EC- 310 Microprocessor and Microcontroller Based Design

Semester Project Report

Course Instructor: Dr. Ishfaq Hussain

Lab Engineer: Engr Usama Shaukat

SUBMITTED BY:

Student Name: Tazeen ur Rehman **Reg #** 466342
Student Name: Rida Zahra **Reg #** 465791
Student Name: Mawa Chaudhary **Reg #** 474121
Student Name: Mahad Umar Qaisrani **Reg #** 482972

Degree/ Syndicate: CE 45 A

Date: 6th May, 2025

Contents

Abstract

A simplified wearable prototype was developed using an ESP32 microcontroller, a breathing sensor, a vibration motor, and a mobile application. The system continuously monitors the user's breathing rate through the sensor. When an abnormal increase in breathing rate is detected, the device automatically activates vibrotactile feedback using the vibration motor. These gentle vibrations follow a slow and steady rhythm designed to help the user subconsciously sync their breathing to a calmer pace. Bluetooth connectivity enables real-time interaction with the mobile app, allowing users to control the system. The system was connected with an already available app (nrF connect) to let user control the system as well. The entire setup was designed, implemented, and tested using Wokwi simulation for proof of concept.

Introduction

Stress and anxiety have become increasingly common due to fast-paced lifestyles and constant exposure to external pressures. Wearable technologies are being explored to monitor physiological signals such as heart rate, breathing rate, and skin conductance to detect stress in real time. In response, various devices have been developed to help users manage stress through guided breathing, meditation, or tactile feedback. Among these, vibrotactile feedback is gaining attention as a non-intrusive method to regulate breathing and promote calmness.

The main purpose of this project is to design a wearable system that can detect elevated heart rates using a sensor and provide vibrotactile feedback to help regulate and reduce anxiety. By guiding users toward slow-paced breathing, the system aims to offer a discreet and effective solution to manage stress in real time without needing visual or audio cues.

This project involves the development of a prototype using an ESP32 microcontroller, a heart rate sensor, and a vibration motor. It includes real-time breathing rate monitoring, wireless communication via Bluetooth, and vibrotactile feedback. A mobile application allows users to interact with and personalize the system. The project was designed, simulated, and tested virtually using Wokwi, focusing on core functionality and demonstrating proof of concept.

Problem Statement

- Many individuals experience stress or anxiety that causes a sudden increase in heart rate.
- Most calming tools rely on audio or visual cues, which may not be effective or practical in all environments.
- There is a need for discreet, wearable systems that can detect elevated heart rate and provide immediate calming feedback without drawing attention.

- A rapid heart rate is a common physiological sign of stress and anxiety, and if not managed, it can lead to discomfort or health issues.
- A wearable device that provides real-time, silent feedback can help users calm down naturally and effectively.
- This solution promotes self-regulation and mental well-being without the need for external tools or conscious effort.

Objectives

The primary goal of this project is to develop a wearable system that detects elevated heart rate and provides vibrotactile feedback to help users manage stress. The system is designed to be semi-automated, allowing users to either manually activate the calming feedback through a mobile app or rely on the device to respond automatically when an increased heart rate is detected. This dual-mode functionality ensures flexibility, making the system suitable for different user preferences and real-life situations. The aim is to offer a discreet, real-time, and user-friendly solution for stress regulation using minimal effort.

The project aims to achieve several key objectives: integrating a heart rate sensor with the ESP32 microcontroller, establishing Bluetooth Low Energy (BLE) communication between the wearable device and the mobile app, and generating vibrotactile feedback based on real-time heart rate data. Additionally, a mobile app will be developed for user control and personalization of the system, while the entire setup will be simulated and tested using Wokwi to ensure functionality.

Literature Review

Stress and anxiety are prevalent conditions in modern society, often leading to both short-term discomfort and long-term health issues. Physiological responses to stress, such as increased heart rate, shallow breathing, and muscle tension, can exacerbate anxiety if not managed effectively. Over the years, various technologies have emerged to address these challenges by monitoring and regulating physiological responses. Among the most promising solutions are wearable devices designed to detect signs of stress and provide real-time feedback to help users regain control over their emotional and physiological state.

Existing Solutions for Stress Management

1. **Wearable Devices for Stress Monitoring**

   Wearable technologies have become increasingly popular for tracking physiological indicators of stress, such as heart rate, skin conductance, and breathing patterns. **Fitbit** and **Apple Watch** are among the most widely used fitness trackers that offer real-time heart rate monitoring. These devices are designed to help users track their physical activity, sleep, and stress levels by providing insights into heart rate variability (HRV), which is an important indicator of stress and recovery. Similarly, devices like the **Garmin Vivosmart** measure HRV and notify users of changes in their physiological state.

2. **Biofeedback and Relaxation Tools**

   Biofeedback systems like the **Muse Headband** provide real-time EEG feedback to users to promote mindfulness and stress reduction. The device tracks brain activity and gives users auditory feedback to help them stay focused during meditation or relaxation exercises. The **Spire Stone** is another biofeedback device that tracks breathing patterns and provides gentle reminders when the user's breathing rate becomes irregular. Similarly, **HeartMath's Inner Balance** device uses HRV data to guide users through breathing exercises aimed at reducing stress.

3. **Guided Breathing Applications**

   Numerous mobile apps provide guided breathing exercises as a means of stress management. Apps like **Calm**, **Headspace**, and **Breethe** offer users guided meditation and deep breathing exercises that can help manage stress and anxiety. While these apps are effective for many, they require the user to actively engage with the device, which may not be ideal in stressful situations where immediate intervention is needed.

Comparison of Existing Solutions

While current solutions provide useful methods for tracking and managing stress, they often rely on **user engagement** and **active participation**. For example, devices like the **Apple Watch** and **Fitbit** provide heart rate monitoring, but users must consciously use the device's features, such as breathing exercises or relaxation prompts. Similarly, biofeedback systems like the **Muse Headband** require users to be actively engaged in a meditation session for them to be effective. This reliance on active user participation can be a limitation, especially when users are already experiencing high levels of stress or anxiety, where they may struggle to engage with the device consciously.

Furthermore, many wearable devices and applications provide feedback through **auditory or visual cues**, which might not be suitable in all environments. For instance, a user in a public place, during a meeting, or in a quiet setting might find it difficult or inappropriate to use audio or visual feedback for stress relief.

Another common limitation is the **lack of automaticity** in existing solutions. Many stress-relieving devices and apps require the user to initiate a response, which may be too late when experiencing acute stress or anxiety. Users may not always be aware of their elevated heart rate or may struggle to respond effectively in the moment.

Gaps in Existing Solutions

Despite the availability of numerous wearable devices and apps designed to manage stress, several gaps remain in terms of **discreetness**, **automation**, and **real-time intervention**. The current devices often lack the ability to automatically detect stress and provide immediate, subtle intervention without requiring conscious user action. Additionally, the reliance on audio or visual feedback limits the flexibility and practicality of these solutions in situations where discretion is required.

The need for a **semi-automated, discreet solution** is evident, where the device responds in real-time to physiological data (such as heart rate), providing non-intrusive feedback that helps regulate stress without requiring the user to actively interact with the system. Vibrotactile feedback, such as the kind used in haptic devices, has been identified as an effective solution for delivering such intervention without drawing attention or disrupting the user's environment.

Proposed Solution and Contribution

The proposed system aims to fill these gaps by offering a **semi-automated solution** for stress management that uses **heart rate monitoring** and **vibrotactile feedback**. The device automatically detects an elevated heart rate and provides subtle, haptic feedback to guide the user toward slow-paced breathing. This system is designed to operate in a **discreet manner**, without the need for visual or auditory cues, making it suitable for use in any environment. Additionally, the system is flexible, allowing users to choose between **manual** and **automatic** modes of intervention based on their preferences.

This approach not only offers real-time, automatic intervention but also allows users to control and personalize the feedback they receive. The inclusion of Bluetooth Low Energy (BLE) communication between the wearable device and a mobile app further enhances the user experience by allowing for easy control and tracking of heart rate data.

In summary, the proposed system offers an innovative and flexible solution for stress management by combining the strengths of wearable technology, heart rate monitoring, and vibrotactile feedback while addressing the limitations of existing solutions. It is expected to be a discreet, effective, and user-friendly tool for managing stress in real-time.

System Overview

The system consists of three main components: a wearable device, a mobile application, and BLE communication between them. The wearable device includes a heart rate sensor connected to an ESP32 microcontroller, which continuously monitors the user's heart rate. When the sensor detects a heart rate above a set threshold, the ESP32 triggers a vibrotactile motor to generate calming feedback. This operation runs in semi-automatic mode, allowing either automatic response or user-initiated control. The mobile application allows users to monitor real-time data, configure settings, and switch modes. Bluetooth Low Energy (BLE) provides seamless, low-power communication between the device and the app, completing the feedback loop.

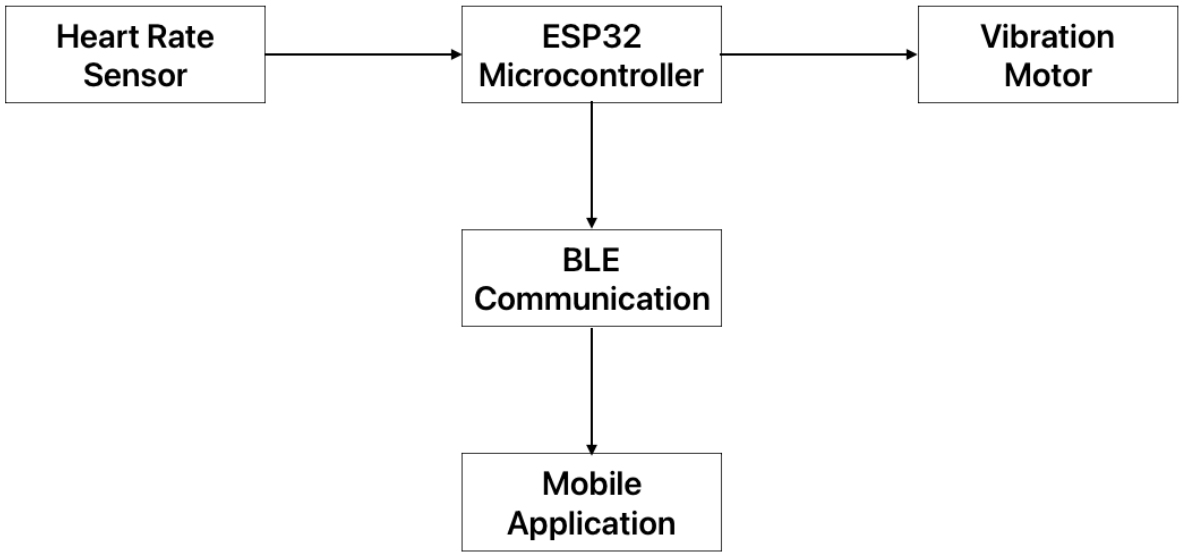


Figure 1_Block Diagram

Hardware Components

- ESP32 WROOM-32 Wifi + Bluetooth MCU Module

- Vibration Coin Motors

- Power source – 300mAh 3.7V 1S Rechargeable LiPo battery

- MAX30100 Sensor

- Veroboard

Software Tools

- Arduino IDE
- nRF Connect
- Wokwi (simulator)

System Design & Implementation

Hardware of the project

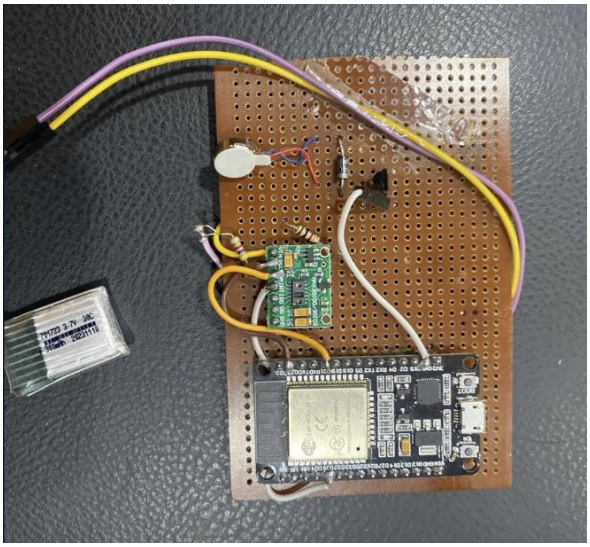


Figure 2_ Circuit Design

Wokwi Test:

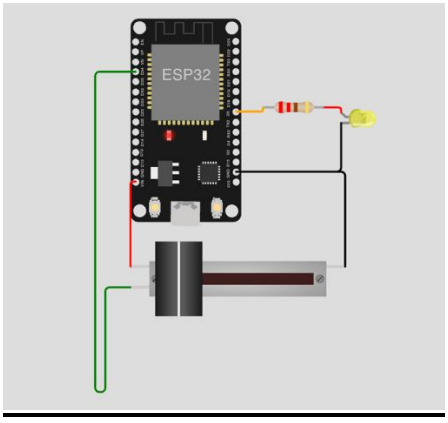


Figure 3_wokwi circuit plan

```
Analog Value: 3374 -> Heartbeat: 109
Motor ON
Analog Value: 3559 -> Heartbeat: 112
Motor ON
Analog Value: 3559 -> Heartbeat: 112
Motor ON
Analog Value: 0 -> Heartbeat: 60
Motor OFF
```

Figure 4 _ Output of wokwi design

Note :Used Potentiometer and LED as wokwi does not support sensor and motor (as we can't see correct simulation through it)

BLE communication details

In our project, BLE communication was implemented using the ESP32's built-in Bluetooth Low Energy module. The ESP32 was programmed to act as a BLE peripheral device that continuously advertises its availability and sends heart rate data to the mobile application. We used custom BLE characteristics to define specific data points, such as the heart rate value and device mode status. The mobile app, acting as a BLE central, connects to the ESP32, reads the data in real-time, and allows the user to configure settings like vibration intensity or control mode (manual/automatic). This setup enabled reliable, low-latency communication between the wearable and the app, ensuring smooth user interaction and system responsiveness.

App interface

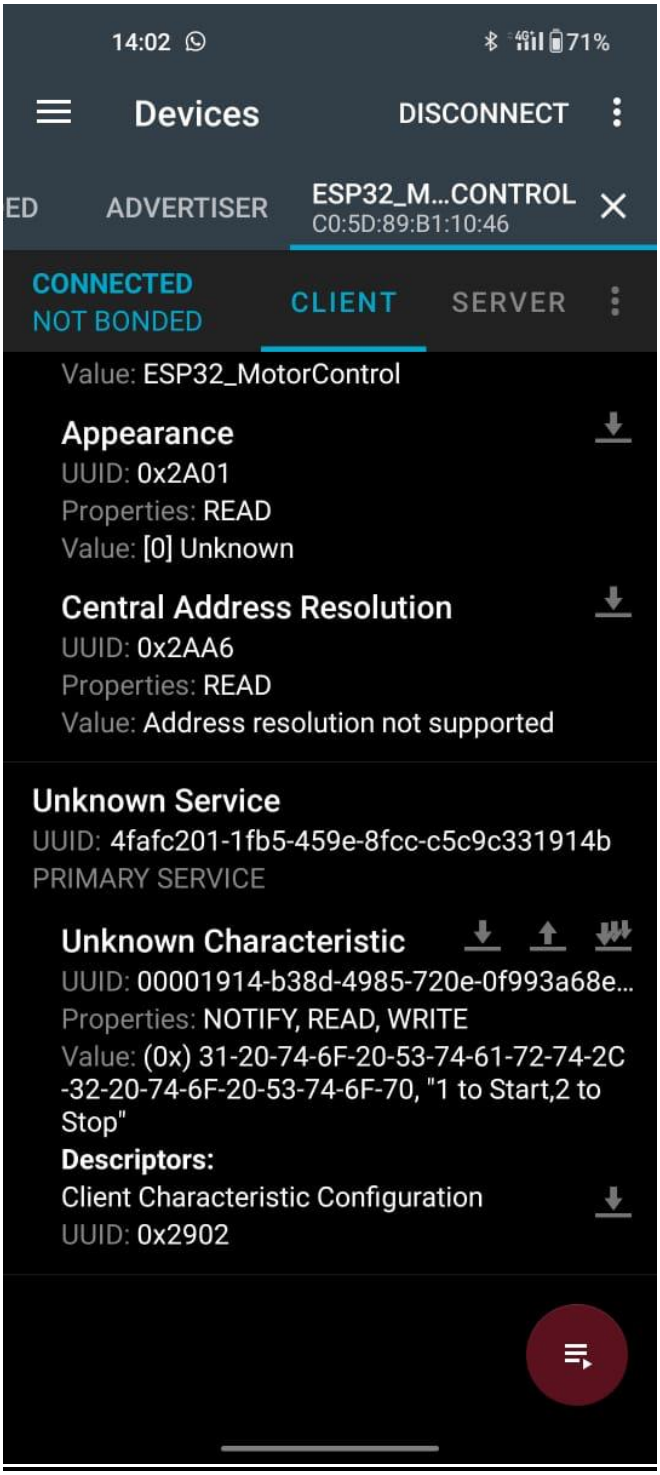


Figure 5_ Add 1 to start and 2 to stop

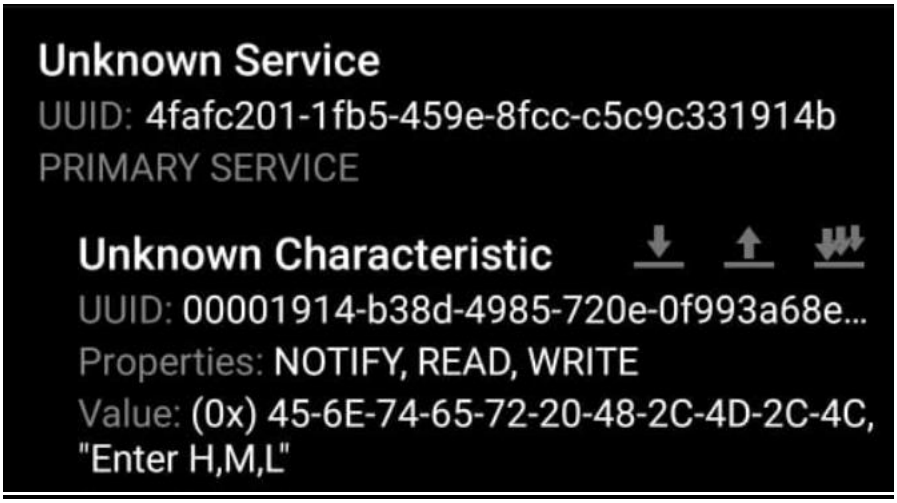


Figure 6_ Enter H, M , L to set intensty

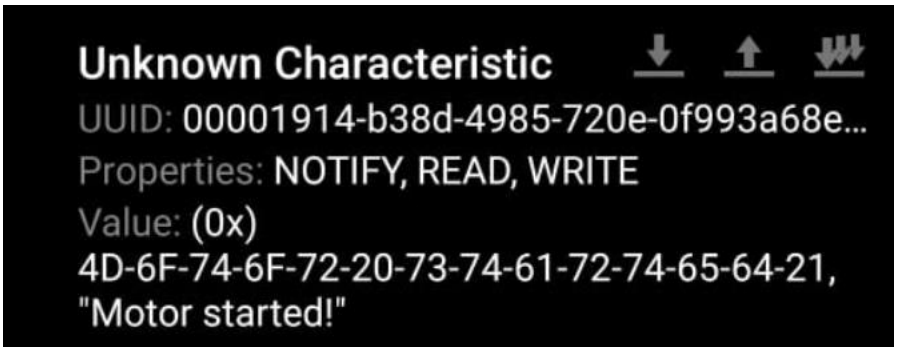


Figure 7_ 1 added and now motor is running

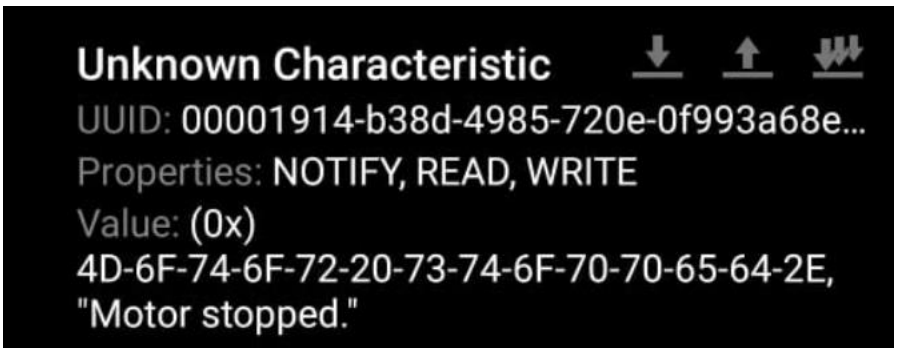


Figure 8__ 2 is entered and motor is stopped

Working Principle

Flow of operation

The operation of the system begins with the user wearing the device, which continuously monitors their heart rate through an integrated sensor. The data is transmitted in real-time to the microcontroller, which processes the information and compares the heart rate to pre-defined thresholds. If the heart rate exceeds a certain threshold, indicating elevated stress, the system automatically activates the vibrotactile feedback mechanism. This feedback is delivered through a vibration motor, generating a series of gentle vibrations that mimic a slow, calming breathing pattern. The device provides subtle, non-intrusive feedback to help the user regulate their breathing in response to the stress trigger. Additionally, the system allows the user to control and personalize the settings via a mobile app, where they can adjust the device's sensitivity or choose between manual and automatic modes. This flow ensures that users can manage stress in real-time, either through automatic intervention or conscious engagement with the device.

The system is designed to offer a smooth and discreet experience for users dealing with stress. Once the device is worn, it operates silently in the background, monitoring the heart rate. If stress is detected, vibrotactile feedback begins automatically, helping calm the user without needing any manual input. The mobile app enhances the experience by allowing users to customize feedback settings, switch between automatic and manual modes, and view real-time heart rate data. The interface is simple and user-friendly, ensuring that even non-technical users can interact with the system effortlessly.

Testing and Results

The system was tested in both simulated and practical conditions to verify its accuracy and responsiveness. Using the Wokwi simulator, we validated BLE connectivity, heart rate data flow, and device response timing. In practical testing with a heart rate sensor, we simulated stress conditions by manually increasing the input rate. The system successfully detected threshold breaches and triggered vibrotactile feedback within seconds. BLE communication remained stable, and the mobile app accurately displayed real-time heart rate and allowed seamless mode switching. Overall, the system performed reliably, confirming the effectiveness of its semi-automated stress response mechanism.

Challenges Faced

During development, we encountered several technical and design challenges. One major issue was ensuring stable BLE communication between the ESP32 and the mobile app, as initial attempts faced frequent disconnections. This was resolved by optimizing the BLE advertising interval and handling reconnection logic in the app. Another challenge was simulating accurate heart rate data for testing since real-time physiological input wasn't always available. We overcame this by manually feeding sensor values through code to test system responses. Designing a subtle yet effective vibration pattern also required fine-tuning the motor control to balance noticeability and comfort. Each issue was resolved through iteration, testing, and reviewing technical documentation.

AI Implementation:

```
## 1. Load and Prepare Data
#df = pd.read_csv('heart_data.csv')
df = pd.read_csv('heart_rate_dataset.csv')
print(df.sample(5))
```

```
     bpm  motor_on  motor_intensity
11   102         1                2
3     75         0                0
91   140         1                2
36    77         0                0
62   163         1                3
```

Figure 9_DataSet we took

```
[7] # Model for motor on/off
    model_on = RandomForestClassifier(random_state=42)
    model_on.fit(X_train, y_train['motor_on'])
    on_pred = model_on.predict(X_test)
    print("\nMotor On/Off Prediction Results:")
    print(classification_report(y_test['motor_on'], on_pred))
```

```
Motor On/Off Prediction Results:
              precision    recall  f1-score   support

           0       1.00      0.80      0.89         5
           1       0.94      1.00      0.97        16

    accuracy                           0.95        21
   macro avg       0.97      0.90      0.93        21
weighted avg       0.96      0.95      0.95        21
```

Figure 10_Model to test if motor should be on or off

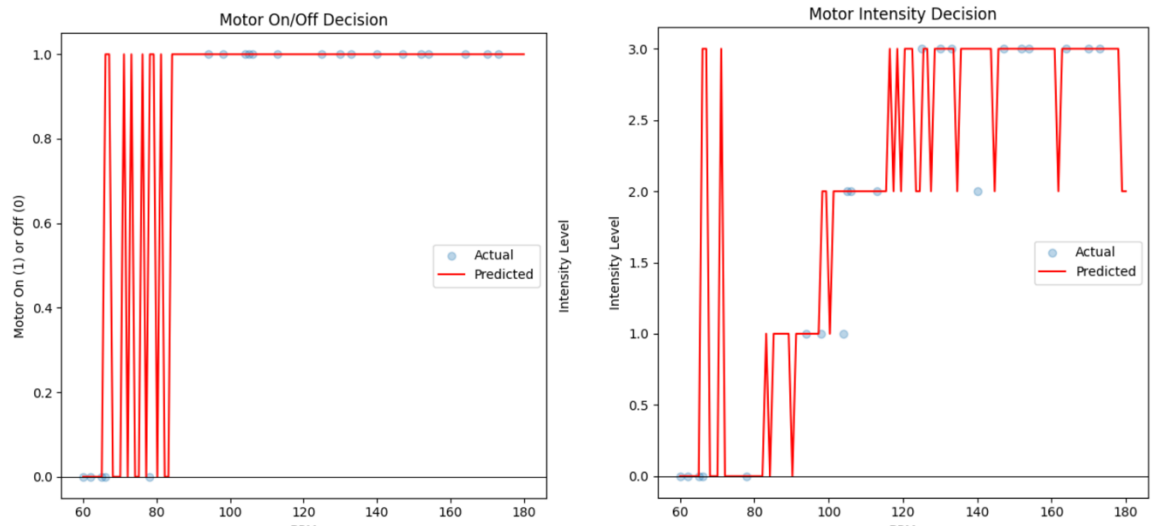
```python
# Model for intensity
model_intensity = RandomForestClassifier(random_state=42)
model_intensity.fit(X_train, y_train['motor_intensity'])
intensity_pred = model_intensity.predict(X_test)
print("\nMotor Intensity Prediction Results:")
print(classification_report(y_test['motor_intensity'], intensity_pred))
```

```
Motor Intensity Prediction Results:
              precision    recall  f1-score   support

           0       1.00      0.80      0.89         5
           1       1.00      0.67      0.80         3
           2       0.60      0.75      0.67         4
           3       0.80      0.89      0.84         9

    accuracy                           0.81        21
   macro avg       0.85      0.78      0.80        21
weighted avg       0.84      0.81      0.81        21
```

Figure 11__model to decide the intensity of the model



Output of our model in graph

```python
print("\nExample Predictions:")
print(predict_motor_action(70))  # Should be Off
print(predict_motor_action(95))  # Should be Low
print(predict_motor_action(110)) # Should be Medium
print(predict_motor_action(130)) # Should be High
```

```
Example Predictions:
{'motor_on': False, 'motor_intensity': 'Off', 'bpm': 70}
{'motor_on': True, 'motor_intensity': 'Low', 'bpm': 95}
{'motor_on': True, 'motor_intensity': 'Medium', 'bpm': 110}
{'motor_on': True, 'motor_intensity': 'High', 'bpm': 130}
```

Figure 12_Tested with sample data (most frequent data from sensor)

Deployed the code on flask and data was received from esp32 and then output of model was sent back.

```
D:\4th Semester\Microprocessor and Microcontroller\Python project related work\flask_api>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 571-209-725
```

The Problem here:

On the ESP32, Wi-Fi and Bluetooth (especially BLE) can interfere with each other because they share the same radio hardware. In practice, simultaneous use of BLE and Wi-Fi on the ESP32 often causes instability or prevents proper operation of one or both features, especially if real-time sensor readings are involved.

We could not remove BLE as it was the major part of or project so now we will deploy ML through TinyML.

Future Work

One of the major future directions of this project is the integration of real-time stress detection through Artificial Intelligence. We have already completed the training phase of a machine learning model that classifies stress levels based on heart rate patterns. The next step is to deploy this model onto the ESP32 microcontroller using TinyML frameworks such as TensorFlow Lite for Microcontrollers. This will enable on-device inference, making the system more efficient and independent of external computation. By enabling the device to autonomously recognize stress in real time, we aim to transform it from a reactive system into a smart, predictive tool that offers proactive calming feedback without user intervention.

In addition to AI, the incorporation of an Inertial Measurement Unit (IMU) sensor is planned to improve the accuracy of stress detection. The IMU sensor can track physical movement, posture, and activity levels, which are important physiological indicators often associated with stress. For instance, sudden movements, restlessness, or prolonged inactivity can be signs of anxiety or discomfort. By fusing IMU data with heart rate information, the system can make more context-aware decisions, improving both the precision and reliability of stress classification. This multi-sensor approach will also help reduce false positives caused by physical exertion or irregular movement patterns.

Furthermore, improvements in the overall system design are planned to enhance usability and aesthetics. A customized hardware casing will be developed to house the components in a compact, wearable form that is both comfortable and discreet. The mobile app will be upgraded with new features such as historical data tracking, stress trend visualization, and deeper personalization

options. Additionally, more robust BLE communication handling and power optimization techniques will be implemented to ensure long-term wearability. These upgrades aim to make the device a complete, user-friendly wellness solution suitable for real-life application.

Conclusion

This project focused on designing and developing a wearable system that helps users manage stress by monitoring heart rate and providing vibrotactile feedback. A semi-automated model was implemented using the ESP32 microcontroller, heart rate sensing, and Bluetooth Low Energy (BLE) communication with a mobile app. The system allows users to either manually control the calming feedback or rely on automatic activation based on elevated heart rate. Through simulation and practical testing, the prototype demonstrated reliable performance, smooth BLE connectivity, and a user-friendly experience.

The key takeaways from this project include the successful implementation of a basic stress-regulation wearable, integration of BLE for real-time communication, and the foundation laid for smart, AI-driven stress detection. The work also highlighted the importance of simplicity, responsiveness, and personalization in designing effective wellness devices. With future enhancements like TinyML deployment, IMU-based motion tracking, and improved casing and app features, this project has the potential to evolve into a fully-functional and market-ready solution for mental well-being.

References

1. **Research Papers**

- Waskita, N. I., Tandungan, H. M., Hafizh, R., Suwaendi, S. J., & Magfirawaty. (2024). ESP32 and MAX30100 with Chebyshev Filter for Enhanced Heart and Oxygen Measurement. *Jurnal Ilmu Fisika*, 14(1), 54–63. https://www.researchgate.net/publication/385299162_ESP32_and_MAX30100_with_Chebyshev_Filter_for_Enhanced_Heart_and_Oxygen_Measurement
- Kuncoro, C. B., Efendi, A., Luo, W.-J., Sakanti, M. M., & Ammarullah, M. I. (2024). Wireless-based portable device heart rate measurement as biomedical devices for stress detection. *AIP Advances*, *14*(4), 045021. https://www.researchgate.net/publication/379779248_Wireless-based_portable_device_heart_rate_measurement_as_biomedical_devices_for_stress_detection
- Iqbal, T., Simpkin, A. J., Roshan, D., Glynn, N., Killilea, J., Walsh, J., Molloy, G., Ganly, S., Ryman, H., Coen, E., Elahi, A., Wijns, W., & Shahzad, A. (2022). Stress Monitoring Using Wearable Sensors: A Pilot Study and Stress-Predict Dataset. *Sensors*, 22(21), 8135 https://www.mdpi.com/1424-8220/22/21/8135
- Pinge, A., Gad, V., Jaisighani, D., Ghosh, S., & Sen, S. (2024). Detection and monitoring of stress using wearables: A systematic review. *Frontiers in Computer Science, 6*, 1478851.

https://www.frontiersin.org/journals/computer-science/articles/10.3389/fcomp.2024.1478851/full

- Pinge, A., Gad, V., Jaisighani, D., Ghosh, S., & Sen, S. (2024). Detection and monitoring of stress using wearables: A systematic review. *Frontiers in Computer Science*, *6*, 1478851. https://pmc.ncbi.nlm.nih.gov/articles/PMC6515276/
- Liang, C.-L., Yeh, Y.-C., & Chen, H.-C. (2024). *Weighted vest combined with vibrotactile stimulations decrease anxiety in college students*. **Brain Sciences**, **14**(2), 153. https://pmc.ncbi.nlm.nih.gov/articles/PMC11558260/
- Adcock, M., Furlong, M., & Ruck, J. (2016). *Vibrotactile and vibroacoustic interventions into health and well-being*. **Journal of Sonic Studies**, **13**. https://www.researchgate.net/publication/312073018_Vibrotactile_and_vibroacoustic_interventions_into_health_and_well-being
- Hollins, M., & Roy, E. A. (1996). *Relaxation measured by EMG as a function of vibrotactile stimulation*. **Perceptual and Motor Skills**, **82**(3_suppl), 1239–1247. https://link.springer.com/article/10.1007/BF01001169
- Valente, A., Lee, D., Choi, S., Billinghurst, M., & Esteves, A. (2024). *Modulating heart activity and task performance using haptic heartbeat feedback: A study across four body placements*. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (UIST '24). https://dl.acm.org/doi/10.1145/3654777.3676435

2. **Websites**
    - Wokwi ESP32 Simulator: https://wokwi.com
    - Bluetooth Low Energy Overview: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/low-energy/
    - Coin Motors : https://www.precisionmicrodrives.com/coin-vibration-motors
3. **Datasheets**
    - ESP32-WROOM-32 Datasheet: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf
    - MAX30100 Pulse Oximeter and Heart-Rate Sensor Datasheet: https://datasheets.maximintegrated.com/en/ds/MAX30100.pdf

Appendix

Complete code

```
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"

#define MOTOR_PIN 5
#define LED_PIN 2
#define SDA_PIN 21
#define SCL_PIN 22
#define HEARTBEAT_THRESHOLD 70
#define REPORTING_PERIOD_MS 1000

PulseOximeter pox;
uint32_t lastReport = 0;

void onBeatDetected() {
  Serial.println("Beat Detected");
}


BLEServer* pServer = nullptr;
BLECharacteristic* pCharacteristic = nullptr;
bool deviceConnected = false;
bool motorRunning = false;
unsigned long motorStartTime = 0;

#define SERVICE_UUID        "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define CHARACTERISTIC_UUID "00001914-b38d-4985-720e-0f993a68ee41"

// PWM intensity values for High, Medium, and Low
int pwmHigh = 250;
int pwmMedium = 180;
int pwmLow = 140;
int motorPWM = 0;

void controlMotor(int pwmValue) {
  analogWrite(MOTOR_PIN, pwmValue);
  digitalWrite(LED_PIN, HIGH);
  motorStartTime = millis(); // Track start time
```

```
  motorRunning = true;
  Serial.print("Motor started with PWM intensity: ");
  Serial.println(pwmValue);
  // Send message to mobile app
  pCharacteristic->setValue("Motor started!");
  pCharacteristic->notify();
}

void stopMotor() {
  analogWrite(MOTOR_PIN, 0);
  digitalWrite(LED_PIN, LOW);
  motorRunning = false;
  Serial.println("Motor stopped.");
  // Send message to mobile app
  pCharacteristic->setValue("Motor stopped.");
  pCharacteristic->notify();
}

class ServerCallbacks : public BLEServerCallbacks {
  void onConnect(BLEServer* pServer) {
    deviceConnected = true;
    Serial.println("BLE Client Connected.");
    // Send a message when connected
    pCharacteristic->setValue("1 to Start,2 to Stop");
    pCharacteristic->notify();
  }

  void onDisconnect(BLEServer* pServer) {
    deviceConnected = false;
    Serial.println("BLE Client Disconnected. Advertising again...");
    BLEDevice::startAdvertising();
  }
};

class WriteCallback : public BLECharacteristicCallbacks {
  void onWrite(BLECharacteristic* pChar) override {
    std::string rxValue = std::string(pChar->getValue().c_str());
    if (rxValue.length() > 0) {
      char cmd = rxValue[0];
      Serial.print("Received BLE input: ");
      Serial.println(cmd);

      if (cmd == '1') {
        // Start motor, prompt user for intensity
        pCharacteristic->setValue("Enter H,M,L");
```

```
        pCharacteristic->notify();
      } else if (cmd == '2') {
        // Stop motor
        stopMotor();
        pCharacteristic->setValue("Motor stopped.");
        pCharacteristic->notify();
      } else if (cmd == 'H' || cmd == 'h') {
        controlMotor(pwmHigh); // High intensity
      } else if (cmd == 'M' || cmd == 'm') {
        controlMotor(pwmMedium); // Medium intensity
      } else if (cmd == 'L' || cmd == 'l') {
        controlMotor(pwmLow); // Low intensity
      } else {
        Serial.println("Unknown command.");
        pCharacteristic->setValue("Unknown command.");
        pCharacteristic->notify();
      }
    }
  }
};

void setup() {
  Serial.begin(115200);
  pinMode(MOTOR_PIN, OUTPUT);
  pinMode(LED_PIN, OUTPUT);
  analogWrite(MOTOR_PIN, 0); // Initial motor state off
  digitalWrite(LED_PIN, LOW);

  BLEDevice::init("ESP32_MotorControl");
  pServer = BLEDevice::createServer();
  pServer->setCallbacks(new ServerCallbacks());

  BLEService* pService = pServer->createService(SERVICE_UUID);
  pCharacteristic = pService->createCharacteristic(
    CHARACTERISTIC_UUID,
    BLECharacteristic::PROPERTY_READ |
    BLECharacteristic::PROPERTY_WRITE |
    BLECharacteristic::PROPERTY_NOTIFY // Enable notifications
  );

  pCharacteristic->addDescriptor(new BLE2902());
  pCharacteristic->setCallbacks(new WriteCallback());

  pService->start();
  BLEAdvertising* pAdvertising = BLEDevice::getAdvertising();
```

```
  pAdvertising->addServiceUUID(SERVICE_UUID);
  BLEDevice::startAdvertising();

  Serial.println("BLE Motor Control Ready.");
  Serial.println("Use nRF Connect to send '1' to start motor, '2' to stop motor,
and 'H', 'M', or 'L' to set intensity.");

  Serial.begin(115200);
  Wire.begin(SDA_PIN, SCL_PIN); // Custom SDA/SCL pins

  pinMode(MOTOR_PIN, OUTPUT);
  digitalWrite(MOTOR_PIN, LOW); // Motor OFF initially

  Serial.println("Initializing MAX30100...");
  if (!pox.begin()) {
    Serial.println("FAILED to initialize MAX30100. Check wiring!");
    while (1);
  } else {
    Serial.println("MAX30100 Initialized");
  }

  pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);
  pox.setOnBeatDetectedCallback(onBeatDetected);



}

void loop() {
  pox.update();

  if (millis() - lastReport > REPORTING_PERIOD_MS) {
    float bpm = pox.getHeartRate();

    Serial.print("BPM: ");
    Serial.println(bpm);

    if (bpm > HEARTBEAT_THRESHOLD) {
      digitalWrite(MOTOR_PIN, HIGH); // Motor ON
      Serial.println("Motor ON");
    } else {
      digitalWrite(MOTOR_PIN, LOW); // Motor OFF
      Serial.println("Motor OFF");
    }
```

```
    lastReport = millis();
  }

  delay(100); // Small delay for stability
}
```

Output: