ECE 445/ME 470

SENIOR DESIGN LABORATORY

FINAL REPORT

Smart Medicine Box

Team #6

RUOLIN ZHAO (ruolinz5)
WENTAO KE (wentaok2)
ZHIYI MOU (zhiyim2)
YUTONG LOU (ylou10)

PROFESSOR: WEE-LIAT ONG
TA: Taocheng Yu

May 29, 2025

Abstract

This project presents the design and implementation of a smart medicine box aimed at improving medication adherence among elderly users. The system integrates four functional modules: a mobile application for prescription recognition and validation, a secure rotating medicine storage tray, a suction-based pill grabbing and dropping mechanism, and a reminder system with drawer retrieval and audible alerts. Utilizing a multimodal AI model and Bluetooth-enabled communication, the system automates the dispensing process while ensuring safety, accuracy, and user accessibility. Iterative testing and hardware optimization addressed key challenges in mechanical reliability and pill detection. The final prototype demonstrates that low-cost, modular hardware combined with intelligent software can provide a robust and user-friendly solution to a critical healthcare need.

Contents

1 Introduction

1.1 Problem Statement

The rapid growth of the global elderly population is intensifying pressure on healthcare systems, particularly in the area of medication adherence. By 2050, the population aged 60 or above is expected to reach nearly 2 billion worldwide, and studies have shown that over 22% of adults aged 40–79 in the United States are engaged in polypharmacy, the simultaneous use of five or more medications [1]. This increases the likelihood of dosing errors, missed medications, and adverse drug events—risks that are especially dangerous for older adults with cognitive decline or limited independence. While existing smart pillboxes provide basic functionalities such as alarms and storage, they still rely heavily on manual pill organization and often suffer from low capacity, poor accuracy in dispensing, or user-unfriendly interfaces [2][3].

1.2 Solution

1.2.1 Overview

To address these limitations, we propose a novel smart medicine box that integrates mechanical automation and multimodal AI assistance to enhance the safety, accuracy, and independence of elderly users in managing their prescriptions. Our design combines an air-pump-based suction mechanism for automatic pill pick and drop, a rotating multi-compartment storage tray, and a mobile application for remote control, inventory management, and real-time prescription validation via vision-language models. Compared to existing solutions, our system emphasizes precision, modularity, and user convenience. To contextualize our proposed solution, Figure 1 presents a visual overview of the system in use. It illustrates how the smart medicine box interacts with the user and external components, including a mobile application and a cloud-based prescription analysis module. This scenario-based diagram highlights key functionalities such as automatic pill dispensing, remote control via smartphone, and AI-driven prescription verification, demonstrating how our solution fits seamlessly into a typical daily medication routine.

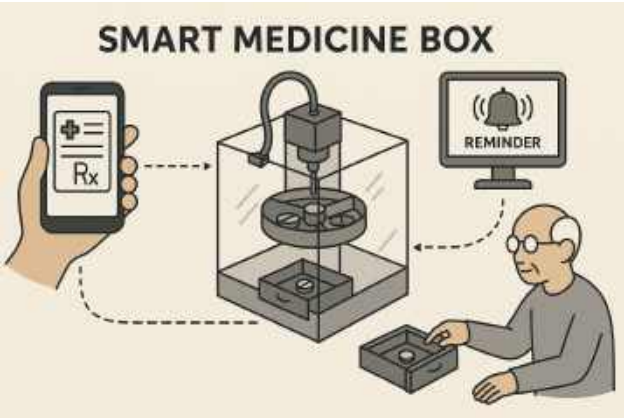


Figure 1: Visual Aid

1.2.2 Block Description

Figure 2 shows how our system works.

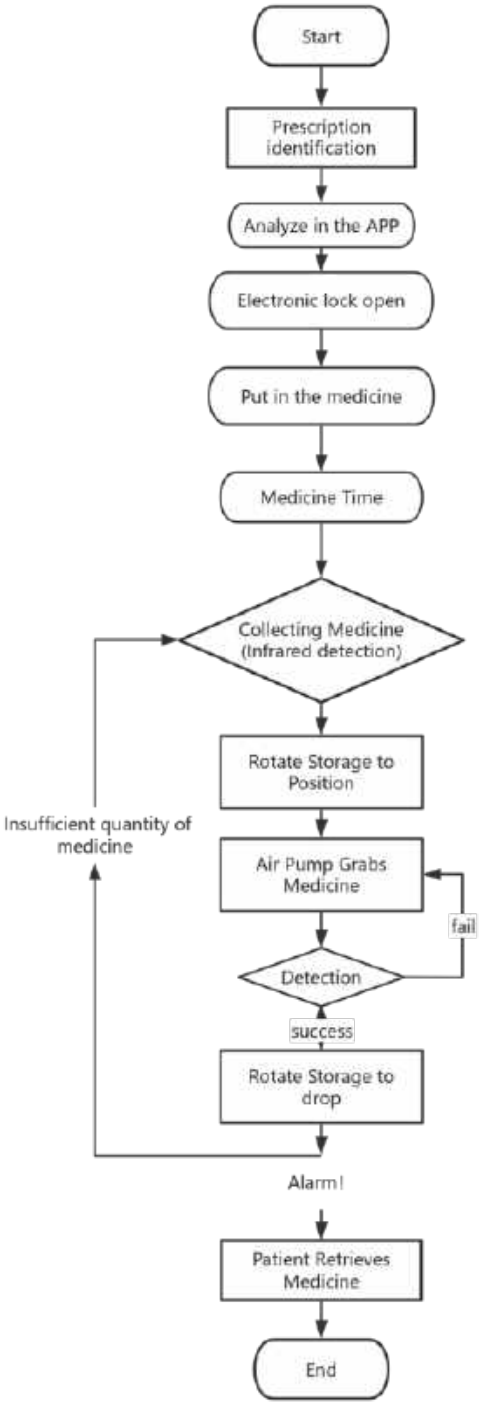


Figure 2: System Workflow

The smart medicine box system begins with the user uploading a prescription. The mobile app identifies the prescription content and analyzes its validity using an AI model. Once the safety of the prescription is confirmed, users can click a button on the app to open the electronic lock on the medicine box. Then, users need to place different types of medicine into the corresponding storage compartments according to the automatically generated list on the app.
After that, users should close the door and confirm on the completion of medicine placement on the app. Then, the processed prescription, which contains types, dosage, and administration time of the patient-needed medicines, will be uploaded to the medicine box through bluetooth.
At the scheduled administration time, the system will begin to collect pills. The storage tray rotates to a specific degree to align the correct compartment beneath the suction nozzle. The air pump is activated to grab a pill, and a detection sensor verifies whether the pickup is successful. If the attempt fails, the system retries until it is successful. Upon successful pickup, the tray rotates again to position the nozzle above the drop slot, and the pill is released into the collection drawer. The system retries this routine until all the medicines needed to take in that time slot are in the collection drawer. The system triggers an alarm to remind the user. And finally, the patient retrieves the medicine, completing one dispensing cycle.

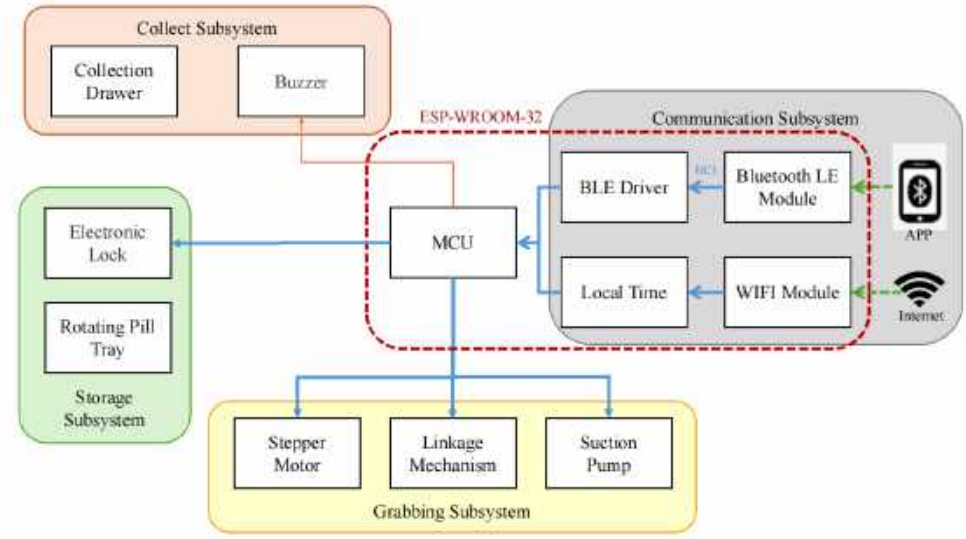


Figure 3: Block Diagram

The smart medicine box integrates four key subsystems, as shown in Figure 3 : the **APP Subsystem**, **Medicine Storage Subsystem**, **Grabbing and Dropping Subsystem**, and **Medicine Collection and Reminder Subsystem**. These subsystems work collaboratively to automate the process from prescription input to pill dispensing. The mobile application plays a central role in the digitization of prescriptions - users can upload a photo of a prescription, which is then analyzed using a multi-modal large model. The system checks for the prescription safety, dosage correctness, and potential interactions before sending

the validated data to the hardware unit via Bluetooth. Once received, the smart box record the prescription into its control system memory automatically.

During the development process, we revised our original subsystem definitions to better reflect functional responsibilities and improve system modularity. Initially, our architecture was divided into traditional hardware-based subsystems—such as Power Management, Communication, Mechanism, and Sensor—based on the type of electrical components. However, as the design evolved, we adopted a task-oriented block structure, grouping components into four functional modules: APP subsystem, Medicine Storage subsystem, Grabbing and Testing subsystem, and Medicine Collection and Reminder Subsystem. This new division aligns more closely with the user experience and software-hardware interaction flow, making debugging and testing more efficient.

Additionally, one key component, the pressure transducer used for air pressure-based pill detection, was discarded after empirical testing revealed its unreliable results due to the small air pressure change the picked-up pills caused. Instead, we restructured the Grabbing and Dropping Subsystem to rely entirely on optical detection using infrared sensors, simplifying the subsystem and improving stability.

To meet project requirements, our system must achieve the following:

- Interpret prescription photos with an accuracy rate of at least 80%.
- Maintain synchronization between the rotating storage and air pump grabbing mechanism.
- Ensure dispensing of a single pill per processing cycle, with a success rate of at least 75%.
- Complete dispensing after receiving validated data which shows required medicine is dispensed.
- Issue timely medicine-taken reminders and ensure design security using an electronic lock.

This smart medicine box represents a robust, scalable solution that combines mechanical precision with intelligent software. It supports secure, accurate medication handling while offering users greater autonomy and safety.

2 Design

Figures 5 to 8 illustrate the design evolution of the smart medicine box through three stages of CAD modeling. The first conceptual model (Figure 5) served as an early prototype to convey the basic operation mechanism. It lacked detailed component geometry—most notably, the medicine tray had no defined bottom slots, the vertical linkage was oversimplified, and motor integration was not yet considered.
As an alternative direction explored during the initial phase, a gashapon-inspired dispensing system was proposed (Figure 4). This concept involved a circular rotating tray equipped with adjustable blades that could regulate pill release based on the dimensions of different medications. The channel width could be fine-tuned by modifying the screw positions in the sliding slots, offering a modular and customizable approach to pill delivery. However, the complexity of fine adjustment, difficulties in maintaining dispensing accuracy, and challenges in integrating with the suction mechanism led to its discontinuation in favor of the vertical pickup scheme.

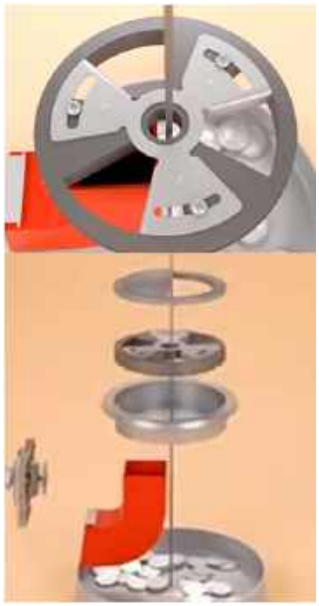


Figure 4: Alternative Design: Gashapon-Style Adjustable Dispensing Mechanism

Following the first CAD version, an intermediate design (Figure6) was developed with more complete motor mount structures and an enclosed cylindrical housing, intended to be fabricated via 3D printing. This version was the first to feature a fully enclosed body and internal mounts for motion modules. Nevertheless, the high cost and print volume limitations of 3D-printed enclosures prompted a redesign using laser-cut acrylic plates, leading to the subsequent box-shaped iterations.
The second finalized version (Figure 7) introduced precise geometries and preliminary placements for key mechanical elements. A DC motor was incorporated to drive the suction linkage; however, testing revealed insufficient torque output and motion instability,

compromising the reliability of pill extraction.
To address these limitations, the final design (Figure 8) adopted a dual-shaft stepper motor, enabling accurate and synchronized vertical motion. The linkage and suction arm were redesigned to reduce friction and improve mechanical stability. Additionally, a transparent acrylic enclosure was added to provide structural support, enhance safety, and facilitate user interaction through an accessible pill drawer. These enhancements significantly improved the system's precision, modularity, and robustness.

Figure 5: First Conceptual CAD

Figure 6: Intermediate Cylindrical CAD Design

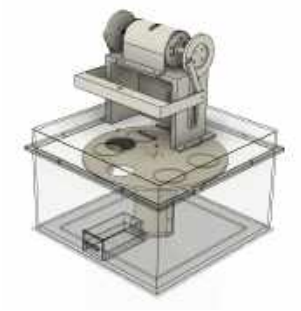


Figure 7: Intermediate CAD Version

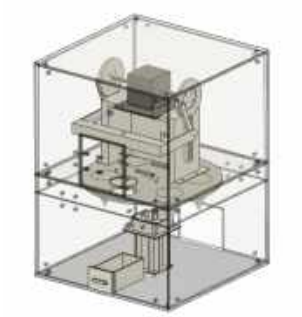


Figure 8: Final Updated CAD Model

In addition to mechanical improvements, the smart medicine box also incorporates a comprehensive electrical control system to enable automation, sensing, and user interaction. The system integrates stepper motor drivers, relay modules, an ESP32 controller, and peripheral modules to support a variety of functions:

- **Stepper Motor Control**: Two stepper motors are employed—one to lift and lower the suction tube via a vertical linkage, and the other to rotate the medicine tray to align compartments.
- **Pump Control**: A relay-based control circuit enables the vacuum pump to start and stop precisely, facilitating reliable pill pickup.
- **Pill Detection**: An infrared sensor is installed along the pill drop path to detect whether the suction process successfully delivered a pill.
- **Reminder Module**: A speaker and a recording module are incorporated to play preset audio reminders to prompt users to take their medication.
- **Remote Control via Mobile App**: The ESP32 microcontroller enables Wi-Fi-based communication with a smartphone application. This supports not only scheduling and real-time monitoring but also remote control of the electromagnetic locking mechanism used for refilling pills.

Figure 9 shows an overview of the mechanical assembly, while Figure 10 illustrates the wiring and layout of the integrated electronic control circuit.

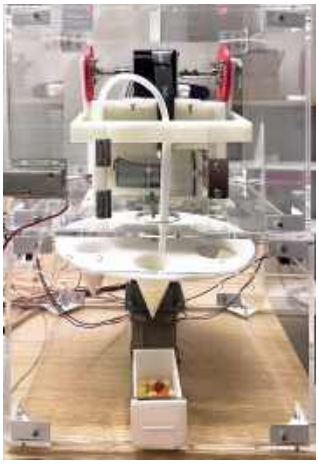


Figure 9: Mechanical Subsystem and Prototype Overview

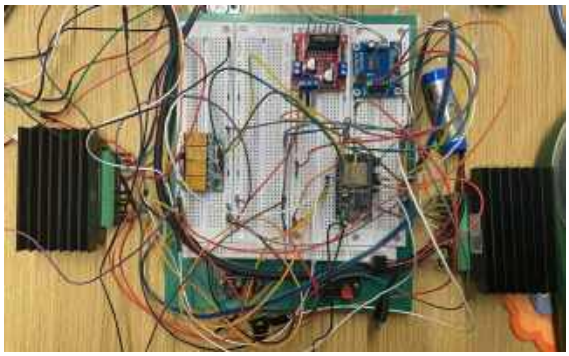


Figure 10: Electrical Subsystem Overview

The **software subsystem** is implemented as a mobile application that allows users to manage prescriptions and communicate with the device. Built-in Python using Kivy and KivyMD, the app supports photo-based prescription input and data transmission via Bluetooth:

Figure 11: Mobile App User Interface: Login and Working Screens

- **User Interface**: The app features a login interface and a working interface. After logging in, users can upload prescription photos and initiate interactions with the hardware.

- **Multimodal Prescription Processing**: Upon image upload, the app calls multiple AI models. The GLM-4V model extracts image text, while DeepSeek-R1 is used for logical parsing, formatting, and medication safety validation.
- **Data Formatting and Security Check**: Prescription content is standardized into structured arrays with drug names mapped to internal identifiers, and passed through a reasoning model to ensure safety and consistency.
- **Bluetooth Communication**: After processing, the app sends the extracted prescription data to the smart medicine box via Bluetooth, triggering the corresponding mechanical dispensing sequence.

Figure 11 presents the app layout and logic flow for prescription recognition and system communication.

2.1 APP Subsystem

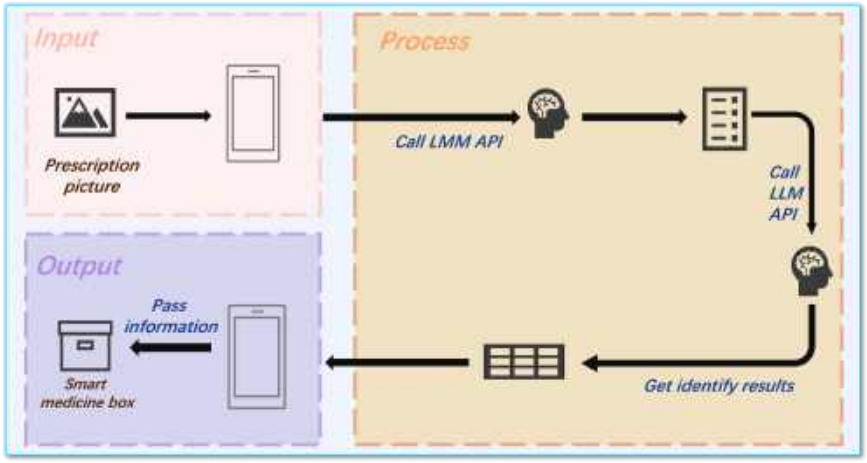


Figure 12: App Subsystem

2.1.1 Design Procedure

The App Subsystem is the module used by the smart medicine box to identify and process prescriptions. The production is mainly accomplished by using the Python language. Specifically, library functions such as kivy, kivymd, and Bluetooth are employed to implement the functions of analysis and transmission. The general subsystem is shown in figure 12, Users can select the prescription they want to take. After uploading the prescription photo to the app and connecting it to the medicine box, they can send the corresponding information to the medicine box.

2.1.2 Design Details

Layout The App mainly consists of two pages (shown in figure 13): one is the login interface, and the other is a feature-rich working interface. The construction of the page

Layout code

Login interface

Work interface

Figure 13: App Layout

is mainly composed of components such as Layout and MDLabel. On the login interface, after entering the username and password and clicking "Login", you can be redirected to the working interface to proceed to the next step. The logged-in user information is displayed at the top of the working page, and there are multiple buttons at the bottom. Click the camera button in the middle to select the prescription photo from the device. After clicking the think button, the large model will be called to process the prescription image. After clicking the link button, the user will be asked to connect to the Bluetooth device. After clicking the transfer button, the prescription information will be sent to the smart medicine box after the user finishes dispensing the medicine.

Process The App mainly processes images and extracts data by calling multiple large models. Once the user wants to perform prescription recognition, the function call_api will be called. The figure 14 show call_api code. Internally, the function call_api will first call the "glm-4v-flash" model to extract all the contents in the image, and then call deepseek-r1-distill-qwen-32b twice. Process the prescription into a fixed format, and finally use deepseek-r1-distill-qwen-32b to conduct a security analysis of the extracted content. We found that the glm-4v-flash model performed well in handling the task of image-to-text conversion, but often performed poorly in dealing with logical and text problems. Therefore, the deepseek-r1-distill-qwen-32b model was selected. This model has logical reasoning ability and can well complete the task of extracting text prescriptions.

Call Api function

Call LMM

Call LLM

Figure 14: Process perscription

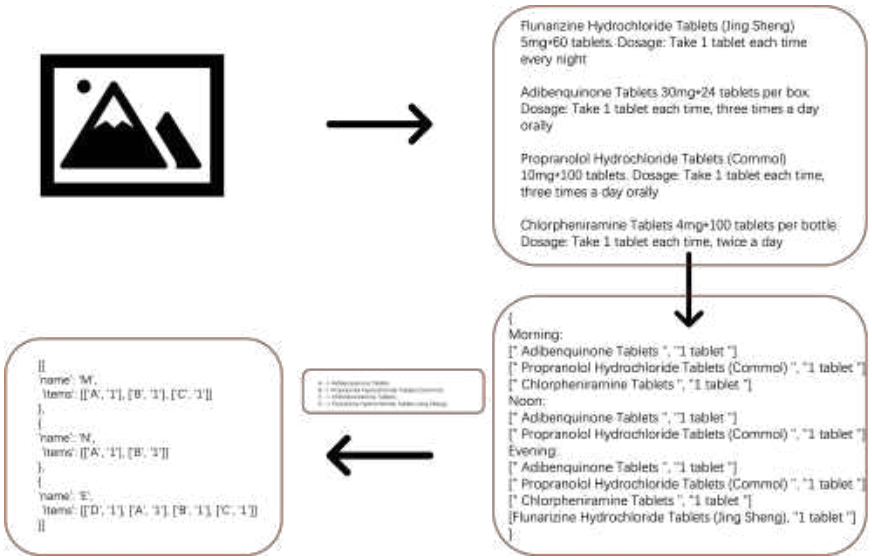


Figure 15: App transmit to smart medicine box

Transmit During the Bluetooth pairing and transmission section, to facilitate the processing of the medicine box, we will match the drug names with their numbers and convert the transmitted content into an array. For example, in figure 15, the output will be an array according to the prescription.

2.2 Medicine Storage Subsystem

2.2.1 Design Procedure

The Medicine Storage Subsystem was designed to ensure accurate pill storage, reliable positioning, and structural protection while maintaining compatibility with the system's electronic and software modules. The design process emphasized fabrication simplicity, mechanical reliability, and modular integration. Key mechanical components—including

the rotating pill tray, stepper motor mounting assembly, and acrylic protective enclosure—were developed iteratively to balance functional performance with manufacturability. Key electrical components-the stepper motor and its controller are then determined to match the torque needed by the mechanical system.

2.2.2 Design Details

Rotating Pill Tray The pill tray adopts a circular disk with six compartments (five storage slots and one dispensing slot) to facilitate indexed rotation for sequential dispensing. To optimize pill centering and suction efficiency, four different compartment base geometries were prototyped and evaluated: hemispherical (bowl-shaped), pyramidal, conical, and flat-bottomed cylindrical.
As shown in Figure 16, these geometries were fabricated using 3D printing and experimentally tested for suction pickup reliability. Among them, the conical base demonstrated the best overall performance—providing consistent pill centering, minimizing jamming, and significantly improving suction success rate. Therefore, the conical design was selected for final implementation in the tray. Figure 17 shows the CAD rendering of the finalized rotating pill tray, including the overall compartment layout and dimensions.

Figure 16: Shape Testing of Pill Tray Compartments

Figure 17: CAD of the Rotating Pill Tray

Stepper Motor Mounting and Flange Coupling To drive the tray rotation, a stepper motor mounting assembly was designed, consisting of a reinforced support frame and upper housing for motor fixation. The motor mount ensures rigid alignment with the pill tray's axis of rotation, with vertical ribs enhancing structural stability.
The tray connects to the stepper motor shaft through a flange coupling. Specifically, the tray is fastened to the flange using four countersunk screws. The flange itself is secured onto the motor shaft with set screws, providing accurate torque transmission and maintaining concentric alignment. This coupling design minimizes rotational backlash, thereby improving dispensing precision.
Figure 18 shows the CAD rendering of the stepper motor mounting assembly, showing the support structure and interface with the tray.

Figure 18: CAD of Stepper Motor Mounting Assembly

Figure 19 and Figure 20 are the real photos of the tray-flange connection highlighting the four-screw mounting, and the assembled tray-flange-motor system displaying overall integration, respectively.

Figure 19: Connection Between Tray and Flange

Figure 20: Flange Coupling to Stepper Motor Shaft

Stepper Motor and its controller To enable precise and repeatable rotation of the pill tray, we selected a **stepper motor** as the actuation mechanism. Stepper motors offer fine angular resolution and reliable 360-degree control, which is ideal for indexing circular trays. To determine the appropriate motor specification, we first **weighed the pill tray with the maximum pill load**, then calculated the required holding torque based on the system's mass distribution and rotation radius. After evaluation, we selected the **57-41 stepper motor**, which provides a holding torque of **0.64 Nm** and a working current of **2.4 A**. To drive this motor, we employed the **TB6600 stepper motor controller**, which supports a wide range of output current (**0.5 A to 4.0 A**) and is well-matched to our motor's requirements. The controller allows for micro stepping and stable current control, ensuring smooth and reliable operation of the tray mechanism. Figure 21 shows the two TB6600 motor controllers used in the system, each connected to a separate motor (one for pill tray rotation, and one for vertical suction linkage control).

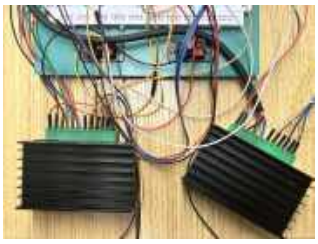


Figure 21: TB6600 Stepper Motor Controllers Connected to the Breadboard System

Acrylic Protective Enclosure with Hinged Cover The enclosure is made of laser-cut acrylic panels assembled into a rectangular box to protect internal components from dust and interference. Panels are joined using angle brackets at the corners, enhancing structural rigidity while simplifying assembly. A rectangular slot on the front panel accommodates a removable pill collection drawer, aligned precisely with the dropping outlet above to ensure accurate collection. The drawer can be smoothly inserted and removed during operation. An upper acrylic cover, attached with metal hinges, completes the enclosure and opens upward like a lid. This design facilitates access for refilling pills, inspecting components, and performing maintenance. All panels are fastened using M4 screws and hex nuts through pre-drilled holes, enabling easy disassembly without adhesives or complex fixtures. Figure 22 shows the CAD model of the acrylic protective enclosure, highlighting panel layout, mounting holes, hinge connection points, and access openings.

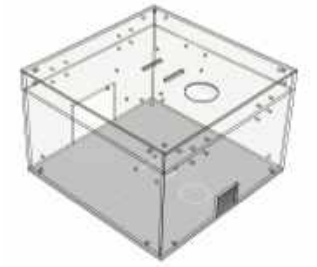


Figure 22: CAD model of the lower acrylic protective enclosure

Figure 23 and Figure 24 illustrate the fastening mechanisms used at different parts of the enclosure: metal brackets for structural frame joints and hinges for lid mobility.

Figure 23: Angle Bracket Fixing between Acrylic Panels

Figure 24: Hinge Connection between Lid and Enclosure

2.3 Medicine Grabbing and Dropping Subsystem

2.3.1 Design Procedure

The Medicine Grabbing and Dropping Subsystem is responsible for transferring individual pills from the storage tray to the collection drawer. It adopts a **suction-based mechanism** powered by an air pump, with vertical motion achieved via a **linkage-crank system** driven by a **dual-shaft stepper motor**. Early concepts such as mechanical grippers and vibration feeders were discarded due to complexity, potential pill damage, and limited space. The suction nozzle was selected for its compactness, reliability, and ease of control. To ensure precise vertical actuation, a stepper motor was chosen over a DC motor for its stable speed and open-loop accuracy. A **57-51 stepper motor** was selected based on torque calculations, paired with a **TB6600 driver** capable of supplying up to 3.5 A. An optical sensor was added for **feedback control**, confirming successful pill pickup before executing the drop, ensuring system accuracy and reliability.

2.3.2 Design Details

Overall Structure Overview The overall structure of the Medicine Grabbing and Dropping Subsystem is shown in Figure 25. The system consists of a dual-shaft stepper motor, rotating crank disks, linkages with sliding guides, a suction hose bracket, and an air pump mounted below the motor support. This mechanism achieves precise vertical motion for pill grabbing, reliable pill release, and seamless integration with the storage module's collection drawer.

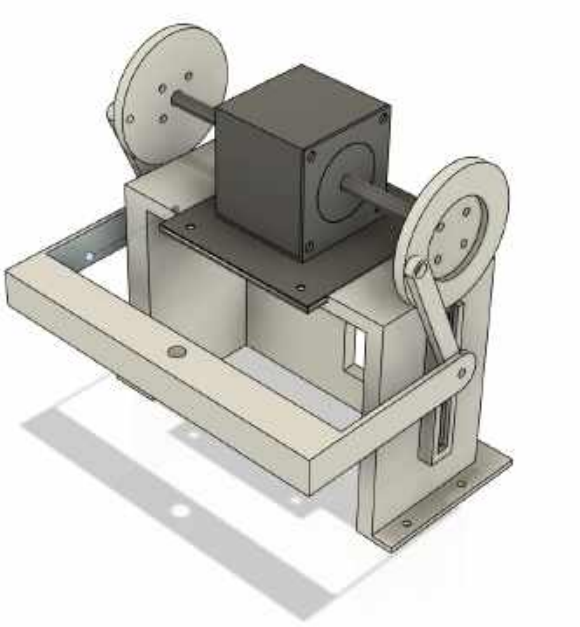


Figure 25: Overall CAD of Medicine Grabbing and Dropping Subsystem

Dual-Shaft Motor and Flange-Driven Rotation At the core of the mechanism is a dual-shaft stepper motor, mounted securely on top of the support structure. The motor is fastened to the frame through dedicated mounting holes, ensuring rigid installation and minimizing vibration during operation.
Each shaft end of the motor is connected to a rotating disk (crank wheel) via a flange coupling. The flanges are fixed to the motor shaft using set screws and are bolted to the crank disks through multiple screw holes. This rigid connection guarantees precise torque transmission, maintaining synchronized bilateral rotation without backlash, which is critical for the smooth vertical actuation of the linkages.
Figure 26 shows the mounting of the stepper motor on the support structure, as well as the flange coupling connection between the motor shaft and the crank disks.

Figure 26: Stepper Motor Mounting and Flange Coupling to Crank Disks

Linkage-Driven Vertical Actuation The rotating disks are connected to a pair of linkages, which convert rotational motion into vertical reciprocating movement. The details

could be found in Appendix C. The tail ends of these linkages feature cylindrical protrusions that slide within vertical guide slots on the sides of the motor support housing. This configuration constrains the motion path, ensuring smooth and controlled vertical actuation.

Linkage-to-Bracket Connection with Dowel Pin Initially, the linkages were connected to the suction hose support bracket using self-tapping screws. However, this method resulted in instability and wear during repetitive operation. To address this, the design was revised to use cylindrical dowel pins, inserted through precisely machined holes in both the linkage tails and the bracket. This connection ensures smooth rotation of the bracket with minimal play, significantly enhancing mechanical stability. Figure 27 illustrates the CAD model of the linkage-to-bracket dowel pin connection.

Figure 27: CAD of the Linkage-to-Bracket Dowel Pin Connection

H-Shaped Support Reinforcement To maintain horizontal alignment of the suction hose support bracket during vertical motion, an H-shaped reinforcement structure was added beneath the bracket. One end of the H-shaped support connects to the bracket, while the other slides within the guide slots. This reinforcement prevents tilting or angular deviation, ensuring the suction nozzle remains correctly oriented throughout its movement. Figure 28 shows the real-world implementation of the H-shaped support structure, demonstrating its effectiveness in maintaining bracket horizontality.

Figure 28: H-shaped Support Structure

Air Pump Suction Mechanism An air pump is mounted beneath the motor support frame, making efficient use of the hollow space. The pump generates negative pressure through a flexible hose connected to the nozzle bracket. When the nozzle reaches its lowest position above the designated pill compartment, suction is applied to grab a pill from the tray. Figure 29 shows the air pump mounting and hose connection interface.

Figure 29: Air Pump

Air Pump Control via Relay Module To enable precise control of the suction-based pill pickup process, a compact **4-channel relay module** is used to switch the **vacuum air pump** on and off as required. As shown in Figure 30, each relay on the board can be controlled independently by a digital output from the ESP32. In this system, only one relay is actively used to control the air pump, while the others remain available for future extensions such as lighting, alarms, or additional actuators.
Each relay (model HK4100F-DC5V-SHG) supports up to **250 V AC 10 A** or **30 V DC 10 A**, making it suitable for switching the air pump's DC load. The relay module is powered via the 5 V rail from the main supply, and the control logic is opto-isolated to ensure safe operation and protect the microcontroller from back-EMF or high current transients.
This setup enables the software to activate or deactivate the suction pump dynamically, for example:

- Turn **on** the pump just before the suction nozzle descends toward the pill tray.
- Turn **off** the pump once the pill is confirmed to be successfully dropped into the collection chamber.

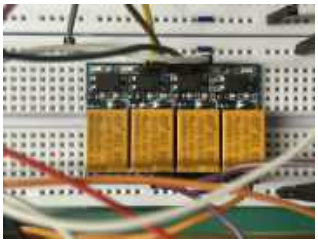


Figure 30: Relay Module Used for Air Pump Control

Fiber-Optic Sensor Feedback Control To ensure reliable pill pickup, an **infrared break-beam sensor** is integrated into the grabbing subsystem. As shown in Figure 31, a pair

of IR transmitter and receiver modules is mounted on opposite sides of the enclosure, aligned along the vertical path of the suction nozzle. When the nozzle ascends after attempting to pick up a pill, the sensor checks whether an object passes through the beam. To avoid false triggering by the nozzle itself, a suction head was specifically selected with a narrow profile that does not interrupt the infrared beam. This design ensures that only the presence of a pill will trigger detection. If the sensor confirms a successful pickup, the system proceeds to the dropping phase; otherwise, the suction action is repeated until confirmation is achieved, ensuring accuracy and reliability in the dispensing process.

Figure 31: Infrared Break-Beam Sensor Positioned Along the Nozzle's Vertical Path for Pill Presence Detection

Pill Dropping and Collection Drawer Interface Once a pill is successfully picked up, the storage tray rotates to align the suction nozzle above a designated dropping slot—an open cavity without a bottom. The air pump then shuts off, releasing the pill. The pill falls directly into the collection drawer, which is housed in the enclosure slot of the medicine storage subsystem. This design provides a clear, user-accessible retrieval point for the dispensed medication. Figure 32 presents the pill collection drawer and its interface with the enclosure opening.

Figure 32: Pill Collection Drawer

Upper Protective Enclosure and Electromagnetic Locking Door To safeguard the motion components and air pump located in the grabbing mechanism, an upper acrylic protective enclosure was added above the rotating disk and linkage system. This enclosure

prevents accidental contact with moving parts during operation while maintaining visibility for inspection and maintenance.
A hinged acrylic door was integrated into one side of the enclosure and designed to be controlled via an electromagnetic locking mechanism. The original intention was to use this door as a refill port for adding pills directly into the rotating tray compartments. However, due to the current positioning of the air pump and its suction hose, the available clearance was insufficient to accommodate the neck of a standard pill bottle.
As a result, the door currently serves only as a maintenance access point. For future iterations, repositioning the pump assembly or rerouting the suction hose could restore the intended refill functionality of this access door. Figures 33 and 34 show the CAD design and real-world implementation of the electromagnetic locking door.

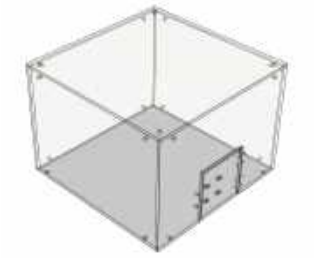


Figure 33: CAD Model of Upper Enclosure and Electromagnetic Locking Door

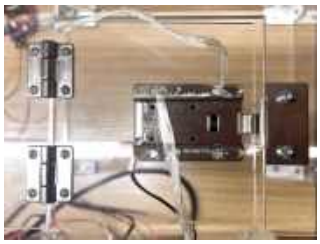


Figure 34: Real Implementation of the Electromagnetic Locking Door

Electromagnetic Lock Control via ESP32 To secure the refill port of the medicine box and prevent unauthorized access, an **electromagnetic locking mechanism** is installed on the side of the protective acrylic enclosure. This lock is designed to remain engaged (locked) by default and can be temporarily disengaged (unlocked) under controlled conditions.
The **ESP32 microcontroller** (shown in Figure 35) manages this lock through one of its GPIO pins, enabling remote unlocking when the user issues a command via the mobile app. This feature supports scenarios such as:

- Allowing caregivers or users to safely refill pills when authorized,
- Preventing unintended tampering with the tray during automatic dispensing,
- Ensuring access control is synchronized with medication management schedules.

The electromagnetic lock is powered via a **12 V supply**, and the control logic is implemented using a **relay or N-MOS transistor switch**, triggered by the ESP32. To unlock, the ESP32 outputs a HIGH signal to energize the lock coil momentarily; once power is cut, the lock returns to its locked state. This feature is tightly integrated with the app-based prescription interface. After the user uploads a prescription and the system processes the

pill types and quantities, the app provides an option to **"Open Lock for Refilling"**. Upon user confirmation, the ESP32 receives the Bluetooth signal and triggers the unlocking sequence.

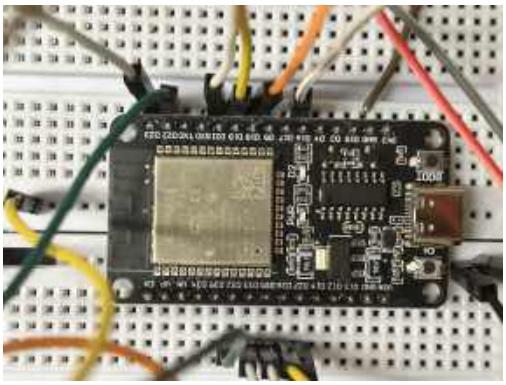


Figure 35: ESP32 Microcontroller Used for Central Control and Bluetooth Communication

2.4 Medicine Collection and Reminder Subsystem

2.4.1 Design Procedure

The collection and reminder module handles the final stage of dispensing by receiving pills and prompting the user to take their medication. Mechanically, this is achieved through a pill drawer located in the lower section of the acrylic enclosure. Pills dropped by the suction mechanism fall directly into the drawer, ensuring reliable delivery to an accessible area. Once the sensor confirms successful pickup, the storage tray rotates to align the nozzle with an open-bottom slot. The air pump is then turned off, allowing the pill to fall into the drawer by gravity. The reminder function is controlled by the ESP32. When the required pills are successfully delivered, the speaker module plays a pre-recorded voice message to alert the user. This provides a clear, real-time prompt for medication retrieval.

2.4.2 Design Details

Pill Drawer The drawer is mounted behind a rectangular slot on the front panel and fits snugly into the opening for smooth sliding. Its shallow design reduces pill bouncing, and it is mechanically separated from the suction system to maintain modularity. This design allows users to retrieve pills easily without disturbing internal components. Figure 32 shows the drawer design and its integration with the enclosure.

Audio Reminder Module To ensure users are promptly notified after successful medication dispensing, the system includes a simple yet effective **audio reminder module** composed of an **ISD1820 voice playback board** and an **8 Ohm 0.5 W speaker**, as shown

in Figure 36. The ISD1820 module supports short audio recording and playback, allowing the system to issue a preset voice prompt after dispensing is complete. In this prototype, a voice message is pre-recorded: "Dear users, it is time to take medicine." This message is automatically played **three times** each time a pill is successfully dropped into the collection drawer. The module is powered via the 5 V supply rail and triggered by the ESP32 through a digital GPIO pin. Upon receiving a "dispense complete" signal, the ESP32 sends a short HIGH pulse to the **PLAYE** pin on the module, activating the playback sequence without requiring mechanical button presses. This audio-based reminder enhances system usability, especially for elderly users who may not always check visual indicators or mobile app notifications.

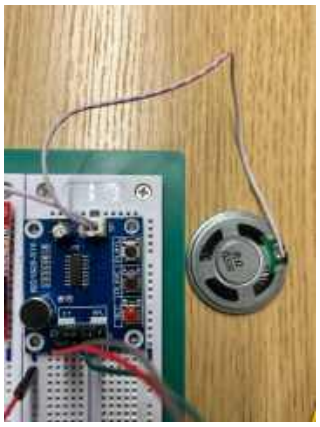


Figure 36: Reminder Module: ISD1820 Voice Playback Board and Speaker

2.5 System Integration and Central Control Code

To coordinate the behavior of all hardware modules—stepper motors, vacuum pump, infrared sensor, electromagnetic lock, and speaker—a unified control logic was implemented on the ESP32 using Arduino IDE. The code integrates: - Bluetooth communication for receiving prescription commands from the app; - Drug identification via keyword matching and mapped storage index; - Sequential actuation of the rotating tray and suction linkage to retrieve pills; - Pill drop verification using an infrared sensor; - Final auditory reminder to prompt users to take the medication. The complete ESP32 control program, which integrates all hardware subsystems including motors, sensors, pump, and Bluetooth communication, is provided in Appendix D.

3 Verification

We summarized the key requirements and whether each was met in the Requirement & Verification Table (see Appendix B).
The APP block successfully met all functional requirements, as shown in Figure 15. Prescription images were captured and transmitted via Bluetooth to the ESP32. The AI model was able to correctly interpret medication names and dosages with a classification accuracy exceeding 80%.

For the Medicine Storage Block, we verified the functionality of the rotating pill tray and the electronic lock mechanism. The stepper motor accurately rotated the tray to the intended compartment within a 5° error margin across multiple cycles. The electronic lock was tested over 50 trials and demonstrated a 90% success rate in securely opening and locking upon receiving commands from the controller. These results indicate that the storage system offers both functional accuracy and secure medication access control.

The Grabbing and Dropping Block was subjected to extensive testing, particularly in suction performance and pill detection. We tested multiple suction head geometries and found that a 4 mm diameter Silicone nozzle achieved the highest pickup success rate across various pill sizes and shapes. The result is shown in Figure 37.

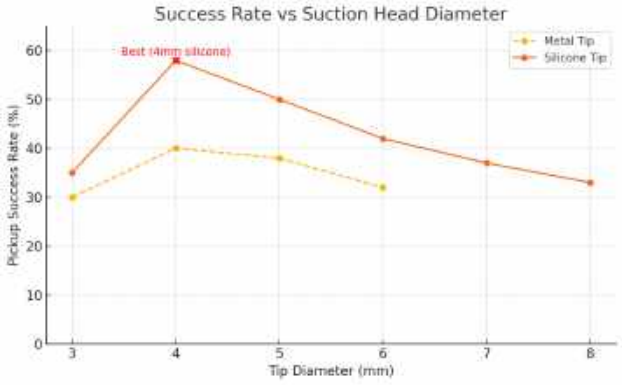


Figure 37: Success Rate Vs. Suction Head

For pills weighing over 0.2 g, the system achieved a grab success rate of 59% in 20 trials.

The original design included a pressure sensor to confirm pickup success; however, empirical tests revealed that it failed to detect zero-pressure states during large pill pickups and exhibited negligible variation for small pills. Therefore, the pressure transducer was removed and replaced with an improved infrared sensor with a wider field and longer range. This sensor achieved consistently high accuracy but struggled with transparent or very small pills. Figure38 shows the success rate of the infrared sensor in detecting different pills at the detection position.

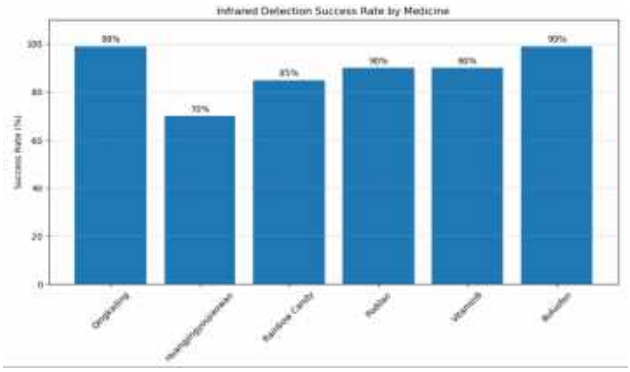


Figure 38: Detection Success Rate Vs. Medicines

To address this, we upgraded to a model with a broader detection beam, which improved detection reliability.
For the mechanical part, we estimated the required torque and selected a suitable dual-shaft stepper motor.

$$T_{need} = F_{load} \cdot R_{disk} \cdot \frac{1}{\sin\theta} = 0.27Nm < T_{motor} = 1.3Nm$$

Meanwhile, we also used Fusion 360 to simulate the forces acting on the entire linkage mechanism during its actual motion, which is shown in AppendixF.
The simulation results confirmed that our design is sufficiently reasonable and that the selected motor provides enough torque to meet our requirements.
The Medicine Collection and Reminder Block was verified through drawer retrieval and alert timing tests. Pills dropped into the drawer were accessible without obstruction in all tests. The buzzer reminder system was evaluated across 20 cycles and activated successfully in 20 out of 20 cases, demonstrating full reliability.

4 Costs

Our fixed development costs are estimated to be 100 ¥/h, 10 hours/week for four people, and the specific schedule is shown in AppendixE. We consider the approximate cost of our final design this semester (16 weeks), neglecting the central server, mesh network optimization, and partnerships with NGOs:

$$100/h \cdot people \times 10h/week \times 16weeks \times 4people = 64000$$

Our parts and manufacturing prototype costs are estimated to be 929.82 ¥ each, as shown in Appendix A:

Since we will only build one smart medicine box, this yields a total development cost of 64929.82 ¥.

5 Conclusion

5.1 Summary and Future Work

This project successfully delivered a functional smart medicine box system integrating mobile prescription management, mechanical pill dispensing, and user alert features. The system comprises four key modules—APP interface, medicine storage, grabbing and dropping, and collection/reminder—which were designed to operate in coordination for a smooth, automated user experience. Throughout the design process, we overcame several practical challenges such as unstable detection methods, inconsistent suction, and mechanical misalignment. Iterative testing and redesign led to substantial performance improvements across all blocks. While some limitations remain—such as detection difficulties with small transparent pills or occasional unlocking errors—the overall system meets its design intent and demonstrates promising real-world applicability.

Future work on the smart medicine box includes mechanically optimizing the grabbing system by redesigning the current linkage into a more stable four-bar mechanism for smoother motion. Enhancing the suction pump power is also planned to improve pickup reliability across a broader range of pill types. On the software side, future development will include releasing an iOS version of the app and implementing cloud-based analytics for long-term medication tracking and user insights.

5.2 Ethics and Safety

Our design adheres to the IEEE Code of Ethics [4] by ensuring safe, responsible, and transparent system behavior. Sensitive medical information processed by the app is transmitted securely and used only for functional execution without long-term storage. We explicitly ensure that the AI prescription analysis tool supports, rather than replaces, human judgment. Additionally, system feedback mechanisms are designed to prevent harm from failed pill pickup or incorrect dispensing, reflecting our commitment to user health and safety.

5.3 Broader Impact

This smart medicine box addresses urgent global needs in elderly healthcare, particularly in supporting medication adherence. Its low-cost, modular architecture promotes affordability and accessibility in both individual households and under-resourced clinical settings. By empowering users with limited mobility or cognitive decline to manage their medication schedules independently, the system enhances quality of life and reduces dependency on external care. Furthermore, its sustainable design—featuring reusable components and reduced waste—aligns with broader environmental goals, making it a socially and globally conscious engineering solution.

References

[1] C. M. Hales, J. Servais, C. B. Martin, and D. E. Kohen, "Prescription drug use among adults aged 40-79 in the united states and canada," *NCHS data brief*, vol. 347, pp. 1–8, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:201631211.

[2] V. Bindu Sree, K. S. Indrani, and G. Mary Swarna Latha, "Smart medicine pill box reminder with voice and display for emergency patients," *Materials Today: Proceedings*, vol. 33, pp. 4876–4879, 2020. DOI: 10.1016/j.matpr.2020.08.400. [Online]. Available: https://doi.org/10.1016/j.matpr.2020.08.400.

[3] Z. Nasir, A. Asif, M. Nawaz, and M. Ali, "Design of a smart medical box for automatic pill dispensing and health monitoring," *Engineering Proceedings*, vol. 32, no. 1, p. 7, 2023. DOI: 10.3390/engproc2023032007. [Online]. Available: https://doi.org/10.3390/engproc2023032007.

[4] IEEE, *IEEE Code of Ethics*, Accessed: 2025-04-18, 2020. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html.

Appendix A Bill of Materials

Item	Quantity	Cost
Rigid Flange Coupling	4	15.31
Right Angle Bracket	20	9.2
Medium Damping Hinge	4	25.16
Brass Reducer Barb Tube	3	5.04
Acrylic Bottom Shell	10	75
3D Printing	8	368
Extra Small Damping Hinge	2	6.79
Screw and Nut Set	1	10.31
Acrylic Top Shell	5	60
Velcro Tape	1	4.01
PP Hose Connector	1	6.5
Rigid Reducer Coupling	2	9.29
Dowel Pin	2	3.12
Suction Cup	1	1.2
Acrylic Adhesive	1	23.84
24V Air Pump	1	44.53
Latch Strip	1	11.42
Single-axis Stepper Motor	1	40
Stepper Motor Controller	1	30
ESP32	1	24.8
Water Drop Sensor	1	29.6
Electromagnetic Lock	1	23.9
Dual-axis Stepper Motor	1	103

Appendix B Requirement and Verification Table

Requirement	Verification
The app can take photos of the prescription.	Given a prescription, the user can take the photo and see the picture on the App Interface.
For the prescription photos that have been taken, the App can realize the function of calling a large model for identification and analysis.	For the prescription photos that have been taken, the user can see the identification and analysis results on the interface.
The app can connect to the esp32 and pass the output of the large model to the esp32 via Bluetooth.	Esp32 When connected to the app, can get the information from the app.

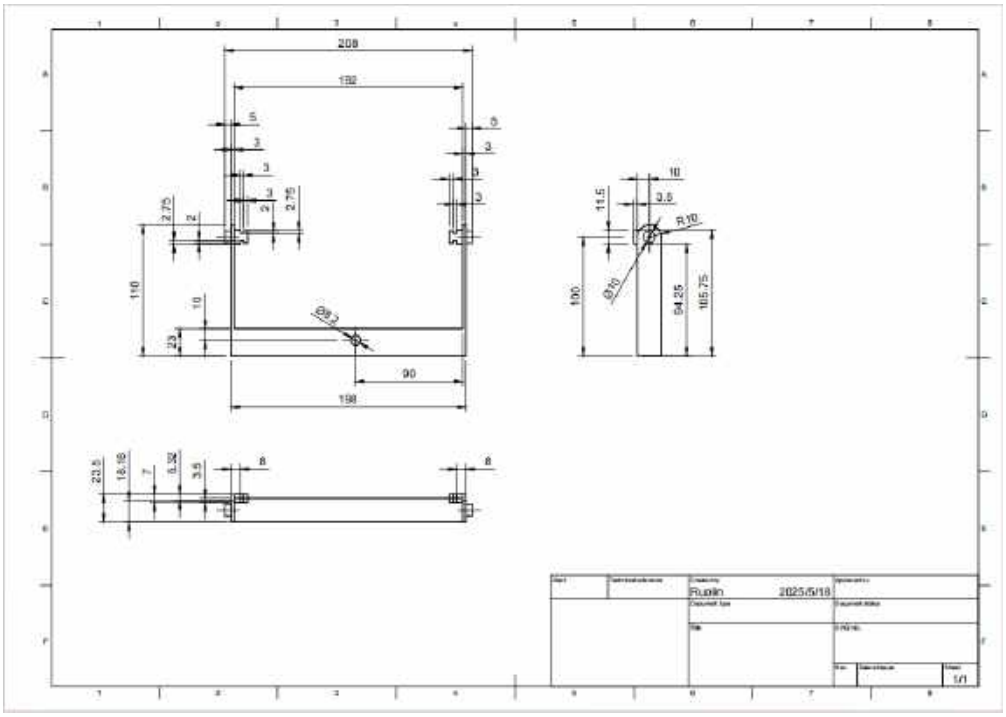
Requirement	Verification
A 3.3 V $\pm$ 0.1 V power is provided to the STM32. Noise should be $\leq$ 50 mV.	Use the oscilloscope to test the STM32 power voltage from the power management subsystem. The result should be a voltage of 3.3 V± 0.1 V with noise should be $\leq$ 50 mV.
A 12 V $\pm$ 0.5% power is provided to the step model and double-ended motor. The maximum current should be 10 A.	Use the oscilloscope to test the step model and double-ended motor power voltage from the power management subsystem. The result should be a voltage of 12 V $\pm$ 0.5% with maximum current 10 A.
A 24 V $\pm$ 3% power is provided to the suction engine with a maximum current 8 A.	Use the oscilloscope to test the suction engine power voltage from the power management subsystem. The result should be a voltage of 24 V $\pm$ 3% with maximum current 8 A.

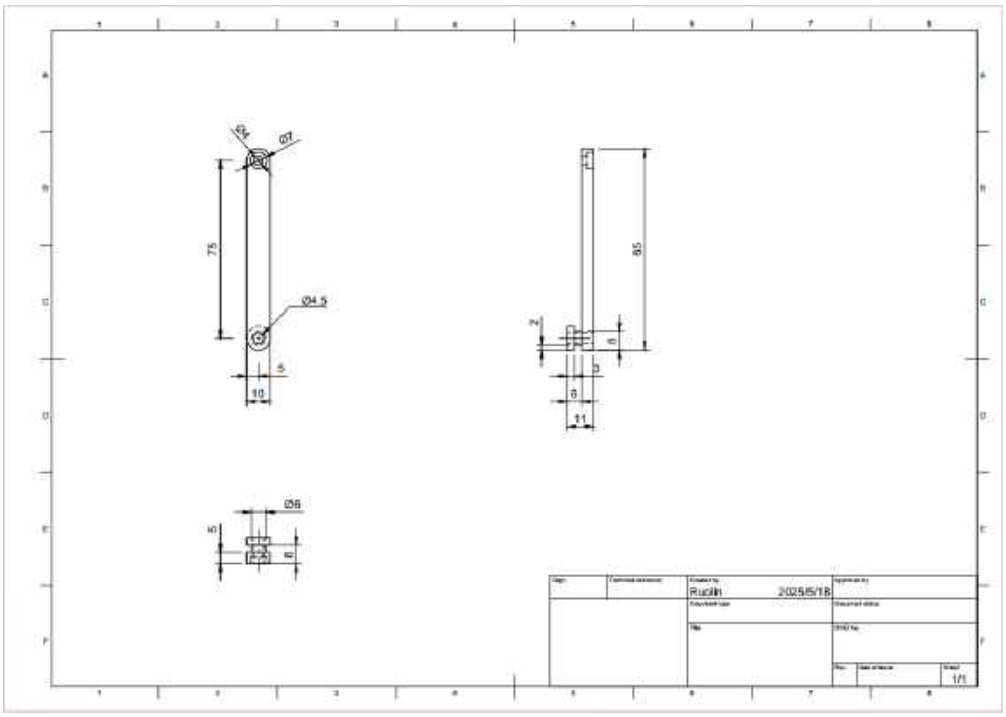
Requirement	Verification
The bluetooth of the STM32 can successfully connect to the APP with a connection time ≤ 5 s and maximum working distance ≥ 10 m. The packet loss rate should be $\leq 0.1\%$.	Connect the APP to our smart medicine box and measure the connection time. Repeat for some time and the average time should be ≤ 5 s. Send 100 instructions to medicine box within a distance of 10 m. Calculate the packet loss rate and it should be $\leq 0.1\%$.
The WIFI module should have the ability to connect to the WIFI signal when the RSSI ≥ -70 dBm.	Try to connect the STM32 to WIFI signal with -90 dBm $\leq$ RSSI ≤ -60 dBm and test its success rate.
The STM32 should get a time $\leq \pm 1$sec within the local time so that we can remind users to take pills at an accurate time.	Connect the STM32 to the WIFI and then calibrate the module time by NTP. Measure the difference between the module time and standard local time. The difference should be within ± 1sec.

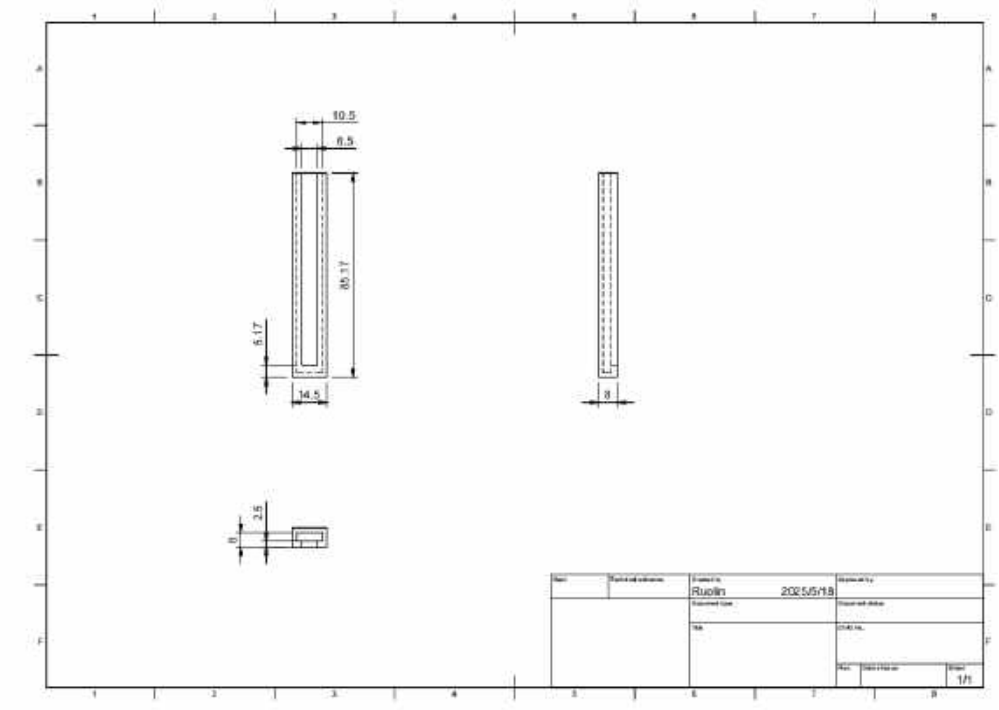
Requirement	Verification
By editing the parameters in the STM32, the pressure transducer should successfully report the suction action at an accuracy greater than 80%	Open the suction pump and connect the pressure transducer to the air-out. Connect the output of the pressure transducer to the STM32, and then the output of STM32 to a LED test circuit. Try to suck one pill onto the straw. The LED should be light on with a probability of 80% .
After program compiling, the Infrared sensor should have the ability to report the tablet drop at an accuracy greater than 80%	After installation, try to drop pills into the collection compartment at the same height as the straw does. Connect the STM32 processed version of the sensor output to the oscillator, then observe and count its success rate.

Requirement	**Verification**
Rotate the storage in specific angle.	Use STM32 to specify a rotation Angle between 0 and 360, the system can be rotated to and error in 5 degrees.
The suction mechanism must successfully grab pills weighing ≥ 0.2 g with a success rate of at least 60% across 20 trials.	Conduct 20 consecutive trials using pills (≥ 0.2 g each). Log the number of successful grabs. A successful grab means the pill is fully lifted and transferred to the output compartment. Success rate = (successful trials ÷ total trials) × 100%.
The system must be able to dispense exactly one pill per activation cycle, with an accuracy of at least 60%.	Trigger 20 grab cycles, each intended to dispense a single pill. Use an optical or weight sensor to count the number of pills actually dispensed each time. The system should dispense exactly one pill in at least 12 out of 20 trials. Record results in a table.

Appendix C Design Drawings






10.5
6.5
65.17
5.17
14.5
8
2.5
Ruolin
2025/5/19
1/1

Appendix D ESP32 Code

We write the ESP32 control code for the medicine box using Arduino IDF.

```
        \caption{Arduino Code for ESP32}
        \label{code:ESP32}
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>

#define SERVICE_UUID        "12345678-1234-1234-1234-1234567890ab"
#define CHARACTERISTIC_UUID "87654321-4321-4321-4321-abcdefabcdef"

#define Step_Motor_0_ENA 18
#define Step_Motor_0_DIR 19
#define Step_Motor_0_PWM 21
#define Step_Motor_1_DIR 26
#define Step_Motor_1_PWM 25

#define Delay_Time_0 10*1000
#define Delay_Time_1 3*1000
#define Step_Motor_0_Resolution 4
#define Step_Motor_1_Resolution 16

#define Step_1 710
#define Step_2 890
#define Step_3 976
#define Step_4 220
#define Step_5 404

#define Lock_Control 32
#define Pump_Control 33
#define Sensor_Out 16
#define Led_Pin 2
#define Speaker_Play_E 22
#define Switch_In 23


bool device_connected = false;
BLECharacteristic *pCharacteristic;

String ble_input;
String my_instruction;
```

```
bool need_close_lock = false;
bool need_update_rx = false;

//Author:MoreThink Dissertation Guidance Store
//created by Dareen@ MoreThink

// callback function, to process the connect and disconnect
class MyServerCallbacks: public BLEServerCallbacks {
  void onConnect(BLEServer* pServer) {
    device_connected = true;
    Serial.println("BLE device connected.");
  }

  void onDisconnect(BLEServer* pServer) {
    device_connected = false;
    Serial.println("BLE device disconnected.");
  }
};

struct KeyValue {
  String key;
  int value;
};

// Self-designed data structure
KeyValue my_pilldict[] = {
  {"square", 5},
  {"capsule", 4},
  {"green",3},
  {"purple", 2},
  {"pilule", 1}
};

struct StepMotorDict{
  int step_motor_num;
  KeyValue control_info[2];
};

StepMotorDict my_stepmotordict[] = {
  {0, {{"dir", Step_Motor_0_DIR}, {"pwm", Step_Motor_0_PWM}}},
  {1, {{"dir", Step_Motor_1_DIR},{"pwm", Step_Motor_1_PWM}}},
  {2, {{"dir", Step_Motor_1_DIR},{"pwm", Step_Motor_1_PWM}}}
};
```

```
void setup() {
  Serial.begin(115200);
  Serial.print("begin!\n");
  pinMode(Switch_In, INPUT);
  pinMode(Step_Motor_0_ENA, OUTPUT);
  pinMode(Step_Motor_0_DIR, OUTPUT);
  pinMode(Step_Motor_0_PWM, OUTPUT);
  pinMode(Step_Motor_1_DIR, OUTPUT);
  pinMode(Step_Motor_1_PWM, OUTPUT);
  digitalWrite(Step_Motor_0_ENA, LOW);
  digitalWrite(Step_Motor_0_DIR, LOW);
  digitalWrite(Step_Motor_1_DIR, LOW);

  pinMode(Lock_Control, OUTPUT);
  digitalWrite(Lock_Control, LOW);
  pinMode(Pump_Control, OUTPUT);
  digitalWrite(Pump_Control, LOW);
  pinMode(Sensor_Out, INPUT_PULLUP);
  pinMode(Led_Pin, OUTPUT);
  digitalWrite(Led_Pin, HIGH);
  pinMode(Speaker_Play_E, OUTPUT);
  digitalWrite(Speaker_Play_E, LOW);
  pinMode(Switch_In, INPUT);
  delay(2000);

/*Blue Tooth Setup*/
  BLEDevice::init("ESP32");
  BLEServer *pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());
  BLEService *pService = pServer->createService(SERVICE_UUID);

  pCharacteristic = pService->createCharacteristic(
  CHARACTERISTIC_UUID,
  BLECharacteristic::PROPERTY_READ |
  BLECharacteristic::PROPERTY_WRITE);

  pCharacteristic->setValue("Test");

  pService->start();

  BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->addServiceUUID(SERVICE_UUID);
  pAdvertising->setScanResponse(true);
  BLEDevice::startAdvertising();
  Serial.println("BLE Peripheral set up. Waiting for connections...");
```

```
}
void loop() {
  // put your main code here, to run repeatedly:
  if (device_connected){
    ble_input = pCharacteristic->getValue().c_str();
    // Serial.println(ble_input);
    if(ble_input != my_instruction){
      my_instruction = ble_input;
      if(my_instruction == "open"){
        need_open_lock = true;
        Serial.println("need_open_lock");
      }
      else if(my_instruction == "close"){
        need_close_lock = true;
        Serial.println("need_close_lock");
      }
      else if(my_instruction.startsWith("[")){
        need_close_lock = true;
        need_update_rx = true;
        Serial.println("need_update_rx");
      }
      else {
        Serial.print("BLE receive:");
        Serial.println(my_instruction);
      }
    }
  }


//Author:MoreThink Dissertation Guidance Store
//created by Dareen@ MoreThink

  if (need_open_lock){
    startInputMode();
    need_open_lock = false;
  }
  if (need_close_lock){
    endInputMode();
    need_close_lock = false;
  }
  if (need_update_rx){

    need_update_rx = false;
  }
```

```
  if(digitalRead(Switch_In) == HIGH){
    executeAll();
  }
}



/*Input Pills Mode Control Function Group*/
/*Input Pills Mode Control Function Group*/
void startInputMode(){
  digitalWrite(Lock_Control, HIGH);
  digitalWrite(Step_Motor_0_ENA, HIGH);
}

void endInputMode(){
  digitalWrite(Lock_Control, LOW);
  digitalWrite(Step_Motor_0_ENA, LOW);
}


/*Self Define Struct Handling Function*/
/*Self Define Struct Handling Function*/
int getPosition(String key) {
  for (int i = 0; i < sizeof(my_pilldict)/sizeof(my_pilldict[0]); i++) {
    if (my_pilldict[i].key == key) {
      Serial.print(my_pilldict[i].value);
      return my_pilldict[i].value;
    }
  }
  return -1;
}

int getPin(KeyValue* info_p, String key) {
  for (int i = 0; i < 2; i++) {
    if ((*info_p).key == key) {
      return (*info_p).value;
    }
    info_p++;
  }
  return -1;
}

KeyValue* getControlInfo(int step_motor_num){
  KeyValue* p = nullptr;
```

```
  //Serial.print("getinfobegin\n");
  for(int i = 0; i < sizeof(my_stepmotordict)/sizeof(my_stepmotordict[0]);
    if (my_stepmotordict[i].step_motor_num == step_motor_num){
      p = my_stepmotordict[i].control_info;
      //Serial.print("getinfodone\n");
      return p;
    }
  }
  return p;
}


/*Step Motor Control Helper Function Group*/
/*Step Motor Control Helper Function Group*/
void rotateNStep(int motor_num, uint8_t dir, int step, int delay_time){
  KeyValue* info_p = getControlInfo(motor_num);
  int dir_pin = getPin(info_p, "dir");
  //Serial.print("dir_pin=");
  //Serial.print(dir_pin);
  int pwm_pin = getPin(info_p, "pwm");
  //Serial.print("pwm_pin=");
  //Serial.print(pwm_pin);
  digitalWrite(dir_pin, dir);
  for(int i = 0; i < step; i++){
    digitalWrite(pwm_pin, HIGH);
    delayMicroseconds(delay_time);
    digitalWrite(pwm_pin, LOW);
    delayMicroseconds(delay_time);
  }
}

void rotateNDegree(int motor_num, uint8_t dir, int degree, int delay_time,
  int step_needed = degree*resolution*5/9;
  int delay_time_needed = delay_time/resolution;
  //Serial.print(delay_time_needed);
  rotateNStep(motor_num, dir, step_needed, delay_time_needed);
}


/*Step Motor Control Function Group*/
/*Step Motor Control Function Group*/
void rotateToBox(int box_num){
  int degree = box_num*60;
  rotateNDegree(0, LOW, degree, Delay_Time_0, Step_Motor_0_Resolution);
}
```

```
void rotateBackFromBox(int box_num){
  int degree = box_num*60;
  rotateNDegree(0, HIGH, degree, Delay_Time_0, Step_Motor_0_Resolution);
}



/*Function Realization*/ /*Function Realization*/ /*Function Realization*/
/*Function Realization*/ /*Function Realization*/ /*Function Realization*/
/*Function Realization*/ /*Function Realization*/ /*Function Realization*/
void fetchPill(String med_type){
  int box_num = getPosition(med_type);
  rotateToBox(box_num);
  delay(500);
  int success = LOW;
  while(success == LOW){
    rotateNStep(1, LOW, Step_1, Delay_Time_1);
    digitalWrite(Pump_Control, HIGH);
    delay(1000);
    rotateNStep(1, LOW, Step_2, Delay_Time_1);
    //do not delete this pause
    //need be shorter
    delay(2000);
    rotateNStep(1, LOW, Step_3, Delay_Time_1);
    delay(500);
    success = Validation();
    //delay(200);
    rotateNStep(1, LOW, Step_5, Delay_Time_1);
    //do not delete this pause
    //need be much more shorter
    delay(2000);
    // rotateNStep(1, HIGH, Step_5, Delay_Time_1);
  }

  //Author:MoreThink Dissertation Guidance Store
  //created by Dareen@ MoreThink
  rotateBackFromBox(box_num);
  //do not delete this pause
  delay(500);
  //do not delete this pause
  digitalWrite(Pump_Control, LOW);
  delay(4000);
}
```

```
int Validation(){
  int sensor_out;
  int success = 0;
  for(int i = 0; i < Step_4; i = i+1){
    sensor_out = digitalRead(Sensor_Out);
    digitalWrite(Led_Pin, sensor_out);
    //Serial.print(sensor_out);
    if (sensor_out == LOW){
      success = 1;
    }
    rotateNStep(1, LOW, 1, Delay_Time_1);
  }
  return success;
}

void executeAll(){
  fetchPill("capsule");
  fetchPill("purple");
  fetchPill("green");
  fetchPill("pilule");
  // fetchPill("pilule");
  fetchPill("square");
  for(int i = 0; i < 3; i++){
    digitalWrite(Speaker_Play_E, HIGH);
    delay(7500);
    digitalWrite(Speaker_Play_E, LOW);
    delay(200);
  }
}
//Kind Remider:Please give a 5star evaluation if you are satisfied for our
//Kind Remider:Please give a 5star evaluation if you are satisfied for our
//Kind Remider:Please give a 5star evaluation if you are satisfied for our

/*Subsystem Test Function Group*/
/*Subsystem Test Function Group*/
void sensorRead(){
  int sensor_out;
  while(1){
    sensor_out = digitalRead(Sensor_Out);
    digitalWrite(Led_Pin, sensor_out);
    // print the results to the Serial Monitor:
    // Serial.print("sensor = ");
    // Serial.print(sensor_out);
    // Serial.print("\n");
  }
```

```
}

void suctionTest(){
  digitalWrite(Pump_Control, HIGH);
  int sensor_out;
  while (100>1){
    sensor_out = digitalRead(Sensor_Out);
    //Serial.print(sensor_out);
    //Serial.print("\n");
    if(sensor_out == HIGH){
      digitalWrite(Pump_Control, LOW);
      break;
    }
  }
}
```

Appendix E Schedule

Week	Ruolin&Wentao	Yutong	Zhiyi
3/17	Brainstorming and design iteration	Design the circuit	Design the workflow of the APP and Select the API to use.
3/24	Complete the design details	Design the circuit and choose appropriate motor	Design the Interface of the APP Login Interface and Work Interface.
3/31	Begine 3D printing and acrylic laser	Complete and test the control code for the step engine to implement angle controlment.	Select the code language to make the App. The Python is the easiest and the fastest way.
4/7	Assemble parts of them and test	Modify the code for the sensor and pump.	Begin to add some buttons and some fonts for Interface.
4/14	Revise the design based on updated Parts. Choose the proper sensor.	Complete the motion control for the dual-axis motor. Realize the stop and motion part of the suction mechanism.	Complete the software front-end design and realize the page jump function.
4/21	Continue 3D printing for delicate parts. Make the shell with acrylic.	Complete the rotation control mechanism. Make sure a specific rotation is based on time and prescription.	App connection API to perform picture analysis and prescription safety analysis.
4/28	Assemble and test. Help with the control section.	Integrate the power transformer circuit.	Realize information transmission between the app and esp32.
5/5	Improve based on tests.	Integrate and refine the prototype.	Improve and modify the App according to the test results.
5/12	Prepare for Mock Demo	Prepare for Mock Demo	Prepare for Mock Demo
5/19	Prepare for Final Demo and Video	Prepare for the Final Demo and Video	Prepare for the Final Demo and Video
5/26	Prepare for Final Presentation and Report	Prepare for Final Presentation and Report	Prepare for Final Presentation and Report

Appendix F Simulation Result

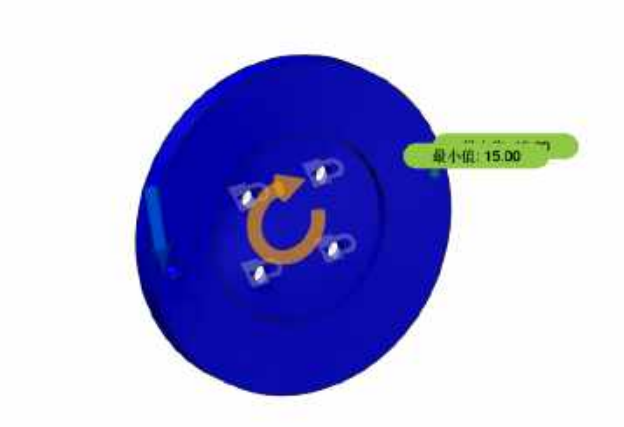


Figure 39: Simulation Results