“FAFA: FOOD ACCESSIBILITY FOR ALL” ENGINEERING MANUAL

A Formal Report for ME102B

Submitted to the Faculty of the

Mechanical Engineering Department
University of California, Berkeley
Berkeley, CA 94720

14 December 2023

Team 13

[illegible]

Ashley Lee

[illegible]

Justin Nam

Table of Contents

Opportunity

Food security and accessibility is intrinsically dependent upon predictable climate and healthy ecosystems. As the threats of climate change, severe weather, fire, and disease continue to increase, poor and vulnerable communities feel the effects disproportionally [1]. There is an increasing demand for food growth in extreme environments with climates insufficient for food growth. Our project aims to address the most ubiquitous influence on agriculture: climate. The project, "FAFA: Food Accessibility For All", aims to mimic the light cycle and exposure plants need to receive in order to optimize growth. In order to ensure accessibility, our group made it fundamental to our research and design that the components be easy to assemble, user-friendly, and cost-effective (see Appendix A). This promotes usage worldwide, especially in regions with extreme climates and poverty-stricken areas.

High-Level Strategy

In the initial stages of design (see Appendix B), we decided the product must be isolated from its external environment to ensure the efficiency of simulating a different climate. This meant the system must be enclosed with a structure covering the overhead of the system. In order to simulate the patterns of the sun, we decided a dome would best produce the orbital of the sun in the sky throughout the various seasons (see Appendix D).

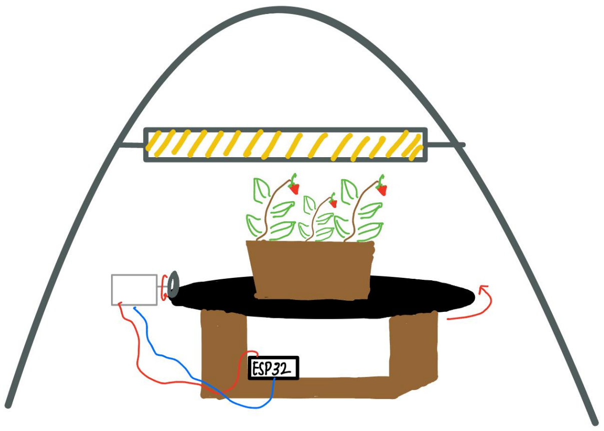


Figure 1. Initial Design of the FAFA Project Device.

We decided that the plant will rest on a rotating disc whose speed and direction is adjustable by the user. Utilizing the ESP32 and a DC motor, we implemented event-driven programming through Arduino. First, our team designed a State Transition Diagram to map out all possible states our system will be in and the inputs which will initiate transitions between states (see Figure 4). The plate speed and direction are adjustable, controlled by a potentiometer. The push button will act as a pause switch for when the device is running. The State Transition Diagram below illustrates how the potentiometer and push button initiate new phases of the system. Refer to Appendix C for the code designed to execute the event-driven programming of the state diagram.

The initial design included the ability for the user to adjust the light sensitivity and brightness to mimic the cycles of the sun throughout the seasons, however, research indicated that the duration of light exposure can be increased or decreased to compensate for light intensity [2]. This became rather a task to add onto the prototype in the future and our group decided for the sake of simplicity to minimize the states of the lights in the enclosure.

In order to protect the electronics in the system, we designed for them to be enclosed by wood supporting the disc with the plant on it.

Integrated Physical Device

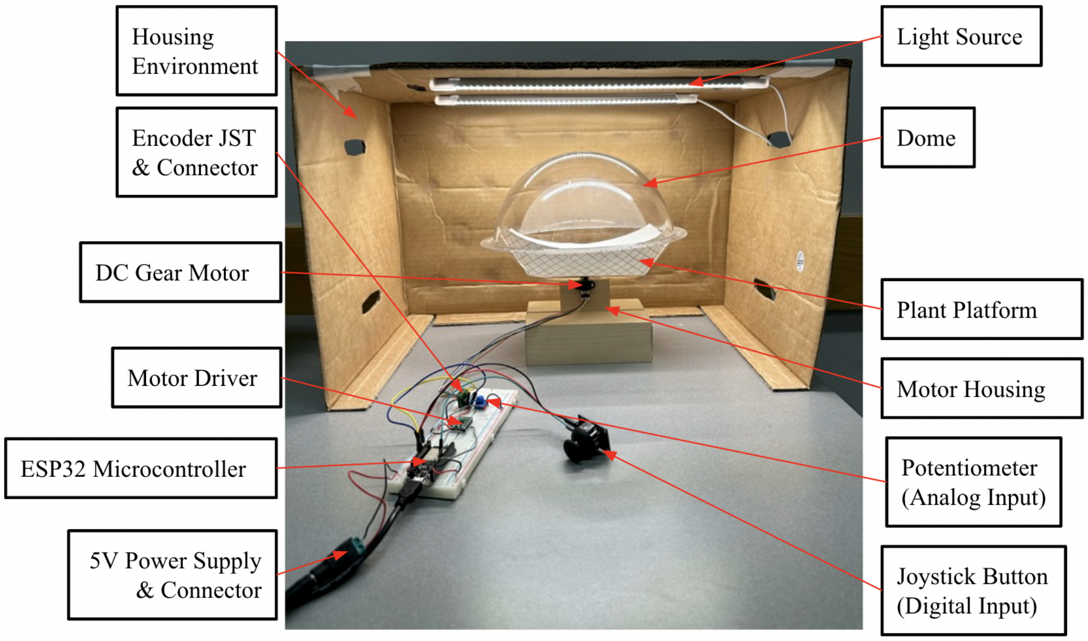


Figure 2. Prototype with the components labeled

Function-Critical Decisions

The initial design was to employ a wheel, connected to a motor to rotate the plate (Figure 1). We instead chose to attach the plate directly to the shaft of the motor in order to have less constraints to our motor. Also, due to the timing of our parts with constant iterations, we were not able to get the correct motor we wanted to support the weight of the wheel we had originally got. We then switched to the Polulu lab motor. But since it's a much smaller motor with a torque value of 0.23kg-cm, we needed a much lighter platform which resulted in us using a paper boat plate with the motor shaft attached vertically to it (Figure 2).

The following calculations are the justification for our selection of the motor to best suit our project goals. We over-approximated our system experience to allow for tolerance in the design.

Project Specs: $\omega_i = 0\, rpm, \omega_f = 20\, rpm, t = 1\, second$ (motor rotates from rest to 20 rpm in 1s)

Disc Specs: $d = 12\, in -> r = 6\, in = 0.1524\, m, m = 15.2\, oz = 0.431\, kg$

Torque Calculations:

Moment of Inertia: $I = \frac{1}{2}mr^2 = \frac{1}{2}(0.431)(0.1524)^2 = 0.005\, kg * m^2$

Angular Acceleration: $\alpha = \frac{\Delta\omega}{\Delta t} = \frac{20-0}{1-0} = \frac{\frac{2\pi}{3}-0}{1} = \frac{2\pi}{3}\, rad/s$

Torque: $\tau = I\alpha = (0.005)(\frac{2\pi}{3}) = 0.0105\, N * m$

Frictional Torque: $\tau_{friction} = mg\mu r = (0.431)(9.81)(0.6)(0.1524) = 0.387\, N * m$

Total Torque Required = Torque + Frictional Torque = 0.0105 + 0.387 = 0.3975 N*m

The motor must produce a torque spec of at least 0.3975 N*m.

Circuit Diagram

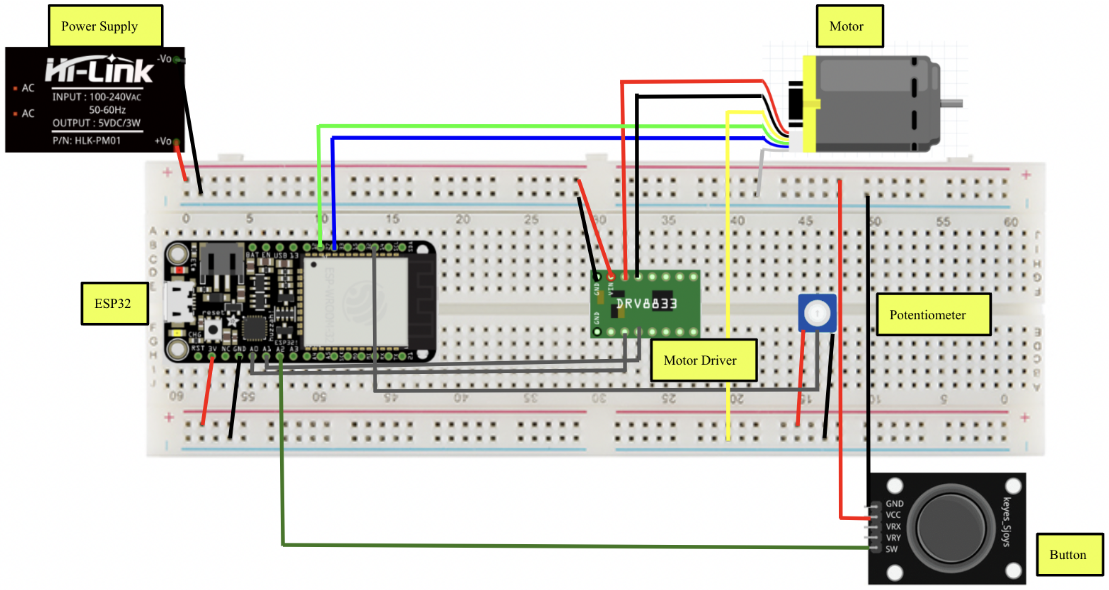


Figure 3. Circuit diagram with labeled components

State-Transition Diagram

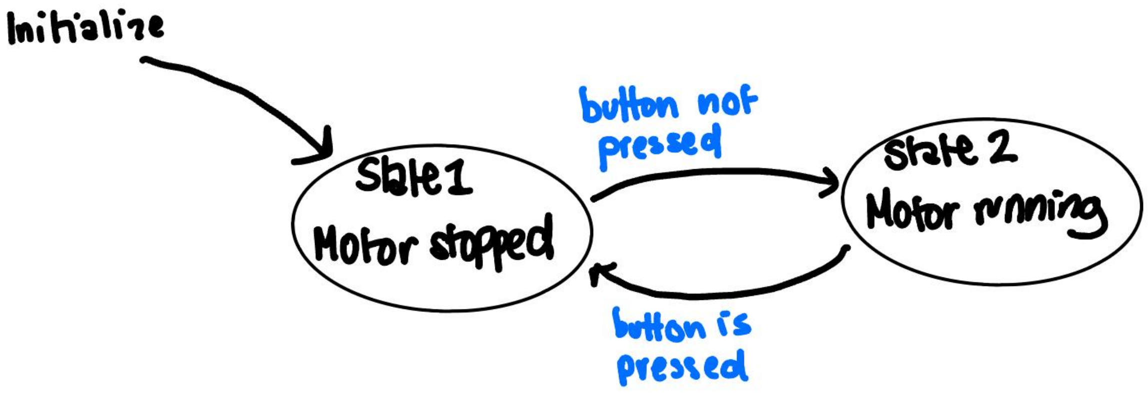


Figure 4. State Transition Diagram for the event-driven programming of the "FAFA: Food Accessibility for All" project

Reflection

Our group viewed communication and consistency as fundamental components of the success of the project. We were proactive about project deadlines and therefore had time to troubleshoot. Our group should have utilized the teaching staff as a resource throughout the project for help on the challenges we faced throughout the semester.

References

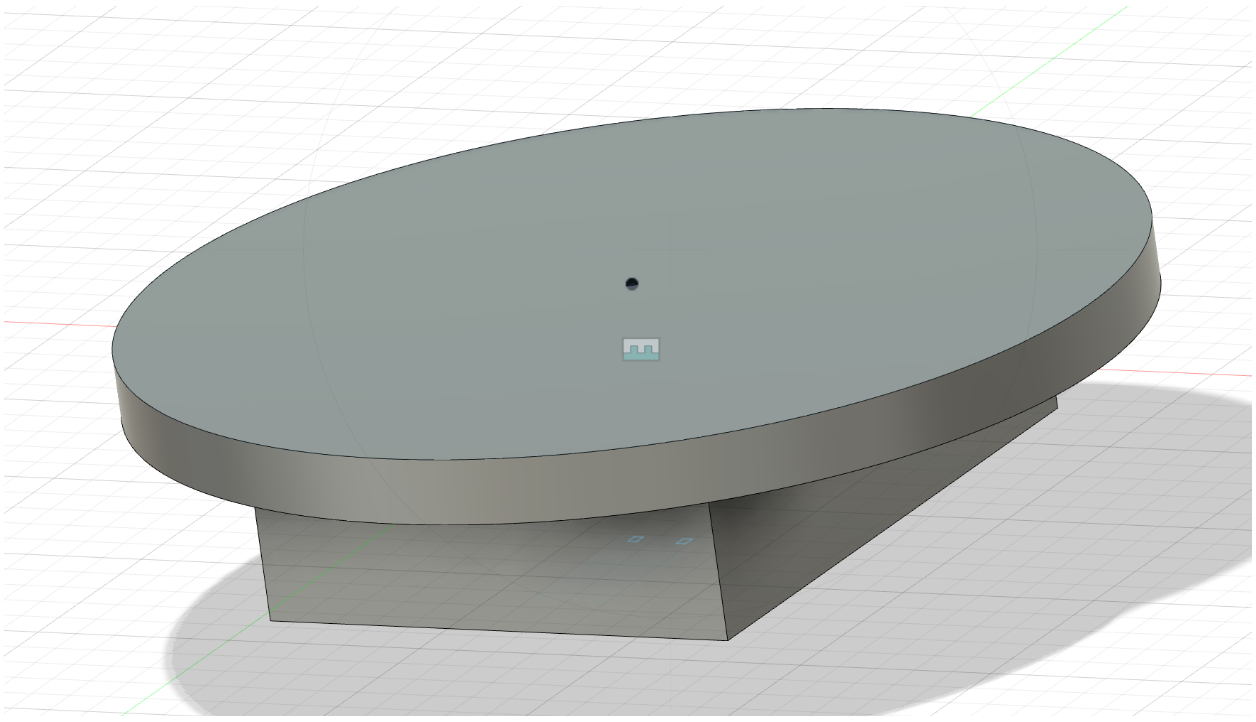
[1] “Climate Change and the Future of Food.” *Unfoundation.Org*, 13 Oct. 2023, unfoundation.org/blog/post/climate-change-and-the-future-of-food/#:~:text=Food%20security%20%E2%80%94%20the%20reliable%20access,of%20food%20around%20the%20world.
[2] “Light, Temperature and Humidity - Ornamental Production Ornamental Production.” *Aggie Horticulture®*, aggie-horticulture.tamu.edu/ornamental/a-reference-guide-to-plant-care-handling-and-merchandising/light-temperature-and-humidity/#:~:text=Increasing%20the%20time%20(duration)%20plants,food%20to%20survive%20and%20grow. Accessed 12 Dec. 2023.
[3] *Sun’s Path Patterns*, YouTube, 20 May 2020, https://m.youtube.com/watch?v=OoXOZl1A7xs. Accessed 12 Dec. 2023.

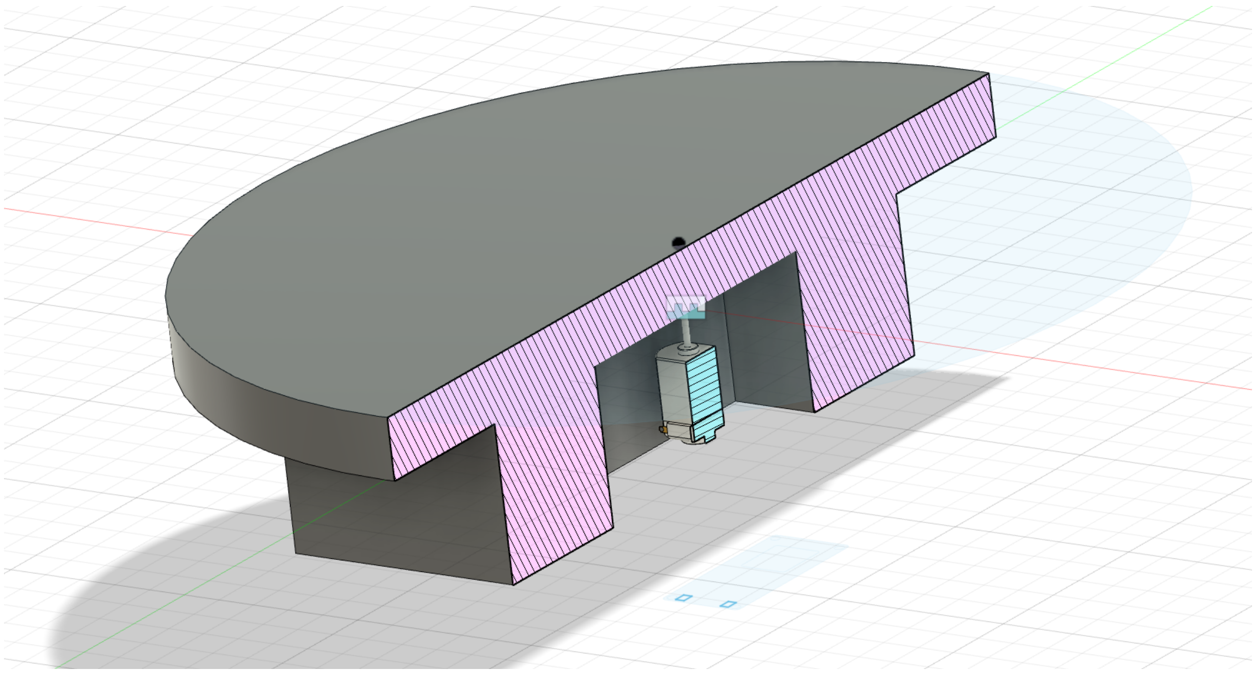
Appendices

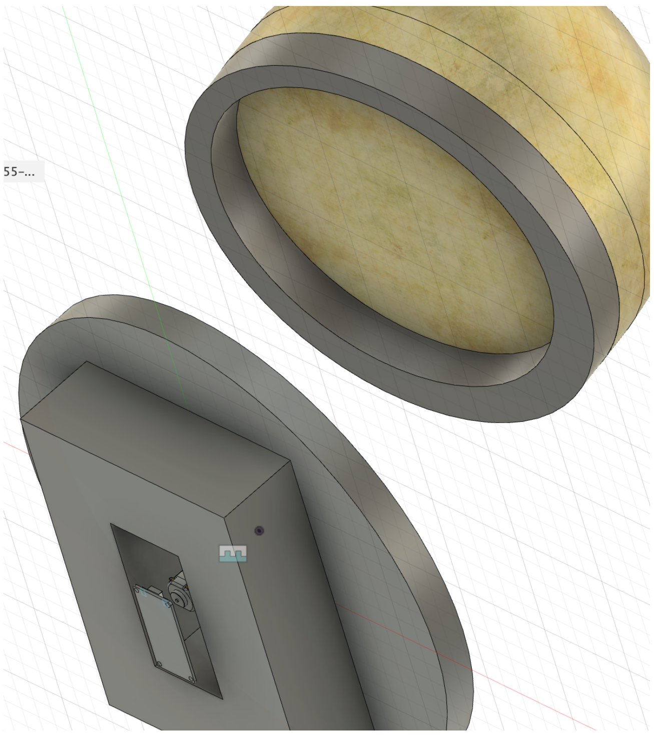
Appendix A: Bill of Materials

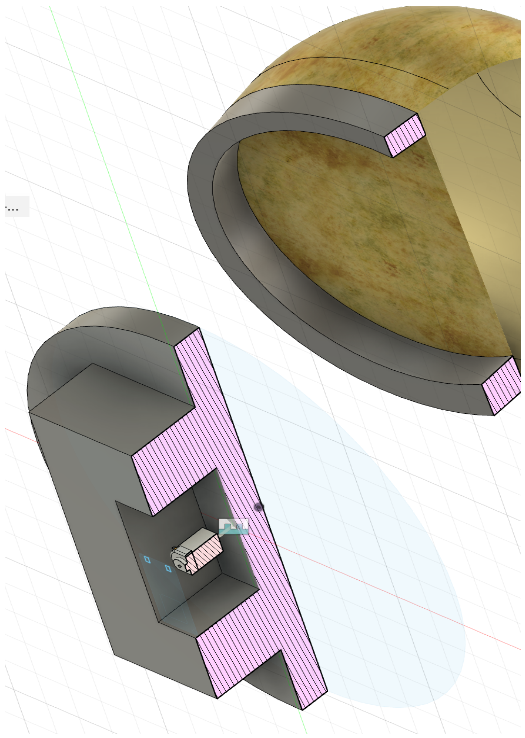
Component	**Part**	**Amount**	**Price**	**Source**	**Link**
Wire Housing	Basswood Carving Blocks 6x2x2 Inch	3x	$17.98	Amazon	Wooden Blocks
Rotating Disk	8 Inch Heavy Duty Rotating Swivel With Steel Ball Bearings Stand	2x	$12.99	Amazon	Lazy Susan Disc
Light Source	LED Grow Light Strips for Indoor Plants	2x	$14.99	Amazon	Sunlight LED Strips
Motor	75:1 Micro Metal Gearmotor HP 6V with Extended Motor Shaft	2x	Free	Polulu	Polulu Motor
Dome Housing	9 Inch Acrylic Dome Cover	1x	$30.79	Amazon	Acrylic Dome Housing
Overall Housing	Amazon Cardboard Box	1x	Free	Home	
Electrical Components	Microkit Components (refer to circuit diagram for details)		Free	Lab Kit	https://microkit.berkeley.edu/
Total			**$76.75**		

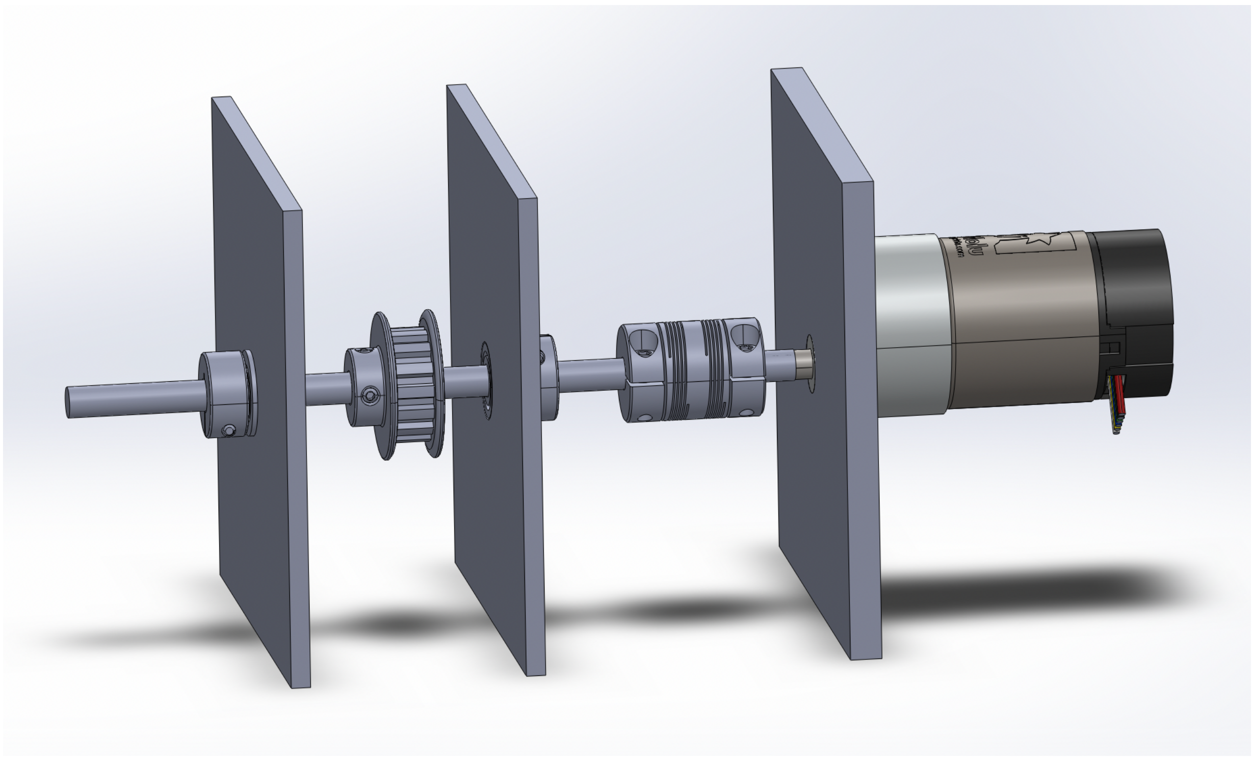
Appendix B: CAD Images






55-...





Appendix C: Relevant Code

```
#include <ESP32Encoder.h>

#define BIN_1 26
#define BIN_2 25
#define LED_PIN 13
#define POT 14
#define JOYSTICK_BUTTON 34  // Joystick button pin

ESP32Encoder encoder;

int omegaSpeed = 0;
int omegaDes = 0;
int omegaMax = 18;
int D = 0;
int potReading = 0;
bool buttonPressed = false; // Stores the state of the joystick button
unsigned long lastDebounceTime = 0;  // Last time the joystick button was toggled
const unsigned long debounceDelay = 50;  // Debounce time in milliseconds

volatile int count = 0;
volatile bool interruptCounter = false;
volatile bool deltaT = false;
int totalInterrupts = 0;
hw_timer_t *timer0 = NULL;
hw_timer_t *timer1 = NULL;
portMUX_TYPE timerMux0 = portMUX_INITIALIZER_UNLOCKED;
portMUX_TYPE timerMux1 = portMUX_INITIALIZER_UNLOCKED;

const int freq = 5000;
const int ledChannel_1 = 1;
const int ledChannel_2 = 2;
const int resolution = 8;
const int MAX_PWM_VOLTAGE = 255;
const int NOM_PWM_VOLTAGE = 150;

unsigned long startTime = 0;
bool motorDirectionClockwise = true;
bool motorRunning = false;  // Motor running state

enum MotorState {
 MOTOR_STOPPED,
 MOTOR_RUNNING,
 FOLLOWING_POT
};

MotorState state = MOTOR_STOPPED;

// Function prototypes
void adjustMotorSpeedBasedOnPot();
void handleJoystickPress();  // Prototype for the ISR

void IRAM_ATTR onTime0() {
 portENTER_CRITICAL_ISR(&timerMux0);
 interruptCounter = true;
 portEXIT_CRITICAL_ISR(&timerMux0);
}

void IRAM_ATTR onTime1() {
 portENTER_CRITICAL_ISR(&timerMux1);
 count = encoder.getCount();
 encoder.clearCount();
 deltaT = true;
 portEXIT_CRITICAL_ISR(&timerMux1);
}

void setup() {
 pinMode(POT, INPUT);
 pinMode(LED_PIN, OUTPUT);
 digitalWrite(LED_PIN, LOW);

 Serial.begin(115200);
 ESP32Encoder::useInternalWeakPullResistors = UP;
 encoder.attachHalfQuad(33, 27);
 encoder.setCount(0);

 ledcSetup(ledChannel_1, freq, resolution);
 ledcSetup(ledChannel_2, freq, resolution);

 ledcAttachPin(BIN_1, ledChannel_1);
 ledcAttachPin(BIN_2, ledChannel_2);

 timer0 = timerBegin(0, 80, true);
 timerAttachInterrupt(timer0, &onTime0, true);
 timerAlarmWrite(timer0, 10000, true); // 10 ms, autoreload true

 timer1 = timerBegin(1, 80, true);
 timerAttachInterrupt(timer1, &onTime1, true);
 timerAlarmWrite(timer1, 10000, true); // 10 ms, autoreload true

 timerAlarmEnable(timer0);
 timerAlarmEnable(timer1);

 pinMode(JOYSTICK_BUTTON, INPUT_PULLUP); // Set the joystick button as an input with
pull-up
 attachInterrupt(digitalPinToInterrupt(JOYSTICK_BUTTON), handleJoystickPress,
FALLING);

 startTime = millis();
}

void loop() {
 switch (state) {
   case MOTOR_STOPPED:
     if (motorRunning) {
       state = MOTOR_RUNNING;
     }
     stopMotor();
     break;

   case MOTOR_RUNNING:
     if (!motorRunning) {
       state = MOTOR_STOPPED;
     }
     handleMotorRunning();
     break;

   case FOLLOWING_POT:
     adjustMotorSpeedBasedOnPot();
     if (!motorRunning) {
       state = MOTOR_STOPPED;
     }
     break;

   default:
     Serial.println("Error: Unknown State");
     state = MOTOR_STOPPED;
     break;
 }
}

void IRAM_ATTR handleJoystickPress() {
 if ((millis() - lastDebounceTime) > debounceDelay) {
   motorRunning = !motorRunning;
   buttonPressed = true;
   state = FOLLOWING_POT;
   lastDebounceTime = millis();
 }
}

void stopMotor() {
 ledcWrite(ledChannel_1, 0);
 ledcWrite(ledChannel_2, 0);
}

void handleMotorRunning() {
 if (deltaT) {
   portENTER_CRITICAL(&timerMux1);
   deltaT = false;
   portEXIT_CRITICAL(&timerMux1);

   omegaSpeed = count;
   potReading = analogRead(POT);
   omegaDes = map(potReading, 0, 4095, -omegaMax, omegaMax);

   motorDirectionClockwise = (omegaDes >= 0);
   D = map(potReading, 0, 4095, -NOM_PWM_VOLTAGE, NOM_PWM_VOLTAGE);

   if (D < 0) {
     D = -D;
   }
   if (D > MAX_PWM_VOLTAGE) {
     D = MAX_PWM_VOLTAGE;
   }

   if (motorDirectionClockwise) {
     ledcWrite(ledChannel_1, D);
```

```
      ledcWrite(ledChannel_2, 0);
    } else {
      ledcWrite(ledChannel_1, 0);
      ledcWrite(ledChannel_2, D);
    }

    Serial.print("Joystick Button State: ");
    Serial.println(buttonPressed ? "Pressed" : "Released");
    Serial.print(" Motor Running: ");
    Serial.println(motorRunning ? "Yes" : "No");
    Serial.print(" Desired Speed: ");
    Serial.print(omegaDes);
    Serial.print(" Actual Speed: ");
    Serial.print(omegaSpeed);
    Serial.print(" Motor Direction: ");
    Serial.println(motorDirectionClockwise ? "Clockwise" : "Counterclockwise");
  }
}

void adjustMotorSpeedBasedOnPot() {
  potReading = analogRead(POT);
  omegaDes = map(potReading, 0, 4095, -omegaMax, omegaMax);

  motorDirectionClockwise = (omegaDes >= 0);
  D = map(potReading, 0, 4095, -NOM_PWM_VOLTAGE, NOM_PWM_VOLTAGE);

  if (D < 0) {
    D = -D;
  }
  if (D > MAX_PWM_VOLTAGE) {
    D = MAX_PWM_VOLTAGE;
  }

  if (motorDirectionClockwise) {
    ledcWrite(ledChannel_1, D);
    ledcWrite(ledChannel_2, 0);
  } else {
    ledcWrite(ledChannel_1, 0);
    ledcWrite(ledChannel_2, D);
  }
}
```

Appendix D: Supplemental Material

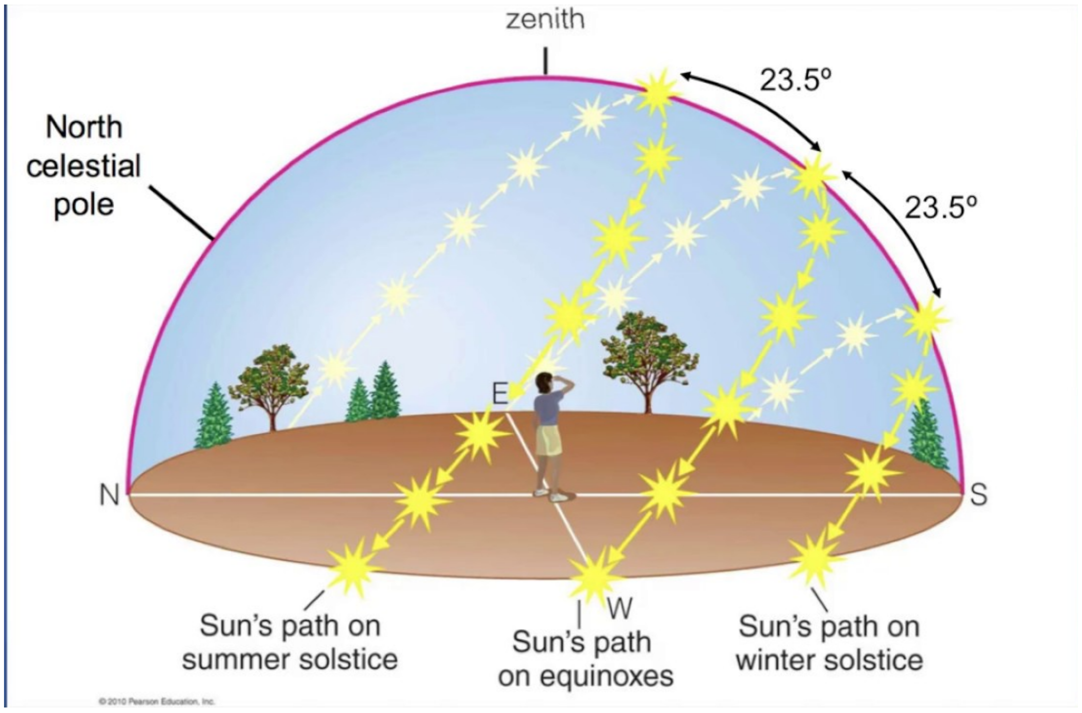

zenith
North celestial pole
23.5°
23.5°
E
N
S
W
Sun's path on summer solstice
Sun's path on equinoxes
Sun's path on winter solstice
© 2010 Pearson Education, Inc.

Figure D-1. The orbital path of the sun [3]