, the high-level processor, motor drivers, rotary encoders, DC gear motors, dual-battery system, Battery Management System, WiFi 6 and LoRa SX1278 communication units, PS2 controller, antenna, and safety mechanisms such as the IP54 enclosure, emergency-stop interface and electrical protection elements. All components shown form part of a proposed design and have not been constructed or tested.

3.6.1 Actuation and Encoder Specification

The proposed actuation configuration lists two DC gear motors fitted with metal gear assemblies and mounted on rubber isolation points. Each motor shaft is fitted with a rotary encoder to provide wheel-speed and direction measurements to the ESP32. Motor voltage is supplied through a two-channel H-bridge motor driver that incorporates current monitoring, temperature monitoring and over-current shutdown. PWM control signals are generated by the ESP32. Encoder pulses are returned to the ESP32 for closedloop control using a Proportional–Integral–Derivative method. All motor wiring uses shielded connectors, and high-current lines are routed separately from low-voltage sensor lines using protective tubing.

3.6.2 Structural Frame and Chassis Layout

The proposed chassis is specified to be constructed from reinforced aluminium or a composite material. In this proposed layout, the motors and battery packs are positioned near the centre of the chassis to maintain a balanced centre of mass. Rubber isolation mounts are specified beneath the motors and sensor brackets to reduce vibration transfer to mounted components. The wheel assemblies are specified as large-diameter units with deep-tread profiles, and sealed hubs are included to limit dust ingress. The external surfaces of the chassis are specified with a protective coating to reduce exposure to moisture and environmental debris.

3.6.3 Sensing System Specification

The sensing configuration specifies an Intel RealSense D455 depth camera housed in a polycarbonate enclosure with an anti-fog coating and infrared lighting ring. An RPLidar C1 unit is mounted beneath a protective dome to reduce direct exposure to airborne material. Ultrasonic sensing modules are positioned at the front, rear and sides of the chassis for shortrange distance measurement. Sensor outputs are routed through a sensor-filter board that performs signal processing and aggregation before forwarding data to the ESP32. All sensor positions and housings are proposed elements and have not been implemented.

3.6.4 Communication Architecture

The proposed design specifies a wired UART connection between the ESP32 and the highlevel processor, listed as either a Raspberry Pi 5 or a Jetson Nano. Operator communication uses Wi-Fi 6 for short-range transmission and LoRa SX1278 for longrange communication. A rearmounted omnidirectional antenna is included for radio coverage. A PS2 controller is listed

for manual operation during testing. All communication elements form part of a design specification only.

3.6.5 Power Supply and Battery Management

The power layout includes a 12-volt lithium-ion battery pack supplying the drive motors and a 5-volt lithium-ion battery pack supplying sensors and processors. A Battery Management System regulates voltage levels, discharge thresholds and temperature limits for both packs and incorporates in-line fuses. Battery levels are displayed on an LCD module mounted on the top surface of the chassis. A cooling fan is positioned near the motor drivers for thermal management. The 5-volt pack is compatible with an optional detachable solar-charging module. XT60 connectors are listed for all battery and power-line connections. These elements represent proposed configurations only.

3.6.6 Processing and Computation Hardware

The proposed processing subsystem specifies either a Raspberry Pi 5 or an NVIDIA Jetson Nano as the high-level processor. These boards manage vision processing and detection algorithms. The processor assembly is mounted on rubber standoffs to reduce vibration transfer and is paired with a cooling fan to maintain an operating temperature within a specified range. The selection of processor and mounting method is part of the design specification only.

3.6.7 Safety and Environmental Protection

The safety configuration lists an IP54-rated enclosure for the electronics to provide resistance to dust and light rain. Water-resistant connectors are specified for external interfaces. An emergency-stop button is mounted on the upper chassis surface, and indicator lights display power, battery and network status. Additional elements include reflective surface markings and forward lighting for low-visibility conditions. Electrical protection is provided through fuses and current-limiting circuits. These features remain part of the proposed design and have not been installed.

3.6.8 Specifications of the Hardware Design

Component Category	**Component Name**	**Quantity**	**Purpose**
Actuation System	DC Gear Motors	2	Wheel actuation using metal gear assemblies
	Rotary Encoders	2	Measurement of wheel speed and direction

	H-Bridge Motor Driver	1	Two-channel motor voltage regulation with current and temperature monitoring
Control System	ESP32 Microcontroller	1	Low-level control, PWM generation, encoder and sensor interface
	High-Level Processor (Raspberry Pi 5 or Jetson Nano)	1	Processing of depth data, system logic and algorithm execution
Sensing System	Intel RealSense D455	1	Depth and RGB sensing; enclosed with anti-fog and IR illumination
	RPLidar C1	1	360° range measurements under protective dome
	Ultrasonic Sensors	4 (front, rear, sides)	Short-range proximity detection
	Sensor Filter Board	1	Data processing, filtering and aggregation from all sensors
Communication Modules	Wi-Fi 6 Module	1	Short-range wireless communication with operator
	LoRa SX1278 Module	1	Long-range wireless communication

	PS2 Controller	1	Manual control during testing
	Omnidirectional Antenna	1	Rear-mounted external radio interface
Power System	12 V Lithium-Ion Battery Pack	1	Motor power supply
	5 V Lithium-Ion Battery Pack	1	Sensor and processor power supply
	Battery Management System	1	Regulation of voltage, temperature and discharge for both battery packs
	Solar Charger (optional)	1	Field charging for 5 V pack
	Cooling Fan	1	Thermal management near motor drivers
	LCD Battery Display	1	Voltage and status indication for both packs
Safety & Protection	IP54 Electronics Enclosure	1	Dust- and splashresistant housing
	Emergency-Stop Button	1	Manual power-cut control
Chassis & Structural Components	Reinforced Aluminium/Composite Frame	1	Structural body of robot
	Large-Diameter Wheels	2	Traction across soil, grass and gravel

	Sealed Wheel Hubs	2	Dust-resistant wheel assembly

Table 3.2: Hardware Components in Nicobot, Including Quantities and Functional Roles

3.6.9 Comparison Between Existing and Proposed system of Nicobot

Table 3.3 summarises the key differences, followed by a justification of the improvements.

Subsystem	Existing Hardware	Proposed Hardware	Justification of Improvement
Processing Control	ESP32 + Raspberry Pi 3B+	ESP32 + Raspberry Pi 5 / Jetson Nano	The Raspberry Pi 5 or Jetson Nano provides significantly higher CPU and GPU capability than the Raspberry Pi 3B+, improving the vision-processing subsystem by enabling realtime depth processing and object detection. This also strengthens the system control subsystem by allowing higher-level decision logic to operate without affecting realtime motor control on the ESP32.
Actuation	DC motors without feedback; BTS7960 driver	DC gear motors with metal gears, rotary encoders and smart H-bridge driver	The inclusion of rotary encoders improves the motioncontrol subsystem by enabling closed-loop speed and position feedback. Metal gear assemblies improve drivetrain durability under load, while the smart Hbridge driver enhances the poweractuation subsystem through current limiting and thermal protection.

Frame Chassis	Basic aluminium frame; small wheels; no vibration isolation	Reinforced aluminium/composite frame; deep-tread large wheels; rubber isolation mounts; sealed hubs	The reinforced frame improves the structural subsystem by increasing rigidity during offroad operation. Larger deeptread wheels improve the locomotion subsystem through enhanced traction on soil and grass. Vibration isolation mounts and sealed hubs protect the sensor and electronics subsystems
			from mechanical stress and environmental ingress.
Sensing System	RealSense D455, RPLidar C1, two ultrasonic sensors	RealSense with IR lighting and enclosure, RPLidar with protective dome, four ultrasonic sensors with sensor filter board	IR illumination improves the vision subsystem by enabling depth sensing in low-light conditions. The protective enclosures improve the environmental resilience of sensing components. Increasing the number of ultrasonic sensors improves the proximity-detection subsystem, while the sensor filter board improves the signal-conditioning subsystem by reducing electrical noise.
Camera System	RealSense D455 + Logitech USB camera	RealSense D455 only (with IR ring); streaming handled by high-level processor	Consolidating vision sensing to a single RealSense camera improves the datahandling subsystem by reducing USB bandwidth contention. Offloading streaming to the high-level processor improves overall system stability by isolating vision processing from realtime control tasks.

Communication	Wi-Fi, LoRa, PS2 controller	Wi-Fi 6, LoRa SX1278, PS2 controller, rear omnidirectional antenna, wired UART between processors	Wi-Fi 6 improves the network communication subsystem through higher throughput and lower latency. The rear omnidirectional antenna improves radio coverage, while a wired UART link improves inter-processor communication reliability by reducing packet loss and timing variability.
Power System	Two groups of 7 Ah 12 V batteries (motor and control separated)	Dual lithium-ion packs (12 V and 5 V), BMS, XT60 connectors, fuses, optional solar charger	The BMS-managed lithiumion packs improve the powermanagement subsystem by enabling controlled charging, discharge and thermal monitoring. XT60 connectors
			and fuses improve the electrical safety subsystem, while the optional solar charger improves field endurance during extended deployment.
Thermal Management	No structured cooling	Active cooling fan near motor driver	The dedicated cooling fan improves the thermalmanagement subsystem by dissipating heat during highcurrent motor operation, reducing thermal stress and protecting power electronics from overheating.

Safety Mechanisms	Limited electrical protection; no emergency stop	IP54 enclosure, emergency-stop button, status indicators, reflective markings, protected connectors	The IP54 enclosure improves the environmental protection subsystem. The emergencystop button enhances the operator safety subsystem by enabling immediate power isolation. Status indicators and reflective markings improve operational awareness and visibility.
& Wiring Connectors	Direct wiring; limited separation of pow er and signal lines	Shielded connectors, separated power and signal routing, protective tubing	Shielded connectors and separated routing improve the electrical integrity subsystem by reducing electromagnetic interference. Protective tubing improves the maintenance and reliability subsystem by reducing cable wear and simplifying fault diagnosis.

Table 3.3: Comparison Between the Existing and Proposed system Configurations of Nicobot

Chapter 4: Evaluation and Redesign of Interface

4.1 Existing Interface

The current user interface of Nicobot is a web-based dashboard that provides manual and limited autonomous control of the robot via a browser. It was built as an early test design and not as a system to be used in the field at scale or for public deployment. The interface is developed using simple web components, which are hosted on the ESP32 web server and displayed in a desktop browser. It is structured in a linear manner, with all features organised in a top-to-bottom layout; therefore, it is easy to follow but visually dense. The design is more functional than aesthetic, indicating that it was developed primarily for engineering evaluation rather than for deployment in real-world field settings with non-technical end users.

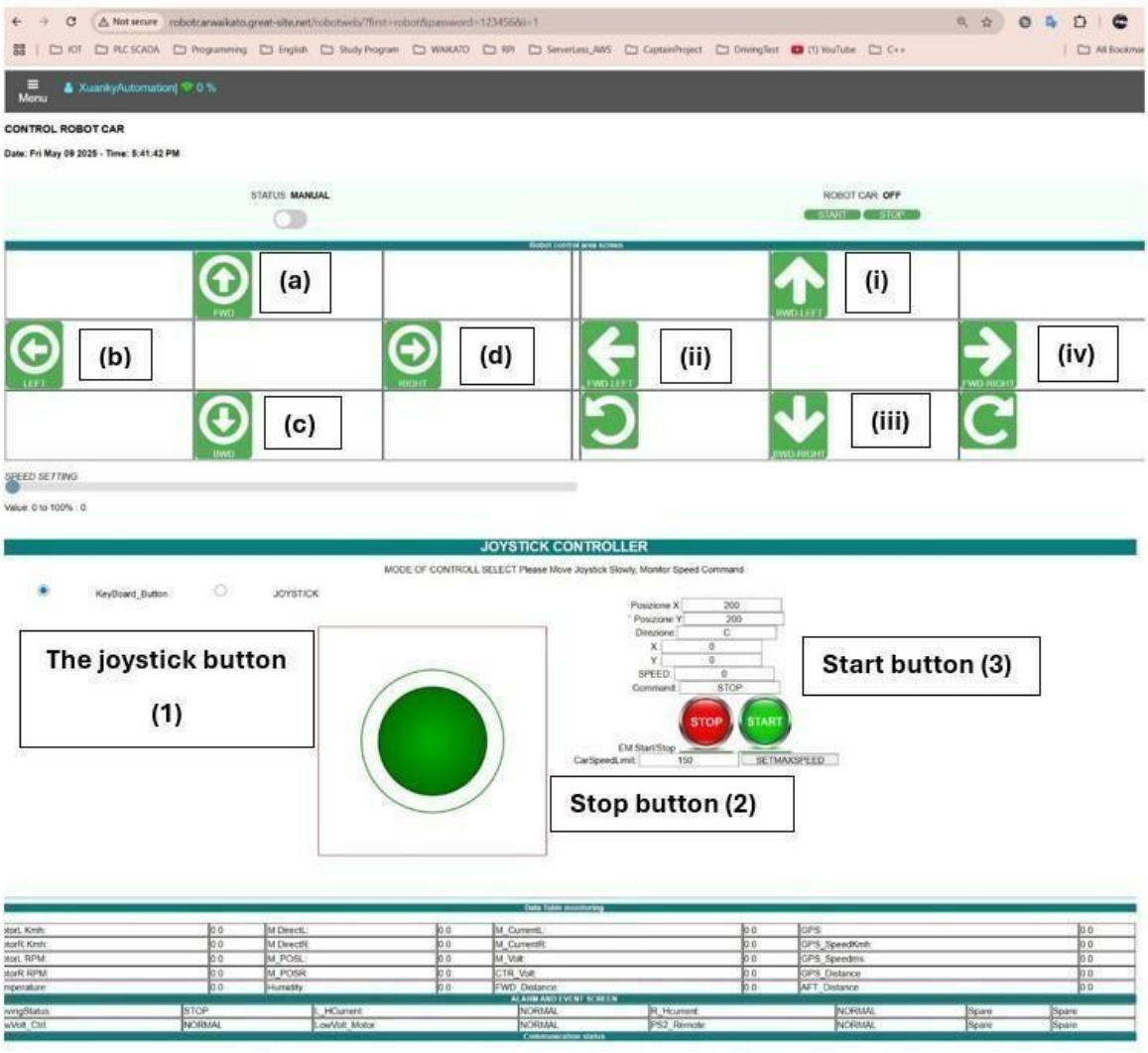


Figure 4.1: Web control interface for Nicobot 2025

Figure 4.1 presents the web-based control interface used to operate Nicobot. The interface is organised into two main sections: a directional control grid for issuing predefined movement commands, and a joystick control panel for proportional, userdriven movement. Each interactive element in the figure has been annotated with a label for ease of description, allowing its on-screen position to be matched directly with its corresponding function. Table 4.1 provides a detailed explanation of each labelled element in Figure 4.1, outlining its

appearance, placement within the interface, and the behaviour that occurs when it is activated. This ensures that the operational roles of all directional, diagonal, and joystickbased controls are accurately described in relation to their visual layout.

Label	Element Name	Description of Current Aspect	Function and Behaviour
(a)	**Forward Button**	A green upward-pointing arrow labelled **"FWD"** positioned in the uppercentre of the movement grid.	Sends an MQTT/WebSocket command instructing the ESP32 to drive both motors forward. Movement occurs only if safety conditions (ultrasonic, depth, and RPLidar checks) permit.
(b)	**Left Button**	A left-pointing green arrow labelled **"LEFT"** displayed on the left side of the grid.	Commands the ESP32 to perform a leftward rotation or lateral movement by reducing rightmotor speed and increasing leftmotor speed. Behaviour depends on realtime communication stability.
(c)	**Backward Button**	A downward green arrow labelled **"BWD"** located in the lower centre of the movement grid.	Sends a backward-movement instruction to the ESP32 using MQTT/WebSocket. Safety sensors (rear ultrasonic and RPLidar) inhibit motion if an object is detected within the set threshold.
(d)	**Right Button**	A right-pointing green arrow labelled **"RIGHT"** shown on the centralright area of the movement grid.	Commands the ESP32 to rotate or move right by adjusting motor speeds.
(i)	**Backward-Left Button**	A green diagonal arrow pointing down-left, shown beneath the forward-left button.	Commands a backward-left diagonal movement. This is achieved through reduced right-motor PWM and reversed motion. Safety sensors prevent motion if an obstacle is detected behind the robot.

(ii)	**Forward-Left Button**	A green diagonal arrow pointing up-left, positioned in the upperright region of the movement grid.	Sends a diagonal forward-left movement command. The ESP32 adjusts motor speeds so the robot moves forward while veering left.
(iii)	**Backward-Right Button**	A green diagonal arrow pointing down-right, located in the lower-right of the movement grid.	Sends a backward-right diagonal command. The ESP32 applies asymmetric PWM to move backward while pivoting right. No visual indicator reflects the robot's movement progress.
(iv)	**Forward-Right Button**	A green diagonal arrow pointing up-right, positioned on the farright of the top row.	Moves the robot diagonally forward-right. This is used for cornering and path adjustment. The movement executes only when communication is stable, and sensors detect a safe path.
(1)	**Joystick Control**	A large green circular joystick element located below the grid inside a square boundary.	Provides proportional control using drag input. Direction and speed values are transmitted continuously to the ESP32.
(2)	**Stop Button**	A red button labelled **"STOP"** positioned next to the joystick area.	Immediately halts the robot by sending a stop command to ESP32. Safety override is instantaneous and overrides any active movement command.
(3)	**Start Button**	A green button labelled **"START"** next to the stop button.	Enables motion by arming the ESP32 control loop. Required before executing movement commands.

Table 4.1: Existing Nicobot Interface Functional Aspects.

4.1.1 General Layout

The interface is arranged as a series of sections presented in a vertical layout. The upper region has the navigation and status buttons and then has a grid of arrow buttons which enable the robot to move in various directions. Beneath this control section is a speed setting bar and below this joystick controller section which has a large circular joystick pad. Below are the tables of data that include sensor readings, system, and communication status. A green horizontal line is used to separate each section and the background colour used is white, Green

and black are used to highlight text and buttons. The colours are selected strictly on the basis of functionality, namely, the green colour signifies mobility or active control, and the grey one is the inactive areas.

4.1.2 Top Bar and Status Area

The upper section of the interface has a menu icon and a label on the header that is called CONTROL Robot car. The name of the interface reflects the primary function of the application, which is the control of the robot. The page has the date and time just below the title, such as the one in the example shown below: Date: Fri May 09 2025 Time: 5:41:42 PM This implies that the system records the live operating session, which is practical for testing and logging, but offers limited interactive value to the operator, although it may still provide reassurance that live data is being received.

Beside the time display, a line shows the current control mode, labelled 'STATUS: MANUAL', with a small toggle switch positioned next to it. The switch can be switched to manual or autonomous mode so that the user can choose whether the robot is being controlled directly by the operator or is on pre-programmed behaviour. Nonetheless, the visual feedback is missing except the word MANUAL and then the user is unable to know whether the system has received the command or is still processing the request. In the right side of the same line, two rectangular buttons marked as START and STOP can be seen, and a label of the robot car is displayed, namely ROBOT CAR: OFF. This displays the activity and power of the robot. The STOP button is red and the START button is green giving the user a simple yet efficient colour variation in the safety functions.

4.1.3 Directional Control Grid

The main section of the upper section is a 3x3 grid of big green arrow buttons. These buttons enable the operator to move the robot in various directions, forward, backward, left, right and four diagonal directions. The buttons are squares with a prominent green arrow symbol in its centre. The labels of the direction include FWD, BWD, LEFT, RIGHT, FWD-LEFT, FWD-RIGHT, BWD-Left and BWD-right, which appear under or on the sides of the arrows. The centre of the grid contains a blank area with a small text label, but it does not provide any functional feedback in the current interface.

All of the buttons in this section are programmed to perform directional movement. Pressing the forward arrow makes the robot move straight ahead, while pressing the diagonal arrows causes it to move forward or backward while turning. The design is symmetrical, logically organized, i.e. the direction of every button corresponds to the physical movement generated by it. Nevertheless, no visual or auditory response is presented when a button is pressed in the interface, which means that users are forced to watch the robot move in response to the command. The arrows are big enough to be easily clicked but are not animated thus it is hard to determine whether a command has been successfully submitted or is being submitted.

4.1.4 Speed Setting Bar

Just under the movement control grid, there is a thin horizontal slider with the name SPEED SETTING written on it. Below the label there is a display of values with the value of 0.0 and 100

percent at 0. This is to show that the slider is used to adjust the speed of the robot. The control can also be dragged in the bar itself so that the operator can alter the speed between zero and maximum setting. Nevertheless, it gives no quantitative data or graphical scale indications that may be used to determine the speed accurately. The existing environment must be assumed based on the position of the bar. Since the adjustment in speed is not associated with the visual image of the movement, the operator can hardly predict the level of the change of the speed before the visualisation of the outcome.

4.1.5 Joystick Controller Section

The second large section of the interface is the joystick controller with the text Joystick Controller written in uppercase text at the top of the panel. There is a short instruction line, which says, "MODE OF CONTROLL SELECT Please Move Joystick Slowly, Monitor Speed Command”. The spelling and grammar in this section indicate that this part was not created to be used by the general users but to be tested internally.

Under this text are two radio boxes which are marked Keyboard Button and JOYSTICK. The latter enables the operator to decide whether to operate the robot with the keyboard or the virtual joystick. In case of a selection of JOYSTICK, a big round pad is displayed at the left side of the section. There is a green stippled circle within the circle which symbolizes the joystick handle. The following aspect of the interface is expected to appear as a virtual joystick in the screen. The user is able to constantly vary direction and speed by dragging the green circle inside the pad instead of having to use discrete arrow keys.

There are a number of small boxes with rectangular shapes, each showing numerical information to the right of the joystick pad. These are "Posizione X”, "Posizione Y”, “Direzione X, Y”, “SPEED” and “COMMAND”. These fields display the live values of the joystick position and the calculated direction because of the input of the user. Beneath these text boxes, there are two circular buttons apparent beside each other. One of them is green with the word "START" on it and the other is red with the word "STOP" on it. The text EM Start/Stop on them hints that these are emergency control buttons where the robot can be activated or stopped very quickly. There is a small label "CarSpeedLimit" and an input box next to which displays the number 150 then a button that is denoted as SETMAXSPEED. This probably enables the user to set a maximum permissible speed to be applied to ensure safety.

Despite the fact that this section is more flexible than the arrow grid, it nevertheless relies on manual numerical input and did not have any visual representation as arrows or lines that represent motion. Numeric values have to be read by the operator to comprehend the direction of movement which makes it less intuitive.

4.1.6 Data Table Surveillance Section

The bottom half of the page has a large rectangular space with the label "Data Table monitoring" in it. There are numerous items in numerical form and variables that are tabularly presented in this section. Each column contains a specific type of measurement or sensor reading. The upper two columns present real-time data relating to the robot’s velocity and direction, which are labelled as “M. DirectL” and “M. DirectR”. The following columns include the current readings “M. CurrentL” and “M. CurrentR”, which represent the amount of electric

current drawn by the left and right motors. Other columns are GPS-related information like "GPS Speed Km/h", “GPSSpeed km” and “GPS Distance”.

A number of status readings like RunningStatus, STOP, M. Current and LowVoltMotor are also present in this section. They give diagnostic feedback in regard to troubleshooting.

This table is organised in such a way that it is oriented to the use of engineers as opposed to the common users. The data is technical and not described or colour coded. An example of this would be M. Current that reports numerical values of current, but nothing is mentioned as regards to what should be considered normal and abnormal. The values are not continually updated but rather updated manually or by the microcontroller.

4.1.7 Alarm and Event Display Area

Under the table of data, there is a small band which is labelled as ALARM and event screen. In this region, there are elementary system notifications. The common messages that are displayed are the words normal, stop, or h. Current. Such short text messages are related to the condition of the robot at that point in time. Nevertheless, the messages are in plain text with no colour contrast and icons. As a result, warning messages are not visually distinguished from normal system messages, requiring the operator to read each message carefully to identify possible issues. This may lead to slow reaction to mistakes in a working environment where the operator may be busy monitoring the motion of the robot within the field.

4.1.8 Communication States Section

There is a tiny field on the very bottom of the interface that is marked with the name Communication states. This part demonstrates the connection status of various subsystems like the PS2 remote controller, motor connection and the network communication links. The entries have either NORMAL or SPARE entries that seem to be placeholders that show whether the module is operational or not. It does not have graphic indicators, just like other sections. A little coloured light or symbol might have clarified it when the connection fails but the design currently used relies solely on text. This simplicity suggests that the interface has been developed primarily to ensure that communication is confirmed in the case of bench testing.

4.1.9 Interaction and Response Behaviour

Each button, when pressed by the user, sends a command to the robot over Wi-Fi through the ESP32 web server. It is a sequential process and thus the interface will wait to receive a response before taking a new command. This brings about short delays between a press of a control and a robot motion. The absence of the progress bars and confirmation indicators allows user to press the buttons multiple times believing that the initial press was not counted. This may result in several overlapping commands. The joystick field reports the position fields gradually and at times freezes when the network traffic becomes more active demonstrating a lack of processing power of the ESP32. The lag is more pronounced when the number of running sensors is large.

4.1.10 Integration with Robot Hardware

Each item indicated in the interface is associated with a hardware component of Nicobot. The directional buttons are associated with motor commands that are fed on the ESP32 microcontroller and transmitted to the BTS7960 motor driver modules. The joystick coordinates are also sent as digital signal to control the same motors. The SPEED SETTING slider varies the amplitude of PWM signal which has a direct effect on the speed of the wheel. The data table contains sensor measurements of the hardware components used, namely the RealSense camera, RPLidar, ultrasonic sensor and GPS module. The values are sent back to the interface by serial communication and shown in the table.

Various sections have the START and STOP buttons which engage the motor control logic. It has a safety feature where when the microcontroller is pressed at a STOP it will cut the power of the motor instantly. Equally, the state of communication shows real-time connection of PS2- or Wi-Fi-modules. But the microcontroller alone has to manage the entire functions such as web hosting, data processing, and sensor communication thus slowing down the interface when all the functions are running.

4.2 Heuristic Assessment of Existing Interface

The current Nicobot control interface was analysed using Nielsen's ten usability heuristics for user interface design (Nielsen, 1994). These heuristics offer an orderly scheme of strengths and weaknesses in an interactive system and aim at such things as visibility, consistency, user control, feedback and error prevention. The assessment was performed via the direct observations and the practical interaction with the web interface as it was observed in a desktop browser and simulating the conditions of a real user. The outcome of the heuristic analysis is summarised in Table 4.2, and each heuristic is discussed in relation to the redesigned interface in the sections that follow.

Heuristic	**Positives**	**Negatives**
1. Visibility of System Status	Robot status (Manual/Off) visible. Speed slider and numeric values displayed. Real-time telemetry shown (speed, distance, current, RPM).	Mostly text/numbers; not intuitive for nontechnical users. No visual feedback when commands executed. Joystick movements not reflected in real time.
2. Match Between Robot and Human Understanding	Arrow buttons familiar. Labels like "FWD" and "BWD" are clear.	Technical terms (EM Start/Stop, M. Current, PS2 Remote) confusing. Raw sensor data table not user-friendly.
3. User Control and Freedom	Start/Stop buttons clear. Joystick and keyboard supported. Emergency stop exists.	Emergency stop not visually prominent. No undo/back option for commands.

4. Consistency and Standards	Navigation arrows follow conventional meanings. Colour scheme consistent (green = go, red = stop).	Inconsistent terminology: START/STOP vs EM Start/Stop. Mixed abbreviations and full words (FWD vs FORWARD).
5. Error Prevention and Recovery	Stop button allows halting. Speed limit prevents unsafe operation.	No confirmation for risky commands. Connection drop not handled.
6. Recognition Rather than Recall	Arrow buttons reduce memory load. Joystick shows position.	Sensor table requires technical knowledge. Robot mode/status small and easy to miss.
7. Flexibility and Efficiency of Use	Keyboard and joystick both supported. Speed adjustment available.	No shortcuts for experienced users. Interface cluttered for beginners.
8. Safety, Trust, and Transparency	Emergency stop exists. Speed limit enhances safety.	No live camera feed is provided, safetycritical alerts are not visually prominent, and speed controls are unclear.
9. Aesthetics and Minimalism	Clean background with large buttons.	The interface contains excessive text, with spelling errors and inconsistent sizing and styling.
10. Help, Documentation, and Learning Support	Joystick instruction line available.	No online help, guidance, or tutorials.

Table 4.2: Summary of heuristic analysis

4.2.1 Visibility of System Status

Existing interface partially provides visibility of system status by using textual cues like "STATUS: MANUAL" and "ROBOT CAR: OFF" message. The fact that a speed slider and the realtime data values of such parameters as current, RPM, and GPS readings are included also contribute to conveying internal states. These outputs, however, are shown more or less as raw numbers, which have very little meaning to non-technical users. No moving visuals like coloured lights, icons and motion indicators to indicate active commands. A user is not given any visual feedback when pressing a directional button or when controlling a joystick other than what the robot moves. Such lack of feedback decreases confidence and predisposes repetition of command. The interface could be more transparent and reassuring by making the design more visual, i.e. the animated status indicators or progress bars.

4.2.2 Match Between Robot and Human Understanding

The system partially uses real-life conventions. The movement arrow buttons are also selfexplanatory, and the colour of the green suggests movement that is forward or active. Yet, there are numerous other components that are using technical terminologies that are not known by the simple users. The labels like EM Start / Stop, M. Current and PS2 Remote, are engineering words and not operator words. The huge data table in the bottom does not provide any context and interpretation of parameters such as voltage, current and humidity. This causes users to memorise technical definitions and not identify simple states. In one case, such as LowVolt_Motor could point to low battery but this is not visually represented. The redesigned interface should replace these words with simple language and easy-tounderstand icons that show information such as battery charge, GPS signal, or motor temperature.

4.2.3 User Control and Freedom

Basic controls including "START", "STOP" and emergency stop feature are available in the interface which will enable users to have some form of freedom when it comes to controlling the start and stop of the robot. The joystick is provided as well as the keyboard, and it provides various control options. Nonetheless, the emergency stop button is not a dominant visual element which may easily be merged with other elements. Undo and neutral button do not exist to undo the last movement and so there is no chance of quickly reversing a mistaken direction command without pressing the stop button. This restricts user freedom when carrying out fine adjustments. To help in this area, it would be beneficial to make the emergency stop button large with a high contrast in colour and addition of a rapid quick return to neutral functionality that will safely move the robot to a stationary point.

4.2.4 Consistency and Standards

Directional controls are in the form of universally recognisable arrow shapes and colours, and they follow primitive visual standards. However, gaps are found in the naming conventions and in the formatting of the text. The interface switches between the abbreviations (FWD, BWD) and full words (START, STOP) and combines the lower- and upper-case fonts. It also provides hybrid words like EM Start/Stop and simple STOP which can be confusing to the operator. Lack of standardised labels enhances lack of predictability. It would be helpful to create consistent terminology, use regular capitalisation, and work with general interface conventions to create a better understanding and professionalism.

4.2.5 Error Prevention and Recovery

The speed-limit setting and the stop button help to avoid the accidental damage since they prevent the robot from moving too fast. The system, however, does not provide warnings before performing potentially dangerous commands like the rebooting or reset of the robot. In case there is a failure in communication between the interface and the microcontroller no explicit error message is shown. Connection drop may result in confusion on the part of the user or unsafe behaviour of the robot due to lack of feedback. The introduction of confirmation dialogues in essential actions and the disconnection warning would be visible would go a long way towards enhancing reliability and avoiding unintentional actions.

4.2.6 Recognition Rather than Recall

The large directional arrows are quite effective to facilitate recognition: the users can guess the direction of the movement without the need to recall complicated instructions. Otherwise, the rest of the interface is primarily based on recall. The large numerical data tables at the bottom require the operator to remember what each parameter represents and how it relates to the robot's performance. Also, the mode indicator
(MANUAL/AUTONOMOUS) is tiny and is easily missed. The interface must employ graphical displays in the form of gauges, coloured bars and icons to indicate the conditions such as battery charge or the connection status and this will ensure less cognitive load to the users.

4.2.7 Flexibility and Efficiency of Use

There is a certain degree of flexibility provided by the interface by using joystick and keyboard inputs, and speed control. However, it does not have adjustable settings in regard to the user expertise level. Even expert users are subjected to the same manual processes as beginners and a new user is presented with a cluttered display of information. The wide range of users would be served by providing interface modes with one basic mode, used by essential driving, and the other advanced mode, used by technical monitoring. Key shortcuts that can be customised would also help in providing better control of the experts when testing or in the field.

4.2.8 Safety, Trust and Transparency

The safety precautions are available in the form of emergency stop button and variable speed limits; the robot is able to be stopped in a short time. Nonetheless, the interface fails to ensure user confidence as some of the most important data, including the strength of connections, reliability of the sensor, or observation of obstacles, are not explicitly shown. It does not have an internal camera and live feedback to verify the environment of the robot. There is also a case where operators only use text data, hence diminishing situational awareness. Addition of real-time video feeds, distance detectors or visualisation of barriers would enhance security and make the operators confident when operating remotely.

4.2.9 Aesthetics and Minimalism

The visual design of the interface is plain and easy to use and has a white backdrop and action buttons of green colour. This brings a certain primitive order. Nevertheless, there is too much information on the interface and several redundant features like start/stop buttons. The crowded data tables create a visual clutter to the user and flood them with information that is hardly ever necessary in practice. The layout would be cleaner with fewer parameters showing and there would be a great separation between control and feedback areas to enhance readability and reduce distraction. The minimalistic design principles, application of space, regular alignment, and the use of soft colour contrast would help in making the interface easier to use.

4.2.10 Help, Documentation and Learning Support

The interface only has one line of instruction above the joystick region, which is, " Please move joystick slowly, monitor speed command". There are no tooltips, contextual cues or online

assistance. This lack of documentation can serve to make the system challenging to learn to a user with limited technical experience. The system is much more user friendly with the addition of a help button opening a small guide, tooltips on hover on every icon, or even links to a small web-based tutorial.

4.3 Proposed Interface

A redesigned interface was developed following the heuristic analysis presented in Table 4.2, with the aim of improving usability, enhancing real-time responsiveness, and providing greater clarity than the previous version. The new design adopts a well-defined and minimalistic layout in which the dashboard is divided into three main sections: the system status display on the left, the operational control area in the centre, and the sensor and safety data display on the right. Each section provides real-time feedback and visual indicators that allow the user to understand the robot's state at a glance. The web layer is implemented using HTML, CSS, python and JavaScript, and communicates with the embedded controller via WebSocket technology to ensure real-time interaction.

The redesigned interface is presented in Figures 4.2–4.10. The revised design contains multiple pages and interactive components, which are described in the sections that follow.

4.3.1 Visibility of System Status

The key dashboard shows all crucial data concerning the state of the robot in a clear format. Figure 4.2 depicts the system status panel on the left side of the interface, and it displays the name of the robot, version of the firmware, and the mode it is operating in. The connectivity section shows live connection conditions of MQTT, LoRa, and PS2 controllers and each has a real-time latency in milliseconds and a green or orange colour mark to indicate stable or slow communication. The control and motor system battery levels are displayed as horizontal bars that indicate a percentage to enable one to easily identify the power level. The robot has a live GPS location panel that shows the position of the robot on an inbuilt map. All these attributes combined provide the operator with real-time knowledge of the status of the robot without manually checking status.

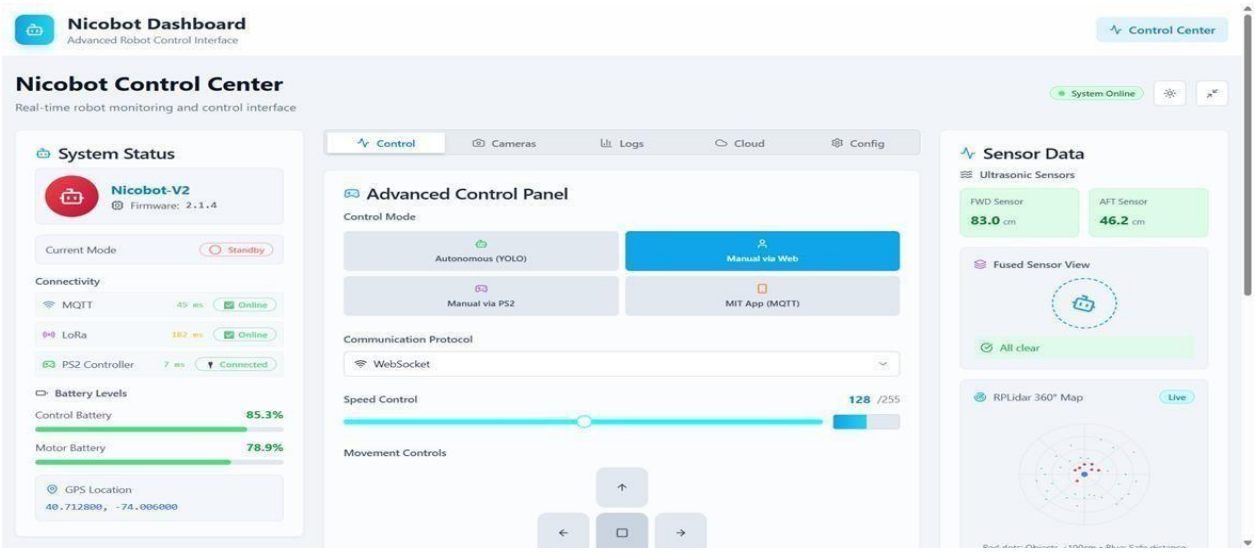


Figure 4.2: The main dashboard of Nicobot Control Centre showing the state of the system and connection.

4.3.2 Match Between Robot and Human Understanding

The redesigned interface improves the alignment between the robot's functions and the user's mental model by presenting information in clear, descriptive language rather than technical abbreviations. As shown in Figure 4.2, the control mode options—Autonomous (YOLO), Manual via Web, Manual via PS2 and MIT App (MQTT)—are displayed using full descriptive labels. This makes the robot's behaviour easier to interpret, especially for users without technical expertise.

In addition, the interface employs intuitive visual icons to represent connectivity, communication protocols, battery levels, and operating modes. For example, connection status for MQTT, LoRa, and the PS2 controller is shown using coloured status indicators and latency values, allowing users to understand system behaviour at a glance. Sensor data is displayed using colour-coded blocks, graphical fused-view icons, and map-based visualisation in the RPLidar panel.

These visual cues reduce the cognitive load placed on users, as they no longer need to recall technical terminology or interpret complex numerical tables. Instead, they can rely on recognisable symbols, labels, and colour coding to understand the robot's state, encouraging a more intuitive and natural style of interaction.

4.3.3 User Control and Freedom

Figure 4.3 illustrates how the user control has been enhanced by making its elements sensible and receptive. Directional movement buttons in the central control section are formed in a basic arrow design and are backed up by an optional joystick having real time coordinate feedback. The design allows users to steer the direction of the robot with ease either on the keyboard or on-screen joystick. Below the control panel are keyboard shortcuts that show commands like W, A, S, D to move and Q to stop and E to emergency stop. The bottom of the dashboard presents a broad red emergency stop bar that is easily visible at all times, and this ensures that the user is able to immediately shut down the robot at any given moment. These safety measures ensure that manual control is safer, and the operator will have confidence when controlling the robot in the field.

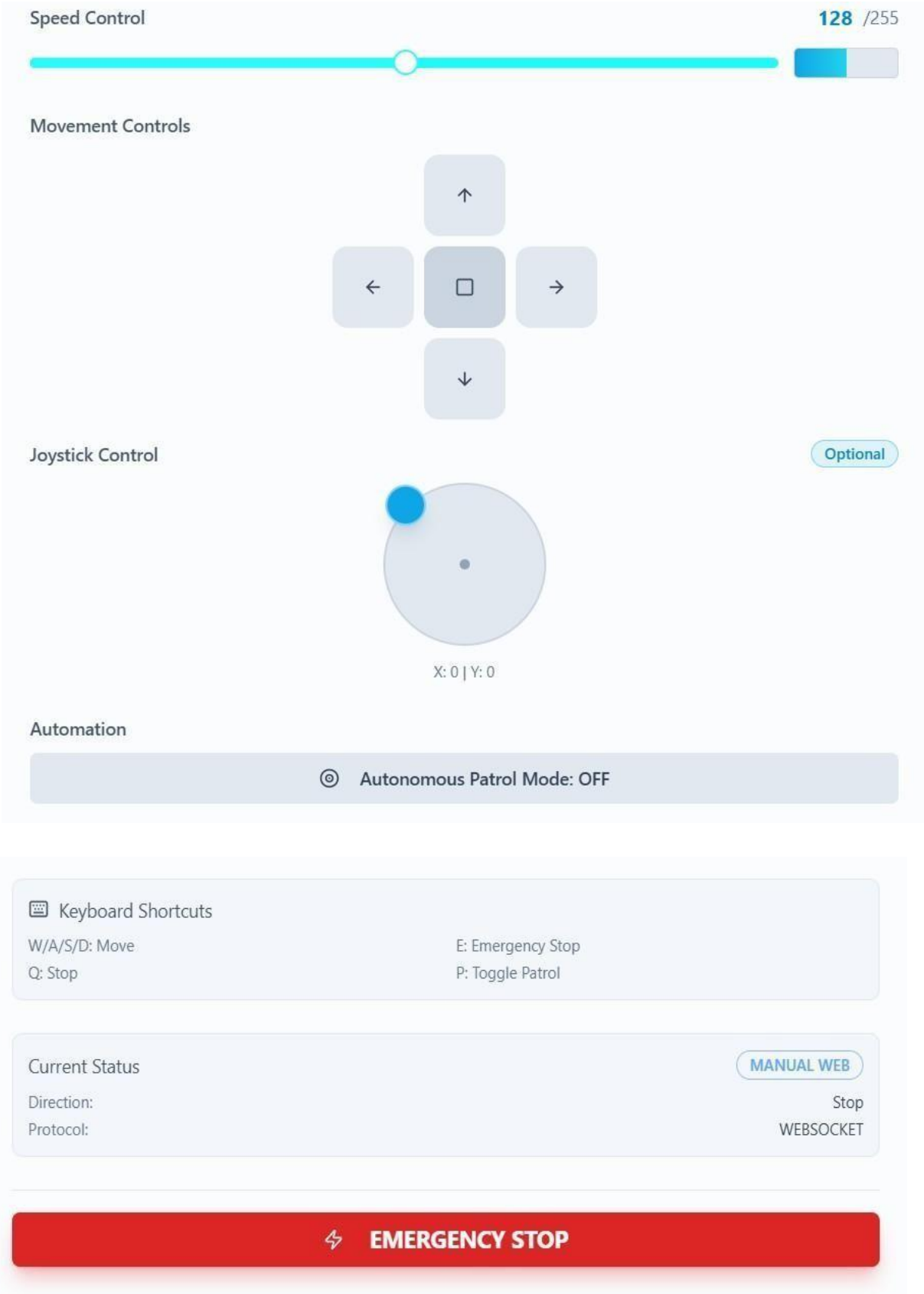


Figure 4.3: *Movement and joystick control area with emergency stop and shortcuts.*

4.3.4 Consistency and Standards

The new interface maintains a consistent layout and visual style across all functional tabs. As Figure 4.4 illustrates, the major navigation bar has five common links, namely Control, Cameras, Logs, Cloud and Config. All the tabs have the same layout principles, typography, and spacing, which gives a feeling of continuity when changing features. The buttons, labels and icons are unified in terms of blue and white theme, with gentle edges and even spacing. The harmonisation of the contents on all screens strengthens conformity as the user can move freely without having to reorient oneself on either side.

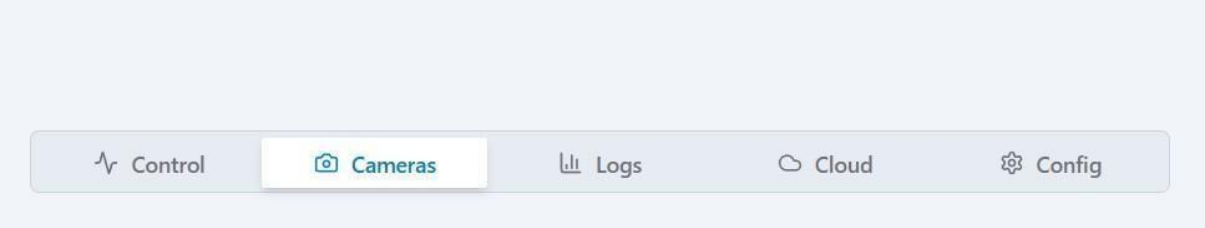


Figure 4.4: *Tab control, camera, logs, cloud, and configuration control is always guided by a tab.*

4.3.5 Recovery and Prevention of Error

The re-engineered system provides better prevention of errors provided with warnings, confirmation, and a log of recovery information. Figure 4.5 depicts the System Logs tab in which each significant action like startup or turning on the camera or a low-battery warning is accompanied by a time and a category of the logs. The information, warnings and communication events are differentiated by colour coding in each entry. Under the logs, there is the Performance Charts which is used to visualise the speed of the robot, its power consumption, and network latency over time, thereby assisting the user to identify abnormal performance trends. These real-time logs and visual patterns help to detect faults in time and minimize the possibility of mistakes made by the users.

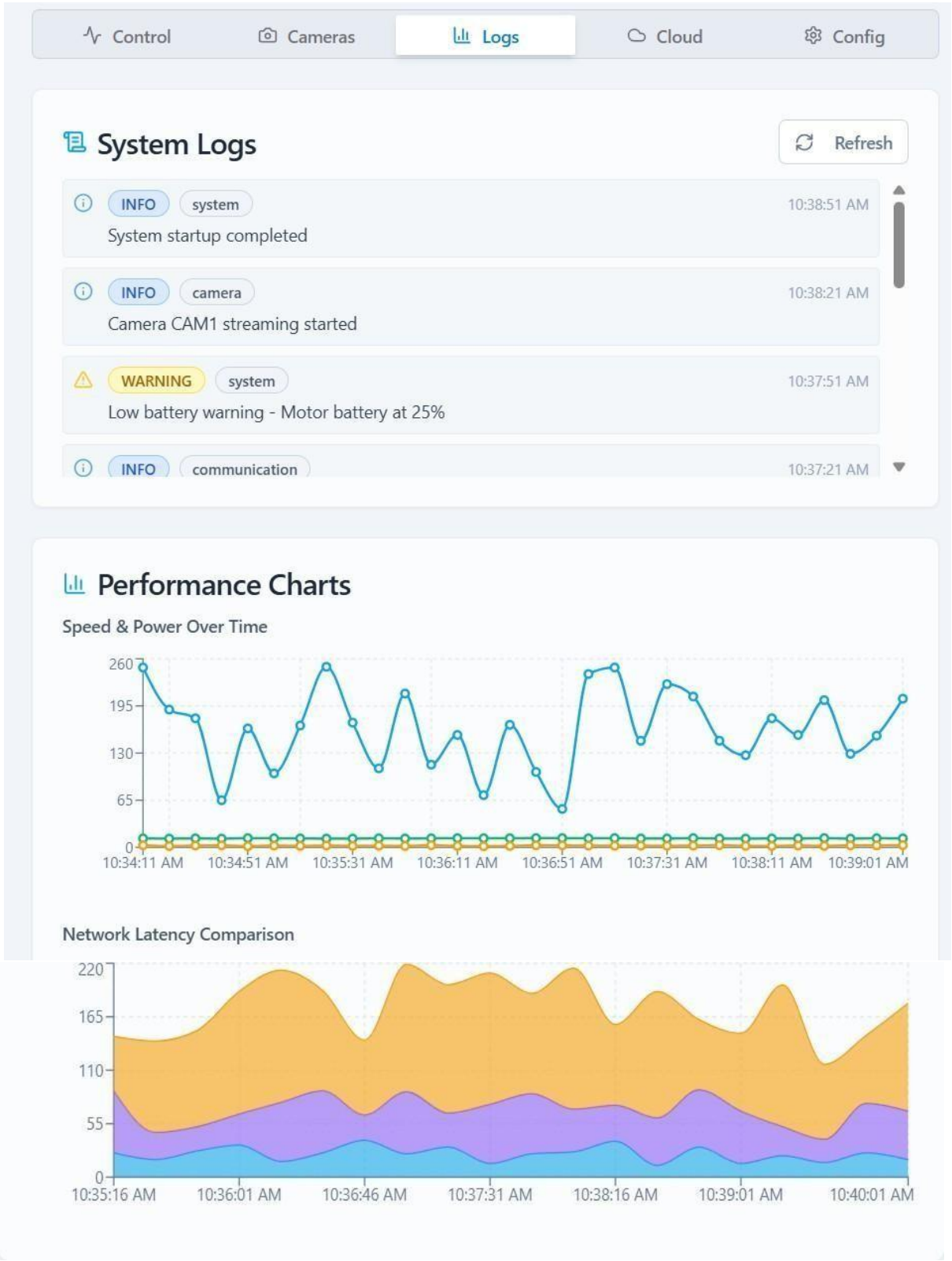


Figure 4.5: System logs and performance charts of error and warning alerts.

4.3.6 Recognition Rather than Recall

The redesigned Nicobot interface integrates sensor data to give a clear view of the robot's immediate environment, with its visual data being easily read (user do not have to memorise

the numerical readings). Figure 4.6 illustrates the Sensor Data panel which shows the live readings of the front and rear ultrasonic sensors in coloured boxes thus enabling the users to identify the distance values immediately. Under this, the Fused Sensor View gives a straightforward All Clear status so one can easily know the environmental safety by looking at it.

These readings can be supplemented with the help of the RPLidar 360deg Map that demonstrates the real-time object detection and distance mapping. Obstacles in one metre are indicated by red dots, whereas safe distances are indicated by blue dots and allow users to quickly and accurately understand the surrounding of the robot. Through this visual method, the operators can comprehend spatial information easily and applications of such variable are in making of quick and effective decisions and enhanced situational awareness in both manual and autonomous navigation.

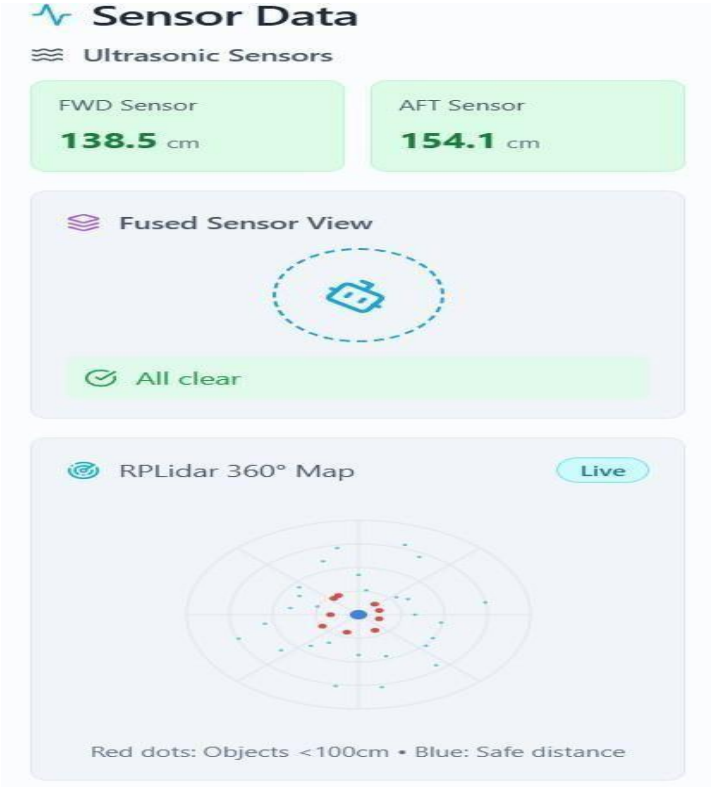


Figure 4.6: Object detection and distance mapping Fused Sensor and RPLidar 360deg view.

4.3.7 Flexibility and Efficiency of use.

The system is able to accommodate simple and sophisticated control requirements and has flexibility among various users. The interface of the Fig 4.7 is known as the Configuration Settings, which enables the user to change the settings of the maximum speed, motor trim, safety distance limits and camera settings. Toggle switches can be used to activate or deactivate sensors rapidly, whereas the sliders can be used to control the sensitivity of the sensors fine. The page also has Save Configuration button and resets to defaults option which give the user an opportunity to test safely without making permanent changes. This design allows novices to use the default settings of the robot and those who are advanced to undertake finer adjustments in sensors, speed and protection limits where needed.

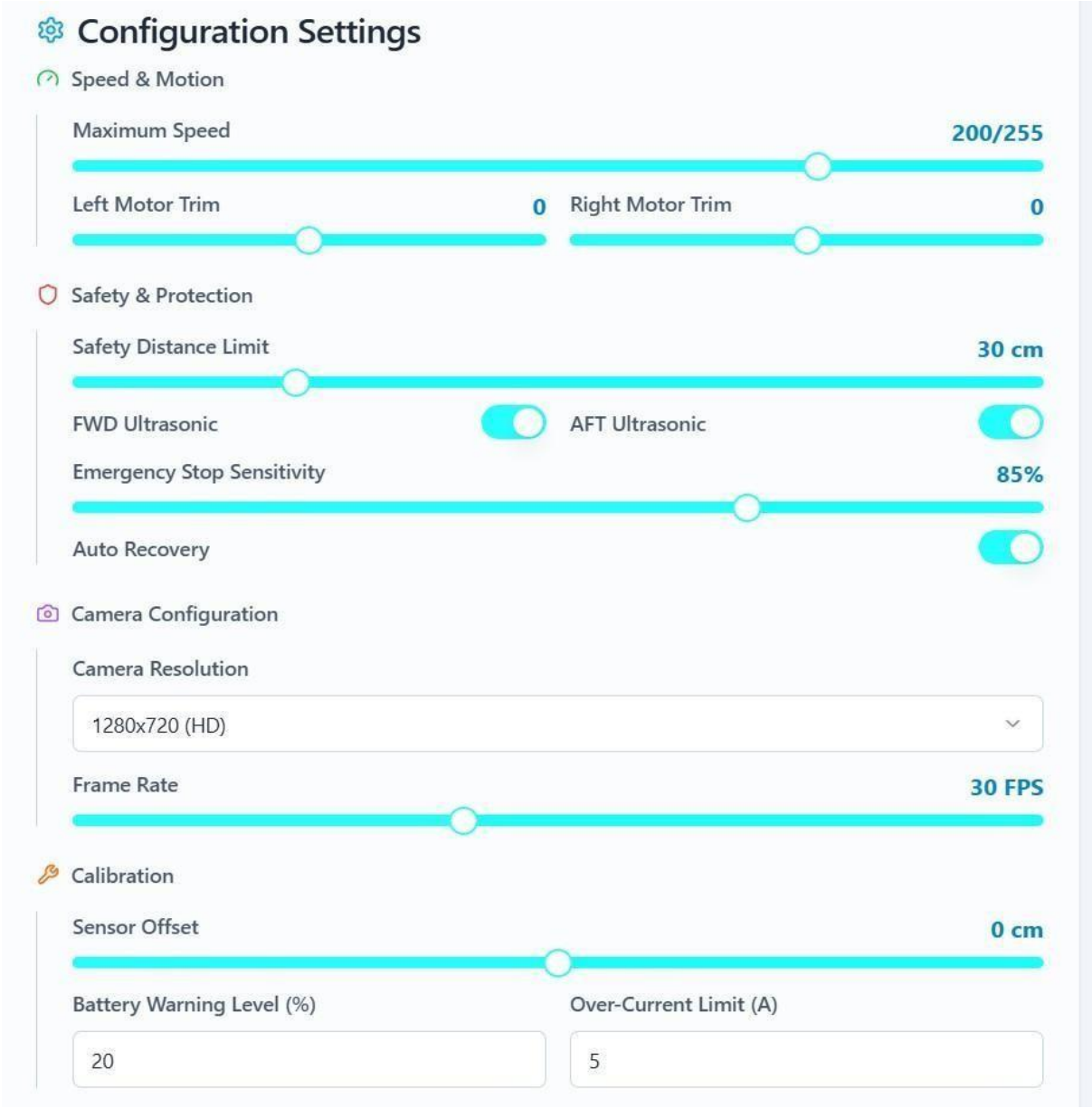


Figure 4.7: Configure page with settings of safety and motion parameters that are adjustable.

4.3.8 Safety, Trust, and Transparency

The interface has enhanced safety and transparency with built in live visual feedback. The camera's tab, which is presented in Figures 4.8 and 4.9 shows video feeds of various cameras including the Intel RealSense D455 and a Logitech USB camera. The system brings out identified objects in form of bounding boxes and labels that display distance and confidence of detection. In case an object or a person is detected, the detection log will be presented beneath the video feed with information like distance and accuracy. Voltage and Current Sensors panel provide the live control voltage and motor voltage simultaneously, whereas the Safety Status indicator indicates that distance-based warnings are on. All these features contribute to the increased trust of the operators as every process is seen in real time.

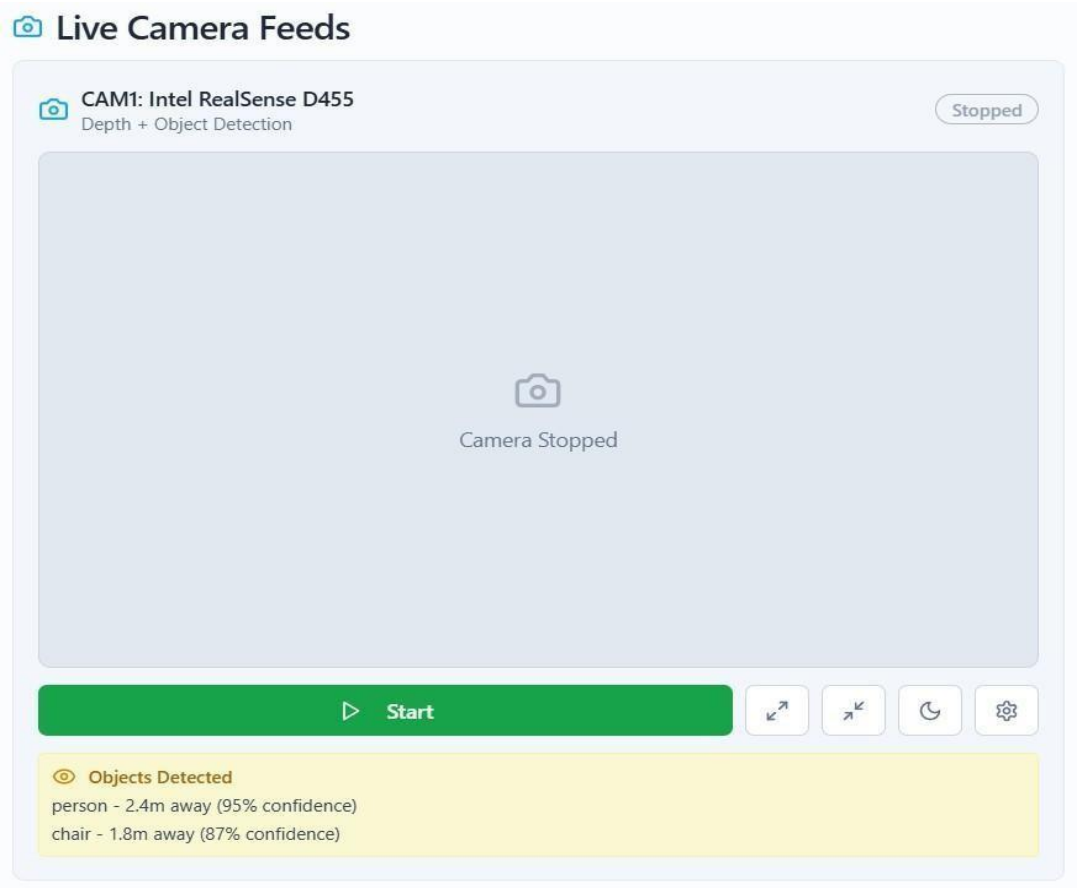


Figure 4.8: camera feed of Intel RealSense D455

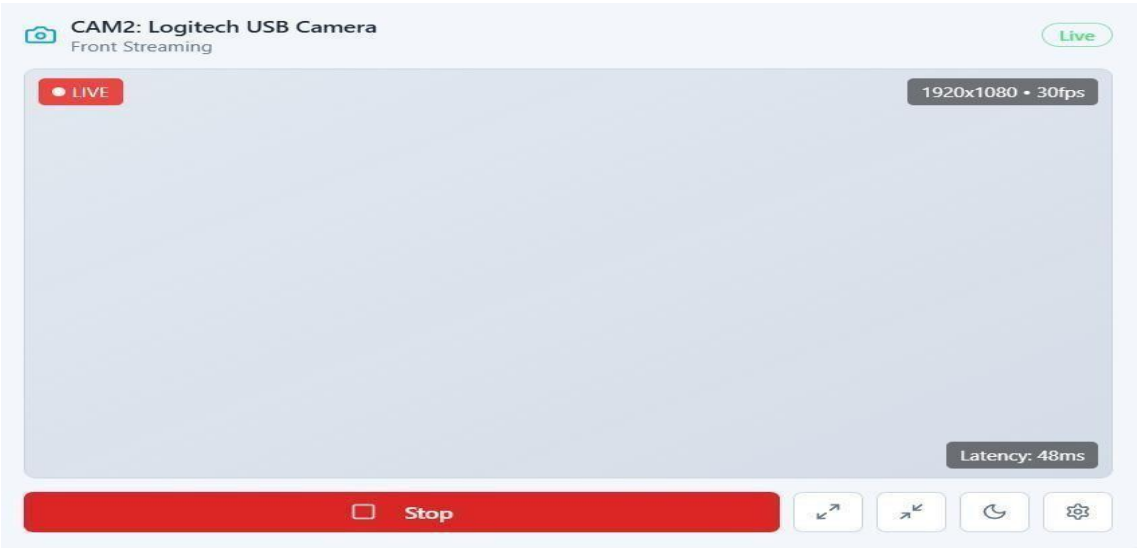


Figure 4.9: *camera view of Logitech USB camera*

4.3.9 Aesthetics and Minimalism

The interface is clean and minimal across any of the screens, avoiding visual saturation. The backdrop is light grey and white panels which avoid distraction of data. The sense of depth and order is made by the use of shadows, rounded corners, and equal margins. Every functional element including charts or sliders is placed rationally to minimize the number of eye movements. Complex technical data, such as network latency and calibration offsets, are

stored within expandable menus, while only essential information is displayed by default.The overall design adopts a clean layout intended to reduce visual clutter and support ease of use.

4.3.10 Help, Documentation and Learning Support

The Nicobot dashboard has been redesigned to support the user in becoming more autonomous, by including inherent support to use the system. There are interactive tooltips, such as when one hovers the cursor over a variety of icons that provide brief explanations about the controls like triggering audible alert and enabling voice control. This makes sure that even a novice user can learn all features in no time without having to consult any outside source.

Figure 4.10 presents the *Integrations and Extensibility* section of the interface, which displays the robot's optional connections to cloud platforms such as ThingsBoard and AWS IoT Core. Connection status is shown through simple coloured indicators—green for active and red for inactive—giving users immediate visual feedback on whether each cloud service is available.

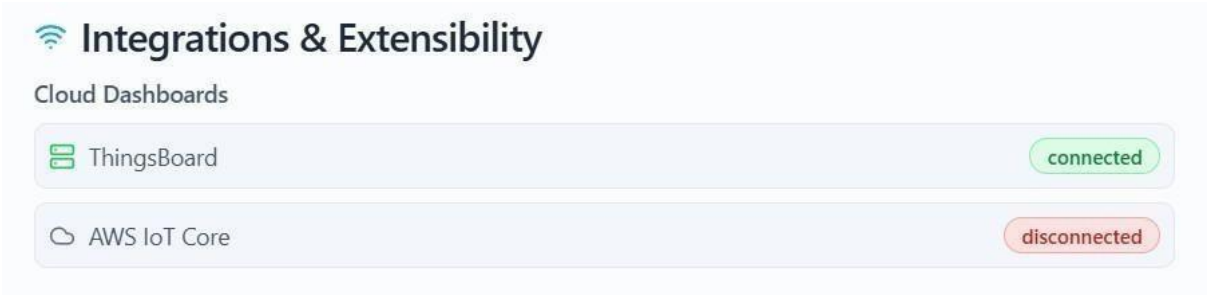


Figure 4.10: Integrations and Extensibility section with the support of cloud connectivity

4.4 Comparison

The original Nicobot interface and the redesigned interface differ primarily in structure, organisation, and presentation of system information. The original interface provides access to all controls and data within a single vertically arranged web page hosted on the ESP32. While this design allows all functions to be accessed from one view, it results in a dense layout with limited separation between operational controls, monitoring information, and safety elements.

The redesigned interface adopts a structured layout that separates system status, movement controls, and sensor-related information into distinct regions. Navigation is also expanded through the introduction of tab-based pages for control, camera monitoring, logs, configuration, and cloud connectivity. This organisation improves clarity and reduces the volume of information presented within any single view.

In the original interface, system state is communicated mainly through text labels and numeric values. In contrast, the redesigned interface displays this information using structured panels, indicators, and graphical elements, enabling operating modes, communication status, and battery levels to be identified more easily. Sensor information is also presented using visual display elements rather than relying only on numerical tables.

Control behaviour differs between the two designs. The original interface relies on directional buttons and joystick input with limited on-screen feedback following user actions. The

redesigned interface provides more structured input controls, keyboard shortcuts, and clearer visual cues. The emergency stop control is also made persistently visible.

Alert handling is minimal in the original interface, where abnormal conditions must be identified by reading numerical values or text messages. In the redesigned interface, alerts are presented through structured logs and visual indicators, improving the visibility of operational states. Configuration management is also extended through a dedicated settings page.

Overall, the redesigned interface presents a more organised and visually structured environment than the original design. The main differences between both interfaces are summarised in Table 4.3.

Aspect	**Original Interface**	**Redesigned Interface**
Deployment Platform	Web interface hosted on the ESP32 web server and accessed through a desktop browser.	Web-based interface implemented using HTML, CSS, Python, and JavaScript with WebSocket communication.
Overall Layout	All controls, sensor data, and status information displayed on a single vertically structured page.	Divided into three main regions: system status, control area, and sensor/safety information.
Navigation Structure	Single-page interface with no separation into functional screens.	Multiple interface pages accessed via a tab-based navigation bar (Control, Cameras, Logs, Cloud, Config).
System Status Display	Displayed using text labels such as *STATUS: MANUAL* and numeric fields.	System state presented using structured panels, indicators, and graphical elements.
Movement Controls	Operated using a directional grid and joystick without visual feedback.	Directional controls supported by onscreen feedback and clearer visual cues.
Speed Control	Slider without numeric scale or visual reference indicators.	Speed controls integrated into configuration settings with structured control layout.

Emergency Stop	Implemented as a standard button near joystick controls.	Persistent emergency stop bar visible during operation.
User Input Options	On-screen buttons and joystick only.	Keyboard shortcuts and joystick supported alongside on-screen controls.
Data Presentation	Sensor values displayed primarily as numeric tables.	Sensor data presented visually through panels, indicators, and graphical display elements.
Alert Handling	Messages displayed as plain text without distortion indicators.	Alerts presented through structured logs and visual system indicators.
Communication Status	Displayed using text-only fields marked as *NORMAL* or *SPARE*.	Connectivity status shown using indicators and labelled sections.
Aesthetic Design	Functional layout with dense text and limited formatting consistency.	Layout uses consistent styling, structured spacing, and reduced visual clutter.
Help and Guidance	Limited to one instruction line near joystick controls.	On-screen tooltips and interface explanations provided.
Cloud Integration Visibility	No cloud-related display.	Interface includes cloud connection sections with status indicators.
Configuration Management	Manual parameter entry via input fields.	Dedicated configuration screen for safety limits, motor trim, and sensor sensitivity.

Table 4.3: Comparison Between Original and Redesigned Nicobot Interface

Chapter 5: Discussion

5.1 Scope of the Discussion

This chapter presents a reflective discussion of the Nicobot project, focusing on insights gained from architectural analysis, user interface evaluation, and the examination of communication design choices. The discussion is structured around the research questions introduced in Chapter 1 and considers how the analysed system design responds to these questions in the context of agricultural robotic systems intended for rural and low-connectivity environments.

5.2 Discussion in Relation to the Research Questions

Discussion of Research Question 1: How does the system architecture of Nicobot support autonomous operation in remote agricultural settings?

In its current form, the Nicobot system architecture provides very limited support for autonomous operation in remote agricultural settings. Although the system includes many of the components that would normally be associated with autonomy, such as multiple sensors, processing units and communication methods, these elements are not used in a way that allows the robot to operate independently. Instead, Nicobot relies heavily on manual control, manual supervision and manual recovery, which significantly restricts its suitability for remote or unattended agricultural use.

A key finding from this study is that autonomy within Nicobot is not achieved through automated decision-making, but rather through direct human involvement at almost every stage of operation. Movement commands, mode selection, speed adjustment, fault handling and system recovery are all initiated by the operator. Even basic actions such as switching between control modes or responding to communication failures require human intervention. As a result, Nicobot cannot currently be described as autonomous in practice, particularly in environments where connectivity is unreliable and constant human oversight is impractical.

In remote agricultural settings, autonomous operation typically implies that a robot can continue functioning safely and predictably without continuous input from a user. This includes the ability to cope with poor network conditions, recover from minor faults, and make basic operational decisions based on sensor data.

The reliance on manual handling is also evident in the way sensor information is used. Although Nicobot collects data from a range of sensors, this information is primarily presented to the user rather than being used by the system itself to guide behaviour. The robot does not autonomously interpret sensor readings to adjust its movement, avoid obstacles, or modify its operation based on environmental conditions. Instead, the operator is expected to observe sensor values, interpret their meaning, and decide how the robot should respond. This places a significant cognitive burden on the user and limits the robot's ability to function independently in the field.

Overall, the findings indicate that Nicobot should currently be viewed as a manually operated and supervised research platform rather than an autonomous agricultural robot. While it can function in remote environments to a limited extent, this operation depends heavily on human control and intervention. The system architecture does not yet support autonomous decisionmaking, automated fault handling or independent operation under low-connectivity

conditions. These limitations directly affect the robot's suitability for real-world agricultural deployment.

Discussion of Research Question 2: How does the design of the Nicobot user interface support usability for non-technical users in agricultural contexts? The current Nicobot user interface offers limited usability support for non-technical users operating in agricultural environments. Although the interface enables manual control and monitoring of the robot through a web browser, its design reflects an engineering-focused development approach rather than the needs of everyday agricultural operators. As a result, the interface requires users to understand technical information, interpret raw data, and continuously monitor the system, which reduces ease of use and increases the likelihood of confusion or error. A key issue is how information is displayed. The interface presents a large amount of numerical data related to sensors, motors, GPS, electrical current, and communication states. These values are shown as raw numbers and technical labels, with no explanation of what the values mean or whether they are within safe limits. For non-technical users, such information does not clearly indicate whether the robot is operating normally or whether a problem exists. Instead of providing clear answers, the interface requires users to interpret data that they may not understand, which increases mental effort and reduces confidence during operation. System status information is only partially clear. Labels such as operating mode and robot power state are displayed, but they do not provide confirmation that user commands have been successfully received or executed. When a user presses a movement button or moves the joystick, there is no visual confirmation on the screen to indicate that the command has been processed. Users must rely on observing the physical movement of the robot to know whether an action has occurred. In agricultural environments, where the robot may be at a distance or partially obscured by terrain or crops, this lack of feedback makes operation uncertain and stressful.

The control elements themselves are logically arranged but lack responsiveness. Directional buttons and joystick controls are visually clear and follow familiar conventions, which initially makes them easy to understand. However, when these controls are used, the interface does not respond visually. Buttons do not change state, animations are absent, and no indicators show that a command is in progress. When network delays occur, users may believe that their input was ignored and may press controls repeatedly. This behaviour can lead to unexpected robot movement once communication resumes, creating safety risks.

The language used throughout the interface also limits usability. Many labels rely on abbreviations and technical terms that are not self-explanatory. Terms related to electrical current, communication modules, or control modes assume prior technical knowledge. For non-technical users, these labels do not communicate meaning clearly. Instead of recognising system conditions instantly, users are forced to remember what each term represents or guess its meaning. This reliance on memory rather than recognition makes the interface harder to learn and increases the chance of misunderstanding.

Overall, the current Nicobot user interface supports usability mainly for technically experienced users who are familiar with robotic systems and development environments. For non-technical agricultural users, the interface requires constant attention, interpretation of technical data, and manual confirmation of system behaviour. While the interface allows

manual control of the robot, it does not sufficiently reduce cognitive effort or support intuitive interaction. This limits its practicality for realworld agricultural use, where simplicity, clarity, and confidence are essential.

Discussion of Research Question 3: How are communication mechanisms incorporated within the Nicobot architecture to support operation in lowconnectivity environments?

Communication is a major challenge for Nicobot when operating in rural and agricultural environments, where network connectivity is often unreliable, intermittent, or unavailable. Although the system includes multiple communication mechanisms, these do not provide reliable or seamless support for operation under low-connectivity conditions. Instead, they expose significant weaknesses that limit effective use of the robot in real-world agricultural settings.

Rather than relying on a single communication channel, Nicobot provides several independent communication options. While this reduces the risk of total disconnection, it does not guarantee consistent operation. The communication mechanisms are not integrated or coordinated, and their effectiveness depends heavily on manual oversight. As a result, maintaining communication becomes the responsibility of the operator rather than an inherent capability of the system.

The primary communication method relies on Wifi to support browser-based control and message exchange between the robot and the operator. This approach assumes the availability of a stable local network, which is rarely the case in agricultural environments. Signal degradation caused by distance, terrain, vegetation, and environmental interference frequently disrupts communication. When connectivity deteriorates, control becomes delayed or unresponsive, making this method unsuitable for dependable operation in remote farming areas.

When Wi-Fi connectivity is unavailable, Nicobot relies on LoRa as a long-range communication option. LoRa allows basic commands and limited status information to be transmitted over longer distances. It introduces severe limitations. Data rates are extremely low, feedback is minimal, and continuous monitoring is not possible. The operator receives only limited information about the robot's state, which makes it difficult to assess whether commands have been executed correctly or whether the system is operating safely.

As a result, long-range communication functions only as a minimal fallback rather than a practical operating mode. It allows basic commands or emergency intervention, but it does not support sustained control or effective supervision. When Nicobot relies on this channel, overall functionality is significantly reduced, increasing uncertainty and operational risk.

In addition to network-based communication, Nicobot also includes a PS2 wireless controller that bypasses network infrastructure entirely. This method is intended for close-range intervention and recovery rather than normal operation. While it provides a safety fallback, it requires the operator to remain physically close to the robot. In agricultural contexts where

robots are expected to operate across large areas, this requirement is impractical and further highlights the system's dependence on human presence.

Although these communication mechanisms form a layered arrangement, this layering does not result in reliable operation. The system does not automatically adapt to changing connectivity conditions. When connectivity degrades, the robot does not switch communication modes on its own or adjust its behaviour accordingly. Instead, the operator must recognise communication failures and manually select an alternative method. This increases cognitive load and creates opportunities for error, particularly for non-technical users.

Overall, while Nicobot includes multiple communication mechanisms, these do not adequately support reliable operation in low-connectivity agricultural settings. The system remains highly sensitive to communication disruption and depends heavily on manual supervision to manage connectivity issues. The presence of multiple communication channels reduces the chance of complete loss of control, but it does not provide a robust or userfriendly solution.

5.3 Integrated Discussion Across the Research Questions

Looking across the three research questions together makes it clear that the main issue with Nicobot is not a single missing feature, but the way responsibility is distributed across the system. Autonomy, usability, and communication are closely linked, and in the current design they collectively assume that a human operator is always present, attentive, and able to intervene.

When considered as a whole, the system design places decision-making outside the robot. Architectural choices do not allow the system to act independently, the interface expects the user to interpret system behaviour, and communication mechanisms exist mainly to keep the operator involved rather than to support independent operation. This combination shapes how Nicobot is used and explains why the system struggles in environments where connectivity is poor and continuous supervision is difficult.

The three research questions also reveal that limitations in one area directly increase pressure on the others. Because the system does not make decisions on its own, the operator must rely heavily on the interface to understand what the robot is doing. When communication becomes unreliable, this dependence becomes a problem, as delays or loss of feedback make it harder for the user to manage the system safely. Rather than compensating for one another, the architecture, interface, and communication design reinforce each other's weaknesses.

Collectively, these findings frame Nicobot as a system based on supervisory autonomy, where control and responsibility are shared between the robot and the human operator rather than fully delegated to the system.

5.4 Implications of Architectural Findings in Relation to the Literature Review

The literature on agricultural robotics consistently emphasises the importance of system architectures that can support autonomous operation under conditions of unreliable, intermittent, or absent network connectivity. Approaches such as edge computing, decentralised control, and hybrid edge–cloud architectures are frequently presented as

foundational strategies for enabling robotic autonomy in rural environments (Xie et al., 2022; Sathya et al., 2024). When the architecture of Nicobot is examined in relation to these established principles, however, a clear gap emerges between the architectural concepts described in the literature and their practical realisation within the system.

Much of the existing literature assumes that placing computational resources closer to the robot naturally leads to increased autonomy. Edge computing is often described as enabling real-time decision-making, reduced latency, and independence from continuous cloud connectivity (Xie et al., 2022). While Nicobot incorporates local processing elements, the architectural findings demonstrate that decision-making authority remains external to the system. Control actions are not generated autonomously by the robot but are instead initiated by the operator through the user interface. As a result, local processing does not translate into autonomous behaviour. This highlights an important implication for the literature: edge computing alone is insufficient to support autonomy unless it is coupled with internal decisionmaking structures that allow the system to interpret sensor data and act independently.

Hybrid edge–cloud architectures described in the literature typically aim to balance local responsiveness with higher-level coordination and optimisation (Sathya et al., 2024). In these models, local systems are expected to maintain safe and basic operation during connectivity loss, while cloud components provide long-term learning, analysis, or coordination when connectivity is available. Although Nicobot includes multiple processing and communication components consistent with this architectural paradigm, their roles are not clearly separated in a manner that supports autonomous operation. The absence of embedded behavioural logic or automated fallback mechanisms means that the system does not degrade gracefully when connectivity is poor, despite the literature identifying such graceful degradation as critical for rural deployment (Xie et al., 2022).

The architectural findings also challenge common assumptions in the literature regarding system resilience. Many studies frame resilience primarily in terms of hardware redundancy or the availability of alternative communication pathways (Sathya et al., 2024). The Nicobot analysis suggests that resilience is equally dependent on how control and responsibility are distributed within the architecture. When interpretation, decisionmaking, and recovery are delegated to the human operator rather than embedded within the system, resilience remains limited regardless of the technologies employed. This observation extends the literature by highlighting the role of architectural responsibility allocation as a key factor in achieving operational robustness.

In this context, the Nicobot project illustrates how partial adoption of architectural concepts from the literature can lead to systems that appear technically capable but remain operationally fragile. While the system incorporates elements associated with modern agricultural robotics architectures, such as local processing and multiple communication mechanisms, these elements are not integrated in a way that enables autonomous behaviour. The findings reinforce the argument presented in the literature that autonomy in agricultural robotics is not achieved through the presence of architectural components alone, but through

deliberate and coherent allocation of control, decision-making, and recovery mechanisms within the system structure (Xie et al., 2022; Sathya et al., 2024).

5.5 Implications of user Interface Findings in Relation to the Literature Review

The literature on human–robot interaction and usability within agricultural robotics consistently emphasises the importance of intuitive interfaces, clear feedback mechanisms, and reduced cognitive load for non-technical users. Numerous studies argue that the successful adoption of agricultural robotic systems depends not only on technical capability, but also on whether users are able to understand, trust, and confidently operate the system within demanding and time-critical field environments (Goodrich and Schultz, 2007; Nielsen, 1994). When the Nicobot user interface is examined in relation to these principles, a clear divergence emerges between recommended usability practices and the current interface design.

The literature strongly advocates for interfaces that prioritise recognition over recall, present information using meaningful and task-oriented terminology, and clearly communicate system status to the user (Nielsen, 1994). In contrast, the Nicobot interface exposes raw technical data and low-level control elements without sufficient contextual explanation. Users are required to interpret system behaviour, sensor readings, and operational state independently, rather than being supported through structured feedback or guidance. This places a high cognitive burden on the operator and conflicts with usability principles commonly recommended for agricultural and rural robotic systems (Goodrich and Schultz, 2007).

Research on agricultural human–robot interaction also highlights the role of feedback in establishing and maintaining user trust, particularly when robots are operated remotely or outside the operator's immediate field of view (Yanco and Drury, 2004). The analysis of the Nicobot interface reveals a lack of explicit feedback confirming command execution or system response. User actions are not consistently acknowledged by the interface, requiring operators to infer robot behaviour through observation rather than interaction. This absence of confirmation increases uncertainty and reduces confidence, directly contradicting the literature's emphasis on feedback as a critical element of trustworthy human–robot interaction.

Another recurring theme within the literature is the need for interfaces to accommodate intermittent connectivity and delayed system responses, which are common in rural and agricultural settings (Sathya et al., 2024). Prior studies suggest that interfaces should clearly communicate communication state, system availability, and potential limitations in order to prevent confusion and misinterpretation. The Nicobot interface does not explicitly indicate connectivity status or distinguish between communication delays and system faults. As a result, users are left uncertain about the cause of delayed responses, increasing mental workload and undermining usability. This omission runs counter to established recommendations for designing interfaces suited to low-connectivity environments.

The user interface findings also demonstrate how architectural limitations directly shape interaction design. Because Nicobot lacks embedded autonomy and does not perform independent decision-making, the interface must expose low-level system details to the

operator in order to maintain control. This observation supports the literature's assertion that usability cannot be treated as an isolated design concern (Goodrich and Schultz, 2007). When autonomy is limited at the architectural level, interface complexity increases, and responsibility for interpretation, monitoring, and recovery shifts from the system to the user.

Overall, the Nicobot interface analysis supports existing literature claims that usability is central to the adoption of agricultural robotic systems, while simultaneously illustrating how these principles can be undermined by architectural design choices. The findings suggest that achieving effective human–robot interaction requires not only careful interface design but also underlying system behaviour that reduces reliance on continuous user interpretation and manual control. In this respect, the Nicobot case reinforces the view that usability and autonomy are closely interconnected, and that limitations in system architecture inevitably constrain interface effectiveness (Nielsen, 1994; Goodrich and Schultz, 2007).

5.6 Summary and Design Implications for Rural Agricultural Robotics

This chapter has examined the Nicobot system through the lenses of architecture, user interface design, and communication mechanisms, addressing each of the research questions in turn. Taken together, the findings indicate that the limitations of Nicobot do not arise from the absence of individual technologies, but from the way responsibility for control, interpretation, and recovery is distributed across the system. The discussion highlights that autonomy, usability, and communication cannot be treated as independent design concerns; instead, they are tightly coupled and must be addressed coherently at the architectural level.

An important implication of these findings concerns the nature of the proposed architectural improvements identified in this research. Although Nicobot already contains many of the hardware components associated with autonomous agricultural robots, the analysis demonstrates that the limitations observed cannot be resolved through incremental additions alone. The proposed architectural changes do not represent simple extensions or plug-in features that can be layered onto the existing system. Rather, they imply a fundamental reorganisation of system responsibilities, particularly with respect to where decision-making, fault handling, and behavioural control are located. This indicates that meaningful support for rural autonomy would require a conceptual redesign of the system architecture, even if some existing hardware components could be retained.

From a broader perspective, the Nicobot case illustrates a common challenge in agricultural robotics development: systems may appear technically advanced while remaining operationally fragile. The presence of sensors, processing units, and multiple communication channels does not, in itself, result in autonomy or usability. What matters is how these elements are integrated, how decisions are generated, and how much cognitive burden is placed on the human operator. This finding reinforces the view that architectural coherence is a more critical factor than technological quantity when designing robots for rural environments.

Several design-oriented insights emerge from the Nicobot project that may inform future agricultural robotics development. First, autonomy in rural contexts should be approached as contextual independence, rather than as full isolation from human involvement. Agricultural

environments frequently operate under unreliable connectivity, making it necessary to treat disconnection as a normal operational condition rather than an exceptional failure state. Systems should therefore be designed to continue operating safely and predictably when communication is degraded, rather than relying on continuous external supervision.

Second, the findings emphasise that autonomy is not solely a software concern. Physical and environmental constraints such as vibration, dust, moisture, and temperature variation directly affect system reliability. As demonstrated through the Nicobot design, mechanical structure, component placement, and environmental protection are integral to autonomous operation. This suggests that architectural decisions must consider material and environmental factors alongside sensing and computation.

Third, the relationship between autonomy and human involvement requires careful consideration. Autonomy in agricultural robotics does not imply the removal of human participation, but a shift from direct control towards supervisory interaction. Interfaces should therefore prioritise transparency, predictability, and clarity of system state, allowing operators to understand what the robot is doing without needing to interpret low-level technical details. When architectural limitations prevent autonomous decision-making, interface complexity increases and usability suffers.

Energy availability also emerged as a fundamental constraint shaping rural autonomy. Rather than treating power management as an optimisation task applied after system design, the Nicobot findings suggest that energy awareness should be embedded structurally within the architecture. Decisions about sensing frequency, computation placement, and communication methods must be informed by finite and variable power resources, particularly in environments where frequent recharging is impractical.

Finally, the discussion highlights the importance of simplicity as a guiding principle for longterm rural deployment. Overly complex systems can become difficult to maintain, understand, or adapt in real agricultural settings. The Nicobot analysis demonstrates that autonomy is supported not by maximal flexibility or feature density, but by clearly defined system behaviour, constrained responsibilities, and interfaces that reduce cognitive effort. Simplicity, in this sense, becomes a prerequisite for sustainable autonomy rather than a limitation.

In summary, the findings of this chapter suggest that designing agricultural robots for rural autonomy requires a shift in emphasis from adding technological components to structuring responsibility within the system. Effective designs should begin by defining what the robot must be able to do independently, what information should be communicated to the user, and how the system should behave when connectivity is unreliable. By grounding architectural decisions in environmental reality, human supervision, and resource constraints, future agricultural robotic systems may achieve forms of autonomy that are practical, interpretable, and resilient in rural contexts.

Chapter 6: Conclusion

6.1 Concluding Overview and Purpose of the Research

This thesis examined the conceptualisation and design of an autonomous agricultural robotic system intended for operation in rural and semi-structured environments. Using Nicobot as a case study, the research focused on how architectural organisation, interaction design, and system integration can be approached when reliable connectivity, controlled conditions, and continuous human oversight cannot be assumed. The purpose of the work was not to validate autonomous performance, but to explore how a robotic platform may be structured to support autonomy as a design objective within realistic rural constraints.

The study was framed as a progressive design investigation in which theoretical considerations from the literature were translated into a coherent system architecture. The literature review identified recurring challenges in agricultural robotics, particularly those related to dependence on continuous connectivity, limited adaptability to field conditions, and insufficient attention to usability. These observations informed the design direction of Nicobot, which prioritised local control, modularity, and supervisory human interaction rather than full automation or optimisation.

Throughout the thesis, autonomy is treated not as a singular technical capability, but as an emergent property of coordinated system organisation. The design work demonstrates how sensing, control, communication, and user interaction can be arranged to support predictable behaviour under uncertain environmental and network conditions. This perspective emphasises balance—between local decision-making and external services, between automated processes and human oversight, and between system flexibility and operational clarity.

The research adopts a reflective and design-oriented methodology rather than an experimental one. Architectural analysis, system modelling, and heuristic usability evaluation were used to reason about system behaviour and identify design limitations. These methods support early-stage robotic research where empirical validation may not yet be feasible, and they allow critical assessment of design choices without overstating system maturity.

A central outcome of this work is the framing of autonomy in agricultural robotics as a collaborative process. Rather than removing the human operator from the control loop, the Nicobot design supports a supervisory interaction model in which autonomy operates alongside transparency and manual intervention. This interpretation aligns with the practical realities of agricultural contexts, where safety, accountability, and situational awareness remain essential.

While Nicobot itself is a research platform, the design principles discussed in this thesis are applicable to a broader range of rural robotic applications. The emphasis on modular architecture, edge-oriented control, and usability-driven interface design provides a conceptual reference for future systems addressing similar environmental and operational challenges.

6.2 Evolution of Nicobot as an Intelligent Field System

The development of Nicobot over the course of this research reflects an evolution in system conceptualisation rather than a progression measured through experimental validation. Early versions of the platform consisted of loosely connected components capable of supporting individual functions, such as manual control or basic sensing, but without a coherent architectural framework. As the research progressed, the focus shifted towards restructuring these components into an integrated system design that could be reasoned about as a unified robotic platform for rural contexts.

This evolution was driven primarily by design reflection informed by environmental and operational constraints identified in the literature and through system analysis. Factors such as intermittent connectivity, uneven terrain, and variable sensing conditions influenced architectural decisions related to local control, modularity, and communication independence. Rather than attempting to eliminate these constraints, the design process treated them as defining characteristics of rural operation that should be accommodated at an architectural level.

As a result, Nicobot transitioned conceptually from a collection of task-oriented subsystems to a structured field system organised around layered responsibilities. Control, communication, and interaction were progressively reorganised to support predictable operation under uncertain conditions. In particular, the allocation of timecritical functions to on-board components and the treatment of external services as supplementary reflected a shift towards autonomy as an architectural property rather than a function of specific algorithms.

Human interaction also played a significant role in shaping this evolution. Early interface concepts prioritised technical completeness over interpretability, which prompted a reassessment of how system state and control should be communicated to operators. Subsequent design iterations emphasised clarity, consistency, and supervisory interaction, reframing the interface as a communication layer rather than a diagnostic console. This shift aligns with the broader aim of supporting autonomy without excluding human oversight.

Importantly, the evolution of Nicobot does not represent the attainment of a fully autonomous or adaptive system. Instead, it illustrates how successive design decisions can refine the coherence and usability of a robotic platform intended for rural research. The system remains a conceptual prototype, but one that embodies principles relevant to intelligent field systems, including modularity, localised control, and transparency of operation.

6.3 Rethinking Human Oversight in Agricultural Robotics

An important design insight emerging from the Nicobot project concerns the role of human oversight within autonomous agricultural systems. Rather than framing the human operator as a passive monitor or an emergency fallback, the system design supports a supervisory role characterised by observation, interpretation, and selective intervention. This shift reflects a broader reconsideration of autonomy in rural robotics, where human involvement remains integral to safe and context-aware operation.

Within this framework, autonomy is not positioned as a replacement for human judgement, but as a redistribution of responsibilities between the system and its operator. Automated

processes handle routine control tasks, while humans retain oversight of system state, environmental interpretation, and decision-making in ambiguous situations. The Nicobot design reflects this relationship through interface elements that prioritise visibility of system status and clarity of control modes rather than continuous manual input.

Communication between the robot and the operator is treated as a bidirectional process. The interface is structured to convey operational state, motion intent, and system conditions through visual indicators that can be interpreted at a glance. This design approach emphasises transparency and predictability, allowing operators to form an understanding of system behaviour without requiring constant interaction. Importantly, this discussion refers to design intent rather than to validated user behaviour or trust outcomes.

The supervisory model also highlights the importance of predictability in autonomous systems. By presenting consistent feedback and clearly defined operational states, the system supports anticipation rather than reactive control. This reduces the cognitive burden associated with supervising complex systems and reinforces the role of the human as an informed decision-maker rather than a direct controller of low-level actions.

Human oversight is further considered in relation to accountability and responsibility. Retaining a human-in-the-loop design acknowledges that contextual judgement, safety assessment, and responsibility for outcomes remain with the operator. The Nicobot architecture supports this by ensuring that control authority can be reclaimed when required and that system behaviour remains observable rather than opaque. These considerations align with contemporary principles of responsible autonomy, without asserting ethical compliance or governance outcomes.

Finally, this perspective on oversight reframes autonomy as a collaborative arrangement rather than a hierarchical one. The Nicobot system is designed to support cooperation between automated processes and human supervision, recognising that agricultural environments are characterised by variability that cannot be fully anticipated in advance. By maintaining human oversight as an active component of system operation, the design reflects a pragmatic approach to autonomy suited to rural contexts.

6.4 Ethical, Environmental, and Socio-Technical Implications

The introduction of autonomous systems into agricultural environments raises ethical, environmental, and socio-technical considerations that extend beyond technical system design. Within the scope of this thesis, these implications are addressed at a conceptual and design level rather than through empirical evaluation. The Nicobot project provides a context in which such considerations can be reflected upon as part of responsible system development for rural robotics.

From an ethical perspective, transparency and accountability are central to the design of autonomous agricultural systems. The Nicobot architecture and interface are structured to prioritise visibility of system state and clarity of operational modes. This design orientation supports human oversight by ensuring that automated behaviour remains observable and interpretable. Rather than asserting ethical compliance or trust outcomes, the project

highlights transparency as a necessary precondition for responsible autonomy, particularly in environments where robotic actions may affect safety, property, or livelihoods.

Environmental considerations are also relevant when designing robotic platforms for rural contexts. Agricultural ecosystems are sensitive to physical disturbance, energy consumption, and prolonged mechanical presence. The Nicobot design reflects an awareness of these constraints through its emphasis on structural moderation, modular construction, and resource awareness. These considerations are presented as design intentions rather than as verified sustainability outcomes, acknowledging that environmental impact can only be assessed through long-term field deployment.

Socio-technical implications arise from the interaction between autonomous systems and established agricultural practices. The Nicobot project frames automation as a supportive tool within a supervisory model rather than as a replacement for human involvement. This perspective recognises that agricultural work involves contextual judgement and responsibility that cannot be fully delegated to automated systems. By retaining human oversight and interpretability as design priorities, the system reflects a cautious approach to the integration of autonomy into rural work practices.

Issues of access and inclusivity also form part of the broader socio-technical context. Rural regions often experience disparities in infrastructure, connectivity, and technical resources. While the Nicobot design does not claim to resolve these challenges, its emphasis on local control and reduced dependence on continuous connectivity aligns with the goal of making autonomous systems more adaptable to diverse deployment contexts. These considerations are framed as design motivations rather than as demonstrated social outcomes.

Data governance and information responsibility represent an additional ethical dimension. Autonomous agricultural systems may generate environmental, spatial, and operational data that hold scientific or commercial value. The Nicobot project acknowledges the importance of data ownership, integrity, and responsible use without proposing or evaluating specific governance mechanisms. These issues are identified as areas requiring further interdisciplinary research involving technical, legal, and policy perspectives.

6.5 Evaluative Summary of Achievements and Limitations

The outcomes of this research are best assessed by considering the relationship between its design objectives and the scope of work undertaken. The thesis did not aim to validate autonomous agricultural performance through experimentation, but rather to explore how autonomy may be approached as a system-level design problem within rural contexts. Accordingly, the primary achievements of the study lie in the development and articulation of a coherent architectural and interaction framework rather than in demonstrated operational outcomes.

One of the key achievements of the research is the consolidation of disparate system elements into a unified conceptual design. The Nicobot project brings together hardware organisation, software architecture, communication strategy, and interface design within a single structured framework. This integration supports a design-oriented understanding of how autonomous

behaviour can be reasoned about in environments characterised by variable connectivity, environmental uncertainty, and the need for human oversight.

The research also contributes methodologically by demonstrating how architectural analysis and heuristic evaluation can be applied to early-stage robotic systems. These approaches enabled critical reflection on design decisions without relying on empirical testing or performance measurement. In doing so, the study illustrates how design quality, coherence, and usability can be examined within the constraints of exploratory research.

At the same time, the limitations of the work are significant and are explicitly acknowledged. The study does not provide experimental validation of system behaviour, durability, energy efficiency, or user interaction. Claims regarding long-term operation, scalability, economic viability, or deployment readiness are therefore beyond the scope of this thesis. Similarly, while the design considers hybrid edge–cloud organisation, the behaviour of such architectures under large-scale or multi-robot conditions has not been examined.

Hardware capability and sensing performance are also treated at a conceptual level. The design does not evaluate alternative processing platforms, perception algorithms, or optimisation strategies, nor does it assess how such choices would affect system behaviour under real-world conditions. These aspects remain open for future empirical investigation.

Despite these limitations, the research offers a structured and transparent account of how autonomy may be approached in rural agricultural robotics as a design challenge. By explicitly recognising what has not been tested or validated, the study avoids overstating system maturity while still providing a foundation for future work. The value of the research lies not in demonstrating a finished solution, but in clarifying design tradeoffs, constraints, and opportunities that shape the development of autonomous systems for rural environments.

6.6 Future Directions for Research, Innovation, and Deployment

This thesis focuses on the conceptual design and architectural organisation of an autonomous agricultural robotic system, and it also highlights several directions for future research that extend beyond its current scope. These directions are presented as potential areas of investigation rather than as planned developments or expected outcomes.

One important area for future work concerns longitudinal field evaluation. Extended observation of robotic systems operating across different seasons, terrains, and environmental conditions would be necessary to examine issues such as component degradation, sensor drift, communication variability, and maintenance requirements. Such studies would provide empirical grounding for design assumptions that are treated conceptually in this thesis.

Another direction for future research involves the coordination of multiple robotic platforms. Although the Nicobot design is centred on a single system, its modular organisation provides a conceptual basis for exploring distributed operation across multiple agents. Research into task coordination, information sharing, and system behaviour under intermittent connectivity could extend this work into the domain of multi-robot agricultural systems.

Further investigation may also examine advances in perception and decision-making. The integration of more sophisticated sensing and interpretation techniques could be explored, provided that their computational and energy demands remain compatible with rural deployment constraints. Any such work would need to carefully balance increased system complexity against the principles of local control and operational continuity emphasised in this thesis.

Human–robot interaction represents another important avenue for continued study. Future research could examine how operators supervise autonomous systems over extended periods and how interface design influences situational awareness and decision-making. Qualitative studies focusing on supervision practices and user interpretation would complement the heuristic-based design approach adopted in this work.

Cybersecurity and data governance constitute additional areas requiring further investigation. As agricultural robotic systems increasingly rely on distributed communication and data exchange, systematic analysis of threat models, access control mechanisms, and data integrity safeguards will be necessary. These challenges extend beyond technical design and would benefit from interdisciplinary collaboration.

Finally, future research may consider how design frameworks such as the one presented in this thesis can be evaluated across different agricultural contexts. Comparative studies involving alternative architectures or deployment environments could help clarify which design principles generalise across rural settings and which are context specific.

In summary, this thesis establishes a structured foundation for ongoing research rather than a complete deployment solution. The future directions outlined above reflect the complexity of rural autonomy and identify opportunities for empirical, technical, and interdisciplinary investigation building on the design-oriented contributions of this work.

References

1. Xie, D., Chen, L., Liu, L., Chen, L., & Wang, H. (2022). Actuators and Sensors for Application in Agricultural Robots: A Review. Machines, 10(10), 913.
2. Goodrich, M. A., & Schultz, A. C. (2007). Human-Robot Interaction: A Survey. Foundations and Trends in Human-Computer Interaction, 1(3), 203–275.
3. Halder, S. (2023). Human-Interactions with Robotic Cyber-Physical Systems (CPS) for Facilitating Construction Progress Monitoring. Virginia Polytechnic Institute and State University.
4. Yerebakan, M. O., & Hu, B. (2024). Human–Robot Collaboration in Modern Agriculture: A Review of the Current Research Landscape. Advanced Intelligent Systems, 2300823.
5. Schraick, L.-M., Ehrlich-Sommer, F., Stampfer, K., Meixner, O., & Holzinger, A. (2025). Usability in Human-Robot Collaborative Workspaces. Universal Access in the Information Society, 24(2), 1609–1622.
6. Adamides, G., & Edan, Y. (2023). Human–Robot Collaboration Systems in Agricultural Tasks: A Review and Roadmap. Computers and Electronics in Agriculture, 204, 107541.
7. Farahani, R., Loh, F., Roman, D., & Prodan, R. (2024). Serverless Workflow Management on the Computing Continuum: A Mini-Survey. Companion of the 15th ACM/SPEC

International Conference on Performance Engineering (ICPE ’24). 8. Gangwani, N. (2024). Serverless Computing in the Edge-Cloud Continuum: Challenges, Opportunities, and a Novel Framework. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 10(5), 337–344.

9. Calavaro, C., Cardellini, V., Lo Presti, F., & Russo, G. (2025). Beyond Cloud: Serverless Functions in the Compute Continuum. SN Computer Science, 6, 194.
10. Zangana, H. M., Sallow, Z. B., & Omar, M. (2024). Cloud Architectures for

Distributed Serverless Computing: A Review of Event-Driven and Function-as-a-Service Paradigms. International Journal of Artificial Intelligence & Robotics, 6(2), 57–64.

11. Patros, P., Spillner, J., Papadopoulos, A. V., Varghese, B., Rana, O., & Dustdar, S. (2021). Toward Sustainable Serverless Computing. IEEE Internet Computing, 25(6), 42–50.

12. Almurshed, O., Patros, P., Huang, V., Mayo, M., Ooi, M., Chard, R., Chard, K., Rana, O., Nagra, H., Baughman, M., & Foster, I. (2022). Adaptive Edge-Cloud Environments for Rural AI. 2022 IEEE International Conference on Services Computing (SCC), 74–83.
13. Mohammed, B. (2025). A Comprehensive Overview of Federated Learning for NextGeneration Smart Agriculture: Current Trends, Challenges, and Future Directions. Informatica, 49(1), 117–136.
14. Kaur, J., Hazrati Fard, S. M., Amiri-Zarandi, M., Dara, R., et al. (2022). Protecting Farmers’ Data Privacy and Confidentiality: Recommendations and Considerations. Frontiers in Sustainable Food Systems, 6.
15. Gupta, S., Arora, S., & Qamar, S. (2025). Federated Learning in Smart Farming: Applications and Challenges. In *Convergence of AI, Federated Learning, and Blockchain for Sustainable Development* (Advances in Science, Technology & Innovation, pp. 121–146). Springer.
16. Rizman Žalik, K., & Žalik, M. (2023). A Review of Federated Learning in Agriculture. Sensors, 23(23), 9566.
17. Zakzouk, S., & Said, L. A. (2025). Federated Learning for Soil Moisture Prediction: Benchmarking Lightweight CNNs and Robustness in Distributed Agricultural IoT Networks. Machine Learning and Knowledge Extraction, 7(4), 132.
18. Shen, S., Zhu, T., Wu, D., Wang, W., & Zhou, W. (2022). From Distributed Machine Learning to Federated Learning: In the View of Data Privacy and Security. Concurrency and Computation: Practice and Experience, 34(16), e6002.
19. Dembani, R., Karvelas, I., Akbar, N., Rizou, S., Tegolo, D., & Fountas, S. (2025). Agricultural Data Privacy and Federated Learning: A Review of Challenges and Opportunities. Computers and Electronics in Agriculture, 232.
20. Ayranci, A. A., & Erkmen, B. (2024). Edge Computing and Robotic Applications in Modern Agriculture. Proceedings of the 2024 International Congress on HumanComputer Interaction, Optimization and Robotic Applications (HORA), Istanbul, Turkey, May 23–25, 2024.
21. Ahmad, W., Ali, F., Ullah, Y., & Almansour, S. (2025). IoUT and Collaborative Cloud Computing-Based UAV Channel Modeling for Agriculture Communication. IEEE Transactions on Consumer Electronics, 71(3), 8067–8081.

22. Kuchuk, H., & Malokhvii, E. (2024). Integration of IoT with Cloud, Fog, and Edge Computing: A Review. Advanced Information Systems, 8(2), 65–78.
23. Benos, L., Tagarakis, A. C., Dolias, G., Berruto, R., Kateris, D., & Bochtis, D. (2021). Machine Learning in Agriculture: A Comprehensive Updated Review. Sensors (Basel), 21(11), 3758.
24. Thilakarathne, K., Salim, H. G., Wijesekara, H., Rajapaksha, A., et al. (2022). Towards Making the Fields Talk: A Real-Time Cloud-Enabled IoT Crop Management Platform for Smart Agriculture. Frontiers in Plant Science, 13, 1030168.
25. Phasinam, K., Kassanuk, T., Shinde, P. P., Thakar, C. M., Sharma, D. K., Mohiddin, M. K., & Rahmani, A. W. (2022). Application of IoT and Cloud Computing in Automation of Agriculture Irrigation. Journal of Food Quality, 2022, 8285969.
26. Yanco, H. A., & Drury, J. L. (2004). Classifying Human-Robot Interaction: An Updated Taxonomy. In *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics*, The Hague, Netherlands, October 2004
27. Szekely, S., Iheme, L. O., O'Driscoll, F., Kypuros, D., Pang, V., & Shmigelsky, G. (2025). Scalable Architecture and Intelligent Edge with 5G-Advanced, MEC, IoT, UAVs and AI for a Sustainable Agriculture and Food Operations. Journal of Engineering and Architecture.
28. Pham, X. K. (2025). Smart Robot Car for Autonomous Navigation: An IoT, MQTT, and Image Processing Approach. Master's thesis, Department of Electrical and Electronic Engineering, University of Waikato, Hamilton, New Zealand.
29. Aldossary, M., & Alharbi, H. A. (2021). Towards a Green Approach for Minimizing Carbon Emissions in Fog-Cloud Architecture. IEEE Access, 9, 131720–131732.
30. Singh, N., & Adhikari, M. (2024). Edge-Centric Collaborative Federated Learning for Irrigation Management of Paddy Fields Using Agriculture Sensor Data Processing. IEEE Sensors Journal, 8(8), 6009104.
31. Kalyani, Y., & Collier, R. (2021). A Systematic Survey on the Role of Cloud, Fog, and Edge Computing Combination in Smart Agriculture. Sensors, 21(17), 5922.
32. Puppala, S., & Sinha, K. (2025). Towards Secure and Efficient Farming Using SelfRegulating Heterogeneous Federated Learning in Dynamic Network Conditions. Agriculture, 15(9), 934.
33. Nielsen, J. (1994). *Usability Engineering*. San Francisco, CA: Morgan Kaufmann.