Graduation Project Report

Faculty of Engineering & Information Technology
Department of Computer Engineering

Graduation Project under title:

Robotic Nurse System

Done by:

Nora Qanazea Saifi (ID: 11923689)
Ghada Qanazea (ID: 11924838)

Supervised by:

Dr. Saed Tarapiah

Presented in partial fulfillment of the requirements for the
Bachelor Degree in Computer Engineering

2025/2026

Acknowledgments

The authors would like to express their sincere gratitude to their families for their continuous support, patience, and encouragement throughout the duration of this project.

Special thanks are extended to peers and colleagues for their constructive discussions, technical feedback, and collaborative spirit, which contributed positively to the development of the system.

The authors would also like to acknowledge the Faculty of Engineering and Information Technology and the university administration for providing the academic environment, facilities, and resources necessary to carry out this work.

Deep appreciation is extended to the supervising doctor, Dr. Saed Tarapiah, for his guidance, technical insight, and academic support throughout the project.

Finally, the authors would like to thank the mechatronics engineering student team for their valuable assistance in the mechanical casing design and 3D printing support, which played an important role in the physical realization of the system.

Disclaimer

Abstract

This project presents the design and implementation of a Robotic Nurse System intended to assist with basic flu assessment and wound treatment procedures. The system integrates sensor-based hardware execution, event-driven software coordination, and image-based wound classification within a unified, modular architecture. A Raspberry Pi is used for high-level decision-making, workflow control, and user interaction, while an Arduino Mega handles all low-level actuation and sensor verification to ensure deterministic and safe physical behavior.

The system incorporates multiple sensing modalities, including temperature, pulse oximetry, proximity detection, and vision-based wound classification using a YOLO-based model trained externally on a GPU-enabled environment. All actions are triggered explicitly through user input or hardware-originated events, with no autonomous or background execution.

Experimental results demonstrate that the system operates reliably within its design constraints and achieves the intended functional objectives using primarily hobby-grade components and a limited training dataset. While further refinement is required for medical deployment, the project validates the feasibility of an event-driven robotic assistance framework and provides a scalable foundation for future development.

Contents

List of Figures

List of Tables

1 Introduction

1.1 General Background

Healthcare services are facing continuous challenges due to the increasing number of patients, limited medical staff, and the need for faster and more efficient medical assistance. Nurses spend a large portion of their time performing routine tasks such as checking vital signs, providing basic first aid, dispensing simple medications, and organizing medical supplies. Although these tasks are essential, they are repetitive and time-consuming, which reduces the time available for critical patient care.

With the advancement of technology, especially in robotics, embedded systems, and automation, it has become possible to develop intelligent systems that support healthcare workers in their daily activities. Robotic assistance systems can improve efficiency, reduce human errors, and provide consistent and reliable services. These systems are particularly valuable in environments such as hospitals, clinics, and emergency centers where rapid response and accuracy are required.

The Robotic Nurse project was developed to demonstrate how modern engineering technologies can be applied to assist in basic medical services. The system acts as a smart medical station that helps users measure vital signs, identify simple injuries, dispense medical supplies, and manage stock levels. It combines hardware components such as sensors, motors, and microcontrollers with software components including a graphical user interface and image processing techniques. This project reflects a practical application of robotics and embedded systems in healthcare.

1.2 Objectives of the Work

The main goal of this project is to design and build a robotic nurse that can assist in performing basic nursing tasks in a simple and efficient manner. The system is designed to measure vital signs such as body temperature, heart rate, and blood oxygen level, and to detect minor hand injuries such as cuts or burns using a camera and image processing techniques.

The system also automates the dispensing of essential medical supplies including water, alcohol, ointment, gauze, plaster, and temperature medication. A tablet-based user interface is developed to ensure ease of use, supporting both Arabic and English languages with clear instructions and smooth navigation. The project integrates an Arduino Mega and a Raspberry Pi to ensure reliable hardware control and effective coordination between system components. Additionally, a stock management feature is implemented to track available medical items, update quantities automatically after each use, and allow manual refilling when required. The final objective is to deliver a functional prototype that demonstrates the practical use of robotics in supporting healthcare services.

1.3 Significance of the Work

This project addresses real challenges faced in healthcare environments, particularly the heavy workload and time pressure experienced by nurses and medical staff. By automating routine and repetitive tasks, the Robotic Nurse helps reduce physical effort, save time, and improve overall efficiency. The system also enhances safety and reliability by minimizing direct contact during certain procedures and reducing the risk of human error during medical supply dispensing.

From an educational perspective, the project is significant because it integrates multiple engineering disciplines, including embedded systems, robotics and automation, computer vision, web-based interface

design, and medical sensor integration. The Robotic Nurse therefore serves as a strong example of a multi-disciplinary engineering project with real-world relevance.

1.4 Organization of the Report

This section will be updated after completing all chapters of the report. It will present a concise overview of the report structure and briefly describe the content of each chapter.

2 Background and Related Work

This chapter provides the theoretical and technological background relevant to the development of the Robotic Nurse System. It introduces the broader context of robotics in healthcare, discusses existing medical robotic applications, and reviews key sensing and automation principles commonly employed in assistive medical systems. The purpose of this chapter is to establish the conceptual foundation upon which the proposed system is designed, without addressing implementation-specific details, which are reserved for subsequent chapters.

2.1 Robotics in Healthcare

Robotic systems have become increasingly prevalent in modern healthcare environments, where they are employed to support medical staff, enhance operational efficiency, and reduce human exposure to repetitive or low-risk tasks. Unlike surgical robots, which operate under strict clinical supervision, assistive medical robots are typically designed to perform auxiliary functions such as patient interaction, monitoring, material delivery, and basic medical assistance.

The adoption of robotics in healthcare has been driven by multiple factors, including staff shortages, increasing patient loads, and the need for improved hygiene and safety standards. Assistive robotic platforms are often deployed in hospitals, clinics, and care facilities to automate routine tasks while maintaining a controlled and predictable mode of operation. These systems are generally designed to complement medical personnel rather than replace them, emphasizing safety, transparency, and user-guided interaction.

2.2 Assistive and Service Medical Robots

Assistive medical robots are typically mobile or stationary platforms equipped with sensing, interaction, and limited actuation capabilities. Their primary function is to support healthcare workflows by performing predefined tasks initiated by users or medical staff. Common applications include telepresence, delivery of medical supplies, patient guidance, and basic physiological monitoring.

A representative example of such systems is shown in Figure 1, which illustrates a commercially deployed service robot operating in a hospital environment. These platforms are characterized by integrated displays for interaction, compact mechanical design, and modular hardware architectures that allow adaptation to different medical settings [7].

Figure 1: Example of an assistive service robot deployed in a hospital environment for interaction and support tasks [7].

While these systems demonstrate the feasibility of robotic assistance in clinical contexts, many are limited to high-level interaction or logistical tasks. Systems capable of performing direct user-facing medical assistance remain an active area of research and development, particularly when safety, determinism, and user trust are critical requirements.

2.3 Physiological Sensing in Medical Systems

Physiological sensing forms a core component of many medical and assistive robotic systems. Non-invasive sensors are commonly employed to acquire vital signs while minimizing user discomfort and risk. Typical measurements include body temperature, heart rate, blood oxygen saturation, and proximity-based presence detection.

Infrared temperature sensors are widely used for non-contact body temperature measurement, particularly in screening and triage applications. Pulse oximetry sensors measure heart rate and blood oxygen saturation through photoplethysmography, providing valuable indicators of cardiovascular and respiratory condition. These sensing methods are well-established and are frequently integrated into automated or semi-automated health monitoring systems.

In assistive robotic applications, physiological sensors are often combined with decision logic that evaluates measurements against predefined thresholds. This approach enables deterministic system behavior and reduces reliance on complex autonomous decision-making, which is particularly important in safety-critical environments.

2.4 Automation and Deterministic Control in Medical Devices

Medical robotic systems are subject to strict requirements regarding predictability, safety, and traceability. As a result, many systems favor deterministic control strategies over fully autonomous behavior. Deterministic systems execute actions only in response to explicit triggers and progress through well-defined operational states.

Event-driven architectures are commonly employed to ensure that actions such as sensing, actuation, or dispensing occur only after validated conditions are met. This approach reduces unintended behavior and simplifies verification and debugging. In the context of assistive medical devices, deterministic control also enhances user trust by ensuring that system responses are transparent and repeatable.

Automation in such systems is therefore typically constrained to low-risk, repetitive tasks, with higher-level decision-making either supervised by users or based on clearly defined logical rules.

2.5 Motivation for a Semi-Automated Robotic Nurse System

Despite advances in medical robotics, there remains a gap between high-level assistive robots and systems capable of providing structured, user-initiated medical assistance. Many existing platforms focus on interaction or logistics, while direct assistance applications are often limited in scope or complexity.

The Robotic Nurse System is motivated by the need for a compact, deterministic, and user-guided platform capable of performing basic nursing-related tasks. By combining established physiological sensing techniques with controlled actuation and a guided interaction model, the system aims to reduce workload on medical staff while maintaining a high level of operational safety.

The conceptual principles discussed in this chapter provide the foundation for the system design and implementation presented in the following chapters.

3 Techniques and Methods

This chapter presents the technical approach adopted for the design and development of the Robotic Nurse System. It describes the system architecture, mechanical and electronic design principles, sensing and actuation strategies, and the software coordination methodology used to achieve deterministic and safe operation.

The chapter focuses on how the system was structured and implemented to meet the defined project objectives and constraints. Emphasis is placed on modularity, separation of responsibilities, and controlled interaction between hardware, software, and the user. Detailed execution logic and workflow behavior are introduced to illustrate how individual subsystems cooperate to form a complete assistive medical platform.

3.1 System Overview

The Robotic Nurse System is designed as a semi-automated assistive medical station capable of performing basic nursing-related tasks in a controlled and user-initiated manner. The system supports two primary operational modes: flu-related assessment and minor wound or burn detection and treatment.

Rather than operating autonomously, the system follows a deterministic execution model in which every action is explicitly triggered either by user input or by validated sensor events. This design approach prioritizes operational safety, predictability, and transparency, which are critical requirements in medical and healthcare-related environments.

From a structural perspective, the system is composed of three cooperating subsystems: a hardware control unit responsible for sensing and actuation, a coordination and logic unit that manages workflows and decision logic, and a human–machine interface that facilitates guided interaction with the user. These subsystems are physically and logically separated to simplify development, testing, and future expansion.

The following sections describe each subsystem in detail, beginning with the overall system architecture.

3.2 Overall System Architecture

The Robotic Nurse System follows a modular and layered architecture designed to ensure deterministic operation, clear separation of responsibilities, and ease of integration between hardware and software components. The architecture was developed with a software-driven approach, where system behavior is primarily governed by explicit control logic rather than autonomous decision-making.

At a high level, the system is organized into three main architectural layers: the hardware control layer, the coordination and control layer, and the human–machine interface layer. Each layer performs a distinct role and communicates with the others through clearly defined interfaces. This separation reduces system complexity, simplifies debugging, and allows individual subsystems to be modified or extended without affecting the overall system behavior.

The hardware control layer is responsible for direct interaction with physical components such as sensors, motors, pumps, and actuators. This layer handles real-time signal acquisition and actuation execution and is designed to operate reliably under strict timing and safety constraints. Low-level control tasks are isolated within this layer to prevent unintended interactions with higher-level software logic.

The coordination and control layer acts as the central system controller. It manages workflow sequencing, decision logic, and event handling based on user input and sensor feedback. Rather than continuously running background processes, this layer follows an event-driven execution model in which actions are triggered only when specific conditions are satisfied. This design ensures predictable system behavior and prevents unsafe or undefined execution states.

The human–machine interface layer provides guided interaction between the user and the system. It presents instructions, visual feedback, and workflow progress information through a tablet-based graphical interface. User inputs collected at this layer serve as the primary triggers for system operation, reinforcing the semi-automated and user-initiated nature of the Robotic Nurse.

Communication between layers is implemented through structured message passing and explicit command-response sequences. This approach ensures that each subsystem operates within its defined scope and that all system actions can be traced back to a validated input or event. The overall architecture therefore supports transparency, safety, and maintainability, which are essential requirements for healthcare-related systems.

3.3 Mechanical Design and Assembly

The mechanical structure of the Robotic Nurse System consists of a laser-cut external body and internally mounted custom components. The main enclosure was digitally designed and fabricated using laser cutting from 6 mm wooden sheets, forming the structural frame of the system. Internal mechanical components and mounting elements were separately designed using CAD tools and fabricated through 3D printing. After fabrication, all parts were manually assembled and finished through surface treatment and painting. This hybrid fabrication approach enabled rapid prototyping, design flexibility, and sufficient structural support for the integrated subsystems.

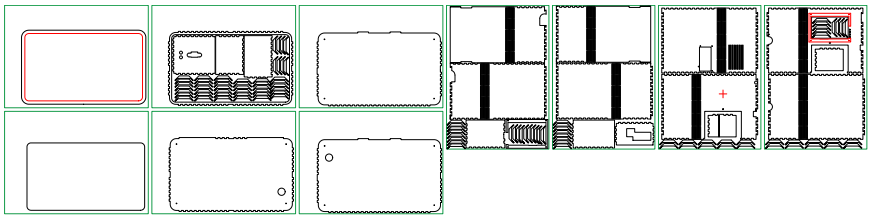


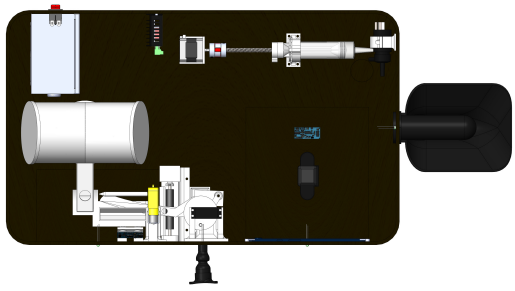
Figure 2: Laser-cut external body of the Robotic Nurse System fabricated from 6 mm wooden sheets.

3.3.1 Overall Mechanical Layout

The mechanical layout of the Robotic Nurse System integrates sensing, actuation, and user interaction within a compact enclosed structure. User-facing elements such as the display, camera, and access openings are positioned on the front panel, while internal components including motors, pumps, storage units, and electronics are arranged to separate fluid handling from control hardware. This layout simplifies assembly, improves accessibility, and supports modular integration without mechanical interference.

((a)) Front view of the complete mechanical assembly.

((b)) Side view.



((c)) Top view.

Figure 3: Overall mechanical layout of the Robotic Nurse System.

3.4 Hardware Architecture and Actuation Systems

This section presents the complete hardware architecture of the Robotic Nurse System, covering mechanical actuation, sensing, processing, and power distribution within a single unified framework. Rather than listing components in isolation, hardware elements are organized according to their functional role in system operation. This approach reflects the mechatronic design philosophy adopted in the project, where mechanical, electrical, and computational subsystems are developed concurrently and integrated through well-defined interfaces.

The section builds progressively from system-level integration, through functional actuation mechanisms, to the underlying electronic components that enable sensing, control, and coordination.

3.4.1 System-Level Hardware Integration

The Robotic Nurse System integrates mechanical subsystems, sensing hardware, actuation units, and control electronics within a single enclosed structure. Components are arranged to ensure clear separation between fluid handling, mechanical motion, and electronic control, while maintaining accessibility for assembly, inspection, and maintenance. This physical separation reduces contamination risk, simplifies debugging, and supports modular expansion.

Figure 4 illustrates the overall internal integration of the system and highlights the spatial relationship between the main hardware subsystems.

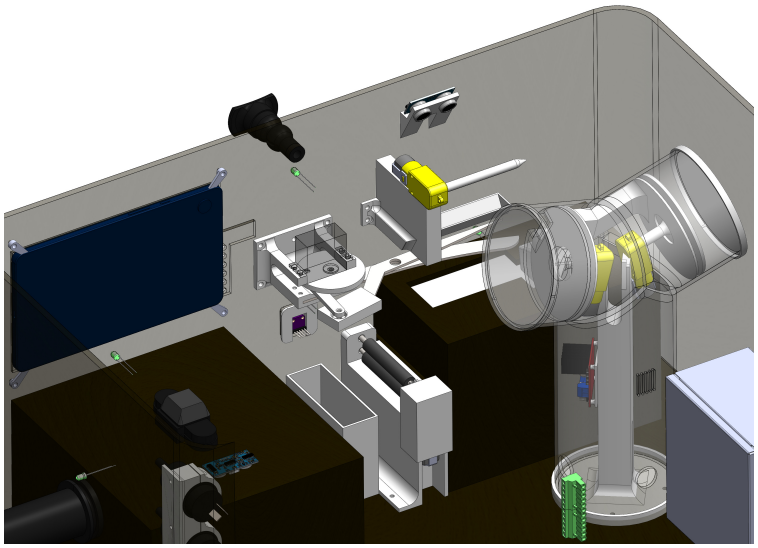


Figure 4: Overall internal integration of the Robotic Nurse System showing mechanical, sensing, and electronic subsystems.

3.4.2 Functional Actuation and Dispensing Mechanisms

All physical interaction with the user and all material handling operations are performed through dedicated actuation and dispensing mechanisms. These mechanisms are executed under deterministic control by the Arduino Mega and are activated only upon explicit commands issued by the coordination layer. No

mechanical action occurs autonomously.

Each dispensing subsystem is designed to perform a single, well-defined function with built-in sensing to confirm correct execution before workflow progression.

Pill Dispensing Mechanism

The pill dispensing mechanism consists of two independent pill containers, each driven by a DC geared motor connected to a rotating disc. The disc aligns individual pills with an exit hole, allowing controlled release. A tube connects the two dispensing stages, with an infrared (IR) sensor positioned between them to detect successful pill transfer. Motor speed is regulated using pulse-width modulation (PWM), and directional control is achieved via an L298N H-bridge motor driver.

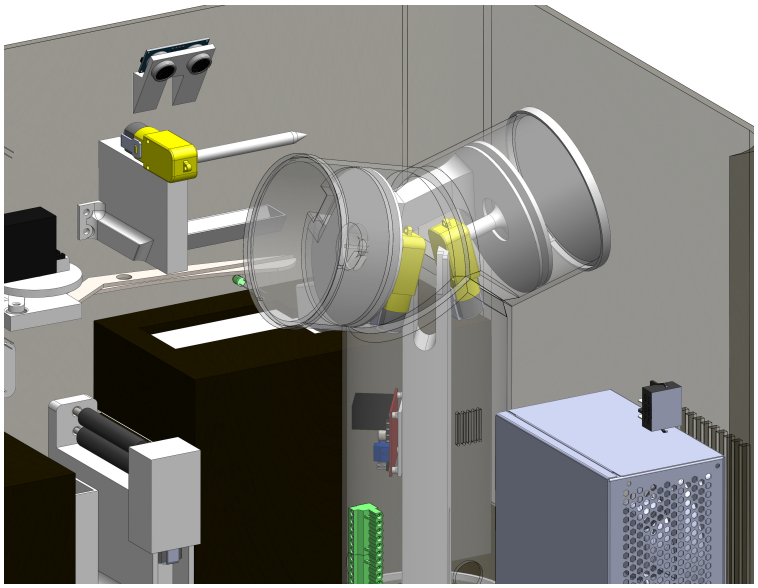


Figure 5: Pill dispensing mechanism with dual DC motors and IR detection.

Water and Alcohol Dispensing Mechanism

Water and alcohol dispensing are implemented using two 12 V DC pumps controlled through a two-channel relay module. Each pump draws fluid from a dedicated container equipped with ultrasonic sensors to monitor liquid levels. Dispensed fluids are routed through tubing to a shared outlet tap. An ultrasonic proximity sensor positioned near the outlet detects hand presence to trigger contactless dispensing, while indicator LEDs provide visual feedback during operation.

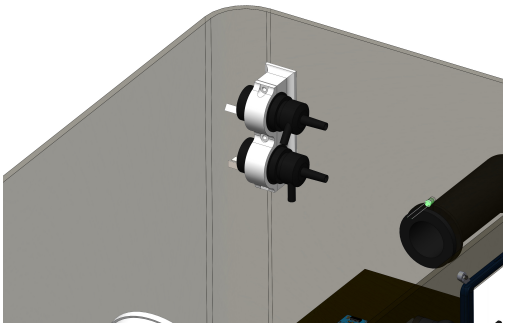


Figure 6: Water and alcohol dispensing system using DC pumps and relay control.

Ointment Dispensing Mechanism

The ointment dispensing mechanism is implemented as a syringe-based extrusion system. A NEMA 17 stepper motor drives the syringe plunger through a lead screw mechanism, enabling precise volumetric control. The motor is driven by a TB6600 stepper driver configured for 1/32 microstepping to achieve smooth and accurate motion. A limit switch defines the physical end-of-travel and prevents mechanical overextension. The extruded ointment is delivered through tubing to the application outlet.

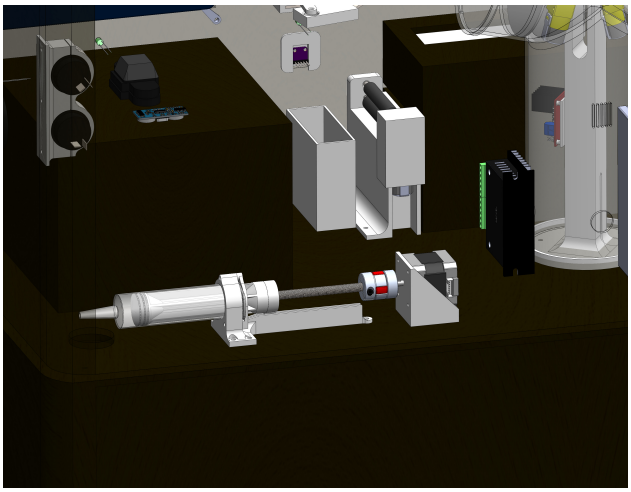


Figure 7: Ointment extrusion mechanism driven by a NEMA 17 stepper motor.

Plaster and Gauze Application Mechanism

The plaster application mechanism consists of a roller driven by a DC motor and gear transmission, enabling controlled dispensing of adhesive plaster. An infrared sensor detects plaster movement to confirm correct feed. Gauze dispensing is implemented using a motor-driven roller that unrolls gauze material. Cutting is performed using a scissor mechanism actuated through a crank-slider linkage driven by a servo motor. A DC–DC buck converter regulates the servo supply voltage to 7.4 V. The DC motors for plaster and gauze dispensing are controlled via an additional L298N driver.

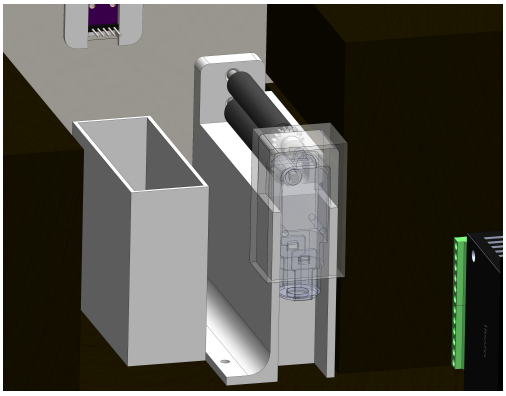


Figure 8: Plaster and gauze dispensing and cutting mechanism.

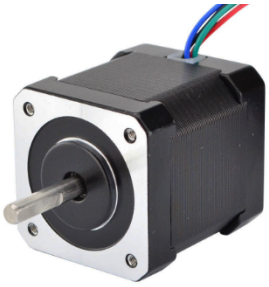
3.4.3 Actuation and Drive Hardware

The actuation hardware provides the mechanical motion required by the dispensing mechanisms described in Section 3.4.2. Components are selected according to torque, precision, duty cycle, and control requirements. All actuators are directly controlled by the Arduino Mega through dedicated driver circuits.

Figure 9 presents the primary actuation and motor drive components used throughout the system.

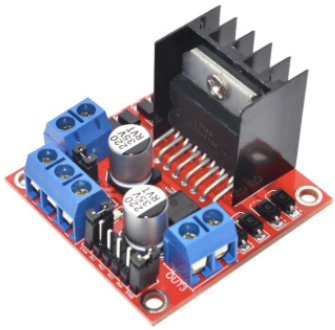
((a)) Micro servo motor

((b)) DC geared motor



((c)) NEMA 17 stepper motor

((d)) TB6600 stepper motor driver



((e)) L298N DC motor driver

Figure 9: Primary actuation and motor drive components used in the Robotic Nurse System.

Stepper motors are used where precise linear displacement is required, servo motors are used for con-

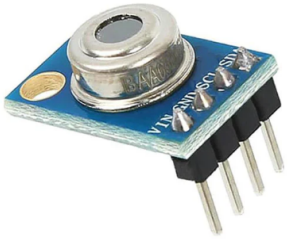
trolled angular motion, and DC motors are employed for continuous rotation tasks such as feeding and rolling operations.

3.4.4 Sensing Hardware

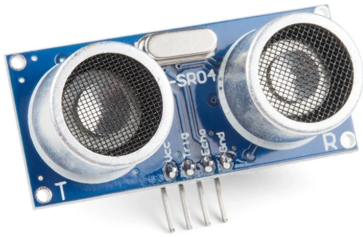
Sensing hardware provides physiological measurements, proximity detection, and execution confirmation signals. These sensors serve as authoritative inputs for workflow decisions and safety validation and are directly interfaced with the Arduino Mega.

Figure 10 illustrates the primary sensing components integrated into the system.

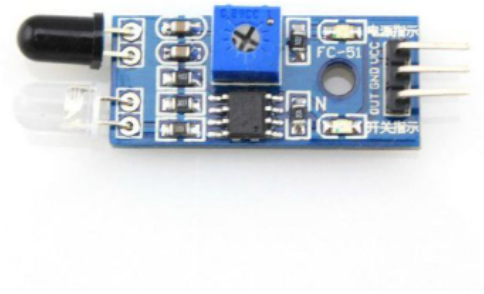
((a)) MAX30100 pulse oximeter



((b)) MLX90614 infrared temperature sensor



((c)) HC-SR04 ultrasonic sensor



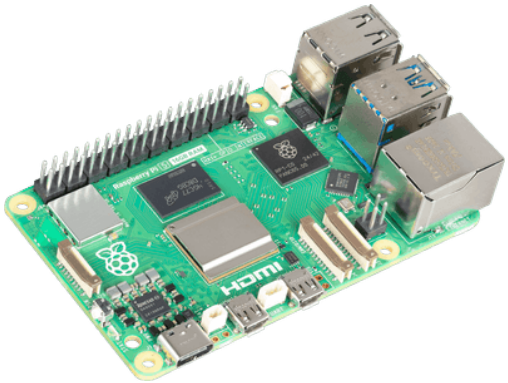
((d)) Infrared detection sensor

Figure 10: Physiological and interaction sensing components used in the system.

3.4.5 Processing and Control Units

The Robotic Nurse System employs a distributed processing architecture consisting of a hardware control unit and a higher-level coordination unit. This separation ensures deterministic low-level execution while allowing flexible software logic and image processing.

((a)) Arduino Mega microcontroller



((b)) Raspberry Pi single-board computer

((c)) Raspberry Pi camera module

Figure 11: Processing, control, and vision hardware used in the Robotic Nurse System.

3.4.6 Power Supply and Distribution

Power delivery to all system components is provided by a centralized ATX power supply unit. The supply delivers regulated voltage rails required by both high-current actuation hardware and logic-level electronics.

Figure 12: ATX power supply unit used as the primary system power source.

The 12 V rail supplies motors, pumps, and motor drivers, while the 5 V rail powers sensors, microcontrollers, and logic circuitry. All components share a common ground reference to ensure stable operation and reliable signal integrity.

3.5 Software Architecture and Control Methodology

The software architecture of the Robotic Nurse System constitutes the coordination, decision-making, and user interaction layer of the overall system. It does not directly drive actuators, read sensors, or execute physical actions. All hardware-level execution is delegated to the Arduino Mega through explicit, validated serial commands issued by the coordination layer.

The software runs on a Raspberry Pi and is implemented primarily using the Python programming language. Its responsibilities include workflow coordination, serial communication handling, logical state management, stock tracking, image-based wound classification triggering, and mediation between the web-based user interface and the hardware control layer.

The architecture is designed around strict separation of concerns. The Raspberry Pi operates as a supervisory controller responsible for logic, validation, and sequencing, while the Arduino Mega operates as a deterministic execution controller responsible solely for physical actions and sensor acquisition. No software component violates this authority boundary.

All software components operate under a strictly event-driven execution model. The system does not perform autonomous actions, background polling, or time-based control. Every software action is initiated exclusively by one of the following events: explicit user input, a hardware-originated serial message, or a validated workflow state transition.

3.5.1 Raspberry Pi Runtime Environment and Deployment

All Raspberry Pi software components reside within a single, self-contained project directory executed inside a dedicated Python virtual environment. This environment contains all required dependencies, libraries, and runtime services for system operation, ensuring reproducibility and isolation from the underlying operating system.

At system startup, the Raspberry Pi automatically activates the virtual environment and executes the main coordination script. This startup process initializes the serial communication link with the Arduino, loads persistent system state files, registers event handlers, and launches the web server. No hardware actions are executed during startup, and the system enters a safe idle state awaiting external commands.

The image processing service is initialized during startup and remains idle until explicitly triggered by a hardware-originated event. No image capture, inference, or background processing occurs unless the corresponding workflow state is reached and a valid trigger is received.

All software files, including coordination logic, workflow controllers, serial communication handlers, image processing modules, and web interface resources, reside within the same runtime environment and are managed under a unified directory structure. This design simplifies deployment, debugging, and system

maintenance.

3.5.2 Software File Structure and Entry Point

The Raspberry Pi software is organized into modular Python files, each with a strictly defined responsibility. This modular structure prevents unintended coupling between components and ensures that hardware interaction is always mediated through validated workflow logic.

The system entry point is the file `main.py`. This file is responsible for initializing the runtime environment, establishing serial communication with the Arduino Mega, instantiating workflow controllers, registering serial message callbacks, and starting the web-based user interface server.

No physical actions are executed within `main.py`. Its role is limited to system initialization and coordination. All action execution requests are issued only after workflow validation and explicit user confirmation.

The primary software modules and their responsibilities are summarized as follows:

- `main.py` – System entry point and global coordination layer.
- `serial_link.py` – Asynchronous serial communication handler.
- `messages.py` – Parsing and interpretation of ASCII serial messages.
- `flu_process.py` – Flu assessment workflow controller.
- `wound_process.py` – Wound detection and treatment workflow controller.
- `stock_manager.py` – Logical stock tracking and persistence.
- `image_processing.py` – Image capture and wound classification logic.
- `web_flu.py`, `web_wound.py`, `web_stock.py` – Web API route definitions.

Each module operates exclusively within its assigned responsibility and does not bypass workflow validation or directly control hardware resources.

3.5.3 Event-Driven Software Execution Model

The Robotic Nurse software follows a strictly event-driven execution model. No periodic tasks, predictive behaviors, or autonomous background loops are implemented anywhere in the system.

Three categories of events are recognized by the software architecture:

- User-initiated events originating from the web interface.
- Hardware-originated events received from the Arduino via serial communication.
- Workflow-generated events representing validated state transitions.

User-initiated events are received through HTTP requests handled by the Flask-based web backend. These events request workflow progression but do not directly trigger hardware actions. All requests are validated against the current system state before execution.

Hardware-originated events are transmitted as ASCII serial messages and represent authoritative physical conditions or execution results. These events may trigger immediate workflow transitions but never initiate actions independently.

Workflow-generated events represent internal logical transitions after validation of conditions such as sensor readings, stock availability, and user confirmation. These events exist solely to advance or terminate workflows in a deterministic manner.

This execution model guarantees predictability, traceability, and safety by ensuring that no action is performed without explicit authorization and confirmed system state.

3.6 Image Processing and Wound Classification Logic

Image processing and wound classification in the Robotic Nurse System are implemented as a dedicated, event-driven software service executed on the Raspberry Pi. This service is responsible solely for image acquisition, preprocessing, and classification, and does not perform autonomous execution, hardware control, or workflow decisions.

The image processing logic is intentionally decoupled from the main coordination logic and is implemented in a separate Python module, `image_processing.py`. This design ensures modularity, maintainability, and strict separation between workflow coordination and computer vision tasks.

3.6.1 Training Environment and Model Preparation

The wound classification model was trained externally using a GPU-enabled Google Colab environment [1]. This approach was selected to leverage high-performance hardware and to decouple model training from the embedded runtime system.

The training process followed a standard YOLO-based object detection workflow [8] and included the following steps:

1. Creation of a GPU-enabled Google Colab runtime.
2. Installation of the YOLO framework and all required dependencies.
3. Uploading and organizing a labeled wound image dataset.
4. Splitting the dataset into training, validation, and test subsets.
5. Configuration of a dataset YAML file specifying class labels and paths.
6. Execution of the YOLO training process.
7. Monitoring training progress using loss curves and validation metrics.

8. Exporting the best-performing trained weights.

Upon completion of training, the final trained model weights were exported as a `.pt` file (referred to as `model.pt`). This file represents a static, frozen model and is not modified or retrained during system operation.

The training methodology and environment setup follow the workflow demonstrated in a publicly available YOLO–Google Colab tutorial, which was used as a reference during implementation [6].

3.6.2 Deployment on Raspberry Pi

The trained `model.pt` file is transferred from the Colab environment to the Raspberry Pi and stored within the system's software directory. During system startup, the image processing module loads the model into memory within the active Python virtual environment.

All image processing dependencies, including the YOLO inference library and image handling packages, are installed within the same virtual environment used by the main system software. This ensures consistent execution and avoids dependency conflicts.

The image processing service remains idle after initialization and does not perform any background computation. Inference is executed only when explicitly triggered by a valid hardware-originated event.

3.6.3 Event-Driven Trigger Mechanism

Image capture and classification are initiated exclusively by the Arduino-originated serial event:

`EVENT|HAND_BOX|DETECTED`

This event is generated when stable hand presence is detected inside the wound imaging box. Upon reception of this event, the wound workflow controller invokes the image processing module. No user input is required to initiate this step, and no image processing occurs without this explicit trigger.

If the corresponding clearance event `EVENT|HAND_BOX|CLEARED` is received before image capture is completed, the operation is immediately aborted to prevent invalid or partial data acquisition.

3.6.4 Image Capture and Preprocessing Pipeline

Once triggered, the `image_processing.py` module executes a deterministic processing pipeline consisting of the following steps:

1. Capture a raw image frame from the Raspberry Pi camera.
2. Save the captured image temporarily to disk.
3. Apply predefined cropping logic to isolate the wound region.
4. Perform image resizing and normalization as required by the YOLO model.

5. Pass the processed image to the trained model for inference.

After inference is completed, all temporary image files are deleted from the filesystem. No historical images are stored, and no long-term image archive is maintained. This design minimizes storage usage and reduces privacy risks.

3.6.5 YOLO-Based Classification and Output Handling

The trained YOLO model processes the preprocessed image and outputs a discrete wound class label selected from a predefined set. Confidence values and bounding box data are used internally by the inference engine but are not propagated beyond the image processing module.

If no valid detection is produced, or if inference fails for any reason, the module returns a default classification value of `NONE`. This conservative behavior ensures that no treatment actions are initiated in the absence of reliable classification.

The final classification result is returned to the wound workflow controller through a direct function call. The image processing module does not trigger hardware actions, modify stock values, or advance workflows independently.

3.6.6 Integration with Workflow Logic

After classification, the wound label is stored in the active workflow context and displayed to the user through the web-based interface. Any subsequent treatment actions require explicit user confirmation and successful workflow validation before execution.

The image processing module serves strictly as a decision-support input and does not perform medical interpretation or autonomous decision-making.

3.6.7 Design Constraints and Safety Considerations

The image processing subsystem adheres to the following constraints:

- No autonomous or background execution
- No direct hardware control
- No stock modification
- No permanent image storage
- No medical decision-making beyond classification

By enforcing strict event-driven activation, modular isolation, and controlled data handling, the image processing logic integrates safely into the overall system methodology without introducing nondeterministic behavior or safety risks.

3.7 Workflow Control Logic and Execution Sequences

The Robotic Nurse System operates under a deterministic, event-driven workflow control architecture implemented on the Raspberry Pi. All system behavior is expressed as explicit workflows that define the order of operations, decision points, and termination conditions. No workflow executes autonomously, periodically, or concurrently with another workflow. Each workflow is initiated only through explicit user intent and progresses strictly in response to validated hardware and software events.

Workflow control logic is responsible for coordinating high-level system behavior, validating preconditions, invoking sensing or actuation requests, and ensuring that all operations terminate in a known safe state. Physical execution, timing, and sensor-level verification are delegated entirely to the Arduino hardware control layer.

Two primary workflows are implemented: a flu assessment workflow and a wound detection and treatment workflow. Both workflows follow the same structural principles, differ only in sensing requirements and actuation sequences, and are mutually exclusive during system operation.

3.7.1 High-Level Software Workflow Structure

At the highest level, the system remains in an idle state awaiting explicit user input through the web-based interface. Upon mode selection, control is transferred to the corresponding workflow controller. After workflow completion or termination, the system returns automatically to the idle state.

Figure 13 illustrates the high-level software workflow control structure implemented on the Raspberry Pi.

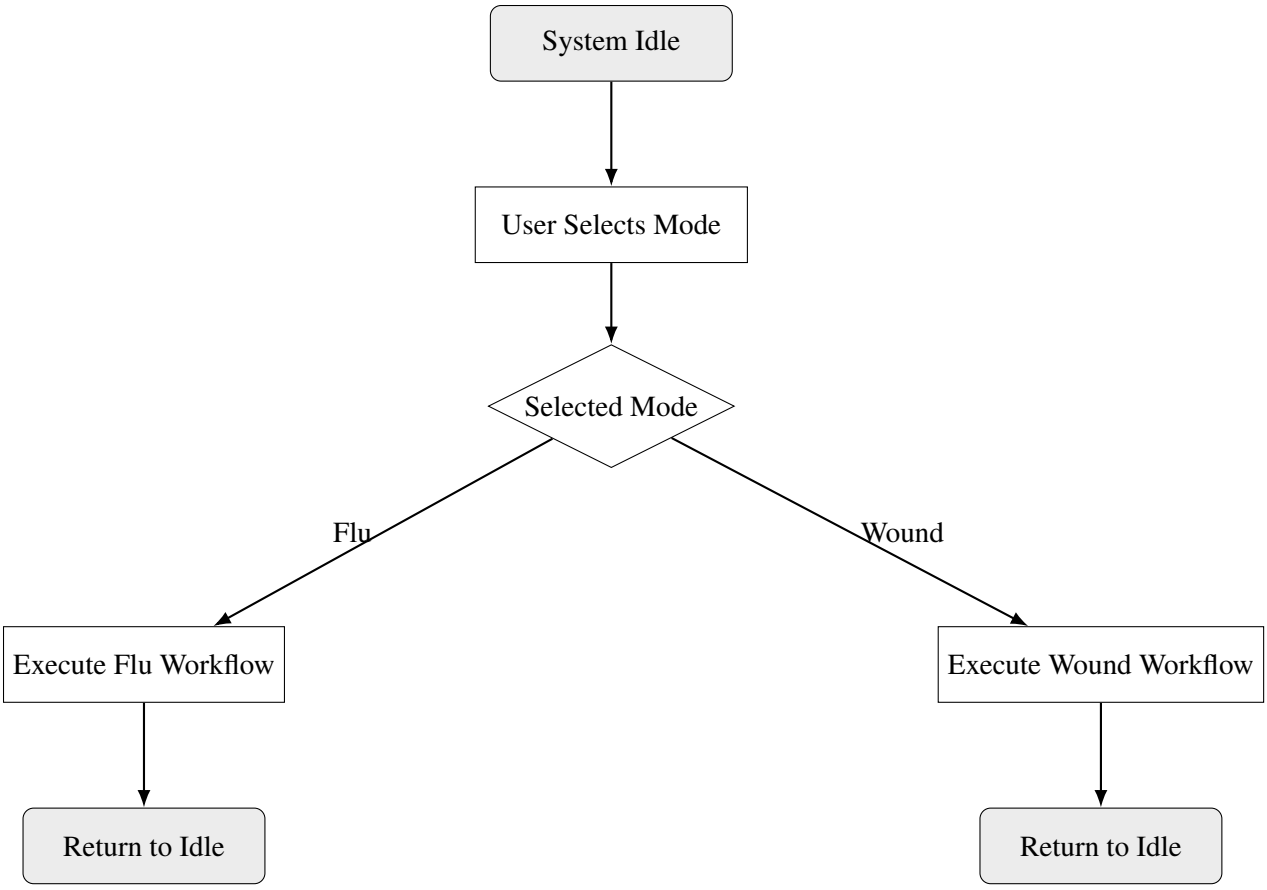


Figure 13: High-level software workflow control logic

This structure ensures that only one workflow is active at any time and that all system states are explicitly defined and recoverable.

3.7.2 Flu Workflow Control Logic

The flu workflow is initiated when the user selects flu assessment mode through the web interface. The workflow begins with non-contact body temperature measurement using an infrared temperature sensor. Based on predefined temperature thresholds, the workflow branches into one of three termination paths.

If the measured temperature is within the normal range, the workflow terminates immediately without requesting further measurements or dispensing medication. If the temperature exceeds predefined thresholds, additional physiological measurements are requested, followed by conditional medication dispensing.

Figure 14 presents the flu workflow control logic.

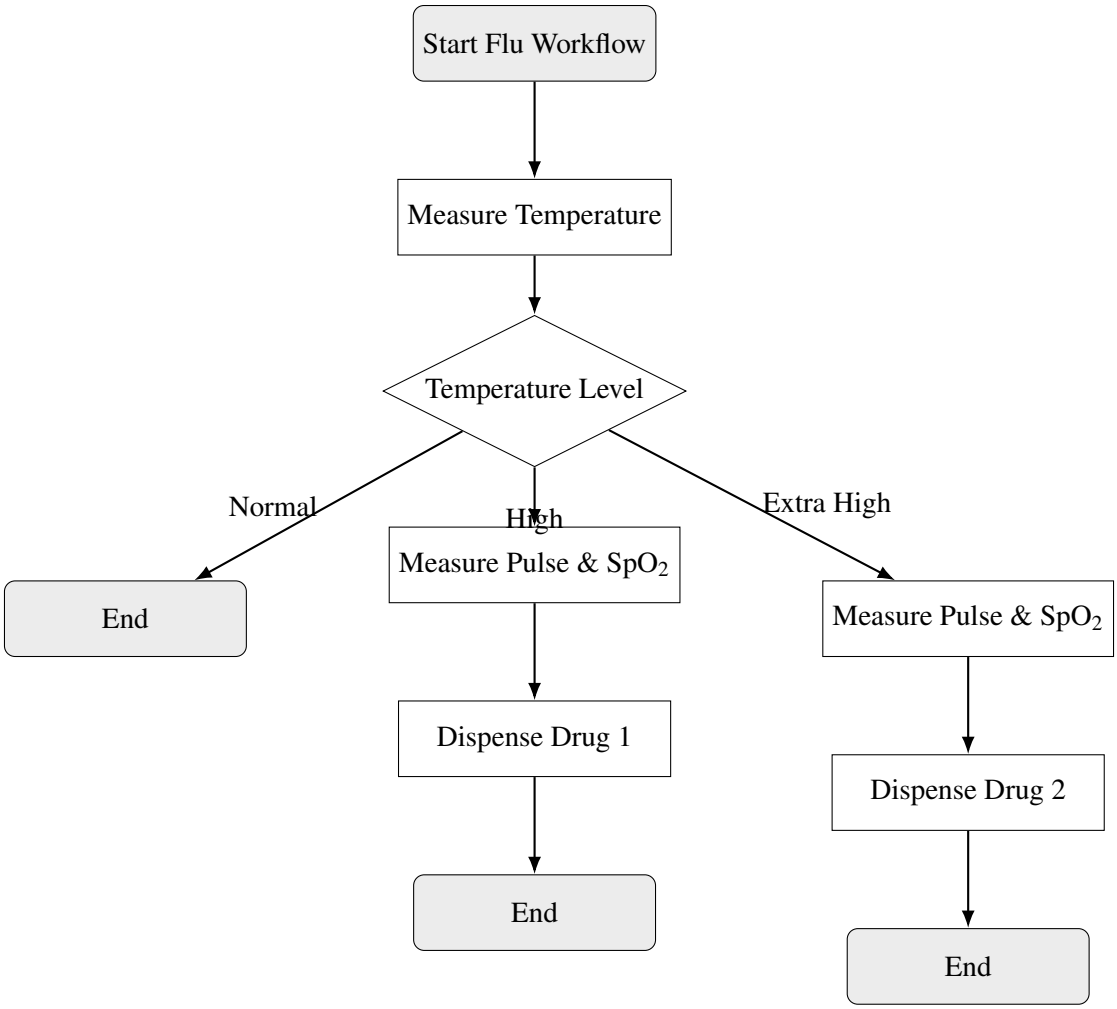


Figure 14: Flu workflow control logic with sensor-based branching

All transitions within the flu workflow are validated by confirmed sensor readings and hardware completion messages before progression or termination.

3.7.3 Wound and Burn Workflow Control Logic

The wound workflow is initiated when the user selects wound treatment mode. The workflow remains idle until hand presence is detected inside the imaging enclosure. This ensures correct positioning and prevents unintended image capture.

Once hand presence is confirmed, an image is captured and processed by the image classification module. The resulting classification determines the subsequent treatment path. No treatment action is executed without a valid classification result and explicit workflow validation.

Figure 15 illustrates the wound and burn workflow control logic.

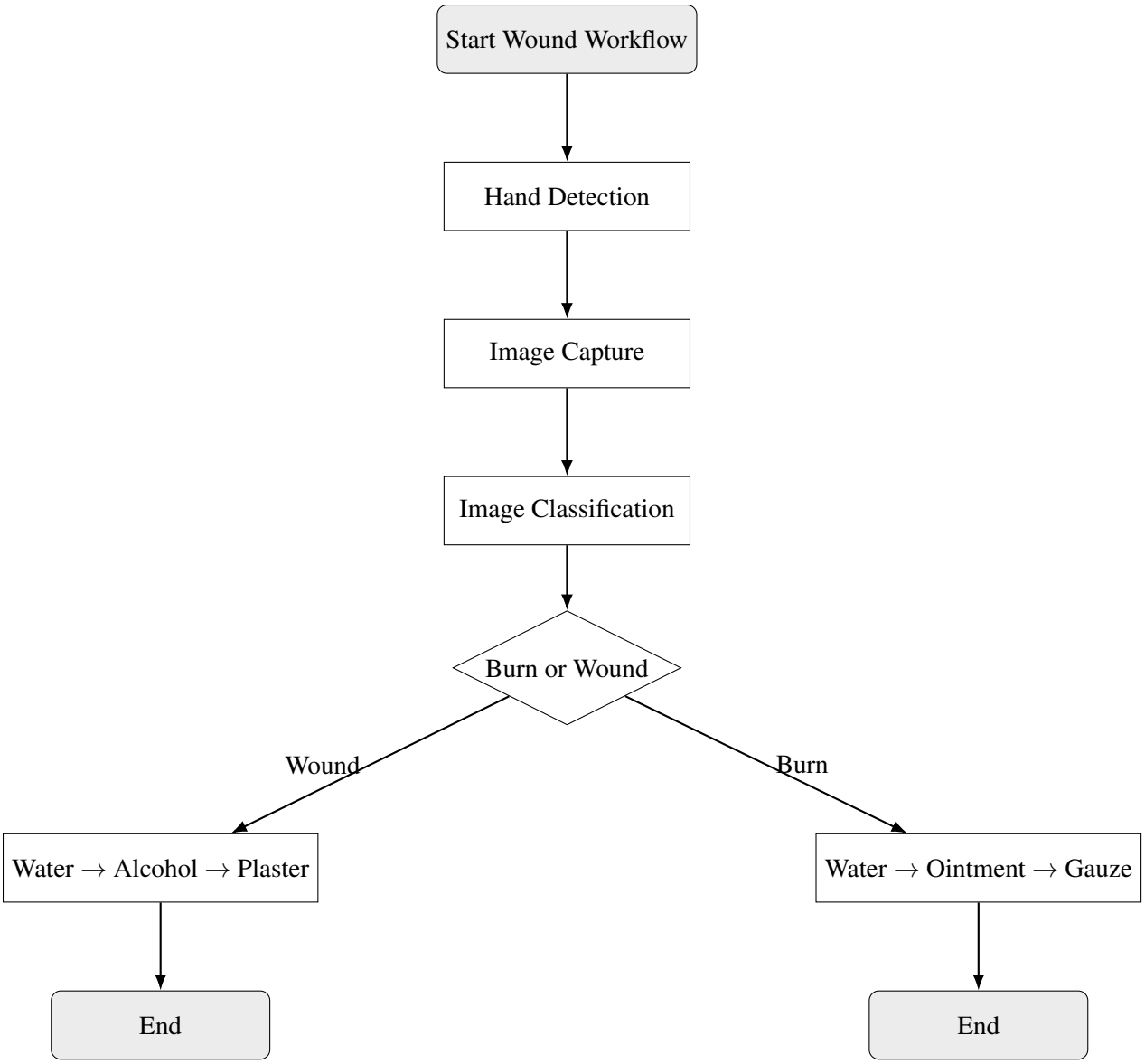


Figure 15: Wound and burn workflow control logic integrating image classification

If classification fails or returns an undefined result, the workflow terminates safely without executing any dispensing actions.

3.7.4 Workflow Safety and Determinism Guarantees

All workflows enforce strict determinism and safety through the following guarantees:

- Only one workflow may execute at any time.
- All transitions require validated events or user confirmation.
- No hardware action is initiated without workflow authorization.
- All workflows terminate in a defined idle state.

- Abort conditions are explicitly handled at every decision point.

These guarantees ensure predictable behavior, fault tolerance, and safe operation under all supported usage conditions.

3.8 Actuation Execution Logic

This subsection presents the low-level actuation execution logic implemented on the Arduino Mega hardware control layer. Each physical action is executed as an isolated, command-driven routine and validated locally using sensor feedback. The following flowcharts represent the authoritative execution logic for all supported actuation functions.

3.8.1 Water and Alcohol Dispensing Execution Logic

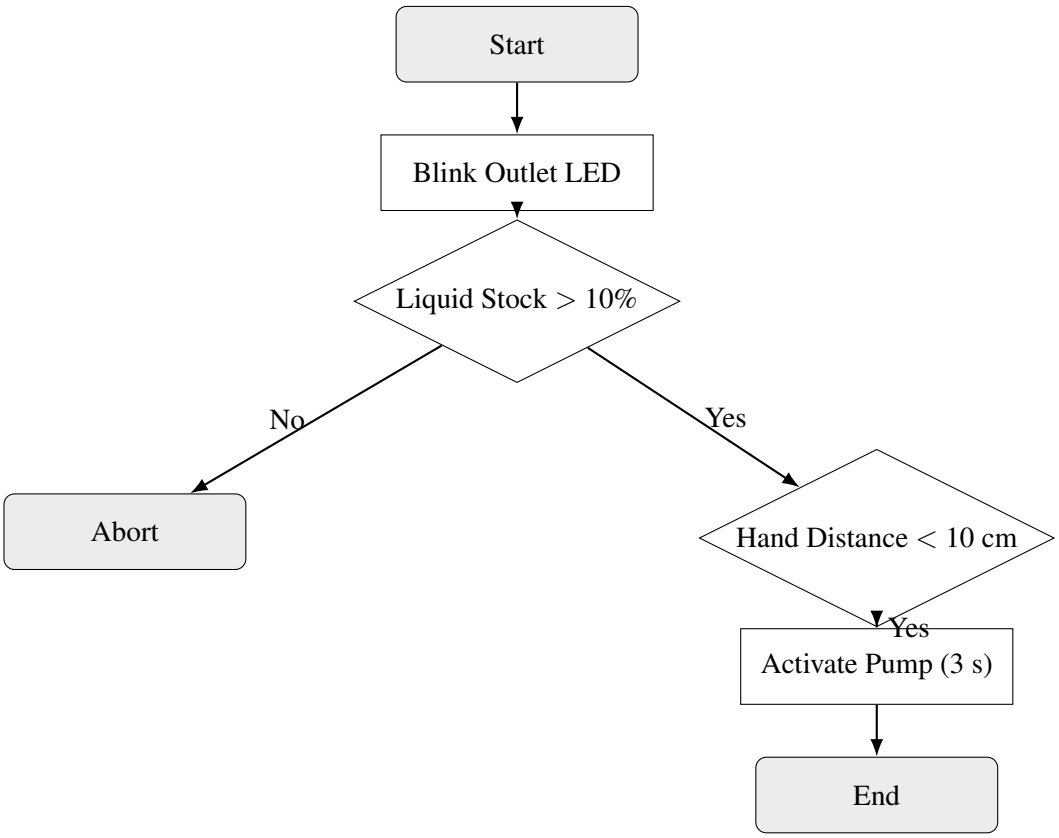


Figure 16: Execution logic for water and alcohol dispensing

3.8.2 Ointment Dispensing Execution Logic

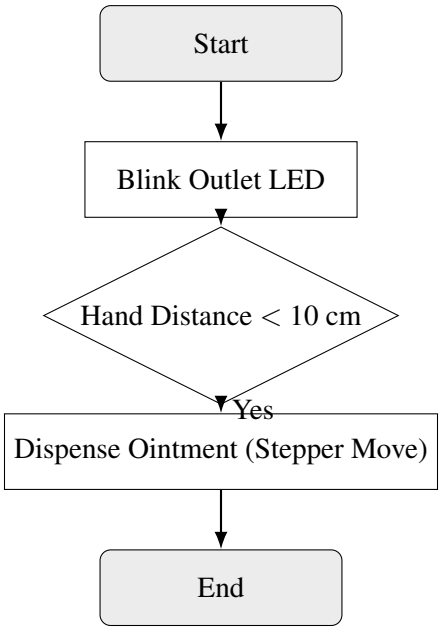


Figure 17: Execution logic for ointment dispensing

3.8.3 Plaster Dispensing Execution Logic

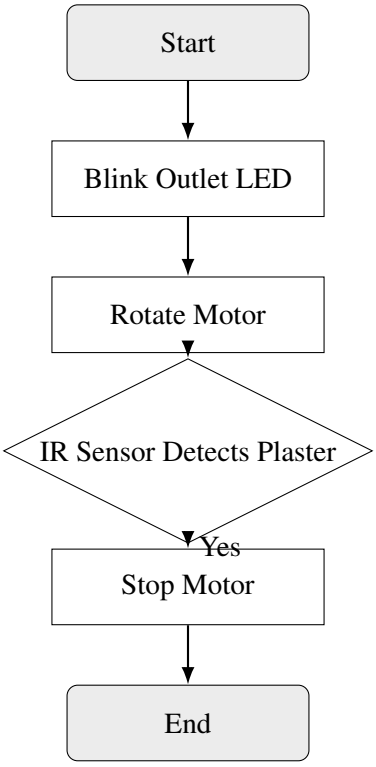


Figure 18: Execution logic for plaster dispensing

3.8.4 Gauze Dispensing and Cutting Execution Logic

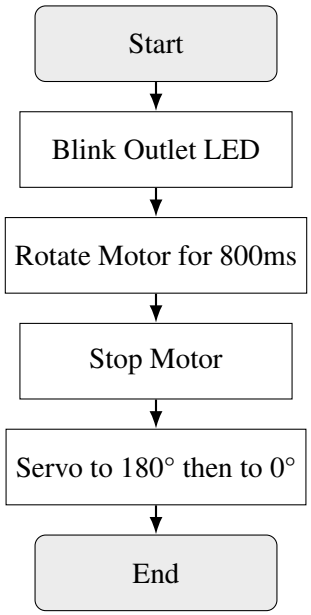


Figure 19: Execution logic for gauze dispensing and cutting

3.8.5 Pill Dispensing Execution Logic

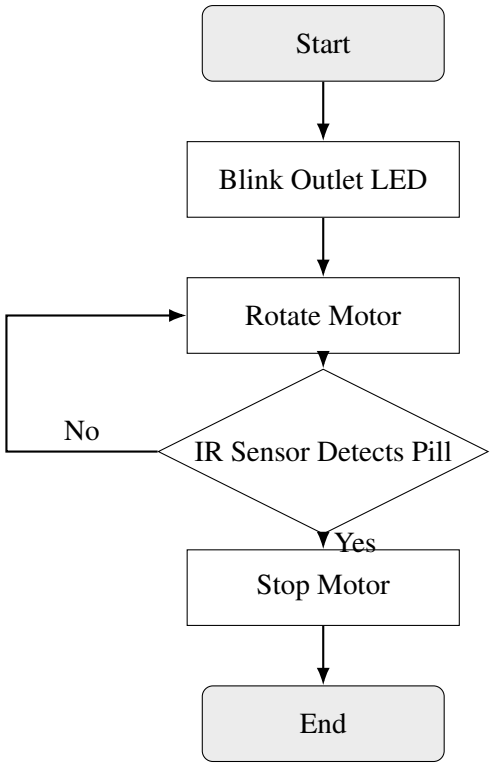


Figure 20: Execution logic for pill dispensing (Drug 1 and Drug 2)

3.9 Engineering Standards, Design Constraints, and Ethical Considerations

This subsection documents the engineering standards, system constraints, and ethical principles that governed the design and implementation of the Robotic Nurse System. These considerations were integrated during the methodology phase to ensure safe, deterministic, and responsible system behavior rather than being evaluated post-development.

The system is developed as a supervised assistive robotic prototype for educational and research purposes. It is not intended for autonomous medical diagnosis or clinical deployment, and therefore does not claim compliance with certified medical device regulations. Nevertheless, relevant engineering and ethical standards were considered to guide safe and responsible design decisions.

3.9.1 Engineering and Electrical Safety Standards

Electrical and electronic design decisions were guided by internationally recognized low-voltage and information technology safety standards. Power distribution, grounding, and insulation practices align with principles described in IEC 62368–1, which governs the safety of audio, video, information, and communication technology equipment [4].

Low-voltage operation (5 V and 12 V) was intentionally selected to reduce electrical risk during human interaction. High-current actuators are electrically isolated from logic-level control circuits using relay modules and motor drivers, preventing fault propagation between subsystems.

Embedded control and firmware development practices follow general guidance provided by IEC 61508 for functional safety in electrical and electronic systems, emphasizing deterministic execution, fault containment, and safe default states [3].

3.9.2 Software Reliability and Runtime Constraints

Software architecture decisions prioritize determinism, traceability, and fault containment. The event-driven execution model eliminates background processes, predictive behavior, and autonomous scheduling, reducing the likelihood of nondeterministic system states.

All software components execute within a controlled Python virtual environment on the Raspberry Pi, ensuring dependency isolation and reproducibility. System startup logic initializes all services into an idle state, with no hardware action permitted until explicitly triggered by validated events.

These design choices align with widely accepted embedded and safety-critical software engineering principles, including separation of concerns, explicit state transitions, and confirmation-based execution [5].

3.9.3 Human Interaction and Operational Constraints

Human interaction safety is enforced through sensor-gated actuation and explicit user confirmation at all decision points. Physical actions such as dispensing, cutting, and motor motion occur only after validating hand presence, stock availability, and workflow state.

The system does not initiate physical actions autonomously. All actuation sequences require either a user-initiated request or a hardware-confirmed event. This constraint ensures predictable behavior and prevents unintended interaction during idle or error states.

The system is designed for supervised operation only. Users are assumed to be present and capable of interrupting operation if unexpected behavior is observed.

3.9.4 Data Handling, Privacy, and Ethical Constraints

Ethical considerations were applied to data handling, image processing, and system decision-making. The system does not store personal identifiers, medical records, or historical physiological data. All sensor measurements are processed transiently and discarded after workflow completion.

Captured wound images are stored temporarily for classification purposes and are deleted immediately after inference. No image archives are maintained, minimizing privacy risk and data exposure.

The image classification subsystem provides only categorical labels and does not perform medical diagnosis or treatment selection autonomously. Final action decisions always require explicit user confirmation, ensuring that the system functions as an assistive tool rather than an authoritative medical agent.

These constraints align with established ethical guidelines for trustworthy and human-centered artificial intelligence systems, including transparency, human oversight, and accountability [2].

3.9.5 Ethical Scope and System Limitations

The Robotic Nurse System explicitly avoids autonomous medical decision-making. It does not claim diagnostic accuracy, clinical validation, or compliance with regulated medical-device frameworks.

The system is ethically scoped as a research and educational platform intended to demonstrate integrated sensing, actuation, and software coordination under controlled conditions. Any real-world deployment would require formal certification, clinical validation, and regulatory approval beyond the scope of this project.

3.10 Budget and Cost Breakdown

This project was developed under a strict cost constraint, with the objective of demonstrating a complete, functional robotic medical assistance system using commercially available and hobby-grade components. Cost efficiency was treated as a design constraint throughout the hardware selection and system integration phases.

All prices are listed in Israeli Shekels (ILS). The total system cost reflects the full prototype implementation, including control boards, sensors, actuators, power electronics, mechanical structure, and fabrication expenses.

Table 1: Robotic Nurse System Budget Breakdown

Category / Component	Unit Cost (ILS)	Quantity	Total (ILS)
Boards			
Arduino Mega	70	1	70
Raspberry Pi 4	330	1	330
Camera and Interface			
Camera Module	90	1	90
Android Tablet	300	1	300
Sensors			
Ultrasonic Sensors	20	5	100
Infrared Sensors	20	2	40
MLX90614 (Temperature)	80	1	80
MAX30100 (Pulse Oximeter)	35	1	35
Actuators			
12V Water Pumps	30	2	60
DS558HV Servo Motor	50	1	50
NEMA 17 Stepper Motor	65	1	65
Stepper Driver	60	1	60
DC Motors	15	4	60
L298N Motor Drivers	30	2	60
Additional Mechanical Actuators	25	2	50
Electronics and Power			
Power Supply	120	1	120
LED Indicators	8	0.5	4
2-Channel Relay Module	20	1	20
LM2596 DC–DC Converter	25	1	25
Fabrication and Materials			
Wood Structure	—	—	270
3D Printing (Plastic Parts)	—	—	1260
		Total Cost	**3029**

The final prototype cost of 3029 ILS enabled the successful realization of a complete end-to-end system, including sensing, actuation, control, user interaction, and image processing capabilities. The selected components prioritized availability, ease of integration, and rapid prototyping over industrial-grade precision.

While the use of hobby-grade components introduces limitations in mechanical tolerance, sensor accuracy, and long-term reliability, the achieved functionality validates the system architecture and methodology within the defined budget constraints. The cost structure also provides a clear baseline for future iterations targeting higher performance, safety certification, and medical-grade compliance.

4 Results

This chapter presents the experimental results obtained from the implementation and testing of the Robotic Nurse System. The results focus on functional verification, system integration performance, and validation of the proposed methodology. The system was evaluated as a prototype under controlled testing

conditions and was not intended for unsupervised medical deployment.

All system components were assembled, integrated, and tested as a complete end-to-end system, including sensing, actuation, software coordination, image processing, and human–machine interaction. The evaluation emphasizes whether the system performed its intended functions reliably within the defined design and budget constraints.

4.1 Overall System Functionality

The Robotic Nurse System successfully executed all primary workflows defined in the methodology chapter, including flu assessment, wound detection, and treatment dispensing. The coordination between the Raspberry Pi software layer and the Arduino hardware control layer operated as designed, with all physical actions executed only in response to validated software commands.

Serial communication between the coordination layer and the hardware control layer remained stable throughout testing. All issued commands resulted in deterministic execution, and all actuation routines reported completion through the defined serial protocol. No unintended or autonomous physical actions were observed during operation.

The web-based user interface functioned correctly, allowing users to initiate workflows, monitor system state, and view stock information. User actions were consistently validated by the backend before execution, preventing invalid transitions or unsafe operations.

4.2 Sensor and Actuation Performance

All sensors integrated into the system operated within their expected functional ranges. Infrared temperature measurement, pulse oximetry, ultrasonic proximity detection, and liquid-level sensing provided stable and repeatable readings during testing. Sensor data were correctly transmitted to the coordination layer and used to drive workflow decisions.

Actuation mechanisms, including pumps, DC motors, stepper motors, and servos, responded accurately to issued commands. Dispensing sequences for water, alcohol, ointment, plaster, gauze, and pills were executed in the correct order and terminated only after sensor-based confirmation or motion completion.

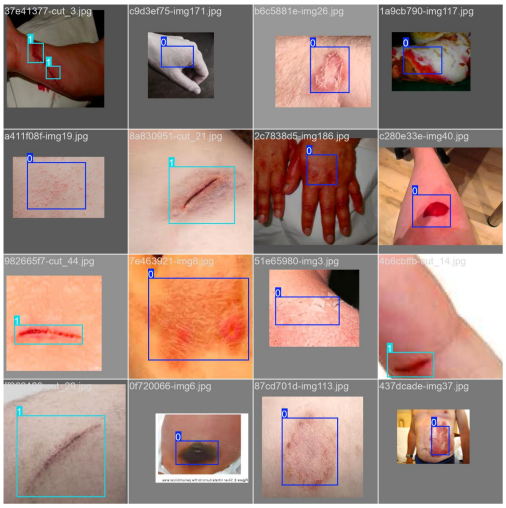
While the mechanical dispensing mechanisms achieved the intended functional behavior, minor variability was observed in the pill dispensing mechanism due to mechanical tolerances inherent to hobby-grade components. These effects did not prevent successful operation but indicate areas for refinement in future iterations.

4.3 Image Processing and Classification Results

The image processing subsystem successfully captured, preprocessed, and classified wound images in response to hardware-triggered events. Image acquisition was reliably initiated upon hand detection inside the imaging box, and no autonomous image capture occurred outside the defined workflow.

The YOLO-based classification model produced valid wound classification outputs during testing. The system correctly propagated classification results to the workflow logic and displayed them through the web interface. All classification outputs required explicit user confirmation before any treatment actions were executed.

Figure 21 shows representative examples of wound detection and classification results produced by the system during testing.



((a)) Multiple wound detection and classification outputs.

((b)) Single wound detection and classification example.

Figure 21: Representative wound detection and classification results generated by the image processing module.

The achieved classification confidence was approximately 40% across the evaluated dataset. This level of performance was sufficient to demonstrate the feasibility of event-driven image-based wound classification within the system architecture. The observed confidence level reflects the limited size and diversity of the available training dataset rather than deficiencies in the system integration or execution pipeline.

4.4 System Reliability and Integration Validation

End-to-end system testing confirmed reliable integration between hardware components, embedded firmware, coordination software, and the user interface. All workflows executed to completion without deadlocks, unsafe states, or inconsistent system behavior.

Logical stock tracking remained consistent with physical execution throughout testing. Stock values were updated only after confirmed actuation events, and persistence across system restarts was verified through repeated power cycling tests.

The event-driven software architecture prevented race conditions, overlapping actions, and unintended hardware activation. This confirms the effectiveness of the adopted control methodology and separation of responsibilities between software and hardware layers.

4.5 Cost-Constrained Performance Evaluation

The complete system was implemented and tested within a total budget of 3029 ILS. Within this constraint, the system achieved full functional operation across sensing, actuation, software coordination, image processing, and user interaction.

The use of commercially available and hobby-grade components enabled rapid prototyping and successful demonstration of the proposed architecture. Despite inherent limitations in precision and durability, the system met its intended functional objectives and validated the design methodology under realistic resource constraints.

The results demonstrate that a modular, event-driven robotic assistance system can be realized effectively at low cost while maintaining deterministic behavior, safety controls, and extensibility for future development.

5 Discussion

5.1 Overall System Performance

The Robotic Nurse System successfully demonstrated the feasibility of integrating event-driven software coordination, sensor-gated hardware execution, and image-based decision support into a single automated medical assistance platform. As a functional prototype, the system achieved its primary design objectives and operated reliably within the defined architectural and budgetary constraints.

5.2 Software Architecture and Control Separation

At the system level, the strict separation between the Raspberry Pi coordination layer and the Arduino hardware control layer proved effective. All physical actions were executed deterministically and only in response to validated commands, preventing unintended actuation and simplifying safety verification. The web-based interface provided clear user interaction without directly influencing hardware behavior, which contributed to overall system robustness.

5.3 Image Processing and Classification Performance

The image processing and wound classification subsystem functioned correctly within its intended scope. Event-driven triggering ensured that image capture and inference occurred only under valid physical conditions. The achieved average classification confidence of approximately 40% reflects limitations in dataset size and diversity rather than deficiencies in the software pipeline itself. With access to larger, clinically sourced datasets and broader class representation, significantly higher accuracy and robustness are expected.

5.4 Mechanical and Actuation System Evaluation

Mechanical subsystems, including liquid dispensing, ointment extrusion, gauze cutting, and plaster application, operated as designed. The pill dispensing mechanism, while functional, exhibited variability due

to mechanical tolerances and the use of hobby-grade components. This limitation highlights the importance of precision manufacturing and higher-quality actuators for applications requiring repeatable and reliable operation.

5.5 Limitations and Future Improvements

Future development of the system would focus on improving mechanical precision, introducing additional safety interlocks, and enhancing the medical validity of the treatment process through expert consultation. On the software side, expanding the image dataset, refining the classification model, and incorporating confidence-based decision thresholds would substantially improve system reliability.

5.6 Scalability and System Outlook

Overall, the results confirm that the proposed architecture is scalable and adaptable. The system serves as a solid engineering foundation for further refinement toward a more robust, medically compliant, and deployment-ready solution.

6 Conclusion

This project presented the design, implementation, and evaluation of the Robotic Nurse System, an integrated hardware–software platform intended to assist with basic medical support tasks through automation, sensing, and guided workflows. The system combined mechanical actuation, embedded hardware control, event-driven software architecture, and image-based wound classification into a unified prototype that was successfully built and tested within realistic academic and budgetary constraints.

From an engineering perspective, the project demonstrated the effectiveness of a layered architecture that separates high-level coordination, perception, and decision logic from low-level hardware execution. By assigning all time-critical and safety-sensitive operations to the Arduino-based control layer and delegating coordination, user interaction, and image processing to the Raspberry Pi, the system achieved deterministic behavior, modularity, and improved fault isolation. This separation proved essential for maintaining predictable actuation behavior while allowing flexible software development and experimentation.

The mechanical and electromechanical subsystems were implemented using primarily hobby-grade components due to budget limitations. Despite this, the system successfully executed all intended physical actions, including pill dispensing, liquid dispensing, ointment extrusion, and plaster handling. While certain mechanisms, particularly the pill dispensing module, require refinement to improve reliability and precision, the overall mechanical design validated the feasibility of the proposed concepts and provided a functional proof of implementation.

A key contribution of this project was the integration of image processing and wound classification into the system workflow. A YOLO-based model was trained externally using a GPU-enabled cloud environment and deployed as a static inference model on the Raspberry Pi. The image processing pipeline, which included controlled image capture, preprocessing, inference, and cleanup, operated as designed and produced valid wound classification outputs. Although the achieved confidence levels were moderate due to the limited size and diversity of the available dataset, the results confirm that the architecture is suitable for

more advanced and accurate models when larger, clinically sourced datasets become available.

The system was tested as an end-to-end prototype, and experimental results demonstrated that the Robotic Nurse System performed efficiently relative to its cost and component quality. All core workflows executed as intended, sensor feedback was correctly used to gate actuation, and software state transitions remained consistent with physical system behavior. The results validate the design methodology and confirm that meaningful automation can be achieved even under constrained resources.

From a broader perspective, this project highlights the challenges and considerations associated with developing automated systems for medical-adjacent applications. Safety, determinism, data privacy, and user confirmation were treated as primary design drivers rather than secondary features. The system intentionally avoids autonomous medical decision-making and instead operates as a supervised, assistive platform, reinforcing the importance of ethical boundaries and responsible engineering in healthcare-related technologies.

Future development of the Robotic Nurse System would focus on several key areas, including mechanical refinement, higher-quality sensors and actuators, expanded and clinically validated image datasets, and enhanced safety mechanisms. Additional work would also be required to align the system with medical standards and regulatory requirements before any real-world deployment. These improvements would significantly increase reliability, accuracy, and clinical relevance while preserving the modular architecture established in this project.

In conclusion, the Robotic Nurse System represents a successful engineering prototype that demonstrates the integration of embedded control, software architecture, and computer vision within a coherent and testable framework. The project achieved its stated objectives, provided valuable technical and practical insights, and established a strong foundation for future research and development in automated assistive healthcare systems.

References

[1] Google LLC. *Google Colaboratory*. GPU-enabled cloud-based Python environment for machine learning and deep learning. 2024. URL: https://colab.research.google.com/.

[2] High-Level Expert Group on Artificial Intelligence. *Ethics Guidelines for Trustworthy Artificial Intelligence*. European Commission, 2019.

[3] *IEC 61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems*. International Electrotechnical Commission, 2010.

[4] *IEC 62368-1: Audio/Video, Information and Communication Technology Equipment – Safety Requirements*. International Electrotechnical Commission, 2018.

[5] Nancy G. Leveson. *Safeware: System Safety and Computers*. Addison-Wesley, 1995.

[6] Roboflow. *Train YOLO Custom Object Detection Model on Google Colab*. YouTube video used as reference for YOLO training workflow. 2023. URL: https://www.youtube.com/watch?v=r0RspiLG260.

[7] Robotnik Automation. *Applications of Robotics in Medicine*. Robotnik Automation. 2023. URL: https://robotnik.eu/applications-of-robotics-in-medicine/ (visited on 01/19/2026).

[8] Ultralytics. *YOLO: Real-Time Object Detection Documentation*. Official YOLO framework documentation. 2024. URL: https://docs.ultralytics.com/.