Couchopy Report

Team 8

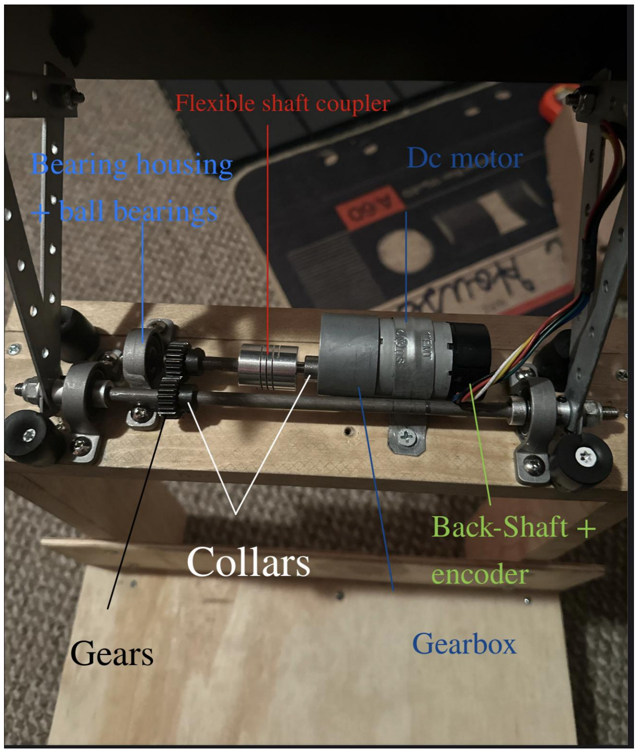
Opportunity

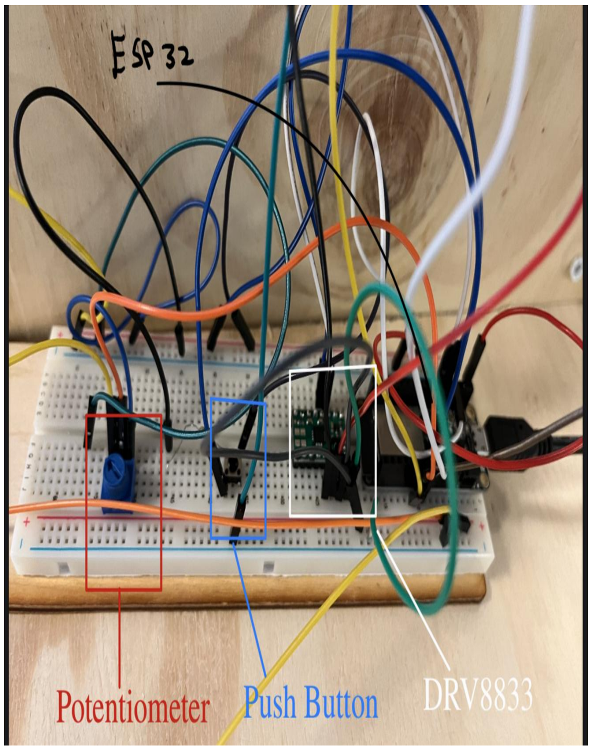
The main opportunity and focus we had for the project was to create a device that can help protect people from an earthquake. We kept at the idea and the project does a good job of shielding incoming debris from an earthquake while people sit on a couch. The project also works great as a recreational use for shielding sun rays on a hot day. The original earthquake opportunity we thought of could also work for a recreational aspect opportunity.

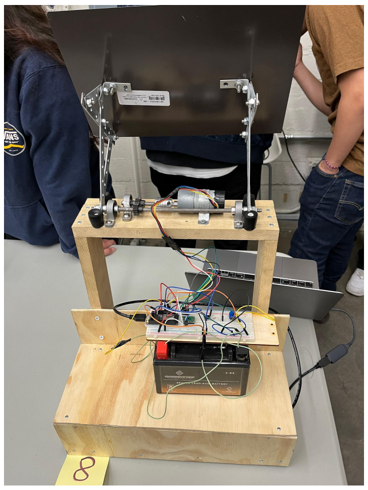
Strategy and functionality

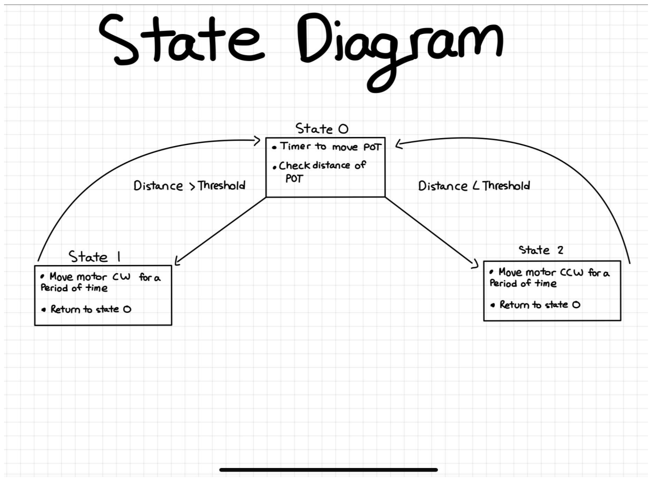
Our updated strategy still had the same approach. We wanted a mechanical system where gears were moved by a motor and that would control the movement of the shielding mechanism. We had the button control when the motor would spin, while the potentiometer controls the direction of the motor (counterclockwise or clockwise). We used one motor which was adjacent to another gear on a shaft. On that same shaft was where the two joints that held the shielding. As the motor spins it would cause the shaft spin which would in turn cause the joints to move. The movement of our joints was that of a crank-rocker, where it would go forward a certain amount then back a certain amount as the shaft spinned a certain amount of time. Initially we thought of using two motors to spin the gears, but we decided that it would be a lot of code to write and it would be hard to control two motors at the same time. The initial desired speed was 255 based on the lab kit motor and we wanted it to move about 3⁄4 of an inch per second. But the lab kit motor did not deliver enough torque to spin the gear. We then used a 12V motor and achieved 3⁄4 of an inch per second at a speed of 195. We also initially wanted the potentiometer to control the switching of between different states but achieved a better solution of having the potentiometer switch directions of the motor. There is practically no delay when using the potentiometer unlike a button that has bounce.

Photos









Calculation

Gear Ratio : Number of input teeth/ number of output teeth = 1

Force on motor = mg(shaft) + mg(shaft) = .5lb + .5lb = 4.48N

Moment on motor = ½ * .907* 3^2(m/s)*.102 = 4.16Nm

Weight of entire system:

Couch = 3lbs

Shield = 2lbs

Gear system = 1.5lbs

Breadboard = .25lb

Total weight = 6.75lbs

Reflection

The project went well and all of our desired functions went practically as a plan. Minor adjustments had to be made as we built the mechanical device like adjusting the speed of the motor tightening screws to bear shaft. What worked well for us was communicating and applying what we learned throughout the semester to our project. Coding wise understanding the lab code and testing it out first helped plan how we want each component to function and integrate the components together. Understanding the code for the lab helps make implementing it to our project much easier. Action we could have done better was to test each component separately and debugging code at a time. We ran into issues where we didn’t know if it was a hardware or software issue. For example, we didn't know if the button was broken or if our code was not working for the button to spin the motor. It ends up being the button.

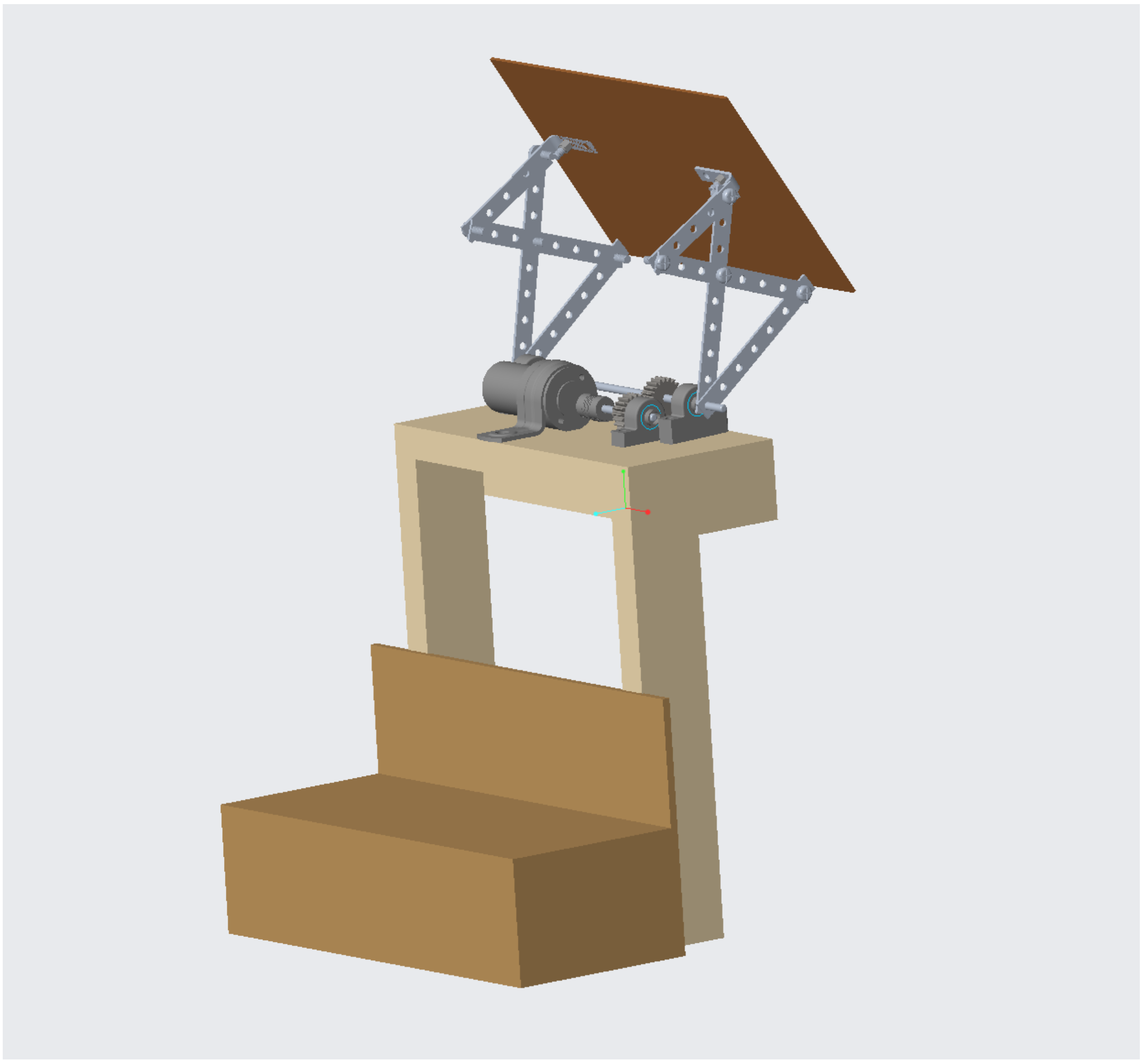
BOM

Name	**Quantity**	**Cost**	**links**
Steel Rod 1/4in X 48in	1	6.38$	link
Plywood 2ft x 4ft	1	17.31$	link
Pine Softwood Board	1	8.63$	link
DC motor	1		
Wood Screws #4 3/4in	2	2.76$	link
Wood Screws #6 3/4in	1	1.38$	link
Wood Screws #6 2in	3	4.14$	link
Hardwood Squares 3/4in x 3/4in x 3ft	1	3.52$	link
J-B Weld	1	7.48$	link
Standard Fitting 2-Hole Electrical Metallic Tube (EMT) Straps	1	2.34$	link
Command Strips	1	9.93$	link
#10-24 Zinc Plated Nylon Lock Nut	4	5.52$	link
#10-24 x 1/2 in. Combo Round Head Stainless Steel Machine Screw	2	2.76$	link
#10 Zinc Flat Washer	1	1.38$	link
1-1/2 in. Galvanized	1	6.27$	link

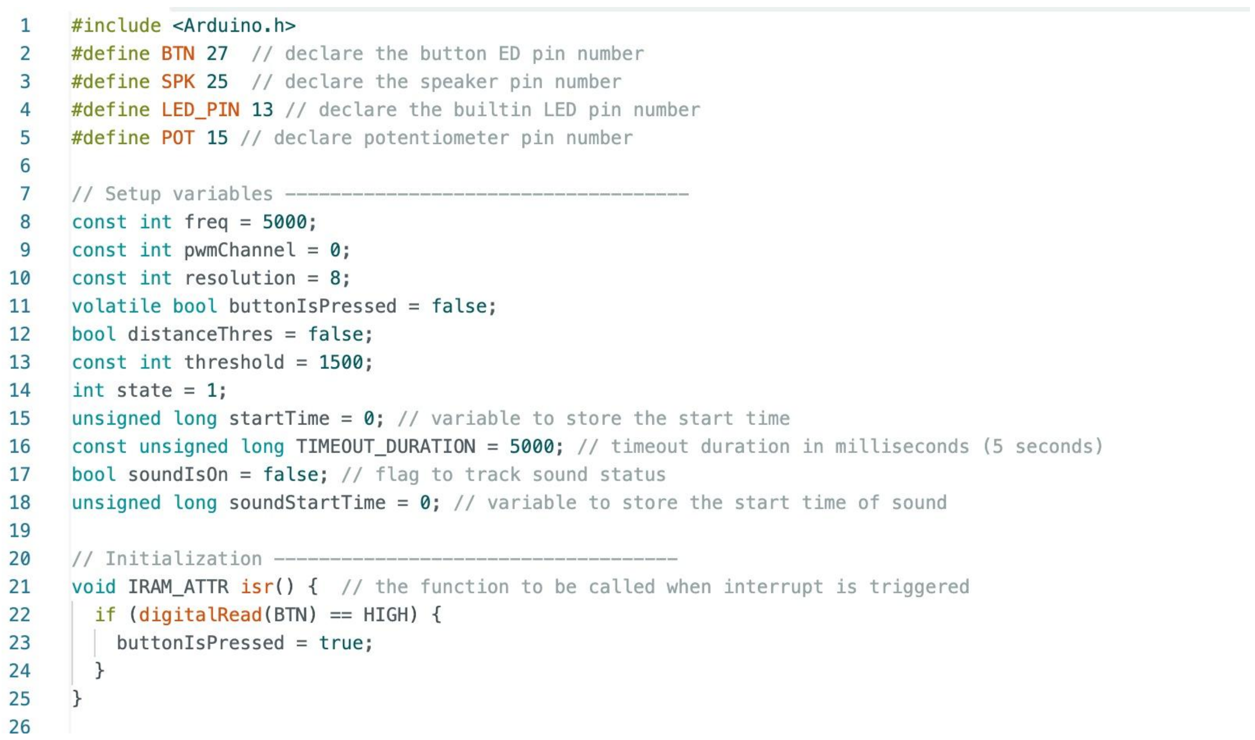
Corner Brace			
5/8 in. x 18 in. 14-Gauge Pipe Support Hyco Bar	3	5.22$	link
8 in. x 12 in. Galvanized Steel Flashing Shingle in Dark Brown	2	3.96$	link
1-1/4 in. x 1 in. Black Rubber Stopper	1	2.75$	link
6.35x6.35mm CNC Flexible Coupling Motor Shaft Coupler 1/4" to 1/4" Shaft	1	8.99$	link
AZSSMUK 1/4" Bore Solid Steel Style Zinc Plated Set Screw Shaft Collars	1	7.99$	link
Apex RC Products 32 Pitch 32P 18T 19T 20T 21T 5mm Hole Pinion Gear Set #9738	2	24.99$	link
HiPicco Pillow Block Bearing, 4Pcs KP08 Flange Mounted Pillow Bearings - Bore 8mm/0.31" ID Self Alignment Zinc Alloy Plummer Block Bearing for Diameter 8mm Linear Shaft Rod	1	8.99$	link

(4-Pack) Aluminum Alloy 8mm Bore Shaft Collars Screw Style, Bore Shaft Collars with 8mm Bore Size, 14mm Outer Diameter, and 8mm Width - Suitable for CNC Machine Tools	1	9.99$	link
Chrome Battery YTX9-BS Maintenance Free Replacement Battery for ATV, Motorcycle, and Scooter: 12 Volts, 9 Amps, 8Ah, Nut and Bolt (T3) Terminal	1	28.63$	link
		Total: 181.31$ w/o Tax	

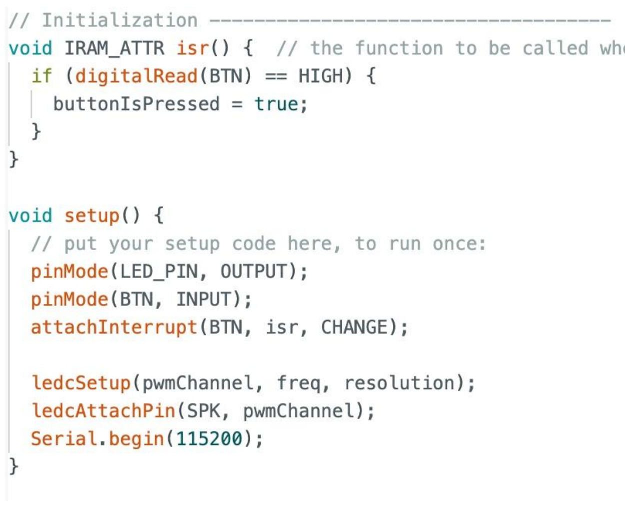
CAD



Code

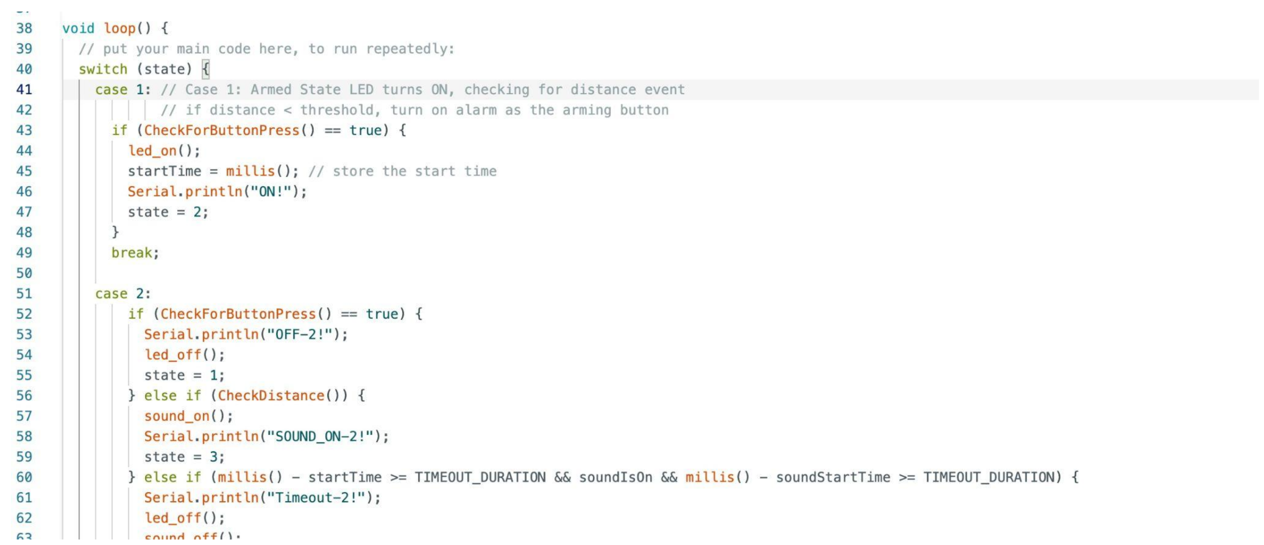
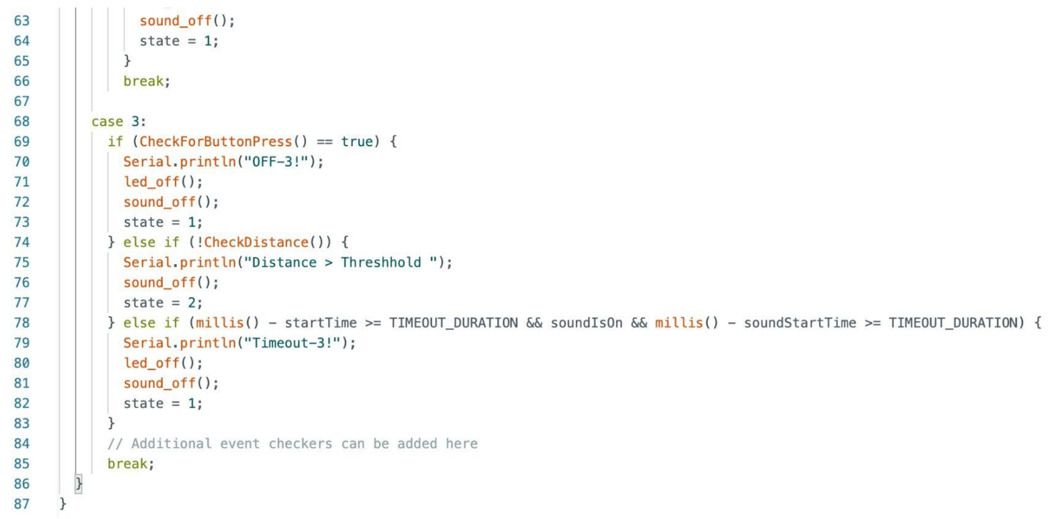


```cpp
#include <Arduino.h>
#define BTN 27  // declare the button ED pin number
#define SPK 25  // declare the speaker pin number
#define LED_PIN 13 // declare the builtin LED pin number
#define POT 15 // declare potentiometer pin number

// Setup variables ------------------------------------
const int freq = 5000;
const int pwmChannel = 0;
const int resolution = 8;
volatile bool buttonIsPressed = false;
bool distanceThres = false;
const int threshold = 1500;
int state = 1;
unsigned long startTime = 0; // variable to store the start time
const unsigned long TIMEOUT_DURATION = 5000; // timeout duration in milliseconds (5 seconds)
bool soundIsOn = false; // flag to track sound status
unsigned long soundStartTime = 0; // variable to store the start time of sound

// Initialization ------------------------------------
void IRAM_ATTR isr() {  // the function to be called when interrupt is triggered
  if (digitalRead(BTN) == HIGH) {
    buttonIsPressed = true;
  }
}

```

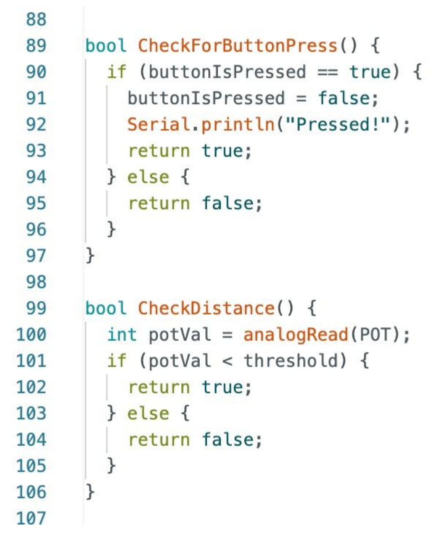


```cpp
// Initialization ------------------------------------
void IRAM_ATTR isr() {  // the function to be called when interrupt is triggered
  if (digitalRead(BTN) == HIGH) {
    buttonIsPressed = true;
  }
}
```

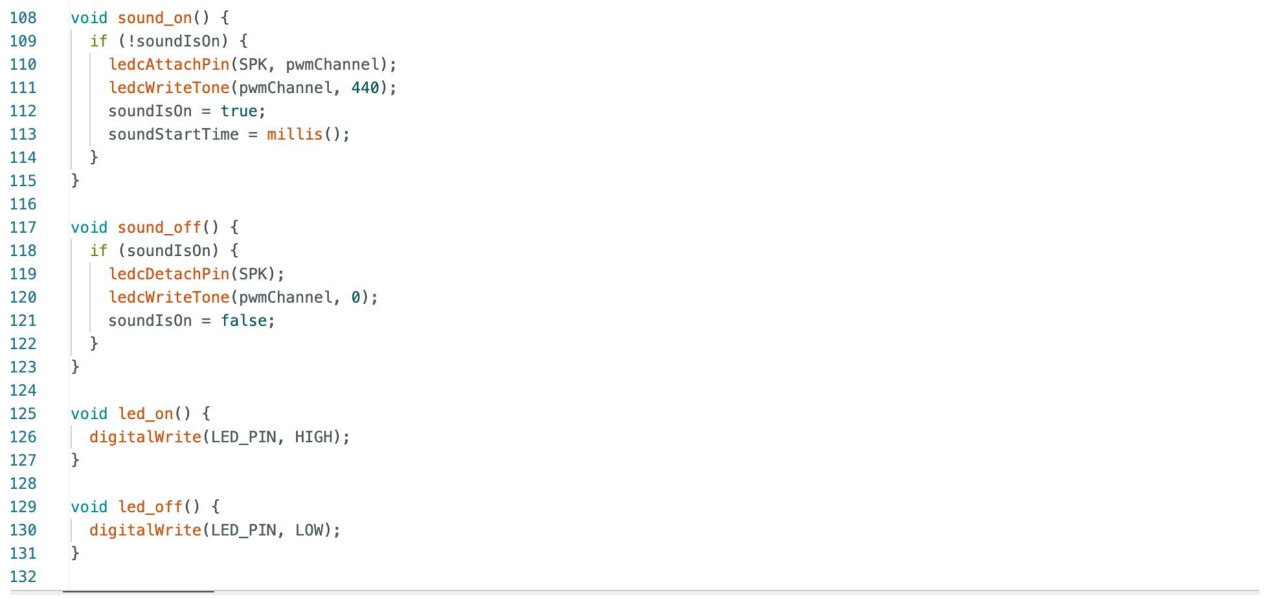
```cpp
void setup() {
  // put your setup code here, to run once:
  pinMode(LED_PIN, OUTPUT);
  pinMode(BTN, INPUT);
  attachInterrupt(BTN, isr, CHANGE);

  ledcSetup(pwmChannel, freq, resolution);
  ledcAttachPin(SPK, pwmChannel);
  Serial.begin(115200);
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  switch (state) {
    case 1: // Case 1: Armed State LED turns ON, checking for distance event
            // if distance < threshold, turn on alarm as the arming button
      if (CheckForButtonPress() == true) {
        led_on();
        startTime = millis(); // store the start time
        Serial.println("ON!");
        state = 2;
      }
      break;

    case 2:
        if (CheckForButtonPress() == true) {
          Serial.println("OFF-2!");
          led_off();
          state = 1;
        } else if (CheckDistance()) {
          sound_on();
          Serial.println("SOUND_ON-2!");
          state = 3;
        } else if (millis() - startTime >= TIMEOUT_DURATION && soundIsOn && millis() - soundStartTime >= TIMEOUT_DURATION) {
          Serial.println("Timeout-2!");
          led_off();
          sound_off();
          state = 1;
        }
        break;

    case 3:
      if (CheckForButtonPress() == true) {
        Serial.println("OFF-3!");
        led_off();
        sound_off();
        state = 1;
      } else if (!CheckDistance()) {
        Serial.println("Distance > Threshhold ");
        sound_off();
        state = 2;
      } else if (millis() - startTime >= TIMEOUT_DURATION && soundIsOn && millis() - soundStartTime >= TIMEOUT_DURATION) {
        Serial.println("Timeout-3!");
        led_off();
        sound_off();
        state = 1;
      }
      // Additional event checkers can be added here
      break;
  }
}
```



```cpp

bool CheckForButtonPress() {
  if (buttonIsPressed == true) {
    buttonIsPressed = false;
    Serial.println("Pressed!");
    return true;
  } else {
    return false;
  }
}

bool CheckDistance() {
  int potVal = analogRead(POT);
  if (potVal < threshold) {
    return true;
  } else {
    return false;
  }
}

```



```cpp
void sound_on() {
  if (!soundIsOn) {
    ledcAttachPin(SPK, pwmChannel);
    ledcWriteTone(pwmChannel, 440);
    soundIsOn = true;
    soundStartTime = millis();
  }
}

void sound_off() {
  if (soundIsOn) {
    ledcDetachPin(SPK);
    ledcWriteTone(pwmChannel, 0);
    soundIsOn = false;
  }
}

void led_on() {
  digitalWrite(LED_PIN, HIGH);
}

void led_off() {
  digitalWrite(LED_PIN, LOW);
}

```