An-Najah National University

Faculty of Engineering and Information Technology

Computer Engineering Department

Hardware Graduation Project

PharmaMatrix

Automated Medication Storage and Dispensing System

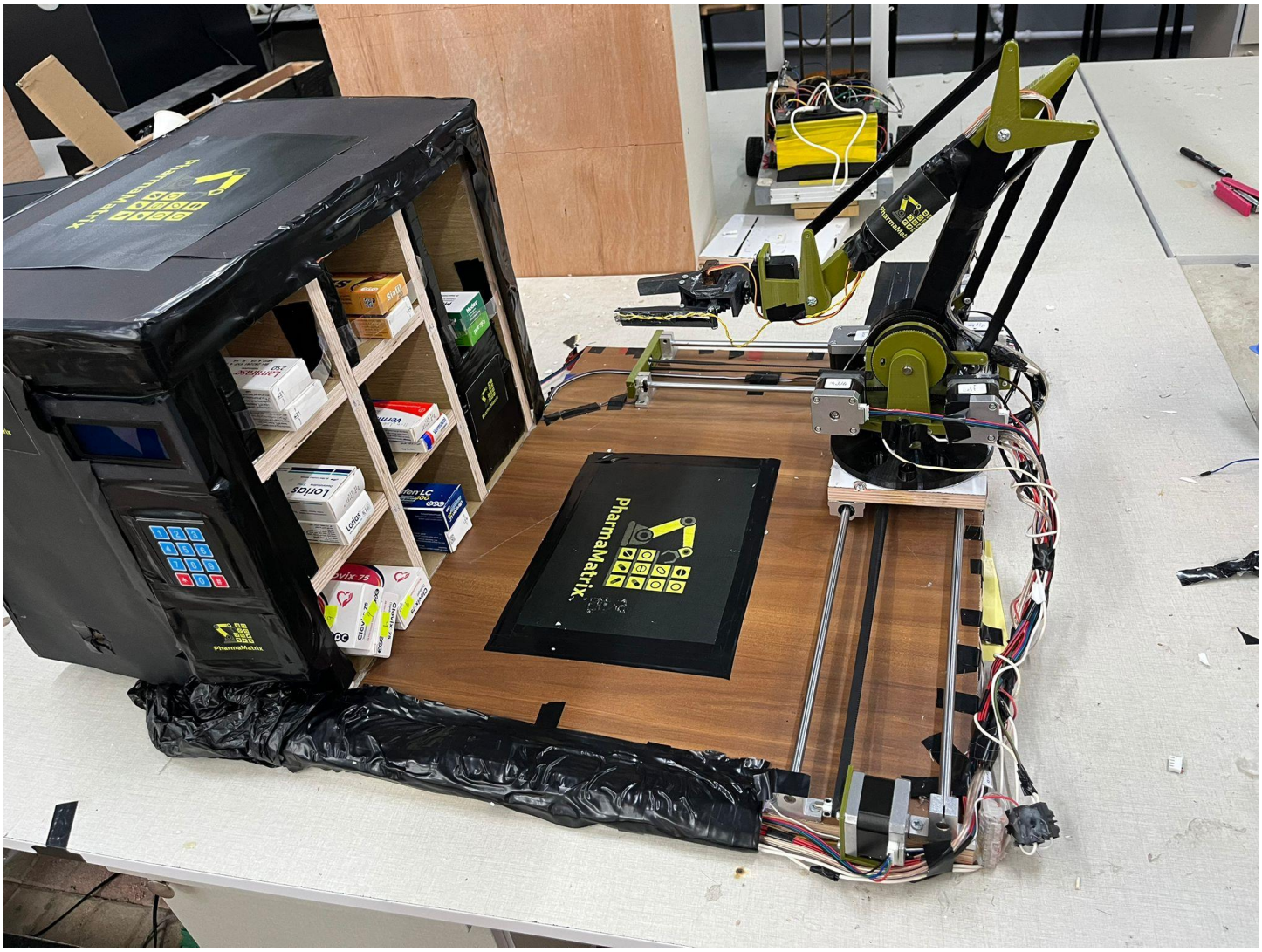
Students: Haneen Alhajali & Deeema Daraghmeh **Supervisor: Dr. Abdallah Rashed**

2025/2026

Presented in partial fulfilment of the requirements for Bachelor degree in Computer Engineering.

Table of Contents

Table of Figures


PharmaMatrix


Stafil
Lorlas
PharmaMatrix

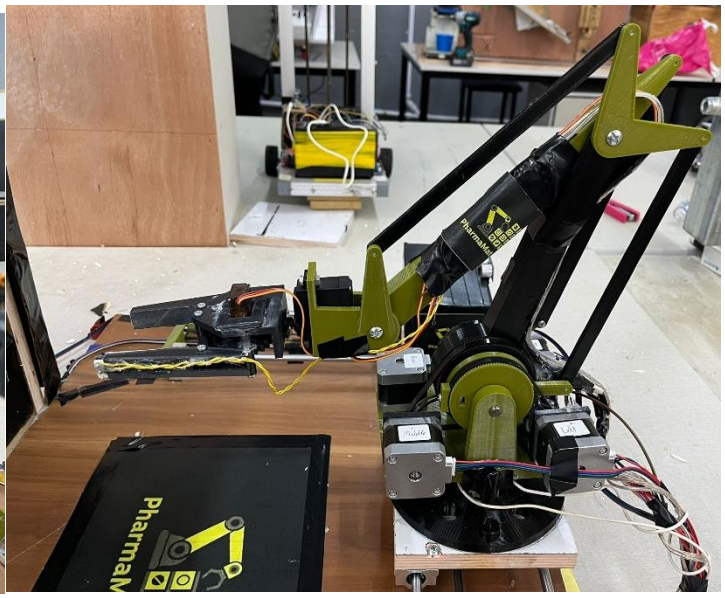

PharmaMatrix

Figure 1 : Complete PharmaMatrix System Assembly - Full view of the automated medication dispensing system

Acknowledgment

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Abdallah Rashed, for their invaluable guidance, constructive feedback, and unwavering support throughout this project. Their expertise and insightful suggestions were instrumental in shaping the direction and quality of this work.

I am also profoundly grateful to my family for their constant encouragement, patience, and understanding during the challenging phases of this project. Their unwavering belief in my abilities gave me the strength to persevere and achieve my goals.

Additionally, I extend my sincere appreciation to the faculty members of the Computer Engineering Department for equipping me with the knowledge and skills necessary to undertake this project. Finally, I would like to thank my fellow students for their valuable feedback and suggestions, which greatly contributed to the development process.

Abstract

The rapid growth of pharmaceutical services has increased the demand for accurate, efficient, and reliable medication storage and dispensing systems. **PharmaMatrix** is an automated hardware-based solution designed to reduce human error, improve dispensing speed, and enhance inventory management. The system integrates a 4-degree-of-freedom robotic arm, dual rail mechanisms, ultrasonic sensors, and embedded controllers to automate medication insertion, retrieval, and stock monitoring.

An **Arduino Mega 2560** manages real-time motion control, sensor processing, and state-based operation, while an **ESP32** enables wireless communication with a mobile application via Bluetooth. The system implements deterministic state machines for ADD, REQUEST, and automatic stock refill operations, ensuring safe execution and robust error handling. A local LCD and keypad interface, along with a mobile application, provides user interaction and system monitoring.

Testing results demonstrate reliable dispensing, accurate inventory tracking, and automatic shelf refilling, highlighting the system's ability to reduce manual workload and improve operational efficiency. PharmaMatrix provides a scalable and low-cost foundation for automated pharmaceutical storage systems with potential for future expansion.

Chapter 1: Introduction

1.1 Problem Statement

In traditional medication storage and dispensing systems, many operations are performed manually, leading to several critical issues:

- Human errors in medication selection and dispensing
- Poor organization and classification of stored medicines
- Slow and inefficient dispensing processes
- Increased workload on pharmacy staff
- Difficulty in maintaining accurate inventory records
- Product loss or theft without proper tracking

The PharmaMatrix project aims to solve these problems for pharmacies, pharmaceutical companies, and healthcare facilities by providing a precise, automated, and controlled hardware system for managing medications.

1.2 Significance

The main goal of PharmaMatrix is to build a robotic system capable of performing input and output operations in a proper and efficient manner. The project contributes by:

- Saving time and effort for pharmaceutical staff
- Improving accuracy in medication dispensing and reducing errors
- Providing automated inventory management and stock monitoring
- Enabling better organization and classification of pharmaceutical products
- Offering a scalable solution applicable to pharmacies, warehouses, and healthcare facilities

1.3 Objectives and Scope

The purpose of PharmaMatrix is to facilitate the complex medication management process through the following objectives:

1. Design and implement a 4-degree-of-freedom robotic arm for medication handling
2. Develop automated storage shelves with intelligent inventory monitoring
3. Create a dual-rail system for medication transport and arm positioning
4. Implement ultrasonic sensors for real-time stock level detection
5. Integrate a user interface through LCD display and keypad
6. Develop a mobile application for wireless medication management
7. Enable automated stock refilling from reserve storage
8. Ensure safe and reliable operation through state management and error handling

1.4 Report Organization

Chapter 2: Constraints - Discusses the challenges encountered during development.

Chapter 3: Literature Review - Examines traditional pharmacy operations and analyzes similar automated systems.

Chapter 4: Methodology - Details the system design, hardware components, software architecture, and implementation approach.

Chapter 5: Results and Discussion - Presents the system performance, problems encountered, and solutions implemented.

Chapter 6: Conclusion - Summarizes the project achievements, improvements, and future work recommendations.

Chapter 2: Constraints

2.1 Inexperience

We created numerous prototypes and tests to validate the feasibility of our design approach. However, all testing required significant time that could have been allocated to other development tasks. Even after extensive testing, research, and time investment, certain decisions remained calculated risks, as we couldn't test all components beforehand and couldn't afford project failure.

2.2 Lack of Funds

Several Budget limitations restricted certain design options and component choices. Some improvements, such as implementing computer vision for medication identification or adding more sophisticated sensors, were postponed due to cost constraints.

2.3 Lack of Time

Time management proved critical throughout development. Research, communication, testing, designing, analyzing requirements, and planning all consumed significant time. Inexperience in certain areas led to time inefficiencies that accumulated throughout the project timeline.

2.4 Lack of Tooling

Specialized tools were required for cutting, preparing, and assembling components. Many tools were either unavailable or too expensive for personal use, necessitating external assistance. 3D

printing services were utilized for custom components, and carpentry expertise was enlisted for the outer frame construction.

Chapter 3: Literature Review

When building this project, our focus centered on addressing speed, accuracy, and efficiency challenges in pharmaceutical operations. We inspected actual pharmacy workflows to understand operational requirements and identify optimal solutions.

3.1 Traditional Pharmacy Operations

In conventional pharmacies and drug stores, operations appear straightforward to observers but present significant complexity for staff. Key challenges include:

- Manual inventory management and accounting
- Sales statistics tracking and profit calculation
- Stock monitoring and planning
- Complete knowledge of medication locations and classifications

The typical workflow involves customers presenting prescriptions, after which pharmacy staff must locate and retrieve the required medication from storage. In large pharmacies with high customer volumes, maintaining accuracy, speed, and service quality becomes increasingly challenging.

3.2 Automation in Pharmaceutical Storage

For large-scale operations, manual processes result in significant inefficiencies:

- Financial losses in storage and classification operations
- Product damage during handling
- Theft and loss without detection
- Substantial labor costs

PharmaMatrix addresses these challenges by automating storage, retrieval, and inventory management. The system knows medication locations and quantities, retrieves items upon request, and handles insertion, removal, and categorization automatically, thereby saving time and reducing labor requirements.

3.3 Similar Projects and Innovations

Research into existing automated pharmacy systems revealed several complex commercial solutions, but few documented open-source implementations suitable for academic reference. Most commercial systems are proprietary with limited technical documentation.

We developed PharmaMatrix from fundamental principles, designing the mechanical structure, selecting components, and programming the control system independently. While inspiration came from various automated storage concepts, the specific implementation represents original engineering work tailored to our requirements and constraints.

Chapter 4: System Architecture

4.1 Overall System Design

The PharmaMatrix system is built around a modular architecture consisting of three primary subsystems: the mechanical subsystem (robotic arm and rail systems), the control subsystem (microcontrollers and sensors), and the user interface subsystem (LCD/Keypad and mobile application). These subsystems work together to provide seamless medication management from storage to dispensing.

4.2 Storage Architecture

The system employs a Shelf-based storage architecture with three distinct columns and rows. Column 2 , Row 1 serves as the active shelf (A1) for immediate dispensing, while Column 1 functions as the stock reserve. This dual-level approach enables automatic refilling of the active shelf when medications run low, ensuring continuous availability without manual intervention.

4.2.1 Shelf Configuration

- Column 1: Stock storage with ultrasonic monitoring for reserve inventory
- Column 2, Row 1: Active shelf (A1) with ultrasonic sensor for real-time stock level detection
- Column 3, Row 1: Active shelf (B1)

4.3 Control Architecture

The control system is centered on the Arduino Mega 2560, which manages all real-time operations including stepper motor control, sensor readings, state machine execution, and coordination of the robotic arm movements. The ESP32 provides wireless connectivity for the mobile application interface via Bluetooth, enabling remote medication requests and inventory monitoring.

Chapter 5: Methodology

5.1 Methodology Overview

The methodology of this project focuses on designing and implementing an automated medication storage and dispensing system through a systematic approach. The system integrates mechanical actuation, precision sensing, and embedded control to detect, retrieve, and dispense medications based on user requests. The methodology is structured to include system design, hardware selection and integration, mechanical construction, sensor calibration, software programming with state machine implementation, and mobile application development, ensuring the system is reliable, accurate, and user-friendly.

5.2 Design Principles

- **Precision and Grip Reliability**: Implementation of sandpaper-lined grippers and limit-switch feedback to ensure medication is never dropped or crushed.
- **Automated Stock Replenishment**: A logic-driven workflow where the system monitors shelf levels and automatically pulls from the "Stock Column" to refill empty dispensing slots.
- **State-Machine Logic**: The system operates through defined states (Idle, Busy, Error, Homing) to prevent command overlapping and ensure user safety.
- **Presence Verification**: Triple-point sensing using Ultrasonic sensors to verify the presence of medication on the shelf, the carriage, and the stock column

5.3 Development Phases

The control system is centered on the Arduino Mega 2560, which manages all real-time operations including stepper motor control, sensor readings, state machine execution, and coordination of the robotic arm movements. The ESP32 provides wireless connectivity for the mobile application interface via Bluetooth, enabling remote medication requests and inventory monitoring.

5.3.1 Phase 1: Mechanical Design and Construction

The first phase involved designing and constructing the mechanical components, including the 4-DOF robotic arm with left, right, middle, and base stepper motors. The arm was 3D printed to achieve lightweight yet rigid structure. Two rail systems were designed: one for medication transport to/from the user (medicine rail) and another for arm positioning across columns (arm rail). Each rail incorporates limit switches for precise homing and positioning.

5.3.2 Phase 2: Electronics Integration

The second phase focused on integrating electronic components. This included connecting stepper motor drivers (TB6600 and A4988), installing ultrasonic sensors at strategic positions

(shelf, stock, and medicine rail), integrating the servo-based gripper with limit switches and sandpaper-enhanced fingers for improved grip, and wiring the LCD display with I2C interface and 3x4 matrix keypad for local control.

5.3.3 Phase 3: Software Development

The third phase involved comprehensive software development with a modular architecture. The main controller implements state machines for ADD and REQUEST operations, with separate states for stock refilling. Motion control manages stepper motor coordination with AccelStepper library for smooth movements. Sensor integration handles ultrasonic readings with filtering and verification logic. The inventory management system tracks medicine quantities with database operations, and user interfaces were developed for both LCD/Keypad local control and mobile application via ESP32 Bluetooth.

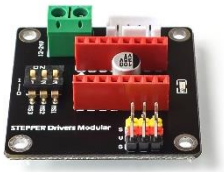
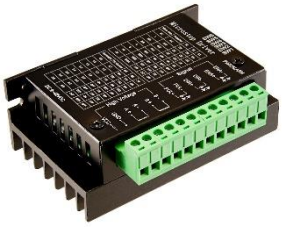
5.3.4 Phase 4: Calibration and Testing

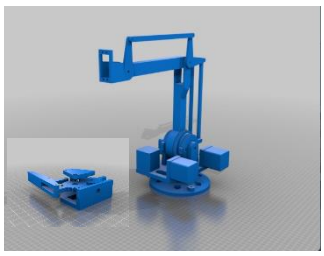
The final phase consisted of extensive calibration and testing. Motor positions were calibrated for precise pick and drop locations across all columns. Ultrasonic sensor thresholds were determined for empty/full detection of shelves and stock. Gripper force was calibrated using limit switch feedback to ensure secure holding without damage. The complete system underwent integration testing with various operation scenarios, retry mechanism validation, and error handling verification.

Chapter 6: Hardware System Design

Table 1 :PharmaMatrix Hardware Component Specification

Category	Component Name	Model/Spec	image	Quantity	Purpose in PharmaMatrix
Control	Arduino Mega 2560	R3 Board		1	Main logic controller; handles motion and sensors.
	ESP32	DevKit V1		1	Handles Bluetooth communication and Mobile App interface.

Actuators	Stepper Motor	NEMA 17		4	Controls the X-rail, Y-rail, and 4-DOF arm joints.
	High-Torque Servo	MG996R		1	Actuates the gripper mechanism in the robotic arm.
Drivers	Stepper Driver Shield	42CH Driver Shield		4	Simplifies wiring; allows A4988 drivers to be mounted easily.
	Stepper Driver	TB6600		1	High-current drivers for the base stepper motor.
	Stepper Driver	A4988		5	Precision drivers for the robotic arm joints.
Sensors	Ultrasonic Sensor	HC-SR04		3	Monitors shelf stock, stock column, and carriage presence.

	Limit Switch	KW11-3Z		9	Used for homing (start/end) on rails, arm steppers and gripper feedback.
Interface	LCD Display	16x2 with I2C		1	Displays real-time status, "Busy" states, and menus.
	Matrix Keypad	3x4 Membrane		1	Local user input for medication selection and admin.
Mechanical	Timing Belt & Pulley	GT2 (6mm)		2 sets	Translates stepper rotation into linear rail movement.
	3D Printed Parts	PLA/PETG		1 set	Custom arm chassis, gripper fingers, and rail mounts.
	Friction Material	Sandpaper		2 strips	Applied to gripper fingers to prevent medicine slippage.

Power	Power Supply	12V / 5V DC		1	Provides high current for motors and stable logic power.

6.1 Mechanical Structure

PharmaMatrix consists of a cabinet-based medication storage system combined with a robotic arm and two linear rail systems:

1. **Medication Cabinet**: Contains vertical columns, each dedicated to a single medication type. Due to the absence of a vision system, each column stores only one known medication.
2. **Medication Rail (User Rail)**: Transfers medication boxes between the user and the robotic arm.
3. **Arm Rail (Cabinet Rail)**: Allows the robotic arm to move horizontally across cabinet columns.

The robotic arm is **3D-printed** and provides **four degrees of freedom**:

- Base rotation
- Left movement
- Right movement
- Vertical (mid) movement

Each axis is driven by a NEMA 17 stepper motor and protected by limit switches for accurate homing.

The gripper assembly is 3D printed with integrated limit switches and sandpaper-enhanced contact surfaces for secure medication handling.

6.2 Controllers and Interfaces

6.2.1 Arduino Mega 2560

The Arduino Mega 2560 serves as the main microcontroller, executing real-time motion control, managing state machines, coordinating stepper motor movements via AccelStepper

library, reading ultrasonic sensors and limit switches, controlling the servo gripper, and communicating with the LCD/Keypad interface. Its 54 digital I/O pins, 16 analog inputs, and 256 KB flash memory provide sufficient resources for the complex control logic required by PharmaMatrix.

he Arduino Mega serves as the central controller, responsible for:

- Stepper motor motion control
- Homing sequences
- Sensor reading and validation
- System state management
- Communication with the ESP32

It is selected due to its large number of I/O pins and ability to handle multiple peripherals simultaneously.

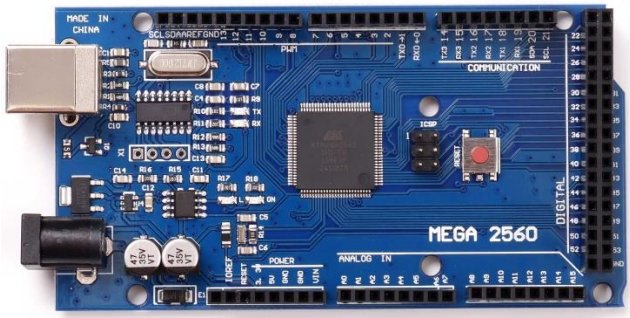


Figure 2: Arduino Mega 2560 - main motion and sensing controller

6.2.2 ESP32 Microcontroller

An ESP32 microcontroller provides wireless connectivity through Bluetooth communication with the mobile application. It receives medication requests from the user's smartphone, transmits commands to the Arduino Mega via serial communication, and sends status updates and inventory information back to the mobile app. The ESP32 enables remote operation and monitoring of the PharmaMatrix system.

Figure 3: ESP32 Module - provides Bluetooth connectivity for mobile app

6.2.3 LCD Display and Keypad

A 16x2 character LCD display with I2C interface provides real-time system status, menu navigation, and operation feedback. The display shows medication selection menus, quantity inputs, operation progress, and error messages. A 3x4 matrix keypad enables local user input for selecting medications, entering quantities, and navigating through menu options without requiring the mobile application, User input is disabled while the system is busy to prevent unsafe interactions.

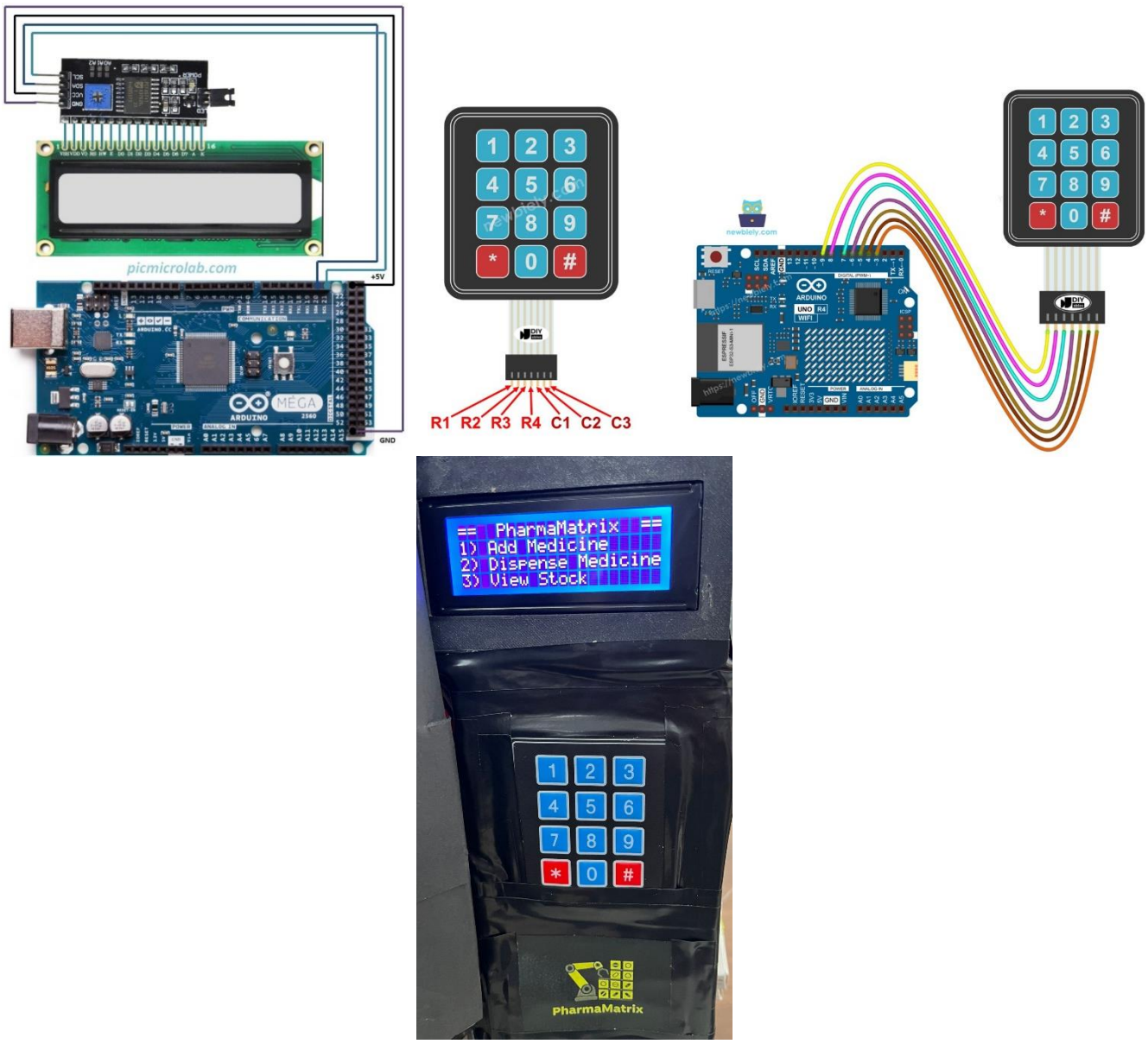


Figure 4: 16x2 LCD Display with I2C and 3x4 Matrix Keypad - local user interface

6.3 Actuators and Drivers

6.3.1 NEMA 17 Stepper Motors

Four NEMA 17 stepper motors with shield stepper driver 42ch to drive the robotic arm axes (left, right, middle joints). These motors provide 200 steps per revolution (1.8° step angle) and deliver sufficient torque for precise positioning while maintaining compact size. The motors are selected for their balance of torque, speed, accuracy, and cost-effectiveness for the arm payload requirements.

- Left Joint Motor: Controls left arm segment movement
- Right Joint Motor: Controls right arm segment movement

- Middle Joint Motor: Controls middle arm segment for reach adjustment

To control the motor, we must use a driver, and for that we used a 4988 driver. To make it easier for us to control with high accuracy, and to protect the motor.

we also used a special shield in that driver to facilitate the process of connecting with the motor and giving it sufficient voltage and to ensure high accuracy.

In order for us to connect the motor with the two-entry coil in the shield driver, where there are four entrances in the shield, especially to the coil, which are (1A,1B,2B,2A) which (1A,1B) mean the first coil, and (2B,2A) the second coil.

To determine the coil, through the wires coming out of the motor, we wrapped each two wires together, and then we twist the motor by hand. If we feel it is difficult to move the motor, these two wires represent a first coil and so on.

Also the shield has 2 input for power supply (12v), input for (step), input for (direction), input for enable, and input for (Vcc,Gnd)-(5v).

Below is a picture of the linking process:

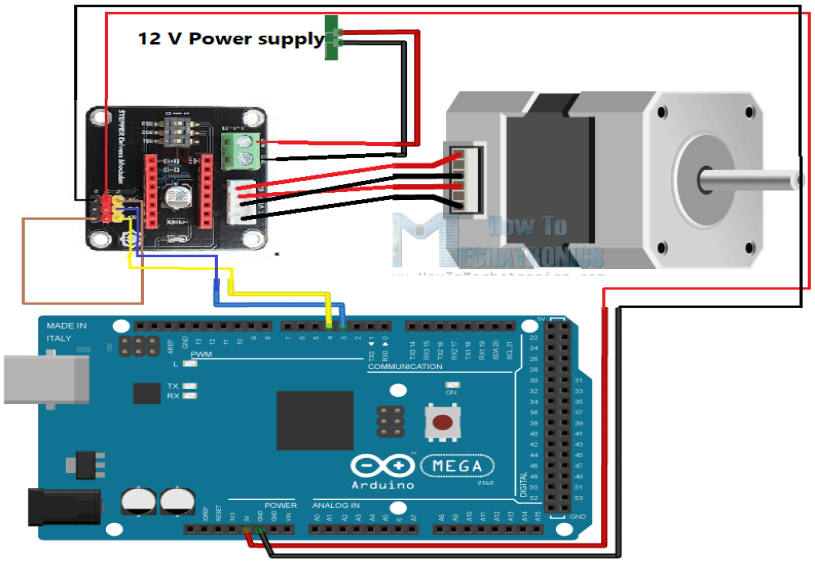


Figure 5: NEMA 17 Stepper Motor Circuit- used for arm joints

6.3.2 TB6600 Stepper Motor Drivers

TB6600 stepper drivers control the arm Base Motor motors to supply sufficient current to handle the load of the robotic arm which supports higher current output and provides better torque and stability for heavy-load applications using PUL (pulse), DIR (direction), and ENA (enable) signals from the Arduino Mega. These drivers support microstepping configurations up

to 1/16 step and provide current control to optimize motor performance. The drivers handle motor currents up to 4A, providing adequate power for smooth and reliable operation.

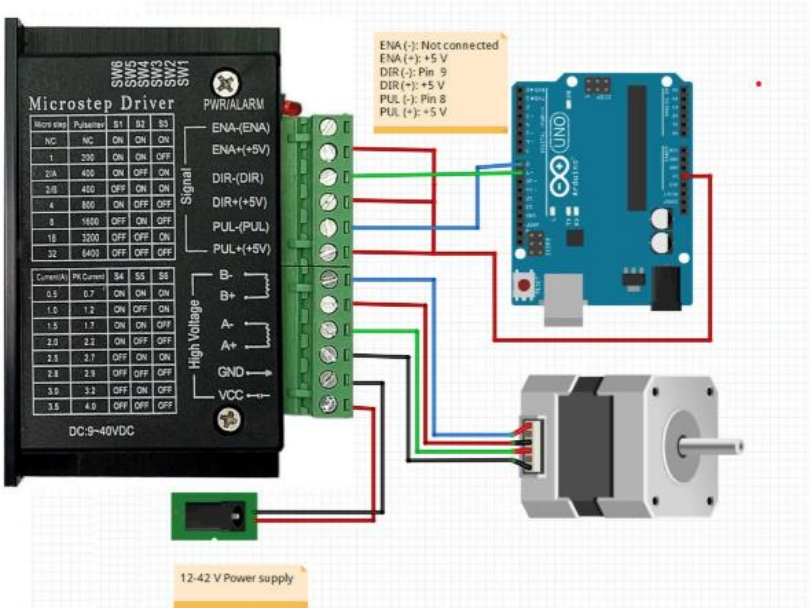


Figure 6: TB6600 Stepper Driver Circuit - controls arm Base stepper motors

6.3.3 MG996R Servo Motor for Gripper

A high-torque MG996R servo motor operates the gripper mechanism. The servo provides approximately 10 kg-cm torque at 4.8V, sufficient for reliable medication grasping and release. The gripper operates between two positions: GRIP_OPEN (100°) for medication entry/exit and GRIP_CLOSE (160°) for secure holding during transport. The gripper fingers are enhanced with sandpaper to increase friction and improve grip reliability.

Moving Servo motor:

We can access the Servo Motor. directly using (Servo.h) library

➔#include <Servo.h>

-we will use a some function the help us to read and write on Servo Motor :

Servo.attach(pin#): to detect which pin (10) is connected with Servo Motor

Servo.Write(angle): to give the angle value to the Servo Motor.

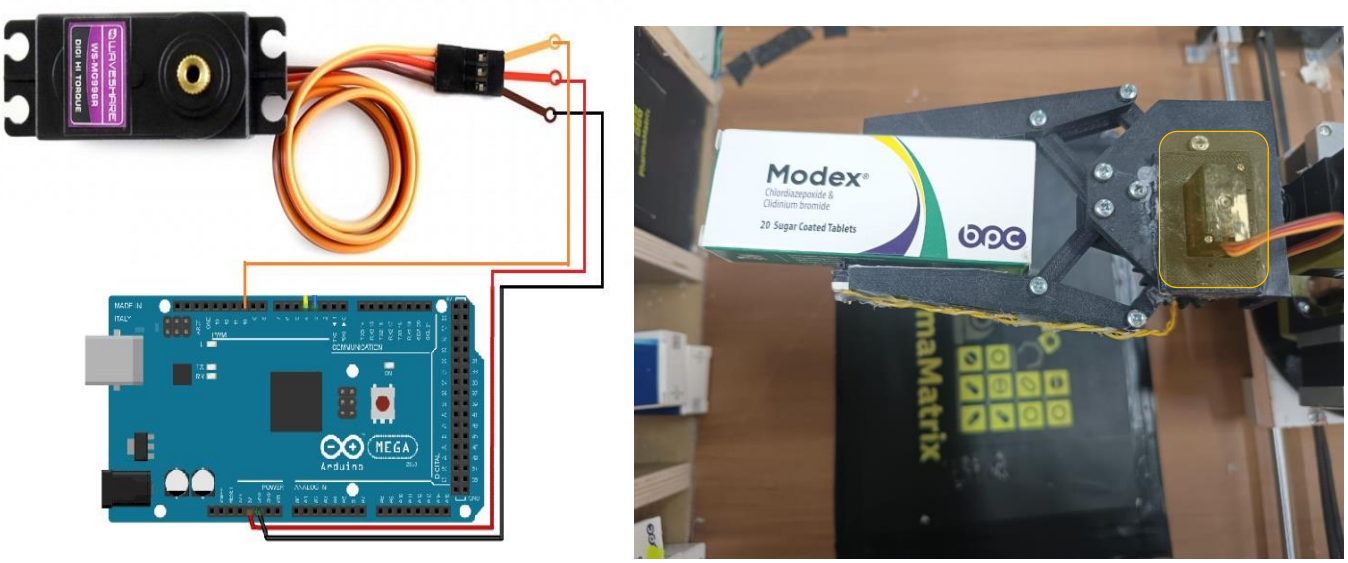


Figure 7: MG996R Servo Motor Circuit- controls gripper open/close mechanism

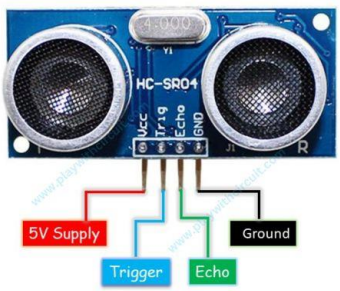
6.4 Sensors

6.4.1 Ultrasonic Sensors

Three ultrasonic sensors (HC-SR04) provide non-contact distance measurement for medication presence verification and stock level monitoring:

- A1 Shelf Sensor: Monitors the active shelf (Column 2) to detect if medications are present (full) or absent (empty). Empty threshold: ≥9 cm, Full threshold: ≤2 cm
- Stock Sensor: Monitors the stock column (Column 1) for reserve inventory levels. Empty threshold: ≥17 cm, Full threshold: ≤5 cm
- Medicine Rail Sensor: Verifies medication presence on the transport rail during ADD and REQUEST operations. Present threshold: ≤8 cm

These sensors enable the system to implement verification logic, retry mechanisms, and automatic stock refilling when the active shelf runs low.



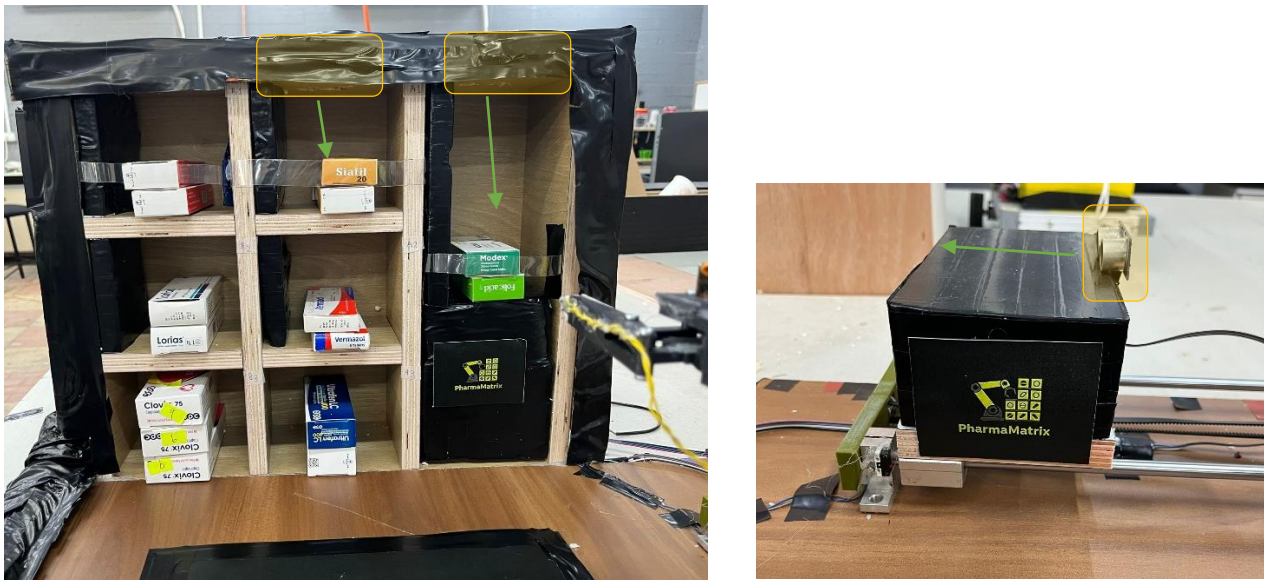


Figure 8: HC-SR04 Ultrasonic Sensor - used for presence verification and stock monitoring

6.4.2 Limit Switches

Multiple limit switches provide position feedback and safety interlocks throughout the system:

- **Arm Joint Limit Switches**: Four switches (left, right, middle, base) define home positions for the arm joints, enabling reliable homing sequences
- **Rail Limit Switches**: Four switches (two per rail) mark the start and end positions of the medicine rail and arm rail for precise positioning
- **Gripper Limit Switch**: Integrated into the gripper mechanism to provide force feedback, confirming when medication is securely grasped or fully released

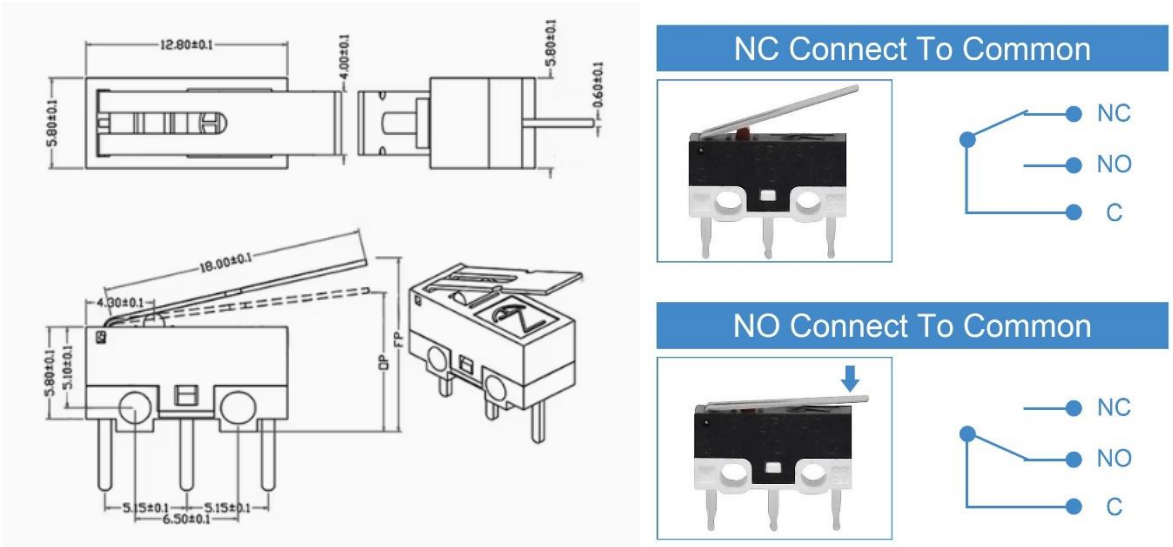


Figure 9: Limit Switch - used for homing and position detection throughout the system

6.5 Power System

The system uses a multi-voltage power distribution architecture. A 12V power supply provides power to stepper motor drivers and motors. A 5V regulated supply powers the Arduino Mega, ESP32, LCD display, and logic circuits. The servo motor operates on a separate 5V supply to prevent voltage drops during gripper operation. All power rails share a common ground to ensure proper signal reference levels across the system.

Figure 10: Power Supply

6.6 Wiring and Integration

All components are interconnected using standard hookup wire with appropriate gauge for current requirements. Signal cables are routed away from high-current motor wires to minimize electromagnetic interference. Connectors are labeled for maintenance and troubleshooting. The wiring is organized to allow easy access to components while maintaining mechanical clearances for moving parts.

Chapter 7: Software Architecture

7.1 Programming Environment

- **Language**: Arduino C/C++
- **Libraries**: AccelStepper, Servo

The software is modular and distributed across multiple files to separate responsibilities.

7.2 Software Modules

The firmware is divided into logical components, including:

- Motion control
- Inventory management
- User input handling
- LCD display management
- System state control

This modular structure improves readability, debugging, and future expansion.

7.3 State Machine Implementation

The system employs deterministic state machines for predictable operation sequences. Three primary state machines handle different operation types: REQUEST state machine for dispensing medications from shelf/stock, ADD state machine for adding medications to shelf/stock, and STOCK REFILL state machine for automatic refilling of the active shelf from stock reserves.

7.4 System States

The system operates using defined internal states:

- **Idle State**: Waiting for user input
- **Busy State**: An operation is executing
- **Operation Done State**: Task completed successfully
- **Error State**: Failure detected, requiring recovery or reset

State flags ensure that no new command interrupts an ongoing operation.

7.5 Operational Workflow

A typical operation follows this sequence:

1. User selects an operation (add or dispense)
2. System enters Busy state
3. Homing is performed if required
4. Rails and arm move to the target shelf
5. Gripper opens and attempts to grab medication
6. Sensor feedback verifies success
7. Medication is transferred to or from the rail

8. System returns to home position
9. Operation is completed and system returns to Idle

Each step executes only after confirmation of the previous step.

Chapter 8: System Operation

8.1 Initialization Sequence

Upon power-up, the system executes an initialization sequence. The Arduino Mega initializes all I/O pins, configures stepper motors with speed and acceleration parameters, attaches the servo and sets gripper to open position, initializes the LCD display and keypad, loads sample inventory data into memory, and displays a welcome message. The ESP32 initializes Bluetooth and waits for mobile app connection. The system then enters an idle state, ready to accept user input from either the LCD/Keypad or mobile application.

8.2 ADD Medicine Operation

The ADD operation allows users to add medications to the system storage. The complete workflow proceeds as follows:

1. User selects Add Medicine from main menu (via keypad or mobile app)
2. System displays medication list; user selects target medication
3. User enters quantity to add and confirms
4. System checks target column (for example if it's A1 or Stock) capacity using ultrasonic sensors
5. If A1 is full, system automatically selects Stock column; if both full, operation is rejected
6. System homes both medicine rail and arm rail
7. LCD displays "Please place the medicine..." prompt
8. System waits for user to place medication on medicine rail, monitoring with ultrasonic sensor
9. When medication detected, medicine rail moves to end position
10. Arm rail positions to pickup location
11. ADD state machine executes: arm homes, rotates to rail, picks medication with gripper, verifies pickup, rotates to shelf, positions arm rail to target column, executes multi-step shelf approach, and releases medication with verification
12. If any verification fails, system retries up to 2 times
13. Medicine rail returns home

14. Arm rail returns home
15. If operation successful, inventory quantity is incremented
16. LCD displays "ADD Completed" or error message if failed

8.3 REQUEST Medicine Operation

The REQUEST operation dispenses medications to users. The workflow includes:

1. User selects Dispense Medicine from main menu
2. System displays available medications with current quantities
3. User selects medication and enters desired quantity
4. System validates quantity against available stock
5. System homes both rails and moves medicine rail to end position to receive medication
6. System checks if A1 shelf is empty using ultrasonic sensor
7. If A1 is empty and stock is available, system initiates two-phase operation: Phase 1 - Serve user directly from stock column, Phase 2 - Automatically refill A1 from stock for future requests
8. If A1 has medication (normal case), REQUEST state machine executes for each quantity: arm positions to target column, picks medication from shelf with verification, rotates to medicine rail, drops medication with ultrasonic verification, and returns to home
9. If verification fails, system retries up to 2 times
10. Medicine rail returns home, making medication accessible to user
11. If operation successful, inventory quantity is decremented
12. LCD displays "Retrieve Completed" or error message

8.4 Automatic Stock Refilling

The system implements intelligent automatic refilling of the active shelf (A1) from stock reserves. When a REQUEST operation detects that A1 is empty, the system first serves the user's immediate request directly from the stock column, then automatically initiates a refill operation. The refill continues in a loop, repeatedly picking from stock and dropping to A1, until either the stock is depleted or A1 is full. This ensures A1 is always ready for the next user request without manual intervention, maximizing system availability and user convenience.

8.5 Error Handling and Recovery

The system implements comprehensive error handling:

- Verification Failures: When pick or drop verification fails, the system logs the error and retries the operation up to MAX_RETRIES times
- Gripper Failures: If the gripper limit switch doesn't detect medication after 2-5 seconds, a timeout triggers retry logic
- Empty Stock/Shelf: When ultrasonic sensors detect empty conditions, appropriate error messages are displayed and operations are halted
- Final Failures: After exhausting retries, the system marks the operation as failed, displays error message to user, does not update inventory, and returns all components to home position
- System Recovery: Failed operations cleanly terminate, allowing the system to accept new requests without requiring manual reset

---

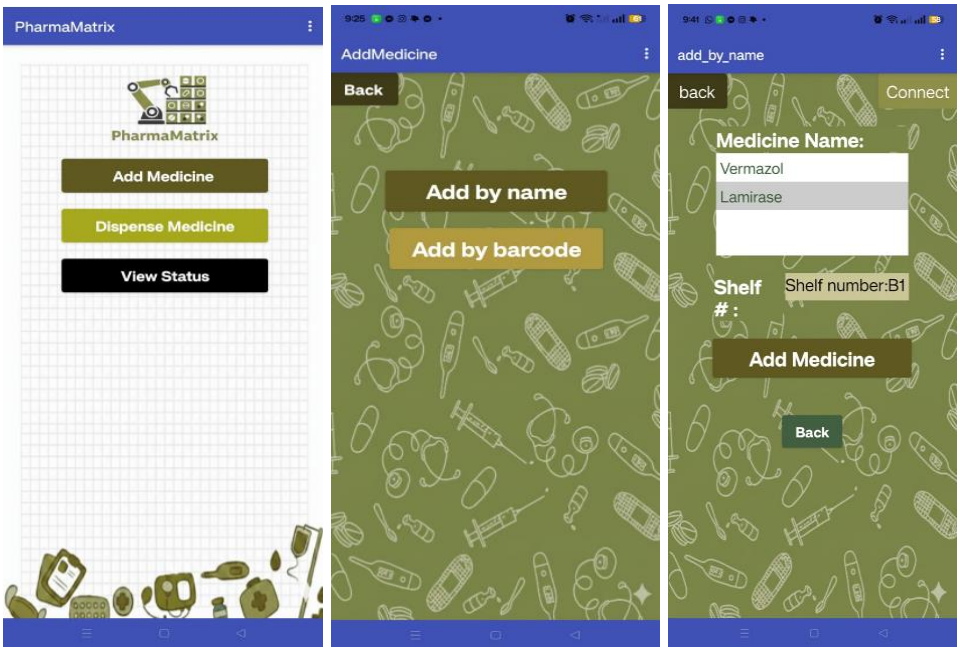
Chapter 9: Mobile Application Integration

A mobile application is developed using MIT App Inventor to provide wireless control and monitoring.

The application includes:

- Home screen for navigation
- Add-medicine screen (manual selection or barcode entry)
- Dispense-medicine screen with inventory list
- Stock-status screen showing quantities, shelf numbers, and barcodes
- Admin screen for automated input operations

The app communicates with the ESP32 via Bluetooth, sending shelf numbers and quantities. The ESP32 forwards these commands to the Arduino Mega for execution.



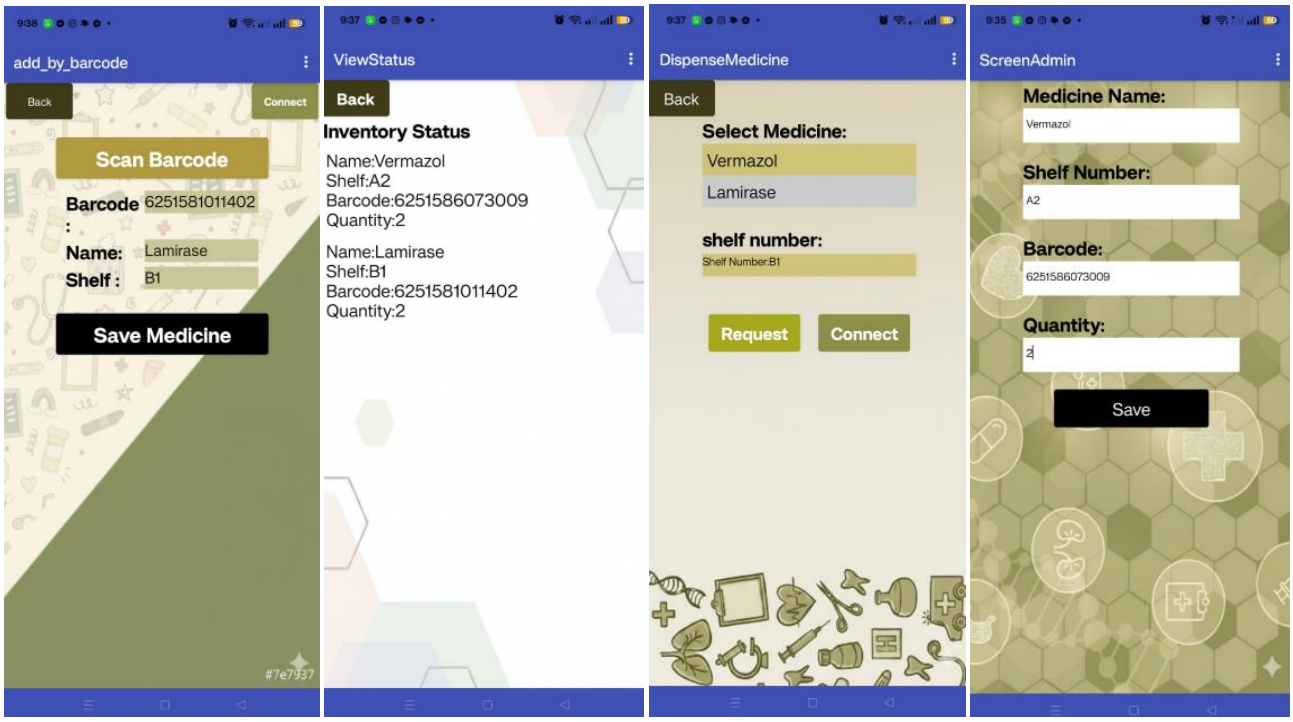


Figure 11: Mobile app screens

Chapter 10: Results and Problems Discussion

10.1 System Performance

PharmaMatrix successfully demonstrates automated medication storage and dispensing capabilities. The system reliably performs add and request operations with the following characteristics:

- Accurate positioning using stepper motors and limit switches
- Successful medication gripping with tactile feedback confirmation
- Automatic inventory tracking and stock level monitoring
- Automated refilling from stock when active shelf becomes empty
- Responsive user interface via LCD and mobile application

10.2 Challenges and Solutions

10.2.1 Motor Driver Issues

Problem: The A4988 motor driver used for the base motor was unable to supply sufficient current to handle the load of the robotic arm. This limitation caused unstable movement and torque loss when rotating the arm, especially under higher mechanical load.

Solution: The issue was resolved by replacing the A4988 driver with a TB6600 motor driver, which supports higher current output and provides better torque and stability for heavy-load applications. This change significantly improved the base motor performance and ensured reliable arm movement.

10.2.2 Shelf Access Complexity

Problem: shelf designs did not accommodate smooth arm insertion medication, Direct linear paths resulted in the arm hitting shelf edges or walls.

Solution: Implemented multi-step positioning sequences to ensure collision-free movement.

10.2.3 Medication Positioning Consistency

Problem: Medications on the rail and shelves could not be guaranteed to maintain consistent orientation, complicating reliable gripper engagement.

Solution: This limitation was accepted as a constraint. The system requires medications to be placed in standard orientations. Future enhancements could include computer vision for position detection and adjustment.

10.2.4 Gripper Reliability

Problem: The initial gripper design using smooth 3D printed surfaces frequently dropped medications during transport, especially when the arm moved quickly or changed direction.

Solution: Sandpaper was adhered to the gripper finger contact surfaces to increase friction coefficient. This simple modification dramatically improved grip reliability. Additionally, a limit switch was integrated into the gripper mechanism to provide force feedback, confirming when medication was securely grasped. The software was enhanced to verify the limit switch state before proceeding with arm movement, preventing premature transport of unsecured medications.

10.2.5 Medication Position Variability

Problem: Medications placed on shelves or rails by the arm don't always land in identical positions. This variability made it difficult for the ultrasonic sensors to consistently detect presence, as medications might be slightly offset from the sensor's detection beam.

Solution: Instead of expecting exact positioning, generous detection thresholds were calibrated based on worst-case positioning. For example, the medicine rail sensor uses ≤8 cm threshold, providing margin for medications positioned anywhere within the rail area. The shelf sensor thresholds (≤2 cm for full, ≥9 cm for empty) similarly account for position variability. This approach prioritizes reliability over precision in medication placement, which is acceptable given the system's purpose.

10.2.6 Ultrasonic Sensor Noise

Problem: Ultrasonic sensors occasionally returned spurious readings due to reflections from shelf edges, angled surfaces, or interference from nearby sensors.

Solution: The sensor reading function includes timeout handling using pulseIn() with a 30ms timeout, returning 0 for no echo. Readings of 0 are handled appropriately in detection logic. While median filtering could further reduce noise, the current threshold-based approach with adequate margins proved sufficient for reliable operation. Future improvements could implement median of multiple readings for critical decisions.

10.3 Testing Results

Extensive testing validated system functionality across multiple operational scenarios:

- Single medication request operations completed successfully
- Multiple sequential requests executed without errors
- Medication addition to active shelves and stock performed correctly
- Automatic stock refilling triggered appropriately when active shelf emptied
- Error recovery mechanisms functioned as designed during grip failures

- Inventory database maintained consistency throughout operations

The final system demonstrates reliable performance meeting project objectives, though limitations exist regarding medication orientation control and scalability to diverse medication sizes.

Chapter 11: Conclusion

11.1 Summary

PharmaMatrix successfully demonstrates automated medication storage and dispensing through integration of mechanical systems, electronic control, and embedded software. The project addresses real challenges in pharmaceutical operations by automating traditionally manual processes, reducing errors, and improving efficiency.

We developed a working system with substantial room for improvement and expansion. The modular architecture facilitates future enhancements, and the core functionality proves the viability of automated pharmaceutical storage for various applications including pharmacies, warehouses, and healthcare facilities.

11.2 Project Achievements

The project successfully achieved the following objectives:

- Designed and constructed 4-degree-of-freedom robotic arm
- Implemented dual-rail positioning system
- Created functional gripper with tactile feedback
- Integrated ultrasonic sensors for inventory monitoring
- Developed modular software architecture
- Implemented state machine control for reliable operations
- Created user-friendly interface via LCD and keypad
- Developed mobile application for wireless control
- Implemented automated stock refilling functionality
- Established error handling and recovery mechanisms

11.3 Improvements and Future Work

Several improvements would enhance system capabilities:

11.3.1 Computer Vision Integration

Adding a camera with Raspberry Pi or similar platform would enable:

- Automatic medication identification via barcode or label recognition
- Dynamic gripper adjustment based on medication size and orientation
- Position verification before pickup attempts
- Support for diverse medication packages without manual orientation

11.3.2 Scalability

- Upgrade to industrial-grade motors and drivers for continuous operation
- Expand storage capacity with additional columns and shelves
- Implement multi-arm configuration for parallel processing
- Design temperature-controlled storage zones for sensitive medications

11.3.3 Optimization

- Develop path optimization algorithms for multi-medication requests
- Implement batch processing for multiple operations
- Optimize motor acceleration profiles for faster operation
- Calculate shortest routes for complex medication retrieval sequences

11.3.4 Barcode Scanner Integration

Adding a barcode scanner would enable automatic medication identification during ADD operations. Currently, users must manually select medications from a list or enter barcode numbers. An integrated scanner at the medicine rail position could automatically read medication barcodes as they're placed, reducing user input requirements and potential selection errors. The existing inventory structure already includes barcode fields, and the findMedicineByBarcode() function is implemented but currently unused due to lack of physical scanner hardware.

11.3.5 Emergency Stop Mechanism

Adding a physical emergency stop button that immediately halts all motor movements would improve safety during maintenance or malfunction scenarios. The button would set a global flag that all motion functions check before executing. Recovery from emergency stop would require deliberate user action to clear the flag and potentially re-home the system.

11.3.6 Collision Detection

Implementing current sensing on stepper motors could detect unexpected resistance indicating collisions or jams. When current exceeds normal levels, the system could

immediately stop movement and alert the user. This would prevent damage to the mechanical system and medications in case of unexpected obstacles or mechanical binding.

11.4 Final Outcome

PharmaMatrix represents a successful integration of hardware automation and embedded systems for pharmaceutical applications. The project demonstrates that automated medication management is achievable with accessible technologies and provides a foundation for future development. The system successfully addresses core objectives of improving accuracy, reducing manual labor, and enabling intelligent inventory management.

The project is applicable across multiple domains including pharmacies, producing companies, healthcare facilities, and general warehousing. With proper enhancements and scaling, PharmaMatrix could significantly impact pharmaceutical operations by providing reliable, efficient, and cost-effective automation.

Chapter 12: Bibliography

Arduino Mega 2560 Documentation. Available at: docs.arduino.cc/hardware/mega-2560/

NEMA 17 Stepper Motor Datasheet. Components101. Available at: https://components101.com/motors/nema17-stepper-motor

TB6600 Stepper Motor Driver Guide. Available at: https://www.makerguides.com/tb6600-stepper-motor-driver-arduino-tutorial/

MG996R Servo Motor Specifications. Components101. Available at: https://components101.com/motors/mg995-servo-motor

HC-SR04 Ultrasonic Sensor Tutorial. Available at: https://www.sparkfun.com/products/15569

AccelStepper Library Documentation. Available at: http://www.airspayce.com/mikem/arduino/AccelStepper/

ESP32 Bluetooth Programming Guide. Available at: https://randomnerdtutorials.com/esp32-bluetooth-classic-arduino-ide/

MIT App Inventor Documentation. Available at: http://ai2.appinventor.mit.edu/

LiquidCrystal I2C Library. Available at: https://github.com/johnrickman/LiquidCrystal_I2C

Keypad Library for Arduino. Available at: https://playground.arduino.cc/Code/Keypad/