AI-POWERED ROBOTICS FOR POST-DISASTER SURVIVOR DETECTION AND RESCUE PATH MAPPING

DIANA NANSUBUGA

S22B23/009

A PROJECT REPORT SUBMITTED TO THE FACULTY OF ENGINEERING, DESIGN AND TECHNOLOGY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE OF UGANDA CHRISTIAN UNIVERSITY

April, 2025

UGANDA CHRISTIAN UNIVERSITY

A Centre of Excellence in the Heart of Africa

Declaration

I Nansubuga Diana do hereby declare that this Project Report is original and has not been published and/or submitted for any other degree award to any other University before.

Date: 5th, May, 2025

Signature: 

Approval

This Project Report has been submitted for examination with the approval of the following supervisor.

Signed: 

Date: 5/5/25

Mr. Ian Raymond Osolo

Faculty of Engineering Design and Technology
Department of Computing and Technology& IT
Uganda Christain University
iosolo.ucu.ac.ug

Dedication

I dedicate this report to the Almighty God without whom we can do nothing. I further dedicate it to my parents and guardians for their unceasing and selfless support throughout my stay at Uganda Christian University.

Acknowledgement

I am deeply indebted to my project supervisor Mr. Ian Raymond Osolo whose unlimited steadfast support and inspiration have made this project a great success. In a very special way, I thank him for every support he has rendered unto me to see that I succeed in this challenging study.

Special thanks go to my friends and families who have contained the hectic moments and stress I have been through during the course of the research project.

Abstract

Natural and man-made disasters often highlight the weaknesses in traditional emergency response efforts, especially in countries with limited resources like Uganda. Challenges such as slow response times, difficulty reaching affected areas, and a lack of real-time information frequently result in lost lives and misused resources. This study introduces a robotic dog system enhanced by artificial intelligence, designed to improve the effectiveness of search and rescue missions following disasters. The system incorporates real-time human detection using AI, SLAM for creating environmental maps, and the A* algorithm to plan efficient rescue routes.

The robot is powered by a Raspberry Pi and managed through a Flask-based web platform. It includes multiple sensors, such as GPS, ultrasonic detectors, a night vision camera, and directional microphones, allowing it to navigate independently, avoid obstacles, and communicate wirelessly with remote rescue teams. Testing the system in a controlled, disaster-like setting confirmed its ability to locate victims, generate accurate maps, and determine safe paths with a high level of reliability. The results suggest that this low-cost, locally adaptable solution could play a vital role in speeding up rescue efforts and minimizing risks to human responders. By addressing key limitations in current practices, the project adds valuable insights to the field of disaster robotics, particularly in settings where advanced tools are not readily available.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Disasters whether triggered by natural forces like floods and landslides or man-made events such as structural collapses continue to challenge emergency response capabilities across the world. The most critical period after such events is the first 72 hours, when the chances of rescuing trapped survivors are highest. In many regions, particularly in developing countries, response efforts are often delayed due to poor infrastructure, lack of information, and limited access to the affected zones. In Uganda, communities in high-risk areas like Bududa frequently suffer loss of life and property due to recurring natural calamities, with rescue teams unable to respond quickly or effectively.

Technology, especially artificial intelligence and robotics, has started to play a vital role in addressing such challenges. Intelligent robotic systems offer the potential to locate survivors, map hazardous areas, and provide real-time data to rescue teams, reducing the need for responders to put themselves in danger. This study explores the development of a robotic system powered by AI and designed to assist in search and rescue missions after disasters, tailored specifically for the Ugandan context.

1.2 Problem Statement

Responding quickly and accurately in the aftermath of a disaster can mean the difference between life and death. However, in many cases, especially in Uganda, search and rescue efforts remain slow, largely manual, and poorly organized. The absence of modern tools, limited access to real-time data, and insufficient planning strategies often cause delays in reaching those most in need. Emergency teams frequently operate without reliable information about where survivors may be located or how to safely navigate hazardous areas. As a result, critical time is lost, resources are used inefficiently, and lives that could be saved are sometimes lost. These challenges highlight the pressing need for a smart, automated solution that can improve situational awareness, support informed decision-making, and speed up rescue efforts during the critical early hours after a disaster strikes.

1.3 Objectives

1.3.1 Main Objective

To design and implement an AI-driven robotic system capable of detecting survivors and mapping safe paths for rescue in post-disaster environments.

1.3.2 Specific Objectives

- To study existing technologies used in search and rescue operations after disasters.
- To identify the limitations of current rescue systems in Uganda with regard to survivor detection and path planning.
- To develop a robotic solution that uses AI and SLAM for real-time mapping and human detection.
- To implement the system using cost-effective, accessible hardware and open-source software tools.
- To evaluate the performance of the robot in a simulated disaster environment.

1.4 Research Questions

- How effective are current rescue systems in locating survivors during the early hours of disaster response?
- What are the main technological limitations faced by rescue teams in Uganda?
- Can a robotic system using AI and real-time mapping improve the speed and safety of rescue operations?
- What hardware and software components are suitable for developing such a system under local constraints?
- How well does the developed prototype perform in a simulated post-disaster setting?

1.5 Hypotheses

- **H1:** A robotic system that integrates AI-based survivor detection and SLAM-based mapping will improve the efficiency and accuracy of search and rescue efforts.
- **H0:** The implementation of the robotic system will have no significant impact on the performance of search and rescue operations compared to manual methods.

1.6 Scope and Limitations

This research focuses on the design and testing of a prototype robotic system intended for use in post-disaster rescue efforts. The scope includes integrating AI models for human detection, SLAM for mapping, and the A* algorithm for pathfinding. Testing is limited to simulated environments due to safety and logistical constraints. The robot is optimised for moderate debris and terrain and is not suited for waterlogged or heavily obstructed areas. Limitations include hardware constraints, reliance on Wi-Fi connectivity, and power consumption challenges.

1.7 Significance of the Study

This project tackles a critical shortfall in Uganda's ability to respond effectively to disasters by offering a practical, locally adaptable technological solution. The proposed system supports rescue teams by enabling faster detection of survivors, mapping out safe routes through hazardous areas, and sharing real-time data to guide decision-making. By minimizing the dependence on slow, manual processes and enhancing coordination on the ground, this robotic solution could play a significant role in saving lives especially in isolated or high-risk regions where conventional methods fall short. In addition to addressing immediate rescue challenges, the project also creates a platform for future innovation and research in the field of disaster robotics tailored to low-resource environments.

1.8 Definition of Terms

1. Artificial Intelligence (AI): A field of computer science that focuses on building systems capable of carrying out tasks that would normally require human thinking, such as recognizing patterns, making decisions, or solving problems.
2. Simultaneous Localization and Mapping (SLAM): A process that allows a robot to create a map of an unfamiliar environment while keeping track of its position within that space, enabling it to navigate effectively without relying on GPS.
3. A Algorithm:* A well-known pathfinding method used to calculate the shortest and safest route between two points, often applied in navigation systems for robots and games.
4. You Only Look Once (YOLO): A fast and efficient computer vision algorithm used to detect objects, including humans, in images or videos especially useful in real-time rescue applications.
5. Rescue Window: Refers to the first 72 hours following a disaster, a crucial period when the chances of locating and saving survivors are significantly higher.
6. Autonomous Navigation: The capability of a robot to explore and move through its environment on its own, using internal logic and sensor input rather than remote control.

7. Geotagging: The practice of linking photos, video, or other data with specific location information, helping rescue teams know exactly where something was detected.
8. Flask: A lightweight web development tool written in Python, used for building simple yet powerful interfaces like the control dashboard for the rescue robot.

Chapter 2

Literature Review

2.1 Introduction

This chapter explores the current landscape of robotics and artificial intelligence as applied to post-disaster search and rescue operations. It examines the evolution of rescue technologies, their strengths, and the limitations that continue to hinder effective deployment particularly in low-resource settings. The review also aligns previous findings with the specific objectives of this research and highlights the existing knowledge gaps that this study seeks to address.

2.2 Review of Three Existing Systems

2.2.1 1. RoboCup Rescue

RoboCup Rescue is a global research initiative that evolved from the larger RoboCup robotics competition. Its primary aim is to encourage the development of autonomous and semi-autonomous robots designed for deployment in disaster scenarios. Participants in this initiative create robotic systems that are tested on their ability to navigate through collapsed structures, identify potential victims using sensors, and generate accurate environmental maps. The program emphasizes simulation environments that challenge a robot's ability to perceive surroundings, move through unstable terrain, and make decisions using advanced path planning techniques. These controlled settings allow researchers to evaluate both software and hardware performance in conditions that mirror aspects of real-world emergencies.

Over time, RoboCup Rescue has had a notable impact on the way robotic systems are designed for urban search and rescue operations. It has underscored the importance of combining navigation, mapping, and human detection capabilities into unified platforms. However, one of its limitations is that the simulations, while detailed, do not fully capture the unpredictability, signal interference, or physical obstacles present in actual disaster zones. As a result, while the competition offers a valuable framework for testing and innovation, the transition from simulation to field deployment remains a challenge. Nonetheless, RoboCup Rescue continues to serve as a key driver of progress in the field

of disaster robotics.

2.2.2 2. DARPA Robotics Challenge

The DARPA Robotics Challenge (DRC), launched by the U.S. Defense Advanced Research Projects Agency, was created to push the boundaries of what robots could achieve in dangerous, human-inaccessible environments. It was developed as a response to the 2011 Fukushima nuclear crisis, which exposed the need for machines that could step in where human safety was at risk. The competition featured a series of demanding tasks such as driving, climbing stairs, using tools to cut through walls, and turning industrial valves each simulating real-world challenges a robot might face after a disaster. The event attracted some of the world's top research teams and led to significant progress in areas like humanoid robot movement, sensor integration, and machine learning for task execution.

While the DRC revealed just how capable modern robotics could become, it also exposed the limits of current technology. On the upside, it accelerated the development of robots with improved human-like motion and the ability to navigate human-centric spaces like vehicles and buildings. However, many robots struggled with stability, often toppling during tasks due to the difficulty of maintaining balance on rough terrain. The systems were also extremely complex and costly, making them unsuitable for widespread use in places with limited funding and infrastructure. Still, the challenge left a lasting legacy by influencing future work in robot mobility, manipulation, and semi-autonomous control systems.

2.2.3 3. PackBot by iRobot

PackBot, developed by iRobot (now Endeavor Robotics), is a field-tested ground robot initially designed for military use and later adapted for disaster response. Its compact design, rugged construction, and versatile mobility allow it to operate in rubble, tunnels, and collapsed infrastructure. Equipped with cameras, sensors, and manipulator arms, PackBot can be remotely operated to conduct surveillance, detect hazardous materials, and locate trapped individuals. The robot was famously deployed during search and rescue efforts at Ground Zero after the 9/11 attacks and again during the Fukushima nuclear crisis.

Unlike many academic prototypes, PackBot has seen real-world deployments, making it one of the most practical and operationally validated systems. Its modular design supports a wide range of missions, and its real-time video transmission allows operators to assess situations from a safe distance. However, one of its major drawbacks is its dependence on human operators, as it lacks autonomous navigation and AI-based detection capabilities. This means the robot is only as effective as the skill and awareness of its controller. As a result, while PackBot has proven invaluable in high-risk operations, integrating AI for autonomous decision-making could significantly enhance its usefulness in future disaster response missions.

2.3 Review Mapped to Objectives

2.3.1 2.1.1 Objective 1: To carry out a preliminary study on the existing systems used in post-disaster search and rescue operations

Over the years, search and rescue (SAR) technologies used in disaster response have seen considerable advancement from entirely manual operations to systems enhanced by robotics and aerial drones. These innovations are intended to help emergency teams reach unsafe or hard-to-access locations while reducing risk to human responders. Among notable examples, the RoboCup Rescue initiative has illustrated the capabilities of robots in simulated earthquake conditions, focusing on their mobility and ability to make autonomous decisions. Likewise, the DARPA Robotics Challenge provided a platform to explore how robots could carry out demanding rescue tasks, such as navigating through rubble and using tools in unpredictable environments.

However, even with these advancements, many SAR systems still fall short in key areas such as autonomy, adaptability, and durability. A significant number continue to rely on remote operation or require direct visual feedback, which limits their usefulness in real-world situations where communication is unreliable or visibility is low. As noted by Murphy et al. [1], robots deployed in major incidents like the 9/11 attacks and the Fukushima nuclear disaster experienced difficulties due to poor sensor accuracy, limited mobility, and a lack of real-time mapping capabilities factors that delayed rescue efforts. In addition, many systems prioritize environmental mapping but do not include mechanisms for detecting human survivors or navigating independently. Madhavan et al. [2] found that most field-tested robots lacked integrated human detection, making them less effective during the crucial initial hours following a disaster. These gaps highlight the need for more robust systems that merge perception, decision-making, and autonomy challenges that this study aims to address.

2.3.2 2.1.2 Objective 2: To identify gaps in current survivor detection and rescue path planning technologies within the Ugandan context

In Uganda, disaster response mechanisms remain largely manual and under-resourced. Teams rely on simple tools like handheld radios and personal observation, with little or no help from intelligent systems. There is a noticeable absence of autonomous technologies such as mobile mapping robots or AI-driven survivor detection systems. Although the National Emergency Coordination and Operations Centre (NECOC) has made progress in organizing national-level coordination, significant limitations still exist in the field, particularly in high-risk regions lacking the infrastructure for digital response systems [3].

Twesigye et al. [4] noted that areas most prone to disasters such as Bududa and Kasese do not have adequate access to critical tools like autonomous mapping technologies or integrated rescue planning platforms. This discrepancy between national policy and field-level implementation severely undermines the efficiency of rescue missions. Moreover,

while universities in Uganda have made strides in robotics and AI, these innovations rarely translate into practical field solutions. Kabuye et al. [5] observed that the gap between research, emergency responders, and humanitarian agencies limits the spread of promising technologies. Without proper funding and collaboration frameworks, many prototypes remain in academic settings, disconnected from the communities that need them most.

2.3.3 2.1.3 Objective 3: To design a robotic system for detecting survivors and mapping safe rescue paths using artificial intelligence

Artificial intelligence now plays a key role in the development of robotic systems, particularly for detecting people in environments that are visually complex or obstructed. Object detection models such as YOLO (You Only Look Once), Faster R-CNN, and SSD have proven effective at identifying human figures, even in low-light conditions or where visibility is partially blocked [6]. When integrated into mobile robots, these models significantly enhance the speed and precision of locating survivors, which in turn reduces the workload and risk faced by emergency responders.

For such systems to be fully effective in disaster-stricken areas, they also need to operate in challenging terrain and without relying on GPS signals. Simultaneous Localization and Mapping (SLAM) techniques like ORB-SLAM and RTAB-Map address this need by allowing robots to map unfamiliar environments while keeping track of their position in real time [7]. When combined with AI-based detection, these tools enable robots to navigate independently, locate survivors, and identify the safest paths through hazardous zones. Studies by Behnke [8] and Chung et al. [9] highlight how such systems improve the overall success of rescue missions and minimize risks to human teams. In countries like Uganda, where resources may be limited and every minute counts, these technologies could have a life-saving impact.

2.3.4 2.1.4 Objective 4: To implement the designed system using appropriate hardware and software platforms

Designing a functional SAR robot requires thoughtful selection of both hardware and software to ensure robustness, flexibility, and performance in real-time conditions. For this project, the NVIDIA Jetson Nano was selected as the core processing unit due to its support for GPU-accelerated deep learning, power efficiency, and compatibility with widely used AI libraries [10]. Other essential components include a camera module for visual input, ultrasonic sensors for proximity sensing, and motor drivers for mobility. All of these were assembled on a mobile robot platform built to withstand moderate terrain irregularities.

From a software perspective, the Robot Operating System (ROS) was employed to manage the integration of key subsystems such as perception, motion, and navigation due to its modular design and extensive support community [11]. Object detection was powered by YOLOv5 models trained on TensorFlow or PyTorch frameworks, while SLAM mapping was handled by RTAB-Map. The system was structured around a development cycle

that involved individual module testing and complete system integration under controlled conditions. The result was a fully functioning prototype capable of processing real-time sensor data, navigating on its own, and producing live maps and survivor alerts.

2.3.5 2.1.5 Objective 5: To carry out the testing and evaluation of the implemented system in a controlled, simulated disaster environment

Testing and validation are key steps in confirming the performance and reliability of a robotic rescue system. The prototype was tested in an artificially created disaster zone that included obstacles, uneven surfaces, and hidden targets, simulating conditions commonly found in collapsed structures or landslide zones [12]. This environment allowed researchers to assess whether the robot could detect survivors, create accurate maps, and navigate through hazards with minimal input.

A range of evaluation metrics was used, including detection accuracy (precision and recall), mapping completeness, and navigation efficiency. The robot performed well in recognizing both stationary and moving human targets and was able to maintain reliable spatial awareness even when faced with sensor noise and partial obstructions [13]. Observational feedback revealed a few areas for improvement, such as real-time map visualization and better obstacle avoidance sensitivity. However, overall results confirmed that the system was capable of operating effectively in dynamic disaster environments. With some refinements, the robot could be scaled for real-world use and potentially assist response teams in saving lives during critical emergencies.

2.4 Research Gap and Justification

Although advancements in robotic disaster response have been significant, the real-world application of such technologies in developing nations has remained limited. Many existing systems are either financially out of reach, difficult to customize for local needs, or lack vital capabilities like live data transmission and multi-sensor detection. In Uganda, these limitations often translate into delayed responses, poor situational awareness, and challenges in quickly finding survivors. This project aims to address those challenges by creating a cost-effective robotic system specifically designed for the local context. It integrates essential features like real-time mapping and human detection to support more efficient rescue operations on the ground.

By using low-cost components such as the Raspberry Pi and leveraging open-source tools like ROS and PyTorch, the system remains both accessible and flexible. Its AI-powered vision allows for rapid identification of survivors, while SLAM-based mapping supports autonomous navigation through complex environments. This solution is not only technologically practical but also directly responds to an urgent humanitarian challenge improving the speed and effectiveness of disaster response in regions where such support is needed most.

2.5 Summary

This chapter has examined previous developments in the field of disaster robotics, focusing on how technologies like artificial intelligence, SLAM, and real-time detection are reshaping emergency response strategies. While substantial progress has been made globally, there is still a noticeable gap when it comes to applying these technologies in low-income countries. The review highlighted the urgent need for a reliable and autonomous system that can locate survivors and support rescue teams with timely, actionable information. Building on existing research, this study puts forward a solution tailored to the realities of Uganda's disaster response landscape one that emphasizes affordability, adaptability, and real-time performance in the face of local challenges.

Chapter 3

Methodology

3.1 Introduction

In response to the ongoing challenges of delayed disaster response and the difficulty in quickly locating survivors, this project presents a robotics-based solution integrated with IoT capabilities. The system has been specifically designed to function in hazardous and difficult-to-access environments, with the aim of detecting trapped individuals and relaying critical information to emergency teams such as the Red Cross. This chapter outlines the design methodology behind the system, the tools and components used in its development, and the approach taken for testing and evaluation. The study focuses on disaster-prone regions in Uganda, with particular attention to high-risk areas like Bududa. The robot is equipped with an array of advanced sensors including a GPS unit that enable it to navigate independently and provide real-time situational updates during search and rescue missions.

3.2 Research Methodology

3.2.1 System Design and Architecture

Structure

The robotic dog has been designed with a strong emphasis on balance and mobility, allowing it to move smoothly across rough or uneven terrain. Its jointed legs are specifically constructed to handle obstacles like debris, giving it the flexibility to climb and adjust its posture as needed. Precise movement is controlled by servo motors, which enable the robot to maintain stability and adapt to changing surfaces, ensuring it can operate effectively in demanding environments.

Sensor Integration

The robot relies on a combination of sensors that work in unison to help locate survivors and determine the safest paths for rescue.

1. Ultrasonic Sensors – These sensors emit high-frequency sound waves to identify nearby objects and obstacles. They play a critical role in avoiding collisions and allow the robot to move safely through narrow or cluttered spaces, especially in conditions where visibility is poor. By continuously assessing the surrounding environment, these sensors provide the robot with the spatial awareness needed to navigate debris-filled or confined areas.

2. GPS Module: The GPS module receives satellite signals to determine the robot's precise real-time location. It aids in route tracking and provides geolocation data to rescue teams to coordinate follow-up operations.

3. Night Vision Camera Module: The camera enables visual detection of survivors and environmental mapping. Combined with AI-based image processing, it improves situational awareness and enhances decision-making.

4. GPS Module: The GPS module is used to measure the Robots current location also aiding in SLAM algorithm.

Control System and Communication

The Raspberry Pi microcontroller handles data processing, navigation logic, and motion control. Wireless communication is managed through WiFi to transmit live video and mapping data to a remote dashboard for operators and rescue teams.

Software Development

1. Navigation and Obstacle Avoidance: Implemented using SLAM algorithms to enable real-time map creation and self-localization. This supports autonomous exploration of unknown terrain and enhances route precision.

   Planning for Rescue Teams: The robot should be able to calculate and recommend the most efficient and safest routes for rescuers to reach identified survivors, using the A* algorithm to generate optimal paths through disaster-affected areas.

2. Survivor Detection: AI-based techniques analyze visual data and sound patterns to identify signs of human presence. The integration of live sensor data allows the robot to adapt and respond dynamically in active rescue scenarios.

3. User Interface: A web and mobile interface visualizes mapping data, provides alerts for survivor detection, and shares geotagged location information with response teams.

3.2.2 Data Collection

Sensor data is continuously collected and stored in a centralized database, including GPS coordinates, visual footage, and environmental mapping. This data supports immediate rescue operations and future system improvements.

The recorded data will also serve as training input for machine learning models. These models aim to improve survivor detection accuracy and refine navigation algorithms.

With time, pattern recognition will allow the robot to predict optimal movement strategies and detect subtle indicators of human presence more reliably, ensuring continuous performance enhancement over successive missions.

3.2.3 Testing and Validation

Simulated Testing:

Conducted in artificial environments simulating rubble piles and collapsed structures.

Performance Metrics:

1. Detection Accuracy: Measures the success of identifying visible and hidden survivors. Accurate detection ensures fewer false alarms and increases mission success rates.
2. Navigation Efficiency: Evaluates the robot's ability to move autonomously through cluttered and unstable terrain without external control, ensuring broader and faster coverage.
3. Battery Life: Measures energy efficiency under continuous operation. Strategies such as sensor prioritisation and low-power modes will be applied to maximise uptime.

Chapter 4

System Study, Analysis and Design

4.1 System study

4.2 Overview of the System

The proposed system is an AI-powered robotic dog designed to support search and rescue operations in post-disaster environments such as landslides, floods, and collapsed structures. The robot is capable of autonomous movement, obstacle avoidance, and survivor detection using onboard sensors and AI algorithms. It is equipped with ultrasonic sensors, a night vision camera with YOLO-based human detection, a GPS module, and a sound direction sensor.

The robot operates using a Raspberry Pi-based control unit and communicates with a Flask-based web interface that enables live video streaming, manual control, and mapping. A SLAM algorithm allows the robot to build a real-time map of the environment, while the A* algorithm generates the most efficient rescue path not only for the robot but also for human responders. When a survivor is detected, a red marker is placed on the interface map, and data about the location is logged. The system offers both autonomous and controlled mechanisms.

4.3 Functional requirements

1. Autonomous Navigation: The robot must autonomously navigate complex, GPS-denied environments such as collapsed buildings or rural disaster zones.
2. Survivor Detection: The system must detect survivors using AI-based analysis of visual (camera) and audio (sound direction sensor) data.
3. Real-Time Mapping: The robot must create a real-time map of the environment using SLAM (Simultaneous Localization and Mapping).
4. Obstacle Avoidance: Ultrasonic sensors must detect and avoid obstacles dynamically, allowing the robot to traverse debris safely.

5. Data Transmission: The system must transmit live camera feeds, survivor geolocation, and mapping data to a remote user interface via Wi-Fi.

6. User Interface Interaction: Operators must be able to view data and control the robot via a Flask-based web dashboard.

7. Path Generation for Rescuers: The robot must compute and suggest the best routes for human rescuers to reach detected survivors using A* pathfinding.

4.4 Non-Functional Requirements

1. **Portability:** The robot should be easy to carry and small enough to fit into tight spaces, making it suitable for quick deployment by rescue teams, especially in areas that are remote or difficult to navigate.

2. Durability:

3. Portability: The robot should be lightweight and compact, allowing rescue teams to carry and deploy it easily especially in tight or hard-to-reach locations.

4. Durability: The system needs to be built to handle tough environments. It should be able to operate over debris, absorb minor impacts, and withstand vibrations, dust, and other harsh conditions typical of disaster zones.

5. Real-Time Responsiveness: The robot must react quickly to sensor input detecting obstacles, identifying survivors, and sending live data to rescue teams without noticeable delay.

6. Battery Efficiency: The system should be capable of running continuously for a minimum of three hours. To conserve energy, it should make use of power-saving methods such as prioritizing essential functions or entering idle mode when inactive.

7. User Interface Simplicity: The web interface must be user-friendly and intuitive, allowing non-technical users to monitor the robot, access video feeds, and view mapped data without requiring extensive training.

8. System Scalability: The architecture must support future upgrades in hardware or software, including new sensors, updated AI models, or additional interface features.

4.5 System Architechture

The system architecture is designed to support both autonomous and semi-autonomous modes of operation for a robotic dog functioning in post-disaster environments. It is composed of hardware and software modules that work together to enable the robot to detect survivors, navigate dangerous terrains, and transmit data in real time to rescue teams. The architecture emphasizes modularity, allowing individual components such as sensors, navigation logic, and control interfaces to be developed and updated independently.

At a high level, the system consists of five core layers:

1. Sensing Layer – This includes ultrasonic sensors, a GPS module, a night vision camera, and a sound direction sensor. These components collect data on surroundings, obstacles, and human presence.

2. Processing Layer – The Raspberry Pi 4 acts as the system's central processor, running navigation algorithms (SLAM and A*), YOLO-based human detection, and data fusion routines.

3. Control Layer – This layer uses servo motors and PID control (via the PiDog kit) to handle robotic leg movements and ensure stable mobility over debris.

4. Communication Layer – WiFi modules are used to send live sensor data and video feeds to a remote Flask-based web application.

5. User Interface Layer – This is a web and mobile-accessible interface where rescue teams can view camera feeds, maps, and survivor locations, and also manually control the robot if necessary.

4.6 Hardware Design

The robotic system consists of several integrated components chosen for performance, efficiency, and compatibility. The following table outlines the main hardware components, their technical specifications, and power requirements.

Table 4.1: Hardware Components and Specifications

Component	**Specification**	**Power Requirement**
Raspberry Pi 4	64-bit quad-core ARM Cortex-A72 processor @ 1.5GHz, 4GB RAM	5V @ 3A
SunFounder PiDog Kit	Servo-controlled quadruped frame, 12 DOF with metal-geared servo motors	6V 7.4V (external battery)
Ultrasonic Sensors (x2)	HC-SR04, detection range: 2cm 400cm, angle: 15°	5V @ 15mA each
Night Vision Camera	5MP IR camera module with 1080p video recording, infrared LEDs	5V @ 250mA
GPS Module	NEO-6M GPS with antenna, accuracy: 2.5m, update rate: 1Hz5Hz	3.3V 5V @ 45mA
Sound Direction Sensor	3-microphone array, analog output, sound source localization capability	5V @ 20mA
Lithium Battery Pack	7.4V 2-cell Li-ion pack, 2600mAh capacity	Supplies main power to all modules
WiFi Module (Built-in)	802.11 b/g/n integrated with Raspberry Pi 4	Included in Pi's power draw

Table 4.2: Complete Hardware Connections and Pin Configuration

Component	**Function**	**GPIO Pins / Interface**	**Power (V)**	**Ground (GND)**
Left Front Leg Servos	Hip and knee movement	GPIO 2, GPIO 3	6V7.4V (via HAT)	Common GND
Right Front Leg Servos	Hip and knee movement	GPIO 7, GPIO 8	6V7.4V (via HAT)	Common GND
Left Hind Leg Servos	Hip and knee movement	GPIO 0, GPIO 1	6V7.4V (via HAT)	Common GND
Right Hind Leg Servos	Hip and knee movement	GPIO 10, GPIO 11	6V7.4V (via HAT)	Common GND
Head Servos	Yaw, Roll, Pitch	GPIO 4, GPIO 6, GPIO 5	6V7.4V (via HAT)	Common GND
Ultrasonic Sensor 1	Obstacle detection	Trigger: GPIO 0, Echo: GPIO 1	5V	GND
Ultrasonic Sensor 2	Obstacle detection	Trigger: GPIO 2, Echo: GPIO 3	5V	GND
GPS Module	Location tracking	TX/RX on HAT UART interface	3.3V 5V	GND
Night Vision Camera	Visual input for detection	CSI Camera Port (Ribbon cable)	5V	GND
Sound Direction Sensor	Survivor audio detection	I/O via analog/audio pins (connected to Pi or HAT)	5V	GND
WiFi Module (Built-in)	Wireless communication	Integrated in Raspberry Pi 4	5V (Pi bus)	GND
Raspberry Pi 4	Main processor	All peripherals connected via GPIO, CSI, and UART	5V @ 3A	GND

4.7 SOFTWARE DESIGN

The software system was developed using Python on Raspberry Pi OS, leveraging a wide range of open-source libraries and frameworks. The table below summarizes the key tools and their respective roles in the system.

Table 4.3: Software Tools and Libraries Used

Software / Library	**Purpose / Function**
Python 3.9	Primary programming language for system development, including control logic, AI integration, and web services.
Flask	Used to create the web-based user interface for live video, map updates, and manual robot control.
OpenCV	Provides real-time camera handling, image processing, and integration with YOLO object detection.
PyTorch	Runs the YOLO deep learning models for visual survivor detection.
RPi.GPIO	Enables GPIO pin control for ultrasonic sensors and other directly connected components.
pySerial	Used to communicate with the GPS module over UART.
NumPy / Pandas	Support numerical computation, data logging, and basic analytics.
RTAB-Map (via ROS)	Implements SLAM for real-time localization and environment mapping.
Geopy	Converts raw GPS data into human-readable location coordinates.
Raspberry Pi OS	Operating system running on the Raspberry Pi, supporting Python development and system libraries.
VNC Server	Provides remote desktop access to the Raspberry Pi for setup and monitoring.

4.8 Use Case Analysis

The use case analysis outlines how the two primary actors the robot and the rescue operator interact with the system in a search and rescue mission. The robot performs core autonomous functions, including navigation, obstacle avoidance, survivor detection, and transmitting data. On the other hand, the rescue operator interacts with the system through a web-based interface that allows them to monitor the robot's location, view live video feeds, receive alerts, and manually control the robot if needed.

Actors:

1. Robot – An autonomous mobile system performing navigation, detection, and data transmission.

2. Rescue Operator – A human user monitoring and guiding the robot remotely.

Use Cases:

1. Survivor Detection: The robot uses onboard AI models and sensors to identify potential survivors based on visual and auditory cues. Detected survivor locations are marked and sent to the rescue operator.

2. Obstacle Navigation: Using ultrasonic sensors and SLAM, the robot avoids obstacles while exploring unknown and cluttered environments.

3. Location Tracking: The robot constantly updates its position using GPS and SLAM-based mapping and transmits that data to the rescue team for situational awareness.

4. Data Upload to User Interface: All captured data, including live video, maps, and survivor alerts, is sent to the operator's web interface in real time.

5. Manual Control (Override): In cases where autonomous operation is hindered, the rescue operator can switch to manual mode using the interface to directly control the robot's movement and behavior.

4.9 System Interaction

The effectiveness of the robotic search and rescue system depends on the smooth coordination of its various components, each playing a role in supporting real-time decision-making during critical missions. At the core of the system is the robot itself, which constantly collects data through onboard sensors including ultrasonic detectors, a GPS module, a camera, and directional microphones. This information is processed instantly by the Raspberry Pi unit inside the robot, using artificial intelligence and SLAM (Simultaneous Localization and Mapping) techniques to determine its location, recognise obstacles, and identify potential survivors.

After processing, the relevant data is transmitted wirelessly to a remote user interface built using the Flask web framework. Through this dashboard, emergency personnel can monitor live video feeds, track the robot's location on a digital map, and receive immediate alerts when a survivor is detected. The interface also allows switching between autonomous and manual control. In manual mode, the operator can guide the robot directly, while in autonomous mode, it navigates independently using internal pathfinding and environmental awareness algorithms.

This dual-mode setup adds flexibility to the system, allowing it to adapt to the unpredictable nature of disaster sites. Autonomous control reduces the burden on human operators, while manual override ensures that rescuers can intervene when necessary to adjust strategy or avoid hazards. The continuous flow of information between the robot, its sensors, the onboard processor, and the user interface forms a closed feedback loop. This integration ensures that all parts of the system remain in sync, enhancing both the speed and accuracy of rescue operations. By enabling timely, informed responses, the system strengthens coordination between technology and human teams, ultimately increasing the chances of saving lives in high-stakes situations.

Chapter 5

Presentation of Results

5.1 Introduction

This chapter highlights the final outcomes of the developed robotic search and rescue system by presenting the main features of its interface and explaining how the system functions under real-time operating conditions. It includes visual examples of the Flask-based web platform, which is used to monitor the robot's activity, control its movement, and access live video and mapping data. The chapter also provides an overview of the software and hardware environment used during development, detailing the programming tools, key libraries, and hardware components that made up the system's architecture.

5.1.1 System Set up and Operation:

To operate the robotic rescue system and access its web-based platform, the following steps must be followed:

1. Connect to Network: Connect to the Network: First, connect the robot to an available Wi-Fi hotspot. This connection enables communication between the robot and the device used for monitoring and control.

   Enable VNC Access: On the Raspberry Pi, open the terminal and enter sudo raspi-config. Navigate to the interface options and enable VNC (Virtual Network Computing) to allow remote desktop access from another device.

2. Launch the Web Application: Once connected, open the folder named RESCUE-DOG located on the Raspberry Pi. Inside this folder, run the file app.py. This script starts the Flask-based web server that powers the system's interface.

3. View the Interface: After launching app.py, a local URL (typically http://0.0.0.0:5000 or similar) will be displayed in the terminal. Copy and paste this link into any web browser on a device connected to the same network. This opens the platform's user interface where live video, maps, and survivor alerts can be monitored.

4. Access Captured Images: Images of detected humans are automatically stored during operation. To view these, open the snapshots folder within the RESCUEDOG directory. Each image is geotagged and timestamped for rescue team reference.

5.1.2 How the Robot Works

The robot follows a multi-step process from initialization to execution of rescue operations. The operational workflow is as follows:

1. **Initialization:** Once powered on, all modules including sensors, motors, and communication interfaces are initialized. This is confirmed through terminal logs displayed during startup (see Figure 7.1).

2. **Interface Launch:** The user connects to the robot via VNC and launches the `app.py` script from the `RescueDog` folder. This opens the robot's Flask-based web interface on a browser (Figure 7.2).

3. **Camera and Map Display:** The interface displays a live feed from the robot's night vision camera and a SLAM map that updates in real time based on the robot's movement and environmental sensing.

4. **Autonomous and Manual Modes:** The operator can toggle between manual control and autonomous navigation using the web dashboard. In autonomous mode, the robot uses SLAM and A* pathfinding to move and avoid obstacles.

5. **Survivor Detection:** When a survivor is detected using the camera and AI detection model, the system highlights the person in red and logs the location (Figure 7.3). A red dot appears on the SLAM map to mark their position (Figure 7.4).

6. **Snapshot Storage:** All images of detected humans are saved in the `snapshots` folder under the `RescueDog` directory, which can be accessed for verification and evidence (Figure 7.5).

7. **Optimal Path Mapping:** Based on the SLAM data, an optimal route is computed using A* and shown on the grid-based path map (Figure 7.6).

Chapter 6

Limitations, Recommendations and Conclusion

6.1 Introduction

This chapter presents a conclusion to the project with limitations, recommendations and Future enhancements.

6.2 Limitations

Despite the success of the developed robotic search and rescue system, several limitations were identified during testing and deployment:

- **Environmental Limitations:** The robot is primarily designed for dry terrain and may face difficulties in muddy or heavily waterlogged environments.
- **Limited Battery Life:** Extended search durations are constrained by battery capacity, especially with continuous camera and sensor usage.
- **Processing Power Constraints:** The Raspberry Pi has limited GPU performance, which can impact real-time image processing during complex operations.
- **Sensor Limitations:** The current system cannot detect deeply buried survivors due to the absence of advanced sensors like thermal cameras or ground-penetrating radar.
- **Network Dependency:** Wi-Fi-based communication is limited in range and may be unreliable in remote disaster zones lacking infrastructure.

6.3 Recommendations

To address the above limitations and enhance system effectiveness, the following recommendations are proposed:

- Integrate higher-capacity batteries and implement intelligent power-saving features.
- Upgrade sensors to include thermal imaging and LiDAR for improved mapping and detection.
- Add support for cellular (5G) or LoRa-based communication for better range and reliability.
- Implement autonomous recovery protocols such as return-to-base and system restart features.
- Explore swarm deployment of multiple robots for coordinated, large-scale rescue coverage.

6.4 Future Enhancements

Several future improvements can be implemented to extend the functionality and real-world applicability of the system:

- **Scaling to Dog Size:** Enhancing the robot's form factor to match the size of a real dog can improve stability and payload capacity.
- **Drone Integration:** Adding an aerial drone can provide aerial views, complementing ground-level detection and navigation.
- **GPR Sensor Integration:** Including a ground-penetrating radar system would allow detection of buried survivors and objects.
- **Return-to-Origin Functionality:** A built-in autonomous return protocol when battery is low would enhance operational safety and reliability.

6.5 Conclusion

This project successfully designed, built, and tested a robotic dog system capable of supporting search and rescue operations in post-disaster scenarios. The robot combines AI-based human detection, SLAM-powered navigation, and real-time communication through a web-based interface, demonstrating how affordable, off-the-shelf technologies can be effectively integrated to address urgent humanitarian needs. Despite a few limitations, the system lays a solid groundwork for continued development and future deployment.

With further enhancements particularly in energy efficiency, advanced sensor integration, and more robust communication capabilities the robot could evolve into a valuable asset for emergency responders. Its practical design and adaptability make it especially promising for use in regions where disaster risks are high and access to advanced rescue tools is limited.

Bibliography

[1] R. R. Murphy, "Trial by fire [rescue robots]," *IEEE Robotics & Automation Magazine*, vol. 11, no. 3, pp. 50–61, 2004.

[2] R. Madhavan, E. R. Messina, and J. S. Albus, "Map-based autonomy for mobile robots in urban search and rescue," *IEEE Intelligent Systems*, vol. 19, no. 2, pp. 17–24, 2004.

[3] O. of the Prime Minister, "Uganda national emergency coordination and operations centre reports," 2020, accessed: 2025-04-10. [Online]. Available: https://opm.go.ug/emergency-coordination

[4] J. Twesigye *et al.*, "A situational analysis of priority disaster hazards in uganda: Findings from a hazard and vulnerability assessment," *Journal of Disaster Risk Studies*, vol. 6, no. 2, pp. 1–9, 2014.

[5] S. Kabuye, V. Namutebi, and J. Lubega, "Coordination and disaster response: A humanitarian organizational network perspective in uganda," *ResearchGate*, 2023, accessed: 2025-04-10. [Online]. Available: https://www.researchgate.net/publication/379763510

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[7] R. Mur-Artal, J. Montiel, and J. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[8] S. Behnke, "Robot rescue teams: From competitions to real-world deployment," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2016.

[9] S.-J. Chung *et al.*, "Navigation and mapping for autonomous search and rescue drones in cluttered environments," *Science Robotics*, vol. 6, no. 52, 2021.

[10] NVIDIA, "Jetson nano developer kit," 2019, accessed: 2025-04-11. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[11] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[12] R. R. Murphy, "Trial by fire [rescue robots]," *IEEE Robotics & Automation Magazine*, vol. 11, no. 3, pp. 50–61, 2004.

[13] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

Chapter 7

Appendix

7.1 Appendix A: Images

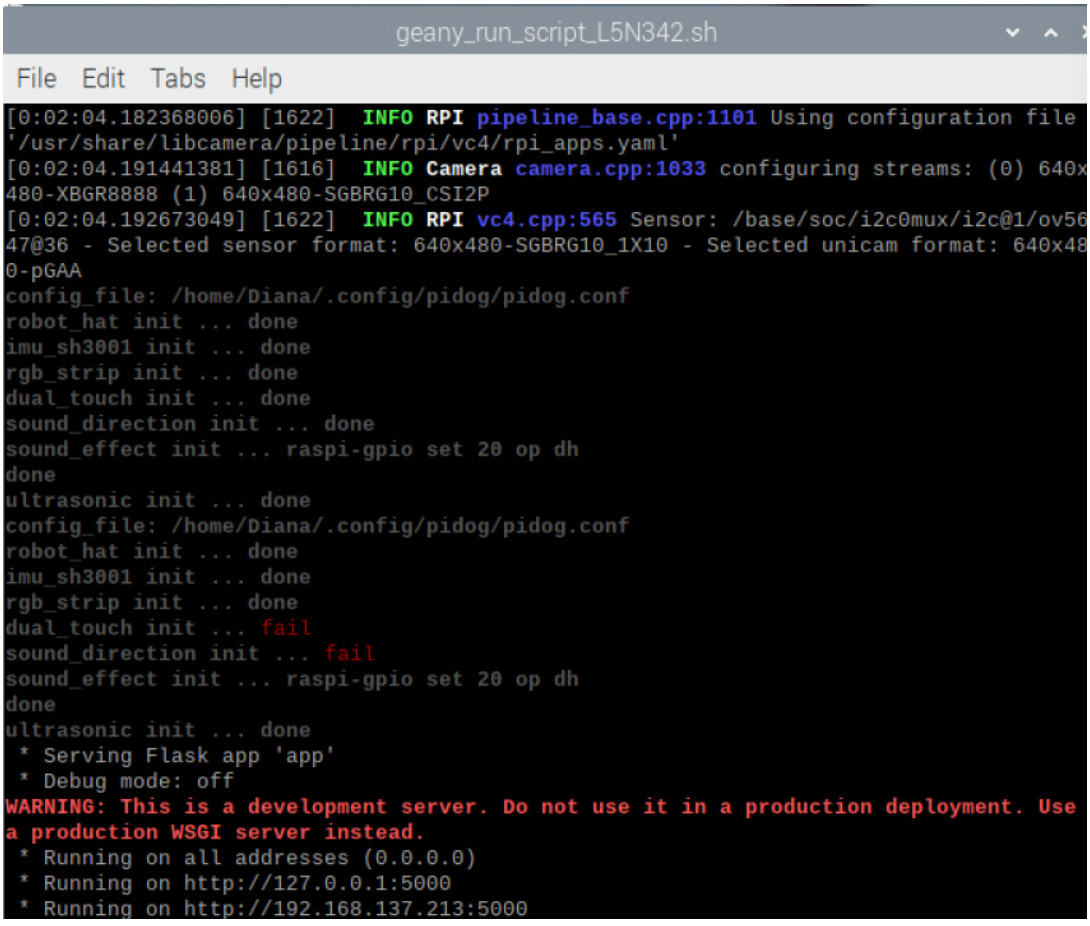

Figure 7.1: System terminal log showing module initialization

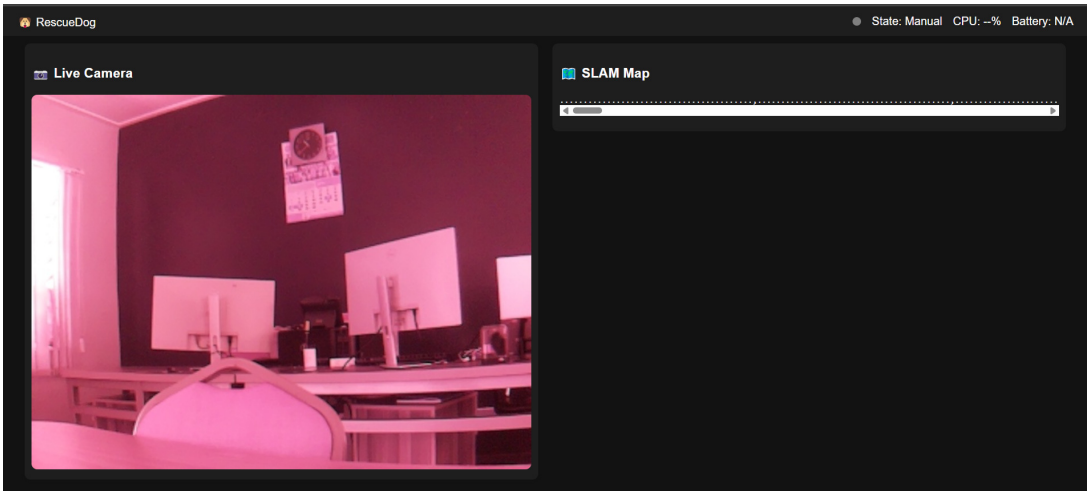

Figure 7.2: Flask web interface displaying live camera and SLAM map

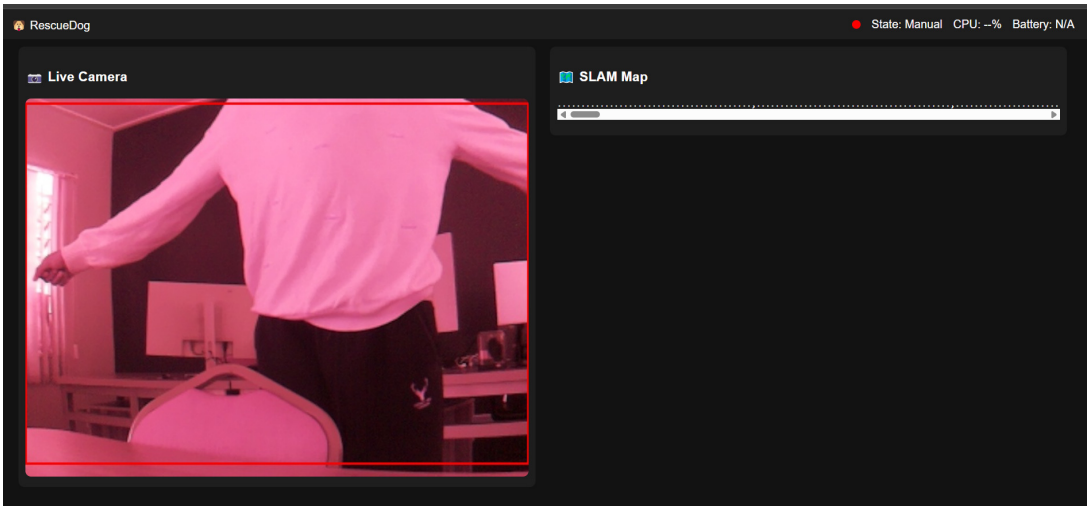

Figure 7.3: Detected survivor highlighted by the AI model

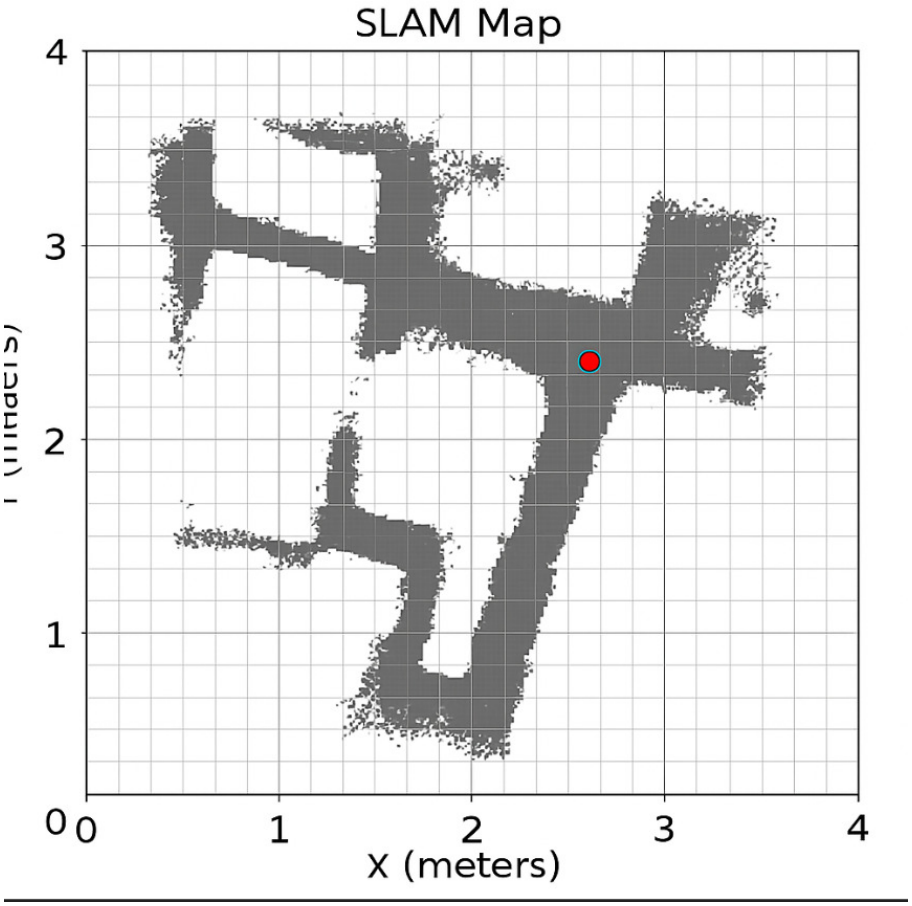

Figure 7.4: SLAM map with survivor location marked in red

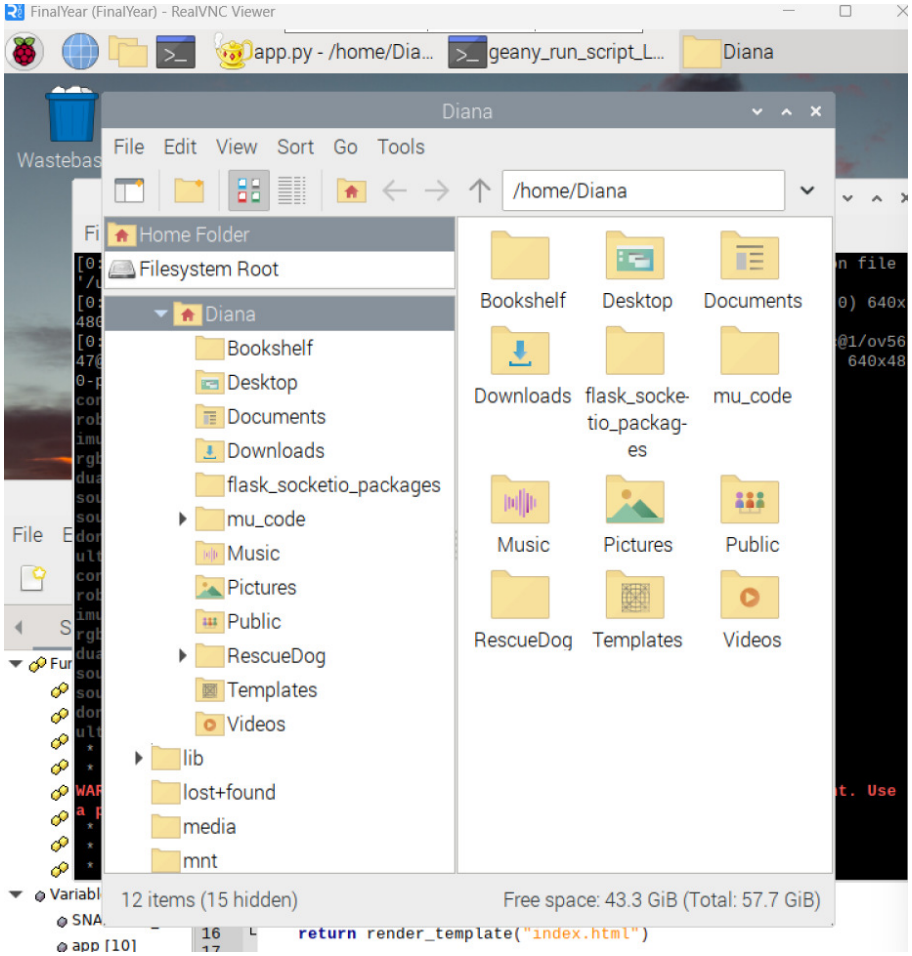

Figure 7.5: Snapshot directory with captured images of survivor

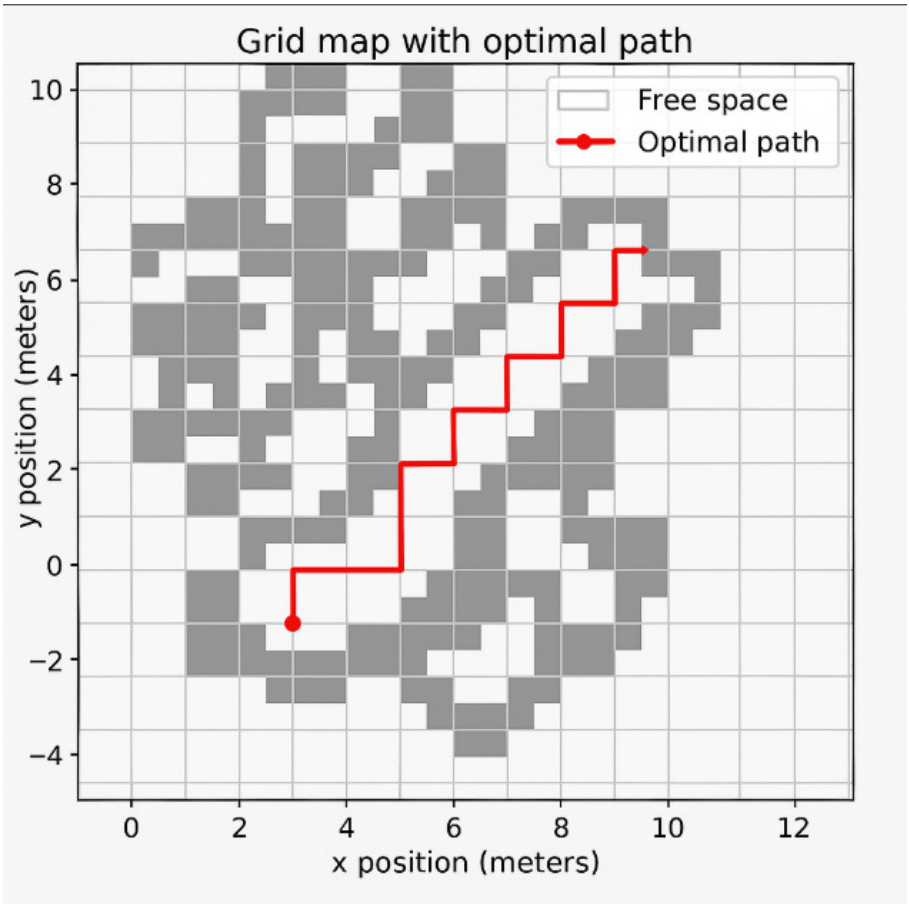

Figure 7.6: Grid-based map showing computed optimal path

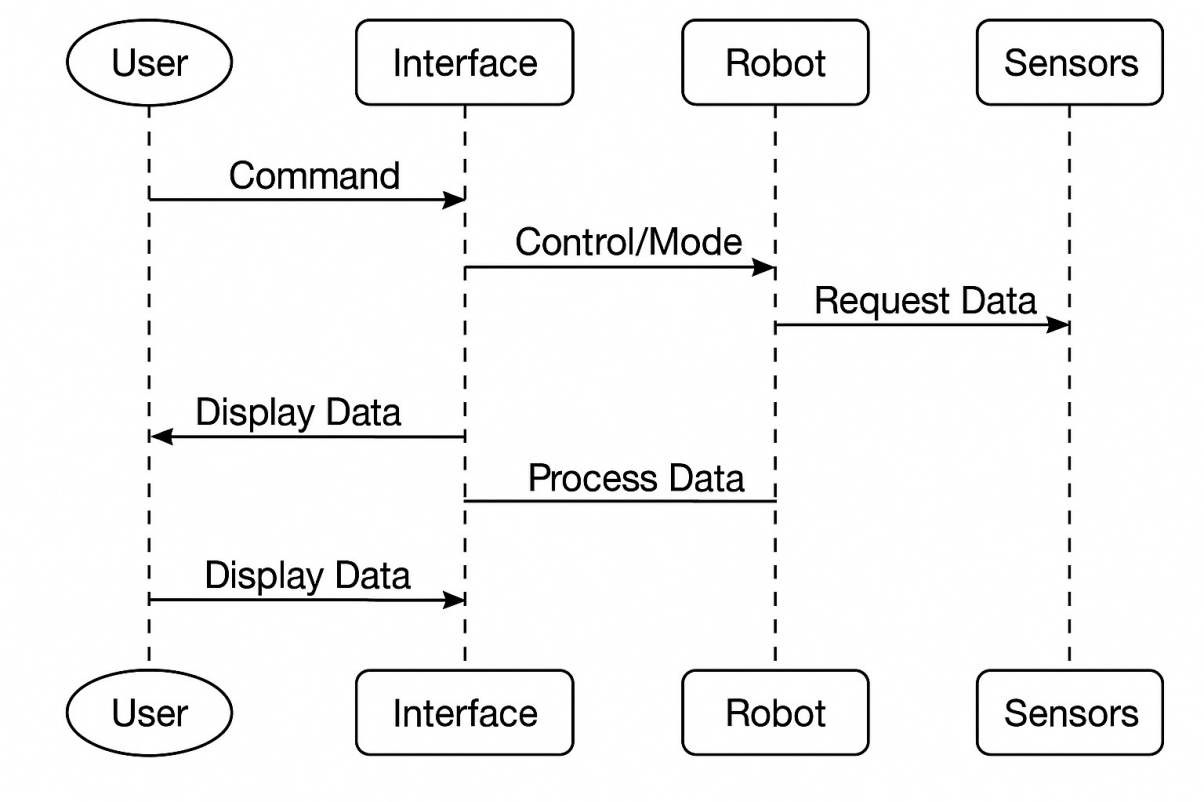

Figure 7.7: Sequence Diagram Showing System Interaction

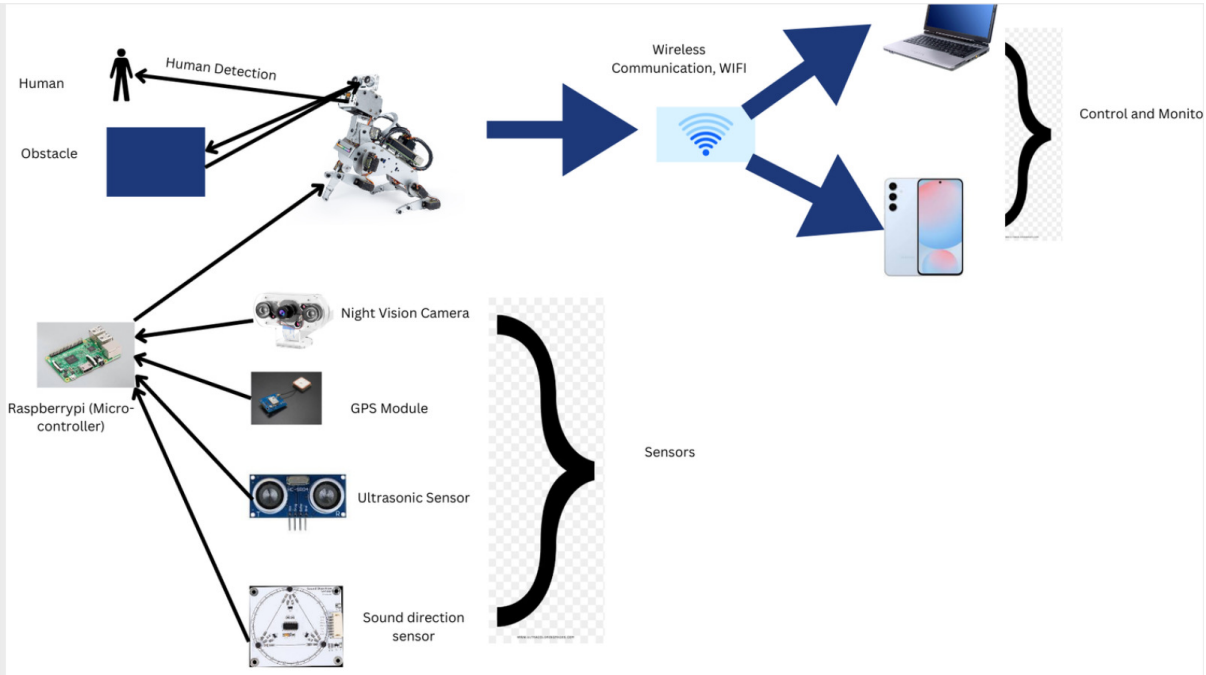

Figure 7.8: System Flow

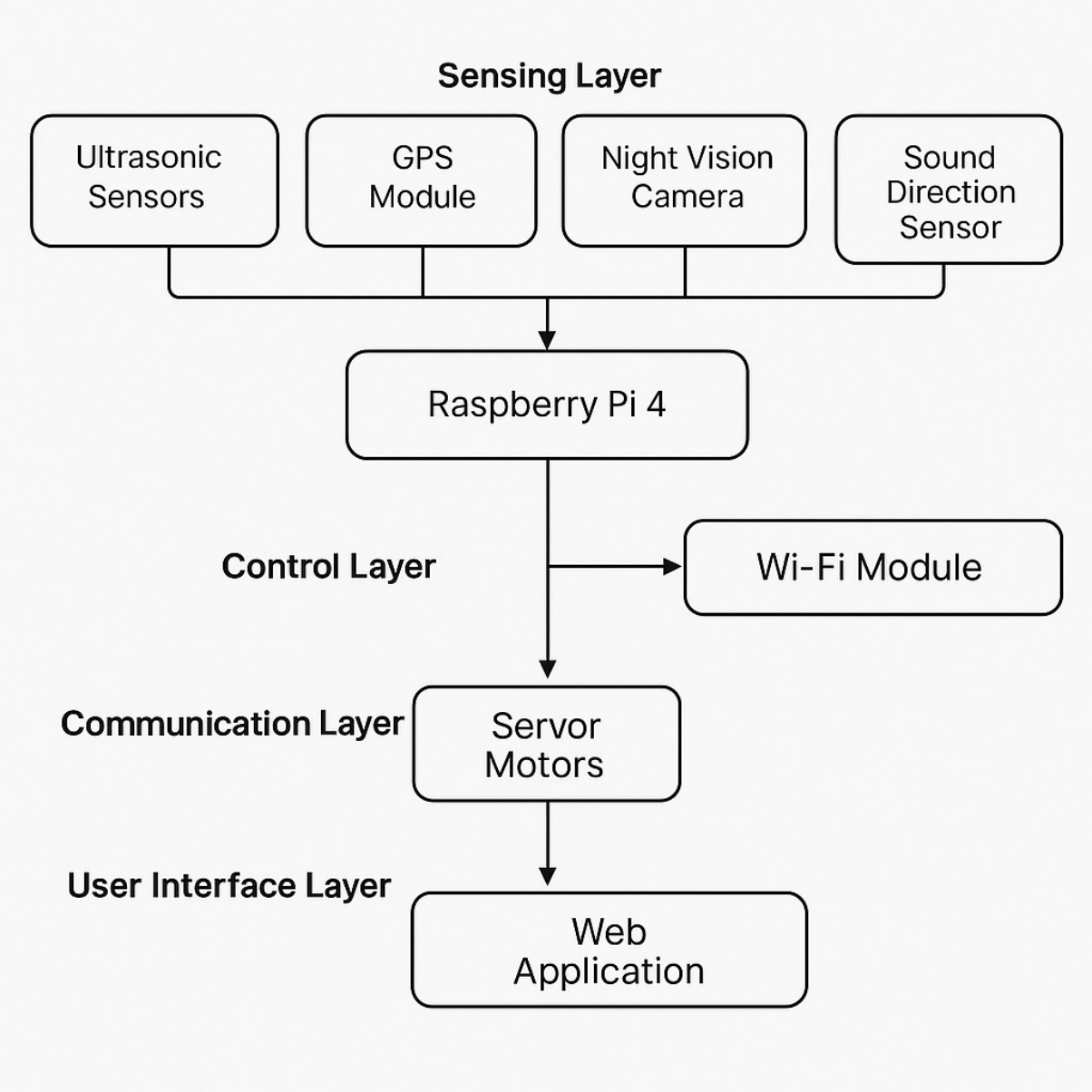

Figure 7.9: High-Level Block Diagram of the Robotic System Architecture

Figure 7.10: Image of robot in field

Glossary of Key Terms

A* Algorithm: A pathfinding and graph traversal algorithm used to determine the shortest and most efficient route between two points. It is used by the robot to generate safe navigation paths in disaster zones.

AI (Artificial Intelligence): The capability of a machine to mimic human cognitive functions such as learning, problem-solving, and pattern recognition. In this project, AI is used for survivor detection and autonomous decision-making.

Autonomous Navigation: The robot's ability to explore and move through an environment without human control, using sensors, SLAM, and AI-based decision systems.

Battery Efficiency: Refers to the system's ability to conserve energy and prolong operational time, especially during continuous rescue missions.

Control Layer: Part of the system architecture that manages the physical movement of the robot using motors and control algorithms like PID.

Durability: The robot's ability to operate reliably under harsh conditions, such as debris-filled terrain or exposure to dust and vibration.

Environmental Mapping: The process of constructing a digital representation of the robot's surroundings to facilitate navigation and path planning.

Flask: A lightweight Python web framework used to build the system's user interface for monitoring live video, maps, and controlling the robot.

Geolocation: The identification or estimation of the real-world geographic location of an object, such as the robot or a detected survivor.

Geotagging: Attaching location metadata to captured images or videos, which helps responders know where survivors were detected.

GPS (Global Positioning System): A satellite navigation system used by the robot to determine its physical location and aid in route tracking.

Hardware Layer: The physical components of the robotic system, including sensors, microcontrollers, motors, and structural frames.

Human Detection: The use of AI and image recognition technologies (e.g., YOLO) to identify people in the robot's camera feed.

IoT (Internet of Things): A network of physical devices embedded with sensors, software, and connectivity to exchange data. In this project, IoT enables communication between the robot and remote operators.

LIDAR (Light Detection and Ranging): A sensor that measures distance using laser light. It can be used for advanced obstacle detection and mapping.

Manual Override: A feature that allows a human operator to take control of the robot remotely when autonomous systems fail or are insufficient.

Night Vision Camera: A specialized camera module equipped with infrared capability that enables visual operation in low-light or no-light conditions.

Object Detection: A computer vision technique used to locate and identify objects (such as humans) within an image or video stream.

Path Planning: The process of computing a viable path from the robot's current position to a target destination while avoiding obstacles.

PID Controller (Proportional–Integral–Derivative Controller): A control system that adjusts motor output to achieve smooth and accurate movement, particularly useful for robot leg motion.

Portability: Refers to the robot's lightweight and compact design, which allows easy transport and deployment in remote or inaccessible locations.

Processing Layer: The component of the architecture responsible for running the AI models, processing sensor data, and making real-time decisions.

Raspberry Pi: A small, low-cost, single-board computer used to process data, control hardware components, and run the robot's software.

Rescue Window (72-Hour Rule): The critical time period following a disaster during which the likelihood of rescuing survivors is highest typically the first 72 hours.

Robot Operating System (ROS): A flexible framework for writing robot software that provides tools and libraries for developing robot applications.

RTAB-Map: A graph-based SLAM technique used to build real-time maps of the environment while keeping track of the robot's position.

Sensor Fusion: The integration of data from multiple sensors (e.g., GPS, camera, ultrasonic) to produce a more accurate and comprehensive understanding of the environment.

Servo Motors: Electric motors with feedback systems that enable precise control of angular movement, used for joint articulation in the robot's limbs.

Simulated Environment: An artificial setting created to imitate real disaster conditions for testing and evaluating the robot's performance.

SLAM (Simultaneous Localization and Mapping): A method by which the robot builds a map of an unknown environment while simultaneously tracking its own location within it.

Software Layer: The part of the system that includes algorithms, control logic, and user interface components.

Survivor Detection: The use of cameras, AI, and sound analysis to identify signs of human presence in disaster-struck areas.

System Scalability: The ability of the robotic system to be expanded or upgraded by adding new sensors, improving software, or enhancing performance.

Ultrasonic Sensors: Sensors that use high-frequency sound waves to detect nearby objects and measure distance, aiding in obstacle avoidance.

User Interface (UI): The platform (usually web-based) through which rescue operators interact with the robot, access maps and video feeds, and issue commands.

VNC (Virtual Network Computing): A tool that allows users to remotely access and control the Raspberry Pi's desktop from another device over a network.

WiFi Module: A built-in or external component that allows wireless communication between the robot and the remote user interface.

YOLO (You Only Look Once): A deep learning algorithm used for fast and accurate object detection in real-time, especially effective for identifying people in disaster zones.