ME102B Final Report: Automated Pill Dispenser

Group 20 - Jason Khatkar, Noah Koch, Kayla Le, Gunit Pabla

Opportunity

As people age, they may experience memory challenges that can make everyday tasks and routines more complex, potentially impacting their sense of independence. Similarly, individuals with disabilities or cognitive differences may face unique challenges in managing daily activities. This creates an opportunity to develop innovative solutions that support people of all abilities in maintaining their routines with greater ease and confidence. Our product is designed to enhance the ability to navigate daily life while respecting autonomy and individuality.

Strategy

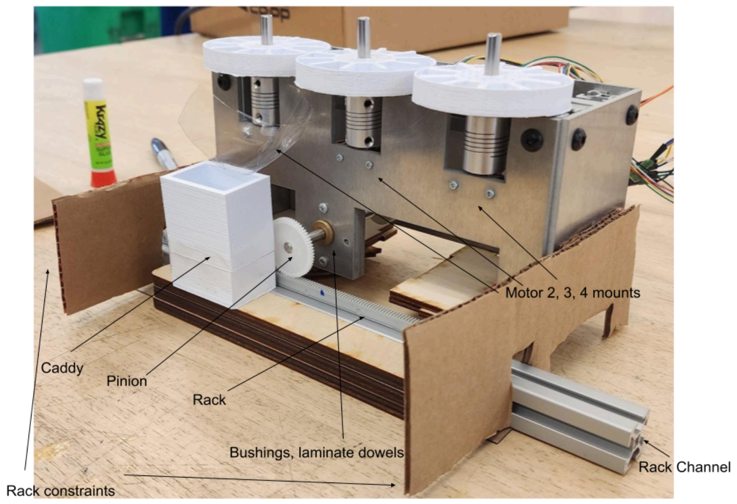
Our main goal was to create a mechanical system that collected pills from each dispenser using a rack and pinion. This would allow our machine to have one well-defined degree of lateral freedom in addition to the dispenser mechanism. We felt that this system, namely the transmission shaft, was the highlight of the project. Implementation of this system required a few alterations in design however; in order to provide support for the cantilever load of the shaft, we had to add two bushings, which required altering the design of the front-plate. Furthermore, the new front-plate design required locating features as well as fasteners; we did this on the fly with ⅛" wooden dowels and 4-40 nuts and bolts, which worked perfectly for creating a laminated part of the front-plate which could support the bushings, which were pressed in.

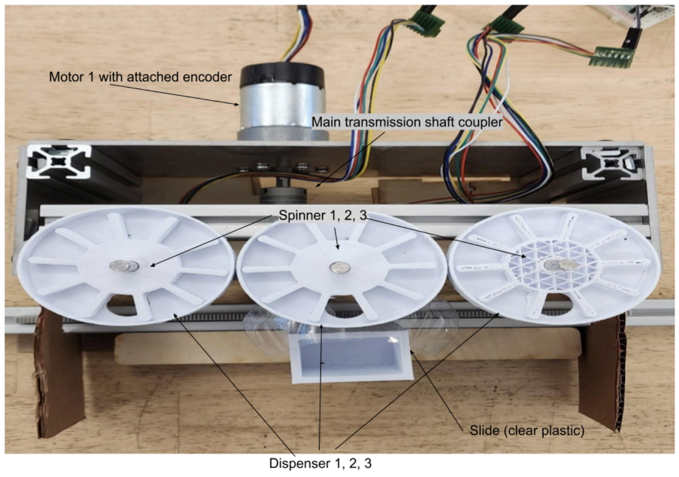
The physical implementation of our design also required us to rework how the rack laterally translated along the ground. We initially wanted it to be free-floating, but this meant it was unconstrained, and it became evident once attached the caddy to the rack that straight lateral movement without constraints was going to be impossible. However, we found that the rack fit within a slot of our leftover 8020 aluminum bars, and we constrained this to the rest of the machine using some handmade cardboard brackets that allowed for an interference fit with the aluminum. After this was implemented, we had no more problems with lateral motion.

Our dispenser system only needed minor adjustments, mainly the PWM values for the spinning, which was adjusted by eye with repeated testing. However, this system could become unreliable with repeated use due to the lack of an encoder; we wanted to use encoders for this system but lacked the pins on our ESP that would accommodate them. We could have added an extra ESP, but this would have required real-time interfacing between ESPs that could have resulted in more issues. The original goal of the project was to keep all electronics to one MCU which was something that we were able to do.

Ultimately, the design worked as expected. We had no real defined requirements for system performance, and chose to tune things by eye. In terms of mechanical design, we were able to realize the mechanisms that we originally intended to use, that being a rack and pinion-driven system that works in conjunction with individual motors that are coupled to rotate at an optimal PWM.

Physical Assembly




Motor 1 with attached encoder
Main transmission shaft coupler
Spinner 1, 2, 3
Slide (clear plastic)
Dispenser 1, 2, 3

<u>Function-Critical Decisions</u>

Our focus was on a functional main transmission, which needed a motor that would be able to transmit the necessary torque to move the caddy at a reasonable speed to each position.

- <u>Motor Specifications:</u> τ_{stall} = 18Kg*cm = 1.7658 N*m
- <u>Mass Contributions:</u> m_{shaft} = 0.02185 kg, $m_{coupler}$ = 0.0158 kg, m_{pinion} = 0.0027 kg, m_{rack} = 0.0473 kg
- <u>Key Dimensions:</u> r_{shaft} = 6mm , $r_{coupler}$ = 9.55 mm , r_{pinion} = 15mm , r_{rack}= 20.9mm, μ:= 0.42
- <u>Travel Time:</u> 1 second <u>Travel Distance:</u> 0.3 m

Torque Calculation:

Mass Moment of Inertia = $\Sigma I = ½\, m_{shaft} * r_{shaft}^2 + ½\, m_{coupler} *(r_{coupler} - r_{shaft)})^2 + ¼\, m_{pinion} *(r_{pinion} - r_{shaft})^2$

$= (0.5*0.022*0.006^2)+(0.5*0.016*(0.0036^2)+(0.25*0.003*0.009^2)$

$= 5.56 \times 10^{-7}$ kg*m^2

Rotational Acceleration: $\theta = ½\, \alpha * t^2$ □$\alpha = 2*\theta/t^2 = 2*(0.3*2\pi/r_{rack})/(1^2)$= 0.18 rad/s^2

$\tau = I*\alpha = 5.56 \times 10^{-7} * 0.18 = 1 \times 10^{-7}$ N*m

Frictional Torque Calculation:

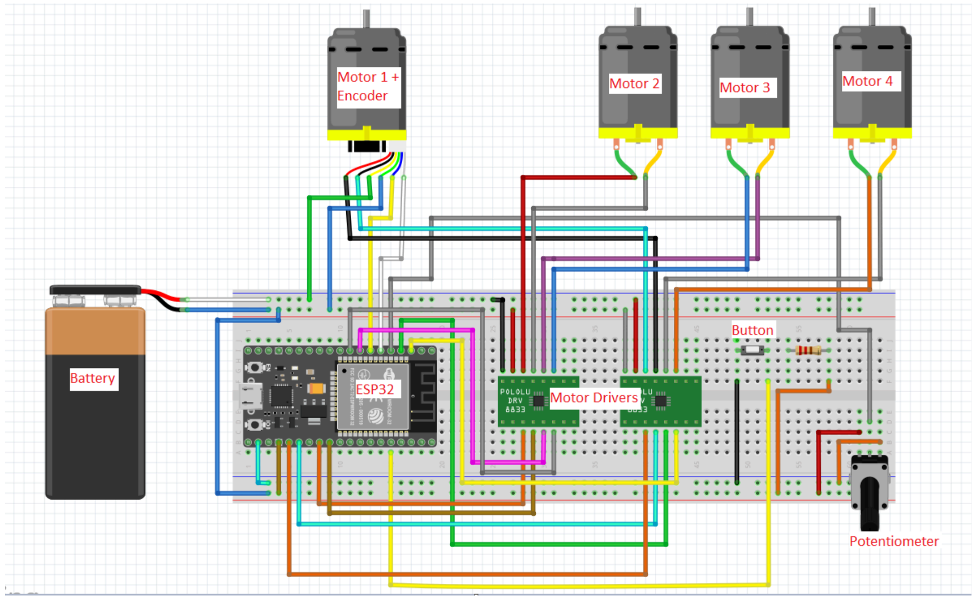
Frictional Torque: $\tau_{load} = \mu * (m_{rack}) * g * r_{rack} = 0.42 * 0.022 * 9.81 * 0.0209 = 0.0019$ N*m

Total Torque:

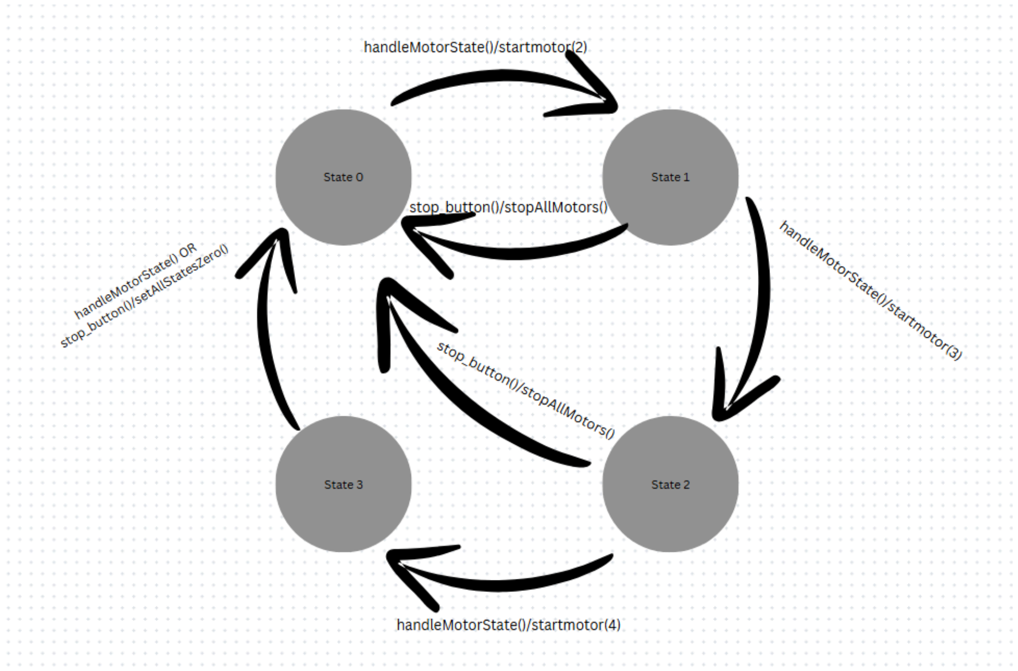
$\tau_{total} = \tau + \tau_{load} = 0.0019$ N*m

This is far lower than the stall torque found on our motor's spec sheet.

Circuit Diagram



State Diagram


handleMotorState()/startmotor(2)
State 0
State 1
stop_button()/stopAllMotors()
handleMotorState()/startmotor(3)
handleMotorState() OR stop_button()/setAllStatesZero()
stop_button()/stopAllMotors()
State 3
State 2
handleMotorState()/startmotor(4)

Reflection

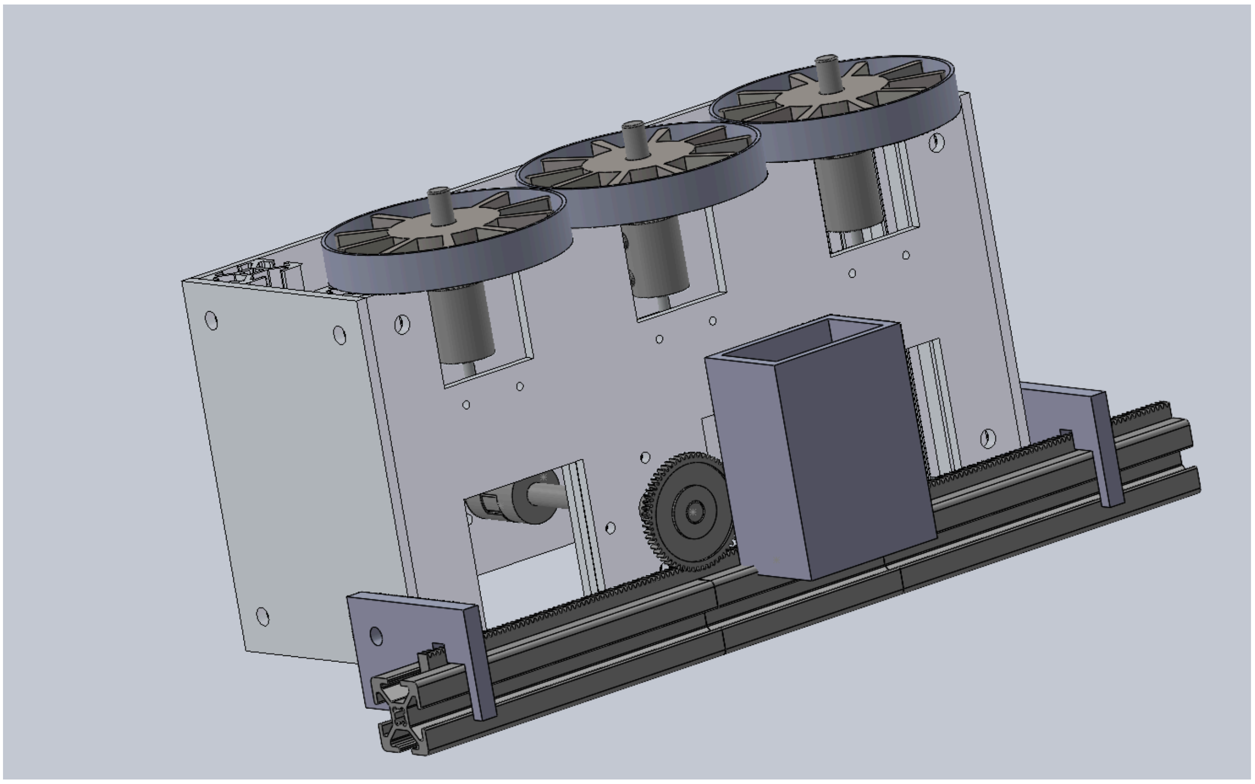
One strategy that worked well for our group was working thoroughly on the CAD design and integrating space for all the requirements early on. It allowed us to meet with the professor and GSIs in office hours for several weeks before physical prototyping. However, we wish we had our prototype functioning sooner as it would have revealed design flaws that had to be solved in makeshift ways. For future students, we recommend getting some sort of prototype early and checking in often with the teaching staff, as we learned of obstacles with our design every time we met up with them.

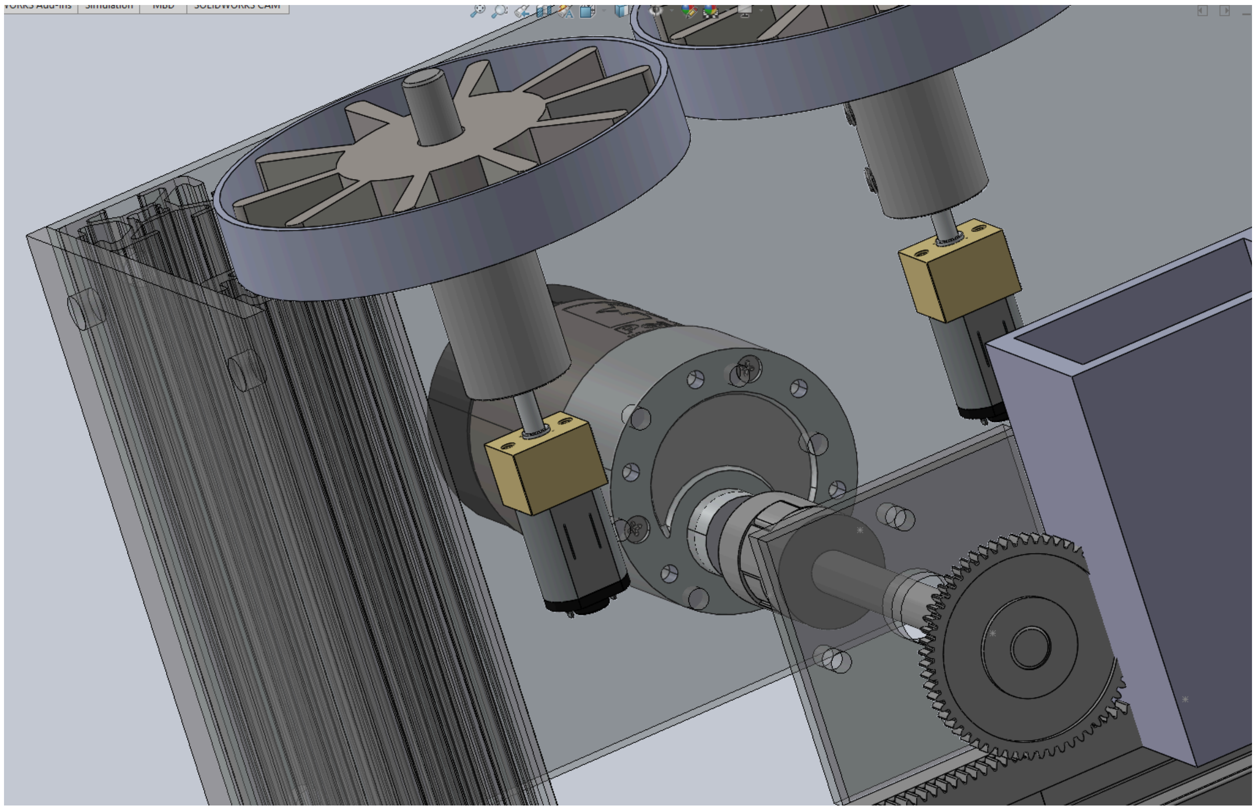
Appendix A: Bill of Materials

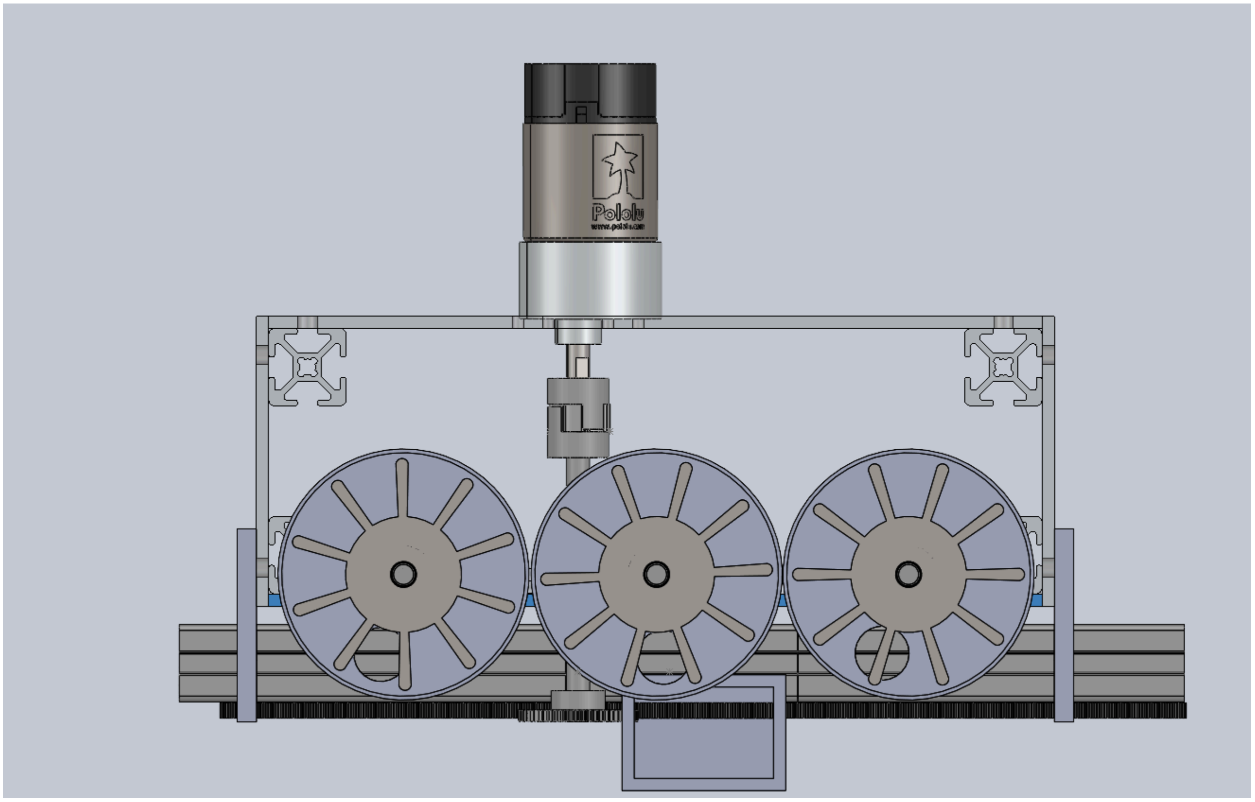
Item	Quantity	Total Cost	Obtained From	Link
2V Metal DC Geared Motor with Encoder (43.8:1, 251RPM, 18Kg.cm)	1	0	Labkit	https://test.dfrobot.com/product-634.html
5:1 Micro Metal Gearmotor HP 6V with Extended Motor Shaft	3	0	Labkit	https://www.pololu.com/product/2215
Polulu mounting brackets	3	0	Labkit	https://www.pololu.com/product/989
#2 screws and nuts	6	0	Labkit	
303 Steel 6mm Shaft 400mm length	1	$21.81	McMaster	https://www.mcmaster.com/4143n13/
6mm to 6mm Flexible Shaft Coupler	1	$57.41	McMaster	https://www.mcmaster.com/4147N144/
Rack	2	$6.88 ($3.44 individual)	McMaster	https://www.mcmaster.com/2662N55/
Pinion	1	$4.63	McMaster	https://www.mcmaster.com/2662N32/
Spinning Gear	3	0	3D printed	
Dispenser Holders	3	0	3D printed	
Pill Caddy	1	0	3D printed	
Aluminum Beam (casing)	1	0	Machine Shop	
⅛" Aluminum Sheet 1'x1' (casing)	1	$14.63	Amazon	https://www.amazon.com/dp/B08YFJ1JV4?psc=1&smid=A3HDY9C2ND66SN&ref_=chk_typ_imgToDp
8020 Aluminum	2	$15.28 ($7.64	8020 dot net	https://8020.net/20-2020.html

Frame (500mm of 20mm)		individual)		
8020 Aluminum Bracket	1	$2.63	8020 dot net	https://8020.net/14057.html
Bolt Assemblies for Bracket	4	$3.20 ($0.80 individual)	8020 dot net	https://8020.net/75-3581.html
ESP32 MCU + wiring	1	0	Labkit	
M3 Screws (dfrobot motor)	6	$1.98 ($0.33 individual)	Ace Hardware	
4-40 Screws + Nuts (frontplate laminate fastener)	2	$0.66 ($0.33 individual)	Ace Hardware	
⅛”-diameter Wooden Dowels	1	$0.28	Jacobs Store	https://store.jacobshall.org/products/wooden-dowel-1-8?_pos=2&_sid=2eefdb9fa&_ss=r
M5 Screws (casing)	16	$12.80 ($0.80 individual)	8020	https://8020.net/75-3581.html?srsltid=AfmBOoqgySc szGfBgMTtqPqF3bNYqSHTmMAQZOjxAyHf1dHH2jBmZkel
Cardboard rack constraints	2	0	Cardboard box	
Motor Driver	2	0	Labkit	
Power supply	1	0	Labkit	
Krazy Superglue	1	0	Had on hand	
Potentiometer	1	0	Labkit	
Bushings (pack of 4, two used)	1	$10.89 ($2.72 for individual bushing)	Amazon	https://www.amazon.com/dp/B095N8MZ8Z
3mm to 6mm flexible coupler (pack of 2)	2	$18.58 ($4.65 for individual coupler)	Amazon	https://a.co/d/2z0ZxxZ
TOTAL:		**$171.66**		

Appendix B: CAD - Mechanical Transmission Elements







Appendix C: Code

```
#include <ESP32Encoder.h>
#include <Arduino.h>

// Pin Definitions
#define BIN_1 25
#define BIN_2 26
#define AIN_1 36
#define AIN_2 4
#define POT_PIN 15
#define STOP_BUTTON_PIN 5
#define CIN_1 12
#define CIN_2 13
#define DIN_1 14
#define DIN_2 32

ESP32Encoder encoder1;

// Motor 1 Control Variables
int theta1 = 0;
int thetaDes1 = 0;
int thetaMax = 43.8 * 46;
int D1 = 0;
// Control parameters
int Kp = 8;
float Ki = 0.005;
float Kd = 0.5;
int IMax = 12;
int e1 = 0;
int e1_prev = 0;
int e_sum1 = 0;
const int deadband = 3;

// Timer service control
hw_timer_t *timer = NULL;
volatile bool timerFlag = false;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;

// Motor timing control variables
volatile unsigned long motor2StartTime = 0;
volatile unsigned long motor3StartTime = 0;
volatile unsigned long motor4StartTime = 0;

// Potentiometer variables
const int POT_THRESHOLD = 2047;
bool systemEnabled = false;
bool skip = false;

// Setting PWM properties
const int freq = 5000;
const int resolution = 8;
const int MAX_PWM_VOLTAGE = 150;
bool stop_button = false;

enum MotorState {IDLE, SPINNING};
MotorState motor2State = IDLE;
MotorState motor3State = IDLE;
MotorState motor4State = IDLE;

void IRAM_ATTR onTimer() {
    portENTER_CRITICAL_ISR(&timerMux);
    timerFlag = true;
    portEXIT_CRITICAL_ISR(&timerMux);
}

void setupTimer() {
    timer = timerBegin(0, 80, true); // 80 prescaler for 1 microsecond
tick
    timerAttachInterrupt(timer, &onTimer, true);
    timerAlarmWrite(timer, 1000, true); // 1ms interval
    timerAlarmEnable(timer);
}

void startMotor(int pin1, int pin2, int V) {
    ledcWrite(pin1, 0);
    ledcWrite(pin2, V);
}

void stopMotor(int pin1, int pin2) {
    ledcWrite(pin1, 0);
    ledcWrite(pin2, 0);
}

void stopAllMotors() {
    stopMotor(BIN_1, BIN_2);
    stopMotor(AIN_1, AIN_2);
    stopMotor(CIN_1, CIN_2);
    stopMotor(DIN_1, DIN_2);

    motor2State = IDLE;
    motor3State = IDLE;
    motor4State = IDLE;

    thetaDes1 = theta1;
    e_sum1 = 0;
}

void handleMotorState(unsigned long currentMillis) {
    if (motor2State == SPINNING && currentMillis >= motor2StartTime) {
        startMotor(AIN_1, AIN_2, MAX_PWM_VOLTAGE);
        if (currentMillis >= motor2StartTime + 50) {
            stopMotor(AIN_1, AIN_2);
            motor2State = IDLE;
        }
    }

    if (motor3State == SPINNING && currentMillis >= motor3StartTime) {
        startMotor(CIN_1, CIN_2, MAX_PWM_VOLTAGE);
        if (currentMillis >= motor3StartTime + 50) {
            stopMotor(CIN_1, CIN_2);
            motor3State = IDLE;
        }
    }

    if (motor4State == SPINNING && currentMillis >= motor4StartTime) {
        startMotor(DIN_1, DIN_2, MAX_PWM_VOLTAGE);
        if (currentMillis >= motor4StartTime + 50) {
            stopMotor(DIN_1, DIN_2);
            motor4State = IDLE;
        }
    }
}

void loop() {
    if (timerFlag) {
        portENTER_CRITICAL(&timerMux);
        timerFlag = false;
        portEXIT_CRITICAL(&timerMux);

        unsigned long currentMillis = millis();

        if (digitalRead(STOP_BUTTON_PIN) == HIGH) {
            stopAllMotors();
            stop_button = true;
        }

        int potValue = analogRead(POT_PIN);
        bool newSystemEnabled = (potValue > POT_THRESHOLD);
        skip = (potValue > 3000);

        if (newSystemEnabled != systemEnabled) {
            systemEnabled = newSystemEnabled;
        }

        // Call motor state handler
        handleMotorState(currentMillis);
    }
}
```