Sit-Fit

Smarter sitting for a Healthier You

Team Members:

Team Lead: Sreehari Nandanan

Member 2 :Kurias Tenson

Member :Aqil Sait

Introduction

As a student, We've seen how sitting for long periods affects concentration and energy levels, especially during study sessions. This experience inspired us to create the Sit-Fit chair, designed to promote healthier habits for those who spend their days at a desk.

Research shows that prolonged sitting can lead to discomfort and various health issues, which made us realize the importance of incorporating movement into our daily routines. We wanted to develop a chair that reminds users to stand up and do simple exercises, helping to break the cycle of inactivity.

By combining ergonomic design with reminders for movement, Sit-Fit aims to support better health and productivity. We believe that encouraging regular breaks and physical activity can make a significant difference in how we feel and perform at work or school. Ultimately, We hope this chair inspires a more active and balanced lifestyle

Working

Once a person sits on the Sit-Fit chair a timer activates and after a particular time period it warns the user with an alarm. In order to turn off the alarm the user needs to stand up and need to go near a distance sensor which is fixed on the back of the chair. Even malpractices like showing the hand infront of the sensor can be prevented with the help of our optimized code.

Components Required

1.Airduino UNO

2.Buzzer

3.Speaker

4.Ultra Sonic Motion Sensor

5.Breadboard

6.Push Button

7.BC547

8.330 Ohm resistor

9.Jumper Wire

For Software:

Arduino ide (embedded c programming)

Installation

Mannually by hand

Run

Displays out put in the serial monitor such as pin out put buzzer status

Future Updates

1.Posture Monitoring

Pressure Sensors: These can be placed in the seat and backrest to detect how weight is distributed, which can indicate slouching or leaning.

Inertial Measurement Units (IMUs): Small IMUs (like accelerometers and gyroscopes) can detect subtle body movements and positioning to track if the user's posture shifts.

Recliners: To set up recliner seats which automatically comes back to normal position after a particular time

2. Heart Rate and Stress Monitoring

Heart Rate Sensors: Optical sensors (similar to those on smartwatches) can monitor the user’s heart rate, which can be an indicator of stress or relaxation.

Electrocardiogram (ECG) Sensors: Embedded ECG sensors in armrests can measure heart rate more accurately if the user’s hands are in contact

3. Breathing Rate Analysis

Respiratory Rate Sensors: Sensors in the backrest can detect slight changes in pressure as the user breathes. The system can measure breath rate and variability, which can be an indicator of stress.

Infrared Sensors: IR sensors could be used to monitor small expansions and contractions in the chest or abdomen area for breath rate.

4. Body Temperature Tracking

Thermocouples or IR Temperature Sensors: These can measure the user’s body temperature to indicate comfort levels or detect signs of potential overheating or discomfort.

5. Reminders for Movement and Stretching

Vibration Motors: To provide gentle nudges or reminders, use vibration motors embedded in the seat or backrest to remind users to stretch, change posture, or stand up.

App Notifications: Connect Sit-Fit to an app or computer to send reminders for movement or exercises after specific sitting intervals.

6. Fatigue Detection

Pressure Redistribution Patterns: If the user’s sitting pressure doesn’t shift over time, it can indicate fatigue. Sensors could prompt the user to shift to prevent strain on muscles and improve blood flow.

7. Environment Sensing

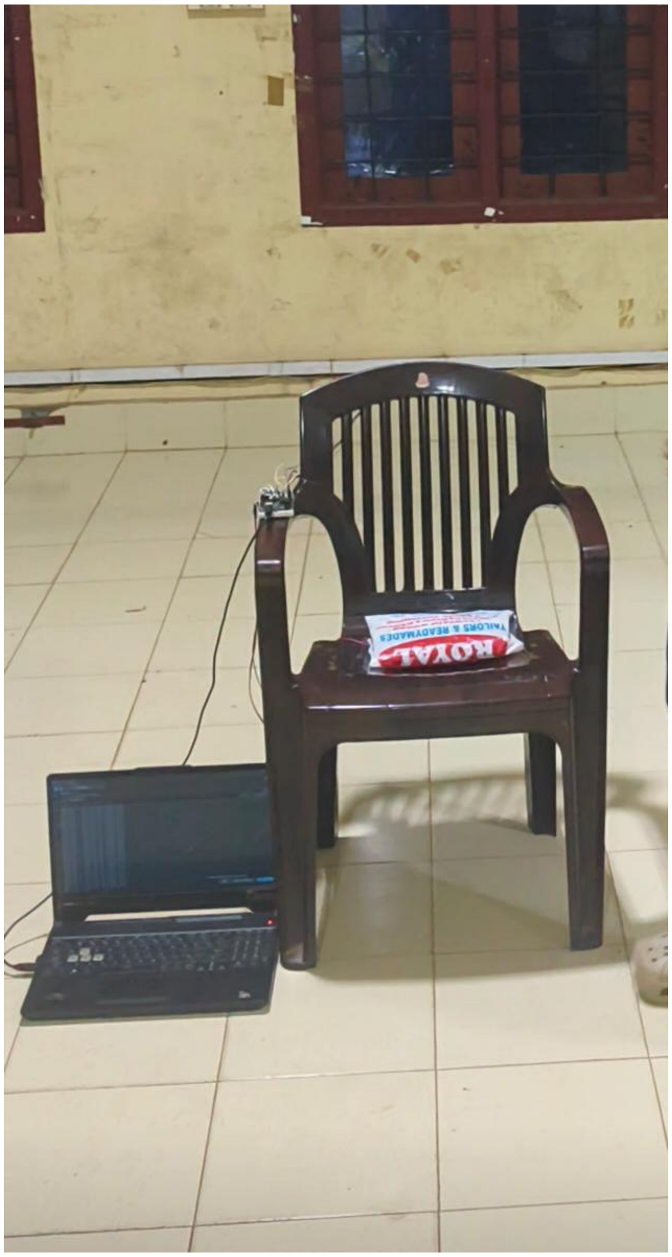
Ambient Light and Temperature Sensors: Measure the surrounding environment to suggest adjustments for optimal working conditions, like adjusting room lighting or seating angle for eye health.

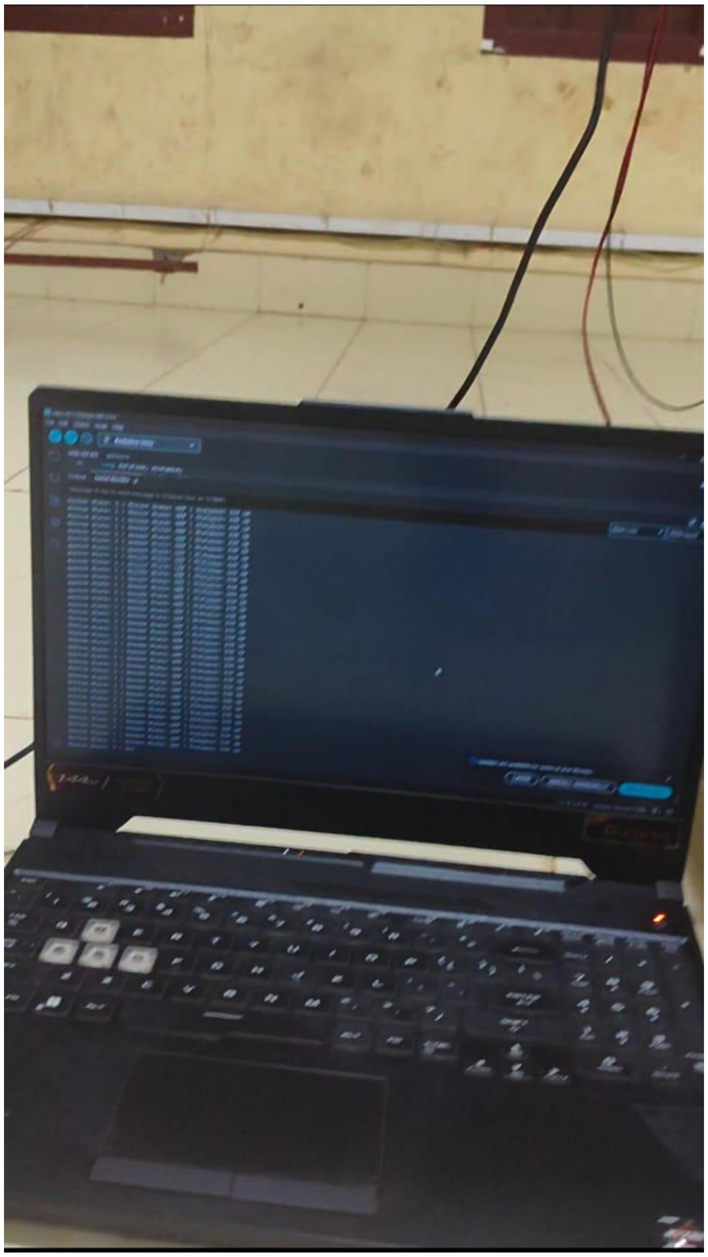
8.Integration and User Experience

Data Collection: Collect health data over time, which can be analyzed to offer personalized suggestions to improve seated habits.

Real-Time Feedback: Using a connected app or display, offer real-time updates on the user’s posture, sitting time, and health metrics, encouraging healthy adjustments.

Screenshots:

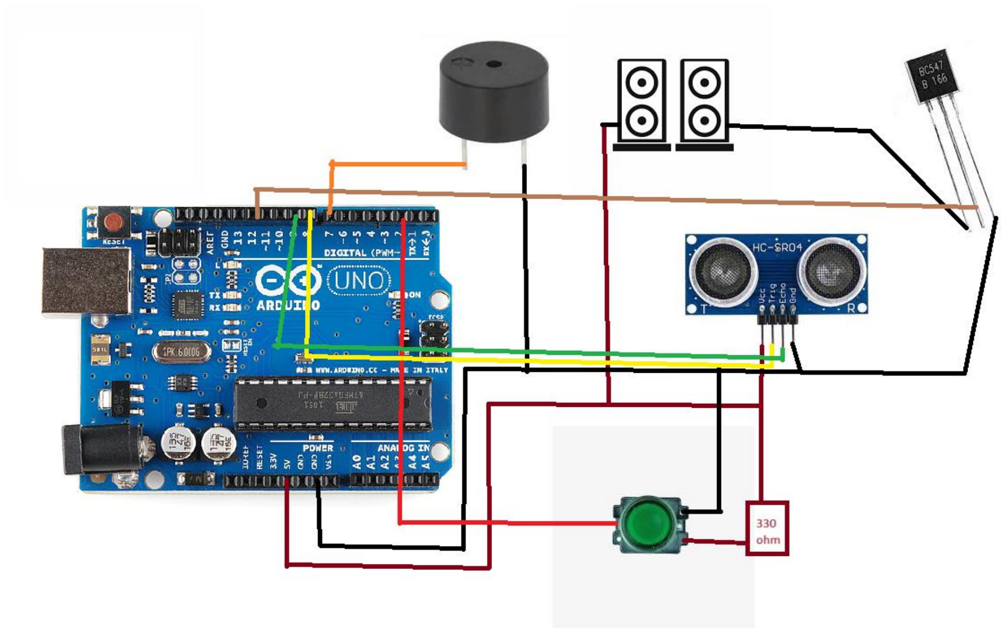

ROYAL
TAILORS & READYMADES



Video:

https://github.com/sreehari-nandanan/Sit-Fit/blob/main/WhatsApp%20Video%202024-11-03%20at%2006.09.24_24530b46.mp4

Circuit:



Team Contributions

Sreehari nandanan: coding

Kurias tenson: circutiry

Aqil Sait: documentation

Code:

```
#include "pitches.h"
int melody[] = {
  NOTE_C7 ,NOTE_D7,NOTE_C7,NOTE_D7,
  NOTE_C8,NOTE_E7,NOTE_C7,NOTE_E7,
  NOTE_C7,NOTE_D7,NOTE_C8,NOTE_D7,
  NOTE_C7,NOTE_D7,NOTE_C8,NOTE_D7
};//melody
int noteDurations[] = {
  8,8,8,8,
  8,8,8,8,
  8,8,8,8,
  8,8,8,8
};//melody duration
const int buttonPin = 2;           // Button connected to pin 2
const int buzzerPin = 7;           // Buzzer connected to pin 7
const int trigPin = 8;             // Ultrasonic sensor trigger pin
const int echoPin = 9;             // Ultrasonic sensor echo pin
unsigned long buttonPressTime = 0; // Time when button is continuously LOW
const unsigned long thresholdTime = 10000; // 30 seconds in milliseconds
bool alertActive = false;          // Flag to indicate if the buzzer should be
active

void setup() {
  pinMode(buttonPin, INPUT_PULLUP); // Set button pin as input with internal
pull-up
  pinMode(buzzerPin, OUTPUT);       // Set buzzer pin as output
  pinMode(trigPin, OUTPUT);         // Set ultrasonic trigger pin as output
  pinMode(echoPin, INPUT);          // Set ultrasonic echo pin as input
  Serial.begin(9600);               // Start the serial monitor
}

void loop() {
  int buttonState = digitalRead(buttonPin); // Read button state

  // If the button is continuously LOW for 30 seconds
  if (buttonState == LOW) {
    if (buttonPressTime == 0) {
      buttonPressTime = millis(); // Start timing if button is pressed
    } else if (millis() - buttonPressTime >= thresholdTime) {
      alertActive = true; // Set the alert to active
    }
  } else {
    buttonPressTime = 0; // Reset timing if button is not pressed
  }

  // Measure distance with ultrasonic sensor
  long duration, distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2; // Calculate distance in cm

  // If alert is active, check distance to stop buzzer
    if (alertActive) {
    if (distance < 70 and distance > 40) {
      alertActive = false;        // Stop the alert if distance is less than 10
cm
      buttonPressTime = 0;        // Reset button press timer
    } else {
      digitalWrite(buzzerPin, HIGH); // Activate buzzer
      delay(300);
      digitalWrite(buzzerPin, LOW);
      for (int thisNote = 0; thisNote < 16; thisNote++) {
        int noteDuration = 1000 / noteDurations[thisNote];
        tone(12, melody[thisNote], noteDuration);
        int pauseBetweenNotes = noteDuration * 1.30;
        delay(pauseBetweenNotes);
        noTone(8);
      }



    }
  } else {
    digitalWrite(buzzerPin, LOW);    // Deactivate buzzer

  }

  // Display pin status and distance in the Serial Monitor
  Serial.print("Button State: ");
  Serial.print(buttonState);
  Serial.print(" | Buzzer State: ");
  Serial.print(alertActive ? "ON" : "OFF");
  Serial.print(" | Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  delay(100); // Small delay to avoid serial flooding
}
```

[redacted]