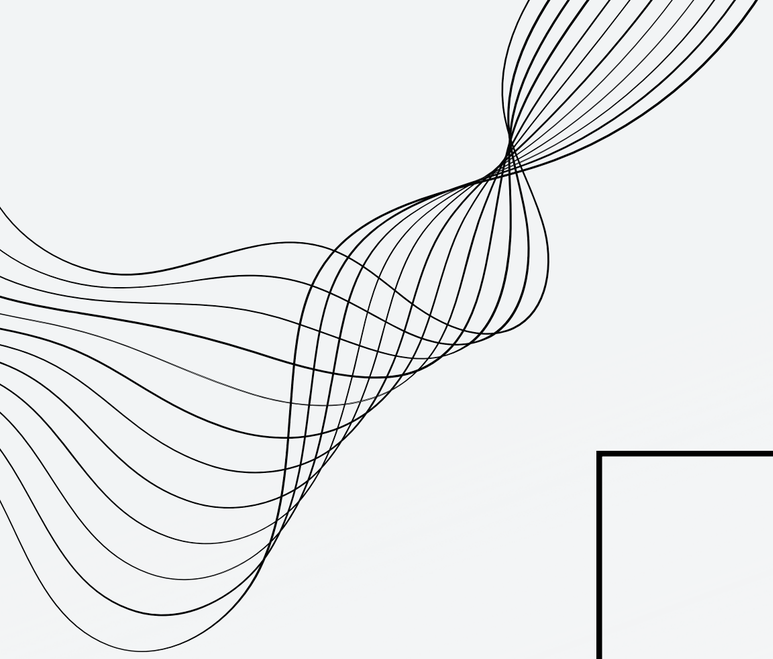
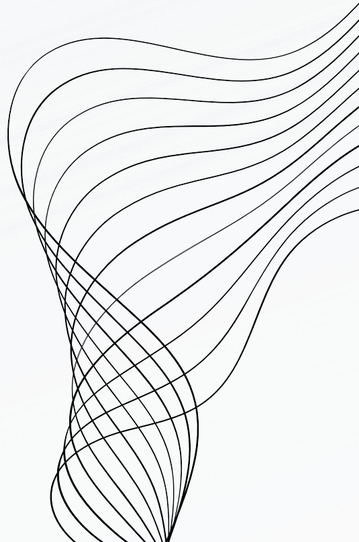


POWERED BY
ADAFRUIT

# MEDITATION

# MART-B

COMPUTER PROGRAMMING FOR ENGINEERS



# CONTENT

# ABOUT MART-B

Project Overview:

- The aim is to develop the MART-B system, an accessible device that enhances meditation by providing real-time biofeedback for stress management.
- Focuses on ensuring user safety and offering customizable feedback options throughout the design, development, and implementation phases.

Benefits of Real-Time Biofeedback:

- Real-time biofeedback offers valuable insights into a user's stress levels, enabling immediate adjustments to meditation techniques.
- This feedback loop enhances the overall meditation experience by providing personalized guidance, helping users understand their physiological state during sessions.

# STATISTICS

In today's fast-paced world, chronic stress has become a prevalent issue affecting individuals' physical and mental well-being. According to the American Psychological Association (APA), more than 70% of adults in the United States report experiencing stress or anxiety daily (APA, 2020). Long-term exposure to stress can lead to adverse health conditions, including cardiovascular diseases, depression, and weakened immune responses (APA, 2020). As people seek ways to mitigate these effects, meditation has emerged as a widely adopted solution due to its ability to reduce stress, promote relaxation, and enhance mental clarity (Smith, 2019).

**70%**

# MART-B ECOSYSTEM

## STRESS MONITOR

A real-time stress monitoring system: MART-B will monitor stress levels
during meditation by collecting biofeedback data - heart rate. This will enable users to receive instant insights into their physiological state, helping them adjust their techniques accordingly.

## MEDITATION ASISTANT

Provide customizable feedback modes: The system will offer multiple modes of feedback, including audio, visual, and haptic feedback, to cater to user preferences
and improve the meditation experience.

## CHATBOT

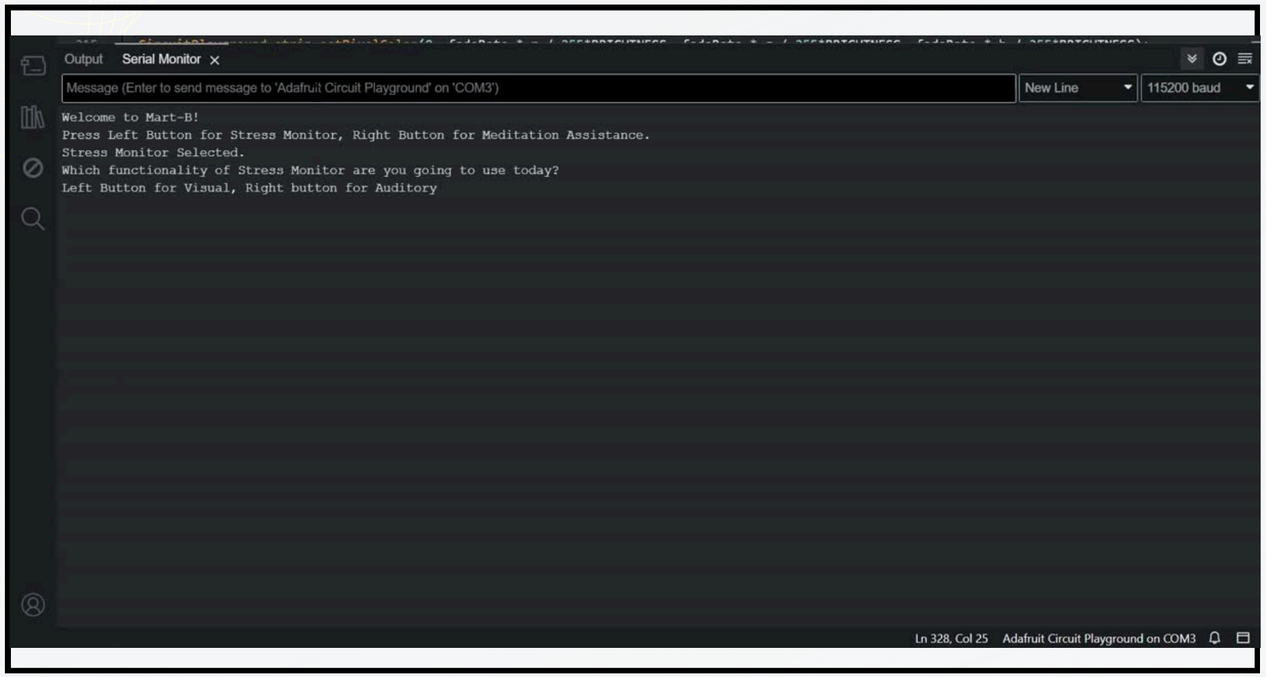
A capable chatbot assistant to help guide users through meditation and engage in
mindfulness exercises as a way to manage stress effectively.

# STRESS MONITOR

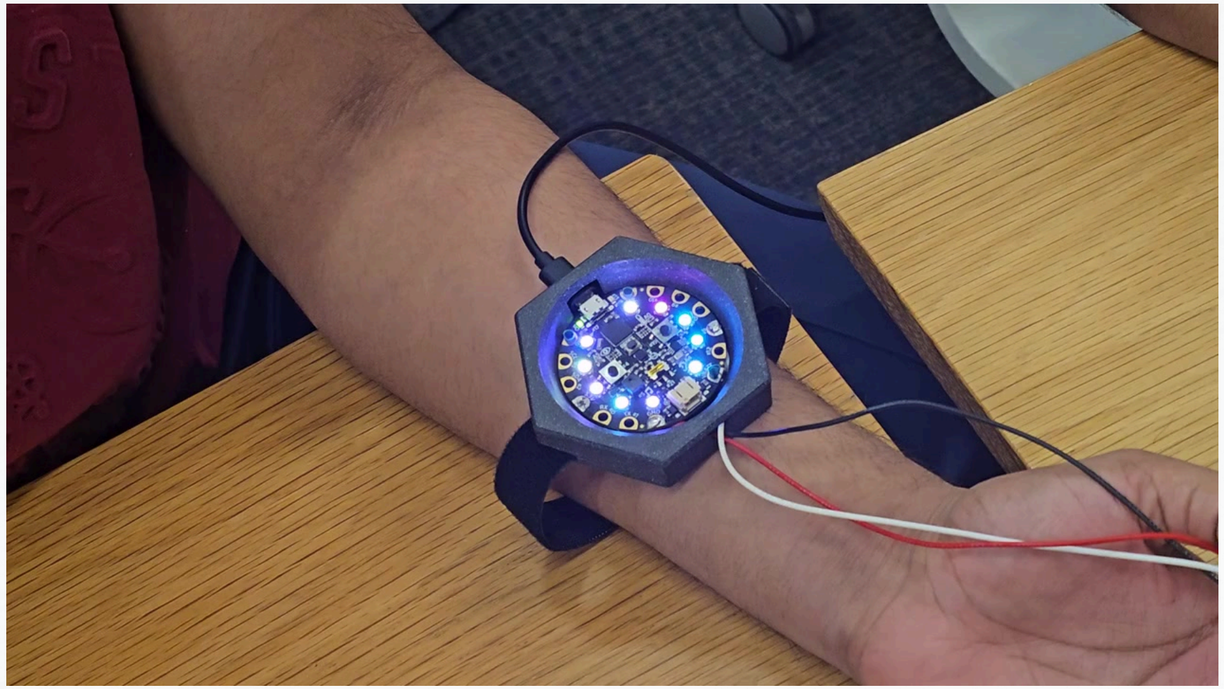
Mode 1:
Visual Mode
Observe your stress with our device with the help of neo-pixel lights

Mode 2:
Audio Mode
Observe your stress with the device as it beeps and vibrates in accordance with your stress levels

# STRESS MONITOR -VISUAL MODE



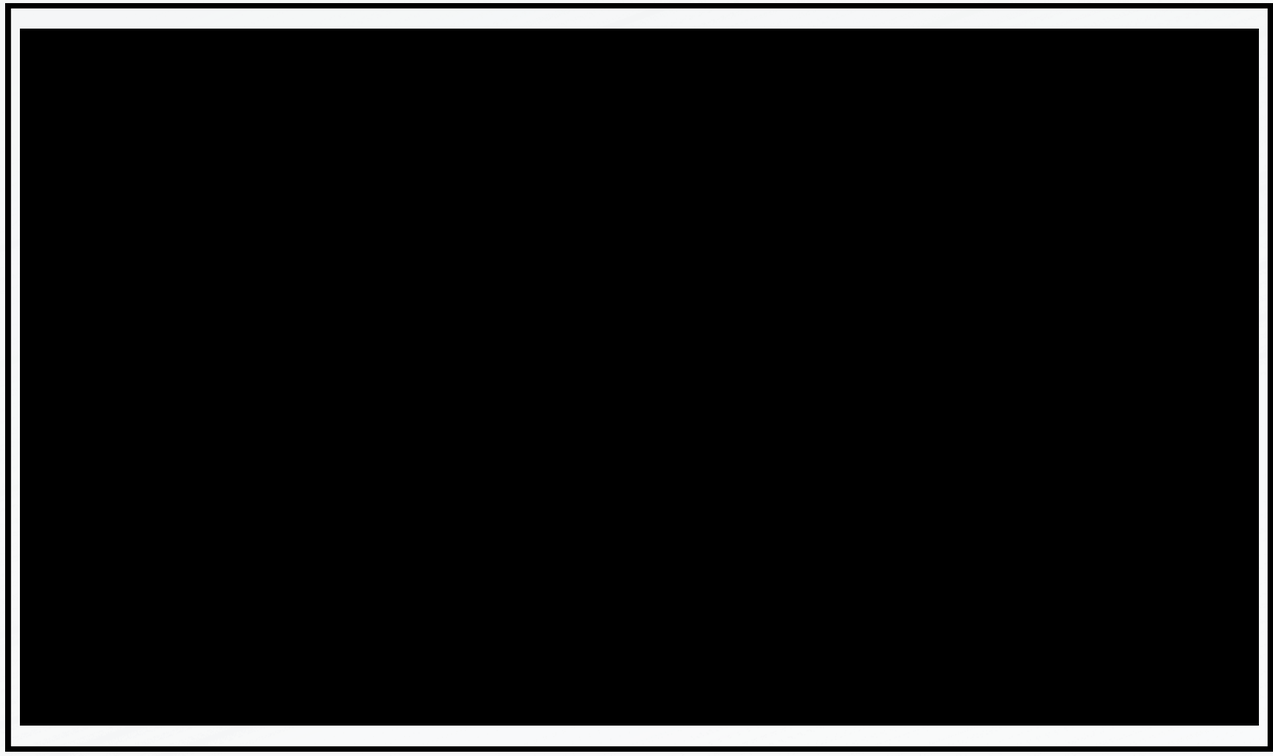

# STRESS MONITOR -AUDIO MODE



# MEDITATION ASSISTANT

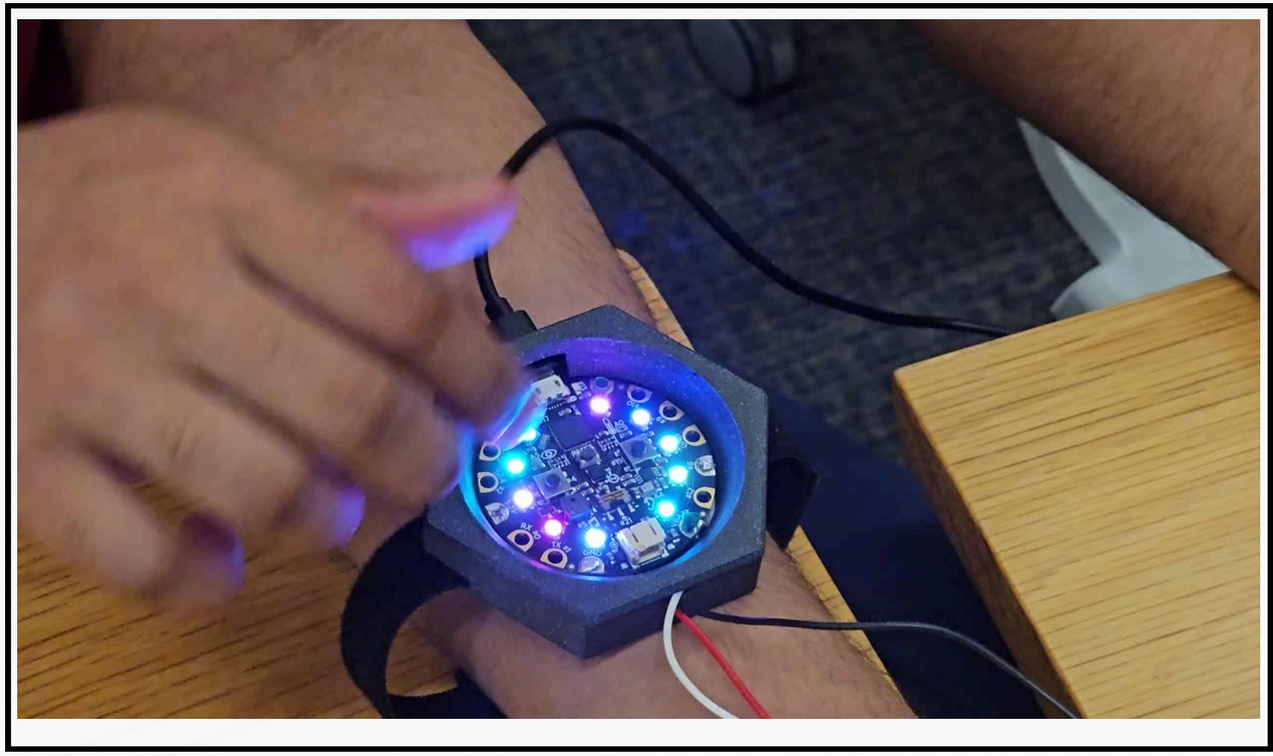
Meditation Assistant with Real-Time Bio-Feedback (MART-B), addresses this need by developing a stress monitoring device that provides users with real-time data on their heart rate and breathing rate during meditation.

# MEDITATION ASSISTANCE - NON AUDIO

# MEDITATION ASSISTANCE - AUDIO

```
Welcome to Mart-B!
Press Left Button for Stress Monitor, Right Button for Meditation Assistance.
```



# IMPLEMENTATION

**Setup & Initialization:**
Initializes Circuit Playground, Pulse Sensor, and modes with startup light animation.

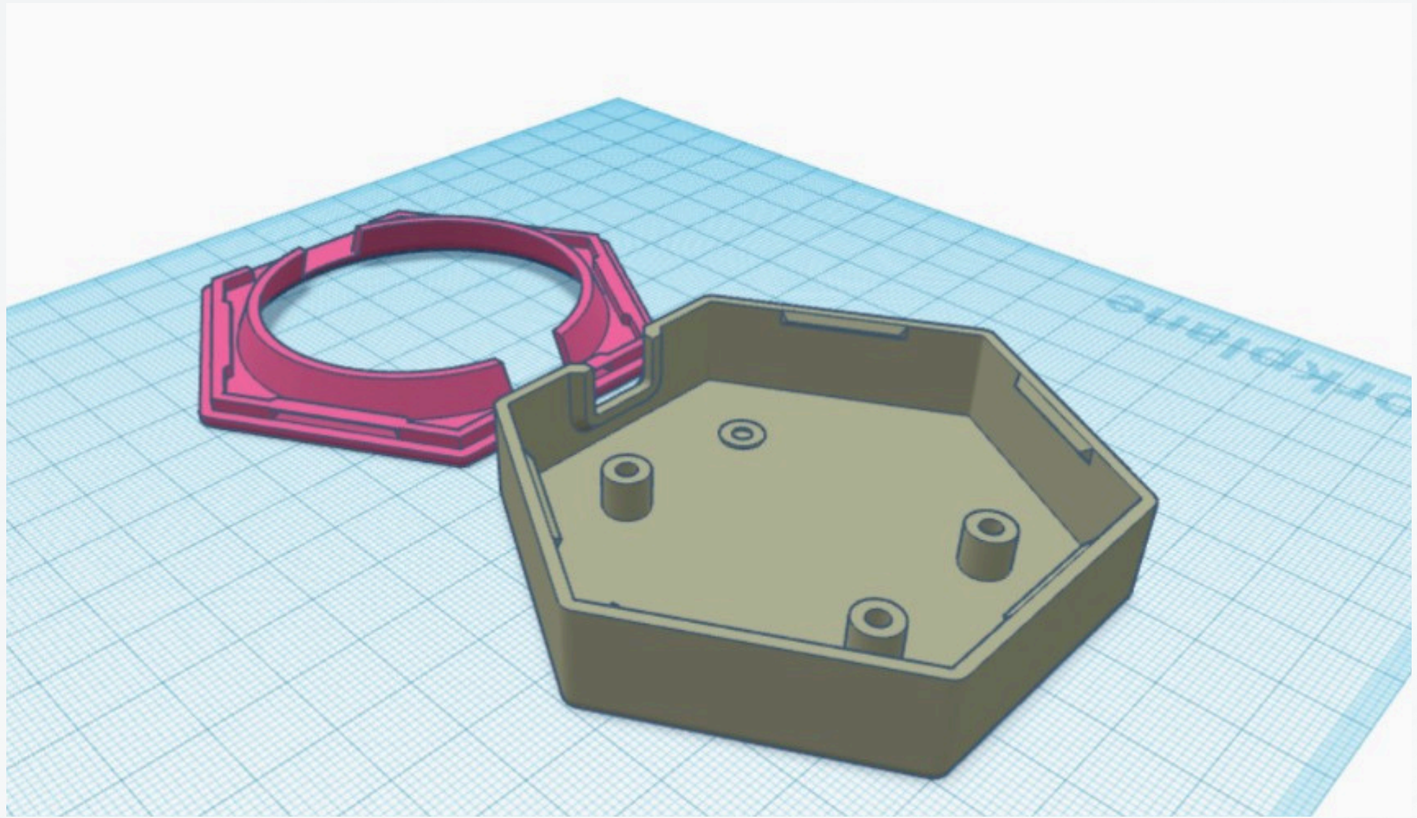
```
void setup() {
  Serial.begin(115200);
  CircuitPlayground.begin();

  // Initialize Pulse Sensor
  pulseSensor.analogInput(PULSE_INPUT_PIN);
  pulseSensor.setThreshold(550);
  pulseSensor.begin();

  // Default light animation on startup
  lightAnimation();

  while (!(CircuitPlayground.leftButton()&& CircuitPlayground.rightButton())){
    continue;
  }
  delay(2000);
  chooseMode();
}
```

```
void loop() {
  if (programMode == 1) {
    stressMonitorLoop();
  } else if (programMode == 2) {
    meditationAssistanceLoop();
  }
  if(CircuitPlayground.leftButton()&& CircuitPlayground.rightButton()){
    delay(2000);
    programMode=-1;
    mode=0;
    mode2=0;
    setup();
  }
}
```

**Mode Selection:**
Left button for Stress Monitor, Right for Meditation Assistance. Further mode choices include audio or visual feedback.

```
void chooseMode() {
  Serial.println("Welcome to Mart-B!");
  Serial.println("Press Left Button for Stress Monitor, Right Button for Meditation Assistance.");
  while (programMode == -1) {
    if (CircuitPlayground.leftButton()) {
      programMode = 1;
      Serial.println("Stress Monitor Selected.");
      Serial.println("Which functionality of Stress Monitor are you going to use today?");
      Serial.println("Left Button for Visual, Right button for Auditory");
      delay(2000);
      while(mode==0){
        if(CircuitPlayground.leftButton()){
          mode = 1;
          Serial.println("You selected the visual mode, yay!");
        } else if (CircuitPlayground.rightButton()){
          mode = 2;
          Serial.println("You selected the audio mode, yay!");
        }
      }
      enterAge();
      startTime = millis();
    } else if (CircuitPlayground.rightButton()) {
      programMode = 2;
      Serial.println("Meditation Assistance Selected. There are 2 modes for Meditation Assistance. Do you want to do it in peace(right button for no music) or beats?(left button for
      delay(2000);
      while(mode2==0){
        if(CircuitPlayground.leftButton()){
          mode2 = 1;
          Serial.println("You selected the audio mode, yay!");
        } else if (CircuitPlayground.rightButton()){
          mode2 = 2;
          Serial.println("You selected the non-audio mode. Good choice!");
```

# IMPLEMENTATION



# IMPLEMENTATION

**Age Input Function**
Takes the age from the user and accordingly sets maximum and minimum heart rates under optimal conditions

```
void enterAge() {
  Serial.println("Please enter your age via the Serial Monitor:");
  while (Serial.available() == 0) {
    // Wait for input
  }
  int age = Serial.parseInt();
  Serial.print("Age entered: ");
  Serial.println(age);
  Serial.println("It begins!");
  // Calculate HRmax and HRrest based on age
  HRmax = 206.9 - (0.67 * age); //https://www.heartonline.org.au/resources/calculators/target-heart-rate-calculator
  if (age <= 1) {
    HRrest = 120;
  } else if (age <= 3) {
    HRrest = 110;
  } else if (age <= 5) {
    HRrest = 100;
  } else if (age <= 12) {
    HRrest = 90;
  } else if (age <= 19) {
    HRrest = 75;
  } else if (age <= 64) {
    HRrest = 72;
  } else {
    HRrest = 70;
  }
}
```

```
void stressMonitorLoop() {
  static unsigned long intervalStart = millis(); // Tracks the start of the current 15-second interval
  static unsigned long monitoringStart = millis(); // Tracks the start of the 60-second duration
  static int readingCount = 0; // Count of BPM readings in the current interval
  static int bpmSum = 0; // Sum of BPM readings in the current interval
  unsigned long currentTime = millis();

  if (pulseSensor.sawStartOfBeat()) {
    alive = true;
    lastAlive = currentTime;

    bpm = pulseSensor.getBeatsPerMinute();

    if (bpm >= 40 && bpm <= 200) { // Filter out erratic values
      Serial.println(bpm);
      bpmSum += bpm;
      readingCount++;
    }
  }

  // Check if 15 seconds have passed
  if (currentTime - intervalStart >= interval) {
    intervalStart = currentTime;

    if (readingCount > 0) {
      int avgBpm = bpmSum / readingCount;
      bpmSum = 0; // Reset for the next interval
      readingCount = 0;
```

**Stress Monitor Loop Function**
Contains two sub-modes and the logic for monitoring the stress over a 60 second on a 15-second interval

# IMPLEMENTATION

**Meditation Assistance Loop Function**

Contains two sub-modes and the logic to help the user meditate. Also contains three sub-functions:

1. Stress Display: Displays stress in 0th and 9th neo-pixels
2. Update Breath LED function: Updates breath LEDs to help the user with a cycle
3. Play music function (only for the audio mode): Plays the octave and reverse-octave according to inhalation-exhalation cycles

```
void meditationAssistanceLoop() {
  unsigned long timeNow = millis();
  float relaxedThreshold = HRrest + 0.15 * (HRmax - HRrest);
  float moderateThreshold = HRrest + 0.35 * (HRmax - HRrest);
  // Update the pulse sensor reading and check if a beat was detected
  if (pulseSensor.sawStartOfBeat()) {
    alive = true;
    lastAlive = timeNow;

    // Calculate BPM and set stress coherence rating
    int bpm = pulseSensor.getBeatsPerMinute();
    if (bpm >= 200) { // Handle false sound values
      Serial.println("Erratic value discarded!");
    }

    if (bpm < relaxedThreshold) {
      stressLights(0,255,0);
    } else if (bpm >= relaxedThreshold && bpm <= moderateThreshold) {
      stressLights(0,0,255);  // Moderate BPM indicates moderate stress
    } else {
      stressLights(255,0,0);  // High BPM indicates high stress
    }

    Serial.print("*** Heartbeat Detected *** BPM: ");
    Serial.println(bpm);

    // Move the breath LED
    updateBreathLED();
  }
```

```
void playSoothingBreathMusic() {
  unsigned long timeNow = millis();
  unsigned long timeSince = timeNow - lastBreath;

  // Determine tone based on breath phase and current LED
  int toneIndex = breathLED - FIRST_BREATH_LED; // Index of the current tone in the array
  int toneFreq = breathToggle ? exhaleTones[toneIndex] : inhaleTones[toneIndex];

  // Play the tone if within the current LED's timing
  if (timeSince <= ledTimes[breathLED]) {
    CircuitPlayground.playTone(toneFreq, ledTimes[breathLED] / 10); // Synchronized tones
  }
}
```

```
void updateBreathLED() {
  for (int i = FIRST_BREATH_LED; i <= LAST_BREATH_LED; i++) {
    if (i == breathLED) {
      CircuitPlayground.setPixelColor(i, 200, 200, 200);
    } else if (i == prevBreathLED) {
      CircuitPlayground.setPixelColor(i, 50, 50, 50);
    } else {
      CircuitPlayground.setPixelColor(i, 0, 0, 0);
    }
  }
  CircuitPlayground.strip.show();
}
```

```
void stressLights(int r, int g, int b) {
    // Gradually reduce fade rate for pulse indicator LEDs
  // Apply fade effect to pulse LEDs
  CircuitPlayground.strip.setPixelColor(0,   r / 255*BRIGHTNESS,   g / 255*BRIGHTNESS,   b / 255*BRIGHTNESS);
  CircuitPlayground.strip.setPixelColor(9,   r / 255*BRIGHTNESS,   g / 255*BRIGHTNESS,   b / 255*BRIGHTNESS);
```

# CHAT BOT

# IMPLEMENTATION

This defines the backbone of your chatbot (the Gemini model) and its empathetic persona.

```
4
5   # Initialize Vertex AI
6   PROJECT_ID = "unified-atom-441618-q6"
7   vertexai.init(project=PROJECT_ID, location="us-central1")
8
9   # Set up the Gemini model with a calming persona
10  model = GenerativeModel(
11      "gemini-1.5-flash-002",
12      system_instruction="You are an empathetic friend and a calming meditation instructor."
13  )
14
```

**This function is crucial for the chatbot, interacting with the Vertex AI model to process prompts and generate responses. It handles errors gracefully with fallback messages, ensuring a smooth user experience and maintaining conversational flow.**

```
def generate_response(prompt):
    try:
        response = model.generate_content(prompt)
        return response.text
    except Exception as e:
        st.error(f"Error generating response: {e}")
        return "I'm having trouble understanding. Can you try rephrasing?"
```

# CONSTRAINTS

Achieving accuracy in biofeedback data collection, particularly from our heart rate sensor, is a critical challenge. The sensitivity and calibration of these sensors directly affected our journey in finding the reliability of the system's feedback.

## HARDWARE LIMITATIONS

As the MART-B system is designed to be portable, maintaining sufficient battery life for real-time monitoring and feedback throughout a meditation session will be a constraint. So, it needs to stay connected to the computer throughout the session

## POWER AND PORTABILITY

Ensuring the device is non-intrusive and comfortable for users during meditation poses design challenges, especially when integrating the sensors and feedback mechanisms.

## USER COMFORT

# OUR TEAM

OMER HAYAT

SAFAYET JAMIL

TAHA SHAFIQ

BILAL AWAN

# THANKS FOR WATCHING

*Questions and Answers Session!!!*

*Ask away!!!*