Class: Exploring Engineering-ESC-100-05

Term: Fall 2024

Instructor: Professor Mafi

DM4c

Table of Contents

1. Introduction

For our design project, we were tasked with proposing an engineering solution to address a human need. Given the vast range of needs that engineering can help solve, we decided to focus on one of the most vulnerable groups in society. We began by listing vulnerable populations—such as the elderly, visually impaired, deaf individuals, and children—and, after careful consideration, chose to address the challenge faced by elderly and blind people in managing their medications. Our design objective is to create an accessible, user-friendly pill dispenser that stores and dispenses medication while notifying users through both sound alerts and app notifications.

2. Design Requirements

Our design criteria are as follows:

- Accessibility
- User-friendliness
- Reliability
- Cost-efficiency

We selected accessibility as one of our primary design requirements because we believe it is crucial for our target audience. Our goal is to simplify the device's functionality as much as possible by using just two buttons—one to start the operation and another to mute the speaker sounds—and a monitor to display the remaining time, making it easy for users to track when it is time to take their pills.

As an extension of accessibility, user-friendliness was also chosen as a key requirement. We believe that our device should be simple and intuitive to ensure users can comfortably interact with our design. Additionally, refilling and cleaning the pill dispenser must be

straightforward and hassle-free.

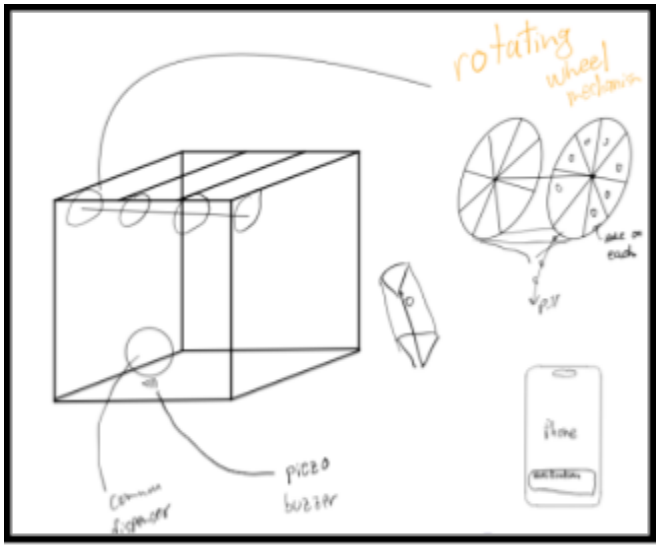
The third design requirement is reliability. Since our primary target audience consists of elderly individuals and people with visual impairments, we feel it is our responsibility to ensure that our invention is durable and functions properly.

Finally, cost-efficiency is an essential design consideration. Making our product accessible to all users includes overcoming economic barriers that may prevent individuals from living more comfortably. As part of our commitment to enhancing the lives of elderly and visually impaired individuals, we are dedicated to making our design both affordable and accessible by carefully choosing materials that are durable and reliable, yet cost-effective.

3. Conceptual Design

a. Designs (Including Alternative Design Combinations)

i. Design 1

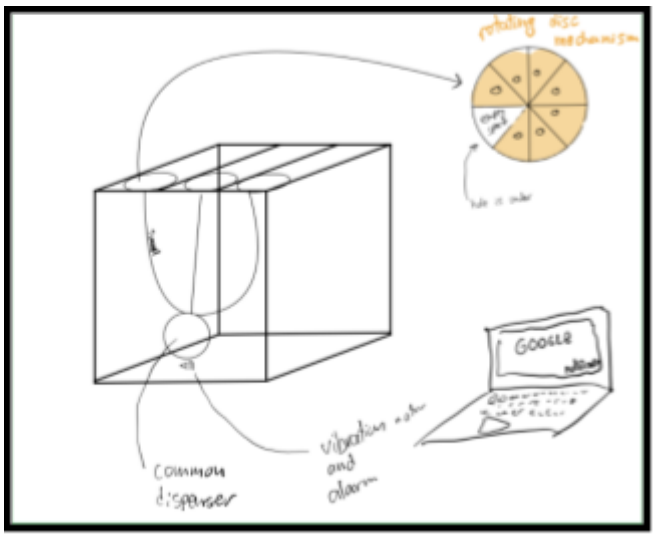


Design 1

The initial design, developed based on our concept matrix, includes a durable box, two piezo buzzers, a rotating barrel system, and an accompanying iOS app. The box is structured to house all the key components, such as the Redboard, the pill barrels, and the pill dispensing container. The mechanism of this design is straightforward: when it is time for the user to take their medication, the pill barrels rotate to release a single pill, while the piezo buzzers sound an alert to notify the user. To enhance convenience, the device is also paired with an iOS

app that provides additional reminders to ensure the user takes their medication on time.

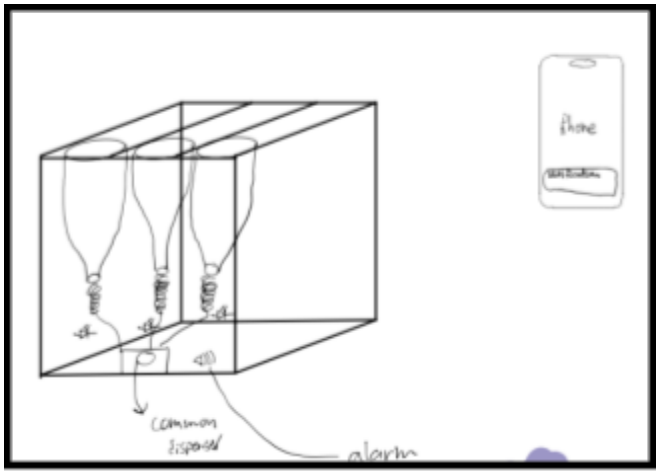
ii **. Design 2**



Design 2

The second design features a durable box, a vibration motor, and a rotating disk system. Similar to the first design, the box holds the key components, including three circular disks, each divided into eight sections—seven for storing pills and one empty for placing them on top. Each disk rotates independently based on the set time, ensuring the user takes the correct pill at the right moment. A vibration motor and alarm are included to help visually impaired users associate vibrations with the pill time. Additionally, the system is connected to a web app that sends notifications when it is time to dispense pills and reminds users if they forget to take them.

iii **. Design 3**



Design 3

The third design features a box housing three water bottles, each designated for different types of pills. Water bottles were chosen for their sustainability and cost-effectiveness, making the design both eco-friendly and economical. Key components include LED lights, an alarm system, a dispenser mechanism, and a

mobile app. Each water bottle is equipped with a spring-loaded piston that compresses to push pills through a small opening when it is time to take them, ensuring the correct number of pills are dispensed. At the same time, the LED light activates and the alarm sounds. The mobile app allows users to set dispensing schedules, receive notifications when it's time to take pills and customize the number of pills per serving. Developed with React Native, the app communicates with the device via Wi-Fi and Bluetooth.

b. Evaluation of Designs (Pugh Matrix)

		Design 1		Design 2		Design 3	
Evaluation Criteria	Wt	Val1	Wt*Val1	Val2	Wt*Val2	Val3	Wt*Val3
Time	0.2	4	0.8	5	1.0	5	1.0
Durability	0.1	5	0.5	4	0.4	3	0.3
Size	0.1	4	0.4	4	0.4	4	0.4
Cost	0.2	4	0.8	5	1.0	5	1.0
Convenience	0.2	5	1.0	3.5	0.7	3	0.6
Aesthetics	0.1	5	0.5	4	0.4	3	0.3
Visible from 10ft	0.1	4	0.4	4	0.4	4	0.4
Total	1.0	31	4.4	29.5	4.3	27	4

Pugh Matrix 4

As a team, we brainstormed potential evaluation criteria and selected seven key factors, as outlined in section 3. We then calculated the weighted average for each criterion, ensuring the total weight added up to 1. Time, cost, and convenience were given the highest weights.

Design 1 performed well across durability, convenience, and aesthetics, with solid scores in time, size, cost, and visibility. It emerged as the most balanced design, earning 31 points.

Design 2 excelled in time and cost and scored reasonably well in durability, size,

aesthetics, and visibility. However, it fell short in convenience, earning 29.5 points—1.5 points behind Design 1.

Design 3 also performed well in time and cost, and scored adequately in size and visibility. However, it struggled with durability, convenience, and aesthetics, earning just 27 points—2.5 points behind Design 2 and 4 points behind Design 1.

We used the Pugh Matrix to evaluate three design alternatives based on key criteria, including time, durability, size, cost, convenience, aesthetics, and visibility from 10 feet. Each criterion was assigned a weight to reflect its relative importance, and each design was scored on a scale of 1 to 5. The scores were then multiplied by the respective weights to calculate the total scores, clearly comparing each design's performance against the project requirements.

Design 1 emerged as the top performer, with a total weighted score of 4.4. It excelled in durability, convenience, and aesthetics, making it a strong choice for long-term use and visual appeal. Design 2 followed closely behind with a score of 4.3, performing particularly well in cost and implementation time. However, it scored slightly lower in convenience and aesthetics compared to Design 1. Design 3, with a total score of 4.0, was the weakest overall, primarily due to lower scores in durability and aesthetics, though it matched the other designs in cost and time efficiency.

In conclusion, Design 1 offers the best balance across all evaluation criteria, making it the most suitable choice for the project. If cost efficiency and implementation speed are the primary concerns, Design 2 may be a viable alternative. However, for optimal durability, user experience, and overall performance, Design 1 stands out as the preferred option.

c. Final Selection

Although each design has its strengths and weaknesses, we have determined that Design

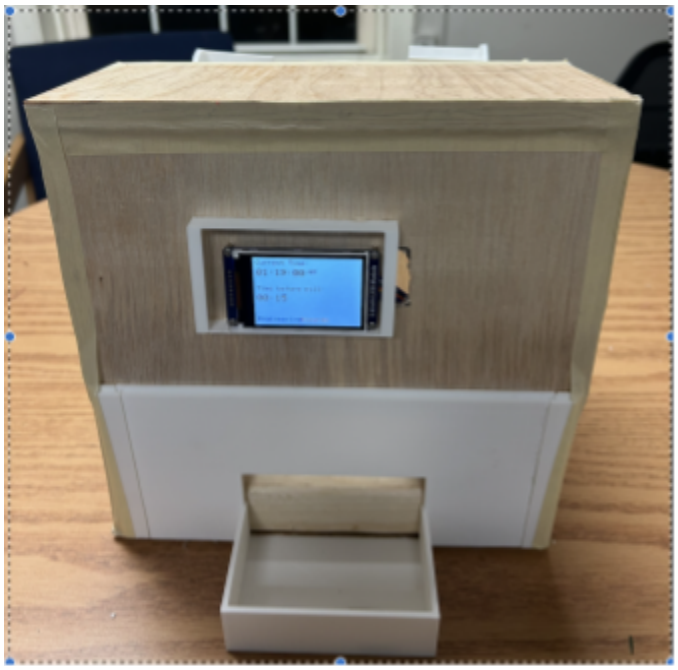
1 will be our final selection, as reflected in the Pugh Matrix with an overall score of 4.4. As shown in section 4.1, Design 1 features a 3D-printed box, piezo buzzer, rotating disk mechanism, and iOS app, making it highly versatile and user-friendly, particularly for blind users. This design ensures that users can take their medications at the right time without having to worry about when or how to do so. Its robust construction also provides notifications every time a pill is dispensed into the designated pill-taking section.

While the design is relatively large and requires periodic refills, it offers significant benefits for blind and elderly individuals. Additionally, Design 1 excels in user-friendliness and accessibility—critical factors for our target audience—making it the most suitable choice for the project.

To improve the convenience of Design 1, our instructor recommended using servo motors instead of standard DC motors. In retrospect, this insightful feedback elevated our concept and took it to a whole new level. To further enhance the convenience of our design, we have opted to replace the iOS app with a cross-platform mobile app.

4. Final Design

a. Overview of Final Design



Front View 5

As shown in Front View 5, the front of our device includes an LCD module, a custom 3D-printed holder to secure the LCD in place, and a container to collect the dispensed pills. We defined the edges of the box to be more aesthetically pleasing and more portable. Additionally, we 3D

printed several key parts, including the white components like the pill dispenser and the LCD display frame. These parts were carefully crafted to ensure an easy user experience, particularly when retrieving pills. To enhance the device's aesthetics and overall quality, we chose 3D printing for precision and design flexibility. The LCD display adds the finishing touch to the front, providing the user with valuable information such as the current time and a countdown to the next pill. We also engraved our logo onto the cover, giving the device a distinctive and professional look.

Top View 6

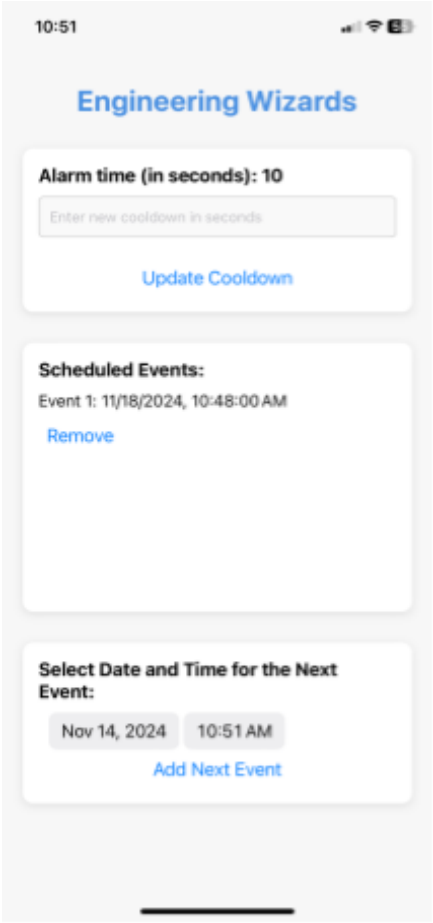
As shown in Top View 6, the top of our device features a lid and an open section that reveals two servo motors connected to the pill barrels. This area is crucial to the device's functionality, as it is where the pills are dispensed. In Top View 6, two white "rotating wheel mechanisms" are visible, which were carefully designed using CAD software to ensure precise alignment with the device's dimensions. These components were then 3D printed in the school's lab. The circular mechanism is divided into five equal sections, with each segment rotated by 72 degrees (360°/5) to ensure that a specific number of pills is dispensed each time. The wooden cover in Top View 6 conceals the internal components, making the design both dustproof and more visually appealing, while enhancing the overall aesthetic of the device.

Back View 7

As shown in Back View 7, the rear of our device features a lid that conceals the internal components, along with two integrated speakers. The back section is designed with the same level of care as the rest of the device. The white 3D-printed part, visible in the picture, not only enhances the device's aesthetics but also carefully hides the internal mechanisms. This part includes three precisely placed holes for cable management. The largest hole, located at the bottom left, provides a pathway for the USB-C cable from the Arduino board, ensuring a smooth and convenient connection to a computer. The two smaller holes on the right are for connecting the speakers to the Arduino board. Additionally, the grid-like structures in the right corner serve dual purposes: they allow sound to escape from the integrated speakers and improve airflow, preventing the device from overheating.

Side View 8

As shown in Side View 8, the side of our device features two buttons—one to start the operation and one to mute the speaker sounds. The edges are secured with tape to ensure the structure remains stable and won't come apart. The main function of this side is to control the two buttons seen in the image. The blue button is used to initiate the program, starting the timer. Once the timer ends and the sound plays, the green button is used to mute the speaker.



Mobile App 9

The **mobile app** *Engineering Wizards* was developed using React Native and Firebase to help users stay on track with their medication regimen, no matter where they are.

Key Features:

- Automatic Time Zone Adjustments

Any date set in the app is automatically adjusted to the user's local time zone.

- Customizable Timer

Set a timer for how long the speaker should play, allowing flexibility in the alert duration.

- Multi-Date Pill Dispensing

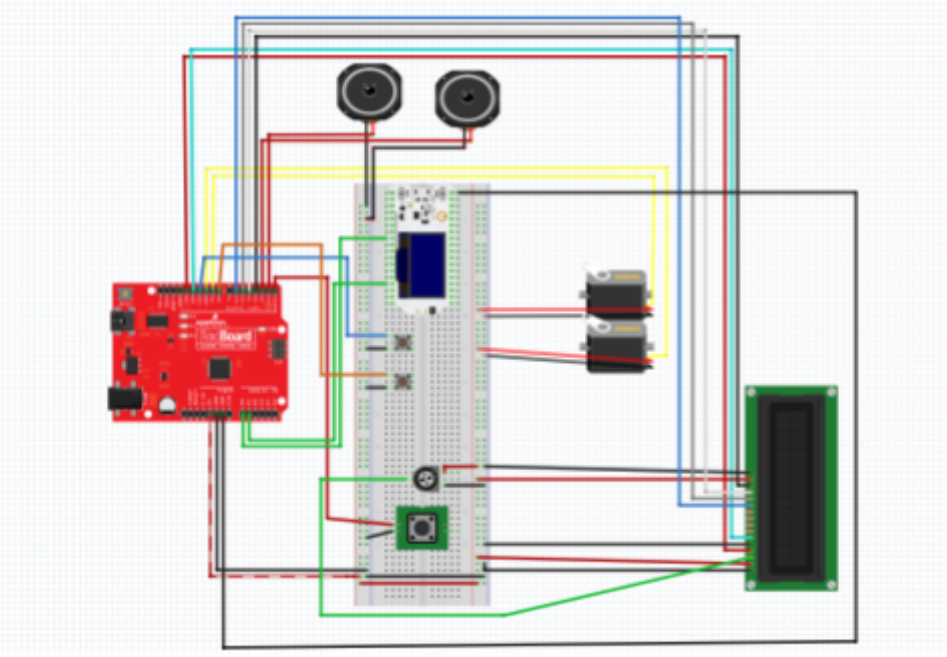
Schedule pill dispensing for specific dates or multiple dates, ensuring you never miss a dose.

- Push Notifications

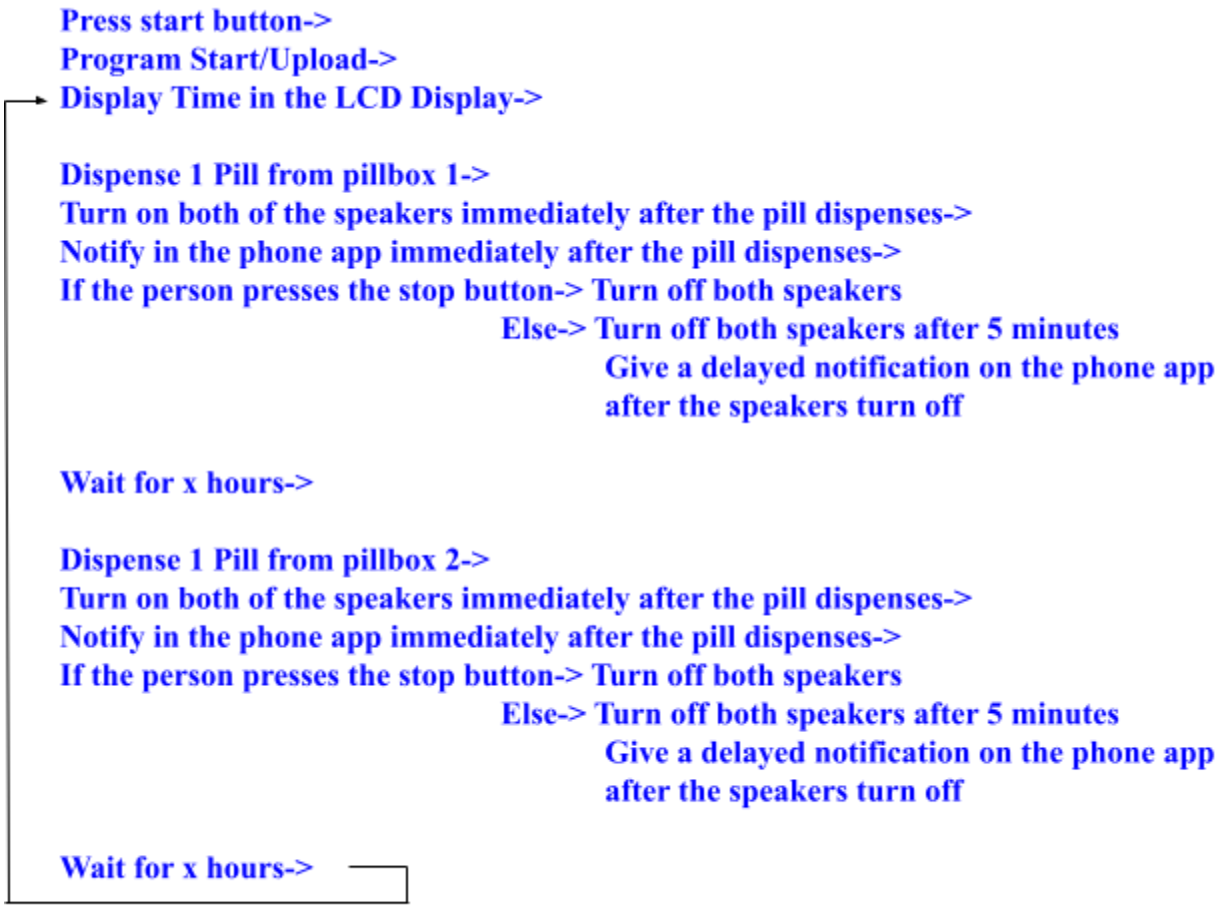
Receive alerts when it is time to take your pill, notifying you when the dispensing event starts.

The app updates data in the Firebase Realtime Database, and the ESP32 board reads this data in real time, triggering the pill dispensing process. Unlike Bluetooth, which has a limited range, the app communicates with the ESP32 board via Wi-Fi. This means the app will work as long as both the phone and the ESP32 are connected to the internet—without needing to be on the same Wi-Fi network.

b. Overview of the Electronics Design



Fritzing Diagram 10



Detailed Functional Decomposition 11

As evident from Detailed Functional Decomposition 11, when the start button is pressed, the program begins, and the LCD starts showing the time in a continuous loop. When the timer reaches zero, the speaker, countdown, and motor will all activate simultaneously. The motor operates once to dispense the pills, while the countdown timer tracks how long the speaker will continue playing before the next scheduled pill dispensing. When the stop button is pressed, both the speaker and the countdown timer will stop and reset.

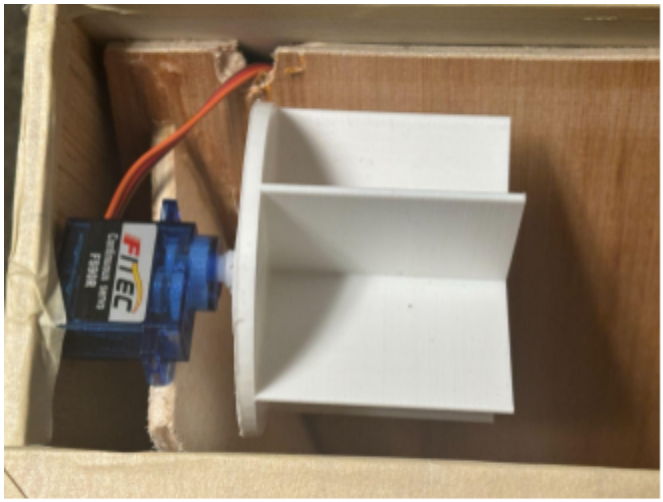
Fritzing Diagram 10 illustrates how all of our components are wired together to ensure seamless integration and proper functionality.

c. First Subsystem: LCD Display System

LCD Display 12

The first subsystem is the LCD display system, which is responsible for showing the current time and a countdown timer indicating when the next pill will be dispensed. Initially, we planned for the LCD monitor to display the message “Time to Take Pill” when it was time for the user to take their medication. However, we encountered a problem: the text would remain on the screen long after the appropriate time had passed. While we explored solutions to limit the display of this message to just five minutes (the time allotted for taking the pill), we found that doing so would interfere with other critical parts of the code.

d. Second Subsystem: Servo Motor System



Servo Motor 13

The second subsystem is the servo motor system, which controls the rotation of the barrels that store the pills. Each barrel has five sections, and the motor rotates 72 degrees to dispense one pill. We use two barrels to allow users to store different types of pills at different times of the day. Among the three subsystems, ensuring consistent functionality of the servo motors proved to be the most challenging. Initially, we set the resting position of the servos to 1500, but

both motors continued to rotate even when they were supposed to be idle. Unsure of the cause, we consulted our professor, who suggested that the servos might be misaligned and that we may need to manually adjust their resting positions. He also recommended trying different values below 1500 to identify specific positions where the servos would stop. We followed his advice and, after several rounds of trial and error, discovered that both servos stopped moving at a position of 1450.

e. Third Subsystem

Speaker System 14

The third subsystem is the alarm and speaker system. In our final code, when the timer reaches zero (indicating it is time for the user to take their pill), the speaker plays a series of musical notes—‘C’, ‘D’, ‘E’, ‘F’, ‘G’, and ‘A’. The sound will continue until the user presses the stop button. While we encountered minimal challenges programming the speaker sounds, we ultimately decided to replace the original buzzer sound with a more pleasant musical tone for a better user experience.

5. Final Evaluation and Results

Performance in DM 3c Evaluation of Full Functionality:

During the DM 3c Initial Evaluation, the automatic pill dispenser performed effectively, achieving our primary goal of dispensing pills on a programmed schedule within a designated 15-second timeframe. Following feedback from the DM 1 Conceptual Design Report, we enhanced clarity in the device’s user interface to improve user interaction and added clear labels

on buttons to guide operation. The servos, speaker, and start and mute buttons all functioned accurately, showing strong initial performance and meeting the core functional requirements of our design.

Action Items from DM 3d Memo:

Based on our DM 3d memo, we focused on two improvements: refining and enhancing the device's durability and making it look more aesthetically pleasing. In terms of design, we positioned labels and buttons externally to improve usability, particularly for unfamiliar users. We also planned to use 3D printed materials to make it look more clean and smooth, although time and budget constraints limited our ability to complete the paint application.

Performance in DM 3e Repeat Evaluation and Overall Assessment:

In the DM 3e Repeat Evaluation, the dispenser again successfully dispensed the correct dosage, confirming its reliability. Most requirements were met; however, a few challenges, such as managing space for pill storage, persisted, with space only for four pills rather than our initial target of ten. Our design demonstrated robustness and usability, although uniform pill sizes proved essential for optimal operation. Overall, the project effectively showed the potential of automated solutions in healthcare, and with future adjustments, the device could evolve into a fully functional product ready for wider application.

6. Conclusion and Lesson Learned

Design and Performance:

The development of our automated pill dispenser has shown the value of integrating technology to address healthcare challenges, particularly for individuals who are elderly or visually impaired. The device successfully met our goal of dispensing pills on a set schedule, which confirms the potential of automated solutions to make daily routines easier and more reliable for those who need regular medication. By streamlining the dispensing process, the device also offers peace of mind to caregivers. Although the current design achieved our functional goals, we recognized that it could benefit from improvements in material durability and the efficiency of its internal mechanisms. Future iterations could explore using higher-quality materials, more streamlined designs to enhance the dispenser's lifespan reduce maintenance needs, and more space to store and dispense the medications. Additionally, user feedback—especially from those who would rely on such a device regularly—could inform further refinements to enhance ease of use and accessibility.

Lessons Learned:

Throughout this project, our team gained valuable insights beyond technical skills, especially regarding collaboration, adaptability, and organization. Communication proved critical; moments of miscommunication occasionally led to duplicated efforts or delays, underscoring the importance of regular check-ins and assigned roles. Flexibility was another key lesson, as we had to adapt to working with new materials and methods, iteratively testing prototypes to find the most effective solutions. This hands-on approach taught us to approach problem-solving with an open mind and to embrace trial and error as part of the design process. Organizational skills also played a significant role: keeping components, tools, and wiring

meticulously organized saved time and minimized errors. Seeking guidance from mentors and making use of available resources, like engineering help desks and office hours, helped us navigate complex issues and led to alternative, sometimes better, solutions. These experiences have equipped us with practical skills and a collaborative mindset, which we look forward to applying in future projects.

7. List of References

SparkFun Inventor's Kit Experiment Guide - v4.1

Arduino Project Hub - The Pill Dispenser 1.0

https://projecthub.arduino.cc/mirandadrummond/the-pill-dispenser-10-abc990

Continuous Rotation Servo Motors and Arduino (Lesson #11)

https://youtu.be/NV6YHZ2RAqc?si=h2W56f7IFV4RAeIQ

Positional vs Continuous Rotation Servo Motors

https://youtu.be/XrEN1oszq_Y?si=hqy2Pv9X-9ZGjYSv

Build an Arduino Automatic Pill Dispenser | Engineering Project

https://youtu.be/00VXq03n_y0?si=TsGgROOcyZ3s7YdK

e-Pill Voice Pro - Locked Automatic Pill Dispenser - Bluetooth Enabled

https://www.amazon.com/MedSmart-Automatic-Dispenser-Bluetooth-Announce/dp/B07C5YDKDL/ref=sr_1_4_sspa?crid=1CG2XCX4RAB1&dib=eyJ2IjoiMSJ9.9Bs9lj5qLwu6uGQkE1W_32NL1bGsOxVx0pC4A70JOZp2ZtQnv7drTdzWCp-K57CkjRo8rZNjpHh3X8z3K3oiIVazWvFi0yoVVc3sTFygNxdozFIJiyLycdP-l7lJaZPc9fL72bf0u4zRmIrrdpleU41WevYjfA645PaDuQoY2bXZ9OZ73Po7zicENu_5E5nIYmPjkbNoBGmRItFRJI2K4CzdJ9pHJG7PUKZklQaSM5B3ZgrpPTjb4xbwJxwgc_1jx2pg41oI4E7FYgGadvK4oRNIlmDIVeA1EQNxXLGRV2I.t2I0bP5k_mh65tL8ZRuWfwj-5JSVzI_bUa9xfGqTUnc&dib_tag=se&keywords=automatic+pill+dispenser&qid=1731577389&sprefix=a

utomatic+pill+dispenser%2Caps%2C316&sr=8-4-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1

Adamson TimelyMed Smart Pill Dispenser Machine with Alarm + WiFi App Monitoring + New 2024 + 28 Day Medicine Dispenser + Automatic Pill Dispenser for Elderly with Alarm & Alzheimers Care + Lock Key

https://www.amazon.com/Adamson-TimelyMed-Dispenser-Monitoring-Alzheimers/dp/B0CLLY2ZTZ/ref=sr_1_33?crid=3G26RN8NARNSE&dib=eyJ2IjoiMSJ9.9Bs9lj5qLwu6uGQkE1W_32NL1bGsOxVx0pC4A70JOZp2ZtQnv7drTdzWCp-K57CkjRo8rZNjpHh3X8z3K3oiIVazWvFi0yoVVc3sTFygNxdozFlJiyLycdP-l7lJaZPc9fL72bf0u4zRmlrrdpleU41WevYjfA645PaDuQoY2bXZ9OZ73Po7zicENu_5E5nIYmPjkbNoBGmRItFRJI2K4CzdJ9pHJG7PUKZkIQaSM5B3ZgrpPTjb4xbwJxwgc_1jx2pg41oI4E7FYgGadvK4oRNlImDIVeA1EQNxXLGRV2I.t2I0bP5k_mh65tL8ZRuWfwj-5JSVzI_bUa9xfGqTUnc&dib_tag=se&keywords=automatic+pill+dispenser&qid=1731577461&sprefix=%2Caps%2C136&sr=8-33

Windtrace Bluetooth Automatic Pill Dispenser for Elderly with Alarm, Smart Pill Dispenser with 31-Slot, Fingerprint& mechanical Dual Lock Medication Dispenser, Timed Pill Dispenser Machine for Elderly

https://www.amazon.com/Windtrace-Bluetooth-Fingerprint-mechanical-Medication/dp/B0D95DLH99/ref=sr_1_22_sspa?crid=6LWV9WXY6LPK&dib=eyJ2IjoiMSJ9.cl8XE2wmGFz3N6N_HeUZgDZqF4a9tyUG8oNhwbTv0xsdU6Sdo2K5AZ3kIcgdc3z0D9EkJloUV_44PefPxNSCMELrUYF4fKGXRTPGox4ZKOJfYCmxPAomDnkkvDqK8I8mORc4QXCMxzYlnBJS5zjta3rzm5p_va_yPeGok1OVVOc1MYUVOFXvFbYmPNePfldxaoJw8U7FCO0rrt-1dmG1H8nbnM91RoWSpdekDSTQ_ECX8d6YCBhX1PVhawjsqMnq-WP51UZ37_d41xfunjn9YOWhiER-E89ndsMKV6m4JzE.M5ZM_8JcdDd2svhpTibkF8AgZdI6rJevEkRewJMSoko&dib_tag=se&keywords=automatic+pill+dispenser+barrel&qid=1731577544&sprefix=automatic+pill+dispenser+barrel%2Caps%2C103&sr=8-22-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9tdGY&psc=1

8. Appendix A: Arduino Code

```
#include <Adafruit_GFX.h>      // Core graphics library
#include <Adafruit_ILI9341.h>   // Library for the display
#include <Servo.h>

// Pin definitions for the display
#define TFT_CS     10
#define TFT_RST    8
#define TFT_DC     9
#define TFT_MOSI   11
#define TFT_CLK    13
#define TFT_MISO   12

// Define the servo pin
Servo myServo;
Servo myServo1;
const int servoPin = 2; // Change this to your servo pin
const int servoPin1 = 5;
const int ServoStop = 1450;
const int ServoStop1 = 1450;

// Define the rotation speeds (adjust these as needed)
const int speed = 255; // Maximum speed
const int servoRotation = 187; // 188 millsec delay

// Color definitions for the display
#define ILI9341_WHITE 0xFFFF
#define ILI9341_BLACK 0x0000
#define ILI9341_BLUE  0x001F
#define ILI9341_RED   0xF800

// Initialize display
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC, TFT_RST);

// Time variables for the display and alarm
unsigned long previousMillis = 0;         // For display update
const unsigned long interval = 1000;      // Update interval of 1 second

int hours = 1;     // Start time: 1:00 PM
int minutes = 19;   // Start minutes
int seconds = 0;    // Start seconds
```

```
int pillTime = 15;  // Countdown for pill time (30 seconds for testing)

int speakerPin = 6;    // Pin connected to the speaker (buzzer)
bool isPlayingAlarm = false;  // Alarm status flag
bool start = false;

// Melody notes and durations (simple melody example)
int melody[] = {262, 294, 330, 349, 392, 440, 494, 523}; // C D E F G A B C (octave)
int noteDurations[] = {500, 500, 500, 500, 500, 500, 500, 500}; // Duration for each
note in ms
int melodyIndex = 0; // To keep track of the current note in the melody

void setup() {
 Serial.begin(9600);  // Initialize serial communication for debugging
 pinMode(speakerPin, OUTPUT);  // Set the speaker pin as an output
 pinMode(4, INPUT_PULLUP);  // Set pin 4 as input with pull-up resistor
 pinMode(3, INPUT_PULLUP); // To make the program start

 // Initialize the display
 tft.begin();
 tft.setRotation(3);  // Adjust rotation if necessary
 tft.fillScreen(ILI9341_WHITE);  // Set background to white

 // Set initial text color and size for the display
 tft.setTextColor(ILI9341_BLACK);
 tft.setTextSize(2);
 tft.setCursor(10, 10);
 tft.print("Current Time:");

 tft.setCursor(10, 100);
 tft.print("Time before pill:");

 // Initially display both the time and pill countdown
 displayTime(hours, minutes, seconds);
 displayPillTime(pillTime);

 // Display the footer with "Engineering Wizards"
 displayFooter();

 myServo.attach(servoPin);
 myServo1.attach(servoPin1);
}
```

```
void loop() {
 while (start == false){
   if (digitalRead(3) == 0){
     start = true;
   }
 }

 Serial.println(digitalRead(3));
 Serial.println(digitalRead(4));
 myServo.writeMicroseconds(ServoStop);
 myServo1.writeMicroseconds(ServoStop1);
 unsigned long currentMillis = millis();

 // Update the display every second
 if (currentMillis - previousMillis >= interval) {
   previousMillis = currentMillis;

   // Update time
   seconds++;
   if (seconds >= 60) {
     seconds = 0;
     minutes++;
     if (minutes >= 60) {
       minutes = 0;
       hours++;
       if (hours >= 24) {
         hours = 0;
       }
     }
   }

   // Update pill countdown timer
   pillTime--;
   if (pillTime < 0) {
     pillTime = 15;  // Reset to 10 seconds for testing
   }

   // Display the updated time and pill countdown
   displayTime(hours, minutes, seconds);
   displayPillTime(pillTime);
```

```
  if (pillTime == 0){
    myServo.writeMicroseconds(1500+speed);
    delay(187);
    myServo.writeMicroseconds(ServoStop);
    myServo1.writeMicroseconds(1500+speed);
    delay(187);
    myServo1.writeMicroseconds(ServoStop1);
    delay(200);
  }
 }

 // Play melody continuously when pill time is zero
 if (pillTime == 0 && !isPlayingAlarm) {
  isPlayingAlarm = true; // Set alarm flag
  melodyIndex = 0; // Reset melody to the beginning
 }

 if (isPlayingAlarm) {
  // Play current note in the melody
  tone(speakerPin, melody[melodyIndex], noteDurations[melodyIndex]);
  delay(noteDurations[melodyIndex] * 1.3); // Delay for the duration of the note

  melodyIndex++; // Move to the next note

  // Restart melody if it reaches the end
  if (melodyIndex >= 8) {
    melodyIndex = 0;
  }

  // Stop the melody if the button is pressed
  if (digitalRead(4) == 0) {
    isPlayingAlarm = false;
    noTone(speakerPin); // Stop sound
  }
 }
}

// Function to display the countdown timer for the pill
void displayPillTime(int pillTime) {
 Serial.println(digitalRead(3));
 int minutesRemaining = pillTime / 60;
 int secondsRemaining = pillTime % 60;
```

```
 // Clear previous countdown
 tft.fillRect(10, 130, 200, 40, ILI9341_WHITE);

 // Display the remaining time in MM:SS format
 tft.setCursor(10, 130);
 tft.setTextSize(3);
 if (minutesRemaining < 10) tft.print("0");
 tft.print(minutesRemaining);
 tft.print(":");
 if (secondsRemaining < 10) tft.print("0");
 tft.print(secondsRemaining);
}

// Function to display the current time
void displayTime(int hours, int minutes, int seconds) {
 Serial.println(digitalRead(3));
 // Clear the previous time
 tft.fillRect(10, 40, 200, 40, ILI9341_WHITE);

 // Display time in HH:MM:SS format
 tft.setCursor(10, 40);
 tft.setTextColor(ILI9341_BLACK);
 tft.setTextSize(3);

 if (hours < 10) tft.print("0");
 tft.print(hours);
 tft.print(":");

 if (minutes < 10) tft.print("0");
 tft.print(minutes);
 tft.print(":");

 if (seconds < 10) tft.print("0");
 tft.print(seconds);

 // Optional: Add AM/PM format
 tft.setCursor(150, 40);
 tft.setTextSize(2);
 if (hours >= 12) {
   tft.print(" PM");
 } else {
```

```
    tft.print(" AM");
  }
}

// Function to display "Engineering Wizards" at the bottom
void displayFooter() {
  // Clear the bottom section
  tft.fillRect(10, 210, 300, 40, ILI9341_WHITE);

  // Set text size for both words
  tft.setTextSize(2);

  // Set the text color for "Engineering"
  tft.setCursor(10, 210);
  tft.setTextColor(ILI9341_BLUE);  // Use blue for "Engineering"
  tft.print("Engineering");

  // Set the text color for "Wizards"
  tft.setCursor(140, 210);
  tft.setTextColor(ILI9341_RED);  // Use red for "Wizards"
  tft.print("Wizards");
}
```

Mobile App Code,

```
import React, { useEffect, useState } from 'react';
import { View, Text, TextInput, Button, StyleSheet, Platform,
TouchableWithoutFeedback, Keyboard, ScrollView, FlatList } from
'react-native';
import { initializeApp } from 'firebase/app';
import { getDatabase, ref, set, push, remove, onValue } from
'firebase/database';
import DateTimePicker from '@react-native-community/datetimepicker';
import * as Notifications from 'expo-notifications';


const firebaseConfig = {
  apiKey: 'AIzaSyDpohEwh7K_0MDNhOSqU0N3tMTtVdXHfsQ', // maybe i should
remove the api key
  databaseURL: 'https://esp32-arduino-39078-default-rtdb.firebaseio.com/',
  projectId: '413645774072',
```

```
  messagingSenderId: '413645774072',
};

const app = initializeApp(firebaseConfig);
const database = getDatabase(app);

const App = () => {
  const [cooldown, setCooldown] = useState('');
  const [selectedDate, setSelectedDate] = useState(new Date());
  const [selectedTimestamp, setSelectedTimestamp] = useState(null);
  const [cooldownSeconds, setCooldownSeconds] = useState(0);
  const [events, setEvents] = useState([]);

  const requestNotificationPermissions = async () => {
    const { status } = await Notifications.getPermissionsAsync();
    if (status !== 'granted') {
      const { status: newStatus } = await
Notifications.requestPermissionsAsync();
      if (newStatus !== 'granted') {
        alert('permissions not granted');
      } else {
        console.log('permissions granted!');
      }
    } else {
      console.log('permission not granted')
    }
  };

  useEffect(() => {
    requestNotificationPermissions();

    const cooldownRef = ref(database, '/timer/cooldown');
    onValue(cooldownRef, (snapshot) => {
      const data = snapshot.val();
      setCooldownSeconds(data ? parseInt(data, 10) : 0);
    });

    const interval = setInterval(() => {
      if (cooldownSeconds > 0) {
        setCooldownSeconds((prev) => prev - 1);
```

```
      }
    }, 1000);

    return () => clearInterval(interval);
  }, [cooldownSeconds]);

  useEffect(() => {
    const eventsRef = ref(database, '/timer/events');
    onValue(eventsRef, (snapshot) => {
      const data = snapshot.val();
      const eventList = data ? Object.entries(data) : [];
      setEvents(eventList.map(([id, eventData]) => ({ id, ...eventData
}))));
    });
  }, []);

  const handleCooldownUpdate = () => {
    Keyboard.dismiss();
    const cooldownRef = ref(database, '/timer/cooldown');
    set(cooldownRef, parseInt(cooldown, 10)).then(() => {
      alert('Cooldown updated!');
    });
  };

  const handleNextEventUpdate = async () => {
    if (selectedTimestamp) {
      const eventsRef = ref(database, '/timer/events');
      const newEventRef = push(eventsRef);
      await set(newEventRef, {
        timestamp: selectedTimestamp,
        message: 'Time to eat Pill',
      });
      alert('Next event added!');

      const notificationTime = new Date(selectedTimestamp * 1000);
      console.log('Scheduling notification for:', notificationTime);

      if (notificationTime > new Date()) {
        await Notifications.scheduleNotificationAsync({
          content: {
```

```
            title: 'Time to eat Pill',
            body: 'The pill has dropped',
            sound: 'default',
          },
          trigger: {
            date: notificationTime,
          },
        });
        console.log('Notification scheduled successfully.');
      } else {
        alert('Cant select past time');
      }
    } else {
      alert('Select a valid date');
    }
  };

  const onDateChange = (event, selectedDate) => {
    const currentDate = selectedDate || selectedDate;
    setSelectedDate(currentDate);
    setSelectedTimestamp(Math.floor(currentDate.getTime() / 1000));
  };

  const handleDeleteEvent = async (eventId) => {
    const eventRef = ref(database, `/timer/events/${eventId}`);
    await remove(eventRef);
    alert('Event removed successfully!');
  };

  return (
    <TouchableWithoutFeedback onPress={() => Keyboard.dismiss()}>
      <View style={styles.container}>
        <Text style={styles.header}>Engineering Wizards</Text>

        <View contentContainerStyle={styles.scrollContainer}>
          <View style={styles.card}>

            <Text style={styles.subHeader}>Alarm time (in seconds):
{cooldownSeconds}</Text>
            <TextInput
```

```
              style={styles.input}
              value={cooldown}
              onChangeText={setCooldown}
              keyboardType="numeric"
              placeholder="Enter new cooldown in seconds"
            />
            <Button title="Update Cooldown" onPress={handleCooldownUpdate}
/>
          </View>


          <View style={styles.card}>
            <Text style={styles.subHeader}>Scheduled Events:</Text>
            <View style={styles.eventListContainer}>
              <ScrollView style={styles.scrollableEventList}
contentContainerStyle={styles.eventList}>
                {events.length > 0 ? (
                  events.map((event, index) => (
                    <View key={event.id} style={styles.eventContainer}>
                      <Text style={styles.eventText}>
                        Event {index + 1}: {new Date(event.timestamp *
1000).toLocaleString()}
                      </Text>
                      <Button title="Remove" onPress={() =>
handleDeleteEvent(event.id)} />
                    </View>
                  ))
                ) : (
                  <Text>No events scheduled yet.</Text>
                )}
              </ScrollView>
            </View>
          </View>


          <View style={styles.card}>
            <Text style={styles.subHeader}>Select Date and Time for the
Next Event:</Text>
            <DateTimePicker
              value={selectedDate}
```

```
              mode="datetime"
              is24Hour={true}
              onChange={onDateChange}
            />
            {selectedTimestamp && (
              <Text style={styles.timestamp}>Selected Date: {new
Date(selectedTimestamp * 1000).toLocaleString()}</Text>
            )}
            <Button title="Add Next Event" onPress={handleNextEventUpdate}
/>
          </View>
        </View>
      </View>
    </TouchableWithoutFeedback>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#f7f7f7',
    padding: 20,
  },
  header: {
    fontSize: 28,
    fontWeight: 'bold',
    color: '#4A90E2',
    textAlign: 'center',
    marginTop: 60,
    marginBottom: 20,
  },
  subHeader: {
    fontSize: 18,
    fontWeight: 'bold',
    marginBottom: 10,
  },
  scrollContainer: {
    flexGrow: 1,
    justifyContent: 'center',
  },
```

```
  card: {
    backgroundColor: 'white',
    padding: 15,
    marginBottom: 20,
    marginTop:10,
    borderRadius: 10,
    shadowColor: '#000',
    shadowOpacity: 0.1,
    shadowOffset: { width: 0, height: 2 },
    shadowRadius: 4,
    elevation: 5,
  },
  input: {
    height: 40,
    borderColor: '#ccc',
    borderWidth: 1,
    borderRadius: 5,
    marginBottom: 20,
    paddingHorizontal: 10,
    backgroundColor: '#f9f9f9',
  },
  eventListContainer: {
    height: 200,
    overflow: 'hidden',
  },
  scrollableEventList: {
    maxHeight: 200,
  },
  eventList: {
    paddingBottom: 10,
  },
  eventContainer: {
    marginBottom: 10,
    alignItems: 'flex-start',
  },
  eventText: {
    fontSize: 16,
    marginBottom: 5,
  },
  timestamp: {
```

```
    fontSize: 14,
    color: '#888',
    marginBottom: 10,
  },
});

export default App;
```