SAFE SIGNAL

A REAL-TIME POST-TRAUMATIC STRESS SYNDROME DETECTION DEVICE

DESIGN DOCUMENT

TEAM NUMBER

sdmay25-13

CLIENT

America's VetDogs

MENTORS

BAE Systems

ADVISOR

Dr. Mohamed Selim

TEAM MEMBERS

Aidan Klimczak - Electrical Engineer
Justin Jaeckel - Computer Engineer
Justin Scherrman - Electrical Engineer
Katerina Zubic - Electrical Engineer
Neil Prange - Computer Engineer
Ty Decker - Cyber Engineer

TEAM EMAIL

sdmay24-13@iastate.edu

TEAM WEBSITE

https://sdmay25-13.sd.ece.iastate.edu

EXECUTIVE SUMMARY

Post-traumatic stress disorder (PTSD) can be debilitating to a Veteran's quality of life. It can negatively affect their psychological health and bring on emotions of stress, alienation, and decreased life satisfaction.

The current market alternatives for monitoring mental health relating to PTSD are only reactive to the current symptoms of the episodes, are costly, and are intrusive. Due to the existing insufficient and expensive monitoring devices, Veterans continue to suffer from their PTSD symptoms.

Tackling this gap in inadequate products, we have been tasked to develop a post-traumatic stress syndrome detection device that proactively reads the biometric information of heart rate and blood pressure to mitigate the chances of a PTSD episode. If onset symptoms are detected, the Veterans' wearable will alert their service dog to provide necessary comfort before the symptoms become crippling.

The design constraints include that this detection device must be wearable for the Veteran and be capable of monitoring heart rate and blood pressure. Additionally, there needs to be another device on the service dog that is non-intrusive and discrete to receive some kind of distress signal to then provide assistance to their Veteran.

To create these devices for the Veteran and service dog, we have selected to use the ESP32 - S3 microcontroller that has Bluetooth compatibility. This is to establish wireless communication between the two parties so that they are effectively wearable and non-intrusive. Additionally, for the biometric data, we are using a photoplethysmogram (PPG) sensor to read heart rate and blood pressure. Finally, to alert the service dog, we have incorporated an eccentric rotating mass motor (ERM motor) that signals the Veteran is experiencing onset symptoms.

This semester we focused on creating a wearable iteration of our design based on the previous semester's breadboard implementation. One PCB is for the service dog side and another PCB was designed for the wearable on the veteran. They both incorporate Bluetooth capabilities, have a rechargeable battery, and are able to load and update code to the S3 microcontroller. The dog side has haptic eccentric rotating mass motors as the alert device. The veteran side incorporates the PPG sensor, LED's as a visual indicator for established Bluetooth connection and if the device has been successfully powered on, and a button that is our pseudo interrupt. The interrupt mimics a PTSD episode in the veteran and sets of the ERM motors.

Our design meets nearly all the requirements of being wearable, incorporating a discrete and humane alert device, and detecting heart rate via ECG data. In future iterations we plan to develop a working algorithm and find a better solution to read blood pressure data. Current market, technology, and research hinders the accuracy of reading blood pressure data. Additionally, to create a housing solution so the veteran side will be able to be worn and the dog side has extra protection.

LEARNING SUMMARY

DEVELOPMENT STANDARDS & PRACTICES USED

- KiCAD
- Gitlab
- Arduino IDE
- VSCode
- Integration / Unit Testing

SUMMARY OF REQUIREMENTS

- The device must be wearable and comfortable for the user.
- The device must record the user's heart rate and blood pressure accurately.
- The device must be able to indicate a PTSD attack to the service animal.
- The device must keep the user's medical data secure per HIPPA.
- The device must be non-disruptive to the user's environment.
- The device must not harm the user or service animal in any way.
- The device Bluetooth detection must be operable at least 15 feet.

APPLICABLE COURSES FROM THE IOWA STATE UNIVERSITY CURRICULUM

- **CPRE 281**: Digital Logic
- **CPRE 288:** Embedded Systems I: Introduction
- **EE 201**: Electric Circuits
- **EE 230**: Electronic Circuits and Systems
- **EE 321**: Communication Systems I
- **EE 185**: Introduction to Electrical Engineering and Problem Solving I
- **EE 285:** Problem Solving Methods and Tools for Electrical Engineering
- **COMS 228:** Data Structures and Algorithms
- **CPRE 381**: Computer Organization and Assembly Level Programming
- **CPRE 308:** Operating Systems, Principles and Practices
- **EE 330:** Integrated Electronics
- **EE 333:** Electronic System Design

NEW SKILLS/KNOWLEDGE ACQUIRED THAT WAS NOT TAUGHT

- Integrating Bluetooth communication
- Interfacing with sensors via I2C
- Power analysis of modules
- Understanding Ethics in Engineering
- PCB schematic design
- PCB footprint design

1. Introduction **6**
1.1. Problem Statement 6
1.2.Intended Users 6
2. Requirements, Constraints, And Standards **7**
2.1 Requirements & Constraints 7
2.2 Engineering Standards 8
3. Project Plan **9**
3.1 Project Management/Tracking Procedures 9
3.2 Task Decomposition 9
3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria 10
3.4 Project Timeline/Schedule 10
3.5 Risks and Risk Management/Mitigation 10
3.6 Personnel Effort Requirements 11
3.7 Other Resource Requirements 11
4 Design **12**
4.1 Design Context 12
4.1.1 Broader Context 12
4.1.2 Prior Work/Solutions 13
4.1.3 Technical Complexity 13
4.2 Design Exploration 14
4.2.1 Design Decisions 14
4.2.2 Ideation 14
4.2.3 Decision-Making and Trade-Off 14
4.3 Final Design 15
4.3.1 Overview 15
4.3.2 Detailed Design and Visuals 15
4.3.3 Functionality 15
4.3.4 Areas of Challenge 15
4.4 Technology Considerations 16
5. Testing **16**
5.1 Unit Testing 16
5.2 Interface Testing 17
5.3 Integration Testing 17
5.4 System Testing 18
5.5 Regression Testing 18
5.6 Acceptance Testing 18
5.7 User Testing 18
5.8 Other Types of Testing (E.g. Security) 18
5.9 Results 18
6. Implementation **19**
6.1 Design Analysis 19

7. Ethics and Professional Responsibility **19**
7.1 Areas of Professional Responsibility/Codes of Ethics 19
7.2 Four Principles 20
7.2 Four Principles 20
7.3 Virtues 22
8. Conclusions **26**
8.1 Summary of Progress 26
8.2 Value Provided 26
8.3 Next Steps 27
9. References **27**
10. Appendices **28**
Appendix 1 - Operation Manual 28
Appendix 2 - Alternative/Initial Version of Design 28
Appendix 3 - Other Considerations 28
Appendix 4 - Code 28
Appendix 5 - Team Contract 28

1. Introduction

1.1 Problem Statement

PTSD episodes can be a hindrance and debilitating to veterans' quality of life. Thus, we have been tasked to develop a PTSD detection device that proactively detects onset PTSD episodes and alerts their service dog to provide comfort before the symptoms become incapacitating.

Current mental health care market alternatives are reactive to symptoms versus aiming to hinder the development of a PTSD attack. Additionally, their market products are costly and cumbersome as they currently entail brain electrodes to monitor activity. Due to the inadequate and expensive monitoring capabilities, Veterans continue to suffer from PTSD episodes which negatively harm their quality of life.

This is why we aim to solve these issues; to create a foresighted product to bring comfort to our nation's warfighters. To implement this solution, we will create a wrist-wearable device that monitors the Veteran's physiological data. This information will include heart rate monitoring via an electrocardiogram as well as blood pressure measuring by a photoplethysmogram. If symptoms arise, a signal via Bluetooth will be sent from the wearable to a receiver on the service dog that will exhibit a vibration to call for comfort.

1.2 Intended Users

Two groups of people will use our product. One group includes those who suffer from PTSD attacks and have service dogs to comfort them. The other group that will work with our product is the professional dog trainers who train service dogs to comfort veterans suffering from PTSD.

Our first user is a veteran with PTSD. This user feels PTSD symptoms, which can occur sporadically and are challenging to manage without help. This user needs comfort from their service dog whenever these PTSD episodes arise. They want to be able to participate in activities of daily life without having to worry about PTSD episodes going out of control.

Another user of this product is people with PTSD. The needs of this user are largely similar to a veteran with PTSD, but the environments or situations in which they may have symptoms can vary greatly from the typical veteran. Examples may include but are not limited to first responders and victims of abuse or other traumatic experiences.

Our last users are service dog trainers. These individuals specialize in training dogs to be able to react to an owner’s PTSD attack and provide comfort to alleviate symptoms. This user is currently faced with the challenge of having to train dogs to detect PTSD episodes purely from owner behavior, without any quantitative or technical measures to diagnose episodes. Therefore,

they need some device that can detect with high certainty that a PTSD episode is about to occur and can inform the dog in such a way that the trainer can then teach the dog what to do when it perceives this PTSD alert. This would provide the trainers the benefit of not having to teach the dog how to read physical signs of a PTSD attack, as they could simply teach the dog to respond to a haptic vibration.

2. Requirements, Constraints, And Standards

2.1 Requirements & Constraints

Functional Requirements

The device must be able to effectively detect the biological markers of a PTSD episode without producing false positive or false negative readings. The device must reliably communicate via Bluetooth with another device secured to the dog without losing signal due to range. The dog device must be able to reliably control a haptic vibration device to alert the dog to a PTSD attack.

Soft Constraints

Dog Side

- The Bluetooth connection between the devices must be keep a connection for at least 15 ft.
- Must fit inside the dog's vest.
- Alert device must be discrete and humane.
- Must be battery-operated.
- The battery must be rechargeable.
- Be able to turn on and off.
- Multiple motors accessible depending on the dog.
    - Larger or thicker fur may require more motors to be able to feel a strong enough vibration.

Veteran Side

- Compact/small enough to be a wearable device.
- Be able to turn on and off.
- Have some kind of visual indicator that it is powered and that Bluetooth is successfully connected.
- Must be battery-powered to be wearable.
- The battery must be rechargeable.
- Photoplethsymogram read blood pressure with a margin of error of up to 15 mmHg

Resource Requirements

The device will require the use of two main microcontrollers: One that interfaces with the veteran, and one that interfaces with the dog. The human-wearable microcontroller must have sufficient processing power to handle process management and semi-complex data analysis algorithms. The dog microcontroller should be sufficient to handle low power draw to enable a vibration device. All devices must have efficient power dissipation to ensure a sufficiently long battery life. More specifically, both batteries for the microcontrollers should last at least a day **(Constraint).**

Physical Requirements

There must be two devices; a wearable device for the veteran and a device that fits in a dog vest. The wearable must be comfortable for the veteran and be nondisruptive to daily activities. The device for the dog must fit inside a dog vest comfortably and humanely relay a signal to the dog.

User-Interface Requirements

For UI we are attempting to make it as simple as possible for our user. Our design is as simple as a button to turn the device on and off. The device also will have multiple LEDs to tell the user if there are problems with the device. One LED will be used for power, another LED will give a visual representation if the Bluetooth has been successfully connected.

Economic / market requirements

We have been given a budget of $5000 for our design. **(Constraint).** The final product should be inexpensive and competitive to current mental health monitoring substitutes. Our goal is to produce and distribute our designed device for sub 200 USD. Having a more affordable product will increase accessibility contributing to an improvement in quality of life. Additionally, will increase the use of PTSD detection devices.

2.2 Engineering Standards

Importance of engineering standards

Engineering standards are set criteria and guidelines ranging in varying industries to ensure the safety of use, production, and quality control.

These standards are crucial for product production as they hold every business, enterprise, and inventor to given principles. Setting safety criteria helps protect consumers from accidents. Encouraging quality assurance promotes consistency and reliability. Giving production protocols increases efficiency as well as assembly safety. In all, these standards are vital for creating a safe, useful, friendly, and viable product.

Current Applied Standards

- IEEE 802.15.1: WPAN / Bluetooth

- ISO/IEEE 11073: Medical / Health Device Communication Standards
- IEEE 360-2022: IEEE Standard for Wearable Consumer Electronic Devices
- IEEE 11073-10407-2020: Health informatics--Personal health device communication Part 10407: Device specialization--Blood pressure monitor

External IEEE Standards

Consumer Electronics

- IEEE 360-2022: IEEE Standard for Wearable Consumer Electronic Devices--Overview and Architecture
    - This standard is intended to address concerns and help industry confidence when it comes to the reliability and quality of wearable electronics. As well as standardizing the safety and security of these wearable devices.
- IEEE 11073-00103-2012: Health informatics - Personal health device communication Part 00103: Overview
    - This standard defines set practices for data communication and standards for mobile health devices and separate computing engines. It is intended to ensure that personal health devices follow secure communication practices and are horizontally integrated.
- IEEE 11073-10407-2020: Health informatics--Personal health device communication Part 10407: Device specialization--Blood pressure monitor
    - This standard is part of a set of standards for personal health device communication specializing in blood pressure monitoring and promotes plug-and-play for the user.

Project Relevance to IEEE Standards

After reviewing the three chosen published standards, we believe that all three have relevance to our project. This is because our product is a wearable electronic device that would directly apply to the IEEE standard for a wearable consumer electronic device. This standard is relevant because we need to design a product that is reliable, safe, and secure. We also feel that the standards for health informatics and device communication apply to our product specifically because of the communication of medical data from the sensor to the microcontroller.

Design Changes Based on IEEE Standards

We plan to implement communication between the human-wearable device and the dog device that adheres to the personal health device communication standards. This includes implementing previously defined Bluetooth protocols to make the communication compatible with other devices. We also plan to adhere to the standard for wearable consumer electronic devices so that the veteran-worn device implements all necessary personal wearable device protocols.

3. Project Plan

3.1 Project Management/Tracking Procedures

To increase proficiency we chose to adopt a management style of Hybrid. We chose to use a hybrid project management style during our project. We chose this management style because it is adaptive and accommodates unforeseen challenges. This allows for the opportunity to reflect on previous steps in our project. Using the hybrid management style, we will be able to improve our work based on updated criteria and constraints from our client. The hybrid management style is a combination of waterfall and agile project management that we use to fully meet deadlines.

To maintain organization and track progress, our team will use a work progress chart made in Google Sheets to keep track of our work throughout the course. We will be using GitLab to maintain our work progress for software development.

3.2 Task Decomposition

To keep our team on track, we decomposed our tasks into major and sub-tasks. This is to ensure we are staying on track and accomplishing all necessary milestones to meet our client's needs with precision and efficiency.

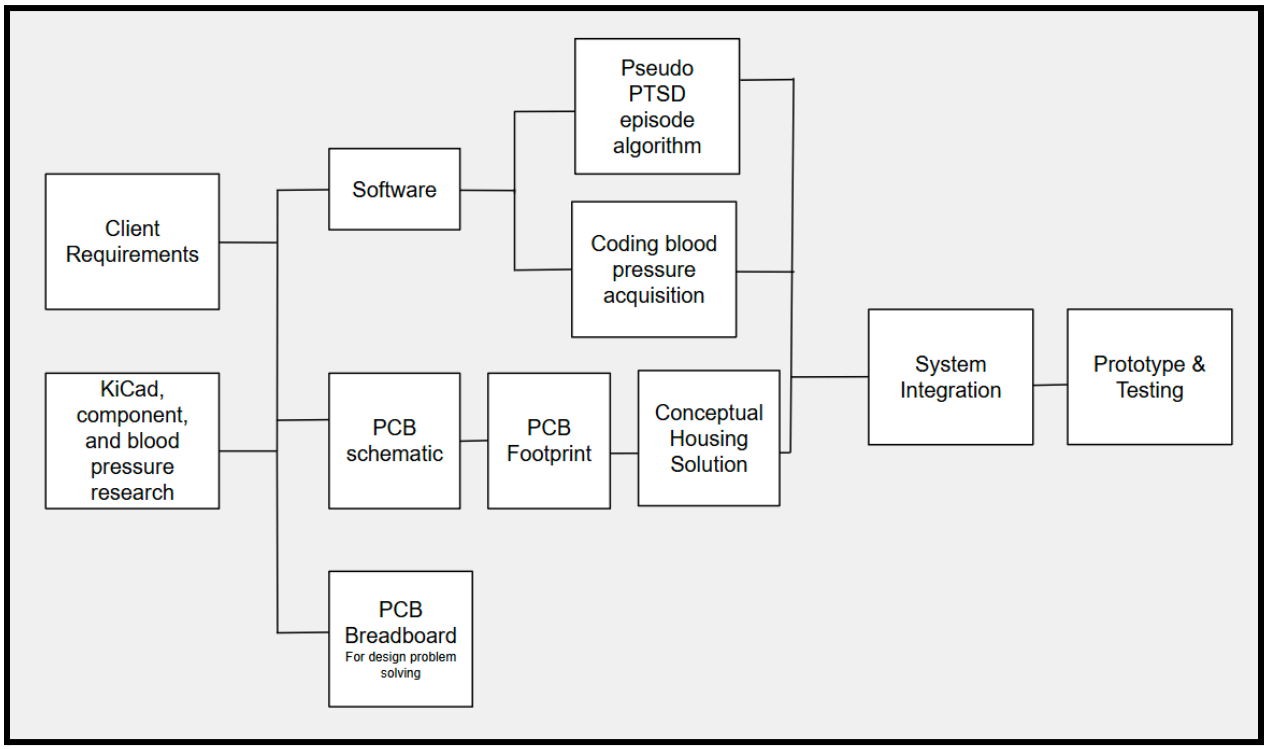


Figure 1: Task Decomposition for PTSD Detection Device.

The main tasks are developing the software algorithms for blood pressure detection as well as PTSD episode detection. These tasks however, have been proven cumbersome due to poor technological advances for non-cuff blood pressure devices. Additionally, due to HIPPA we could not acquire real data for PTSD episodes, so the algorithm is still pseudo.

The other main task was creating the PCB's to make our solution wearable. We went back and forth on many different design considerations, and summed them together to create our final solution. As a way to help debug any issues on the PCBs, we decided to build a PCB breadboard design. This will help solve any problems we may come across.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Illustrating milestones are to help keep our team in the scope of our project as well as decompose what tasks are necessary for a functional project.

Research

- How can blood pressure be pulled from the PPG sensor?
    - How often does this need to happen to ensure accuracy but mitigate power consumption.
- Is heart variability a "good" way to measure a probable PTSD episode?
- How to sample our heart rate acquisition algorithm to reduce power consumption.
- How to use KiCad.
    - Watching videos and reading information on how to create a schematic, footprint, import components, ensuring components are correct size, etc.
    - How to develop a 3D rendering of the PCB.
- What hardware components do we need to make our design work?
    - Reading datasheets.

Developing PCBs

- How do we plan to attach our components to the PCB?
- Adding test points so we can measure the voltage and current for specific areas.
    - Ensure the GPIO pins are only outputting 40mA
    - The microcontroller is only receiving roughly 3.3v.
    - The battery is outputting 3.7V.
    - The LEDs turn on when needed (power is on and Bluetooth is connected).
    - On/Off switch works properly.
        - The LED turns on respectively.
    - The button sends a pseudo code.
        - For debugging purposes.

Software

- Pulling blood pressure data from the PPG sensor.
    - Deciphering if the values correspond to those used with the external blood pressure cuff.
- Reducing the amount of data points taken every minute for heart rate acquisition for power consumption.

Hardware

- The sizes correspond with the constraints given from our clients.
    - The dog side PCB should be able to fit within the dog vest.
        - Maximal design is roughly 3”x3”.
    - The veteran PCB should be able to fit on a wrist.
        - Roughly 2”x2”.
- Antenna location for veteran PCB.
    - Location chosen to cater to right-handed people.
        - The majority of the population is right handed.

Housing

- Conceptual
    - Dog side PCB should have a cut out area where the motors can vibrate freely.
    - Veteran side should have an area where wristbands would be able to connect

3.4 Project Timeline/Schedule

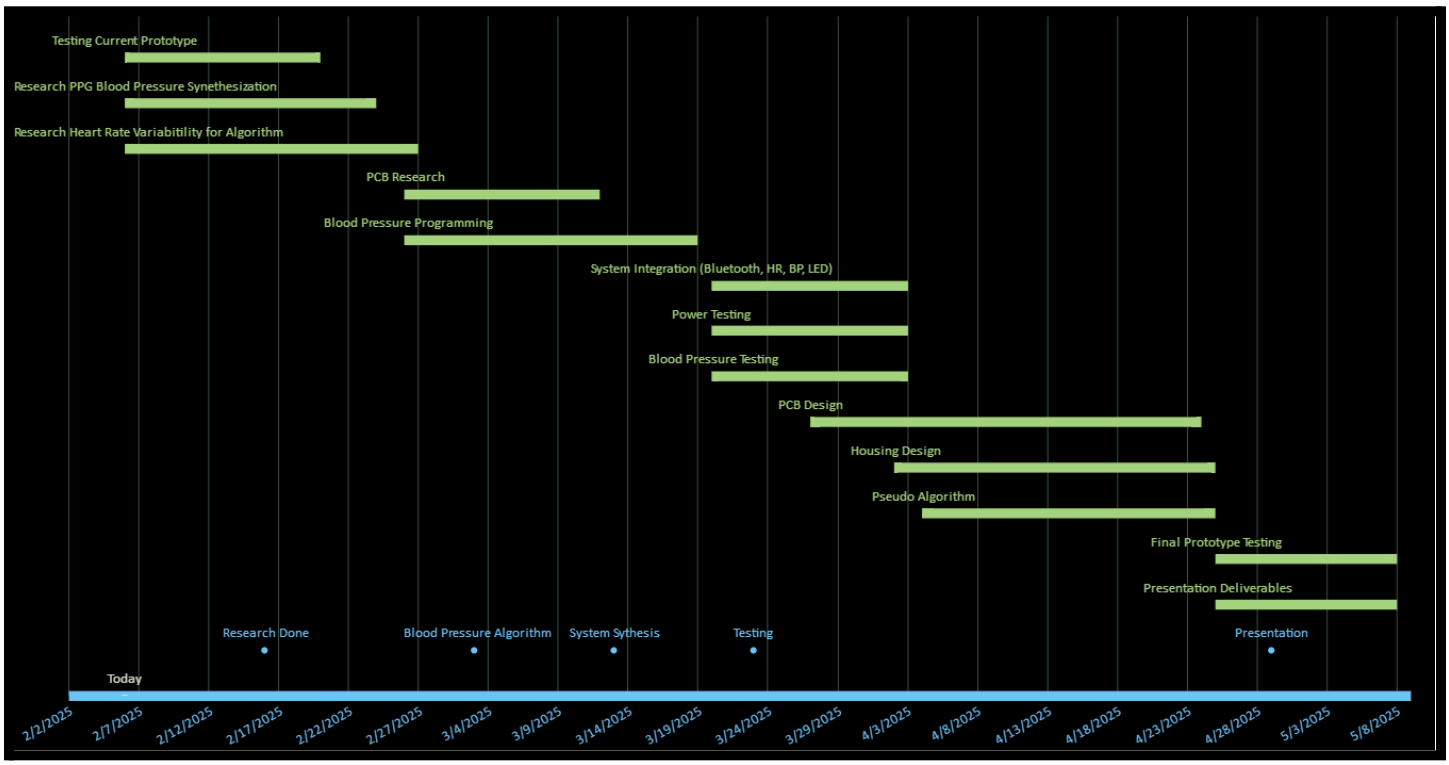


Figure 2: Gantt Chart of Project Timeline

The Gantt chart above details the projected timeline of our project throughout the fall semester. The first phase is going back to research to develop a solution for our algorithm, blood pressure acquisition, and determine how we plan to design our PCBs. The next phase is implementing our research. We’ll begin by programming our blood pressure reading code, design our PCB’s

(starting with the schematics), and order the necessary parts to build our PCBs. The third phase is designing the footprint and the possible housing. The housing will be conceptual due to the time constraints. We'll also continue to develop some kind of pseudo algorithm for PTSD detection. Finally, the last phase will be testing and completing all the necessary final deliverables for the class and presentations.

3.5 Risks and Risk Management/Mitigation

Risk 1: Changing our microcontroller to the S3 model from the Wrover-E - 0.8

- Testing the S3 development board with the original code to see if Bluetooth properly connects to our phones and it sends the appropriate messages.
- Then, testing with the other S3 dev board and seeing if messages can successfully be transferred.

Risk 2: Blood pressure acquisition - 0.9

- Due to the limitations of the current market products, reading blood pressure without using a cuff has not been deemed as a medically accurate and reliable way to read blood pressure.
- Our plan to combat this issue is to build a machine learning algorithm trained on publicly available data, in order to estimate blood pressure up to 15mmHg.

Risk 3: Size - 0.5

- This is only the first iteration, but the size is not yet the most conducive for a wearable device. The veteran side is roughly 3"x3".
- Plan to scale it down to get a working prototype.

Risk 4: ERM motors are too aggressive for the service dog - 0.2

- We will send the prototype to America's VetDogs to get real testing to understand how the dogs respond (negative or positive) to the vibration motors.

Risk 5: Security - 0.6

- Despite we are not sending data over Bluetooth, if someone interrupts the connection, perhaps they can hinder the sending of an alert signal, risking the veteran experiencing a PTSD episode.
- We will incorporate Bluetooth security precautions.

3.6 Personnel Effort Requirements

Task	Estimated Hours Spent
Client Research	20
PTSD Symptom Research	15
Current Market Research	20
Hardware Research	40
Initial Hardware Testing	20
Preliminary Design	30
Modular Hardware Testing	30
Breadboard implementation	20
Testing Communication	15

Figure 3: Allocated Project Hours Table

3.7 Other Resource Requirements

Illuminating other resources aside from financial needs, we will need to acquire several other resources to be successful.

Biometric Control Data:

- Receiving data from America's VetDogs to accurately detect PTSD attacks.
- Finding publicly available health data for blood pressure estimation

Computers/Laptops

- This is to interface with our specific microcontroller IDEs.

Digital Multimeter

- To ensure the electrical integrity of our parts are functioning properly, we need to measure that the values read what we expect.

KiCad

- PCB design software to develop our schematics and PCB designs.

Blood Pressure Cuff

- To test different iterations of the blood pressure monitoring scheme with our own setup.
- To ensure publically available blood pressure data matches our experimental data.

4 Design

4.1 Design Context

4.1.1 Broader Context

The larger context in which our design problem is situated is within communities of people who are suffering from PTSD attacks. The communities include veterans and first responders. Our project addresses the issue of the societal need for veterans who suffer from PTSD to be able to live their everyday lives without needing to worry about suffering a PTSD attack. Our design will act proactively to detect PTSD attacks and alert the service dog to help prevent PTSD attacks.

Area	**Description**	**Examples**
Public health, safety, and welfare	Veterans suffering from PTSD attacks are directly benefited from our device. The device allows for fast and automatic detection of PTSD symptoms to alert a service animal.	Faster PTSD episode detection time for a veteran reduces the magnitude of the episode, and carries the potential to stop the episode altogether if symptoms can be detected before the episode ensues.
Global, cultural, and social	US war veterans are known to have a tightly knit community, with strong emphasis on helping each other out. Our solution supports this culture by allowing veterans more freedom and control over their daily lives through reducing the likelihood of debilitating PTSD attacks.	A veteran using our device would be better able to participate in their community, and spread information of the device's utility to other veterans within the community so as to increase the impact of the device.
Environmental	Like many other mass produced electronic devices, there is some potential for waste and pollution during manufacture of components. The largest potential area of waste would be the batteries, which are known to be hard to dispose of.	Two batteries are used, one for each device in our system. In addition to disposal costs after the device's lifetime, batteries might need to be replaced, further increasing waste. Additionally, manufacturing ESP32 chips can

		have an environmental impact when done at scale.
Economic	The economic impact of our device would affect companies that help veterans and their service animals. The initial cost of our prototype is reasonably low compared to the current market.	Product needs to remain affordable for target users, product creates or diminishes opportunities for economic advancement, and high development cost creates risk for the organization.

Figure 4: Impact Matrix

4.1.2 Prior Work/Solutions

In today's current market, there is no technology available that answers the problem of detecting a PTSD attack before it happens. BAE has proposed efforts to create this product to previous senior design groups.

The previous senior design group from 2023-2024 was assigned this project also proposed by BAE. An advantage of the previous senior design group is that the device is user-friendly for people wanting to track their heart rate and episodes through the app. A disadvantage of the previous senior design group's product is that the prototype seems not to be wearable so the user can't exactly use the device. This prototype just seems to be functional.

S. D. May. "Project Website for SDMay24-15." Iowa State University Department of Electrical and Computer Engineering. [Online]. Available: https://sdmay24-15.sd.ece.iastate.edu/.

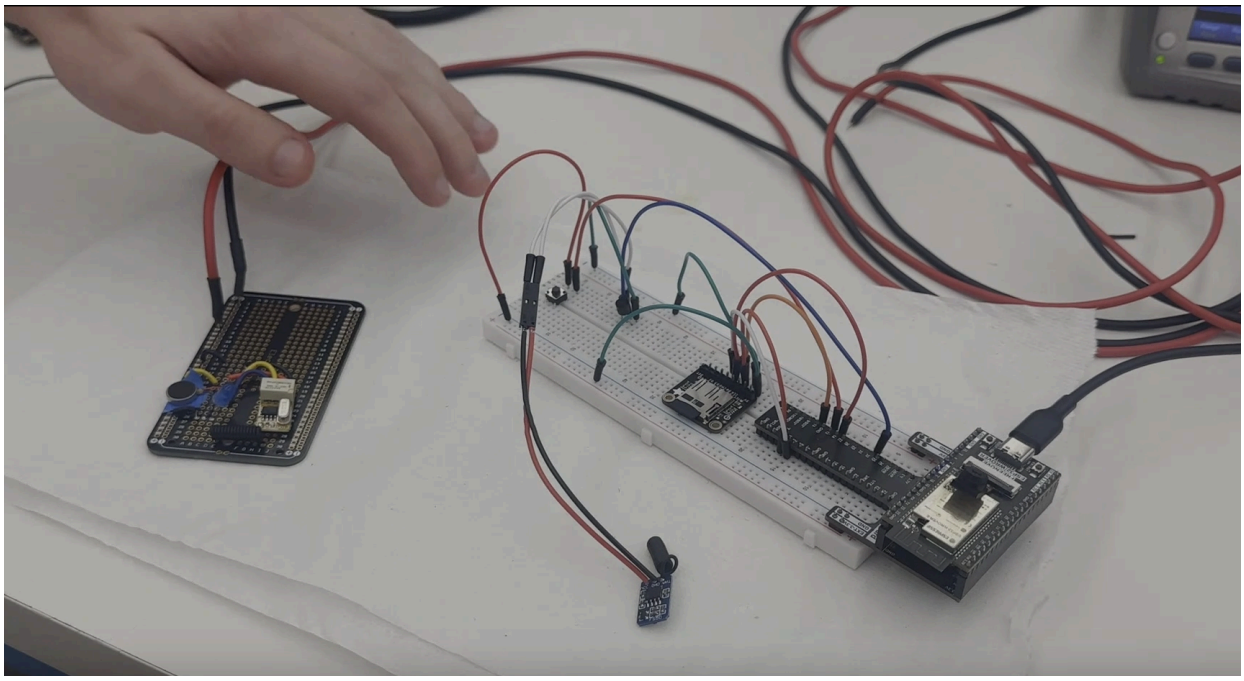


Figure 4.1: Previous Senior Design Team Implementation

Our team's previous semester efforts produced a breadboard prototype capable of successfully establishing Bluetooth connections between devices and performing basic PPG sensor data acquisition. Although this initial version was not fully wearable, it demonstrated critical feasibility for the wireless system and helped refine component selections for the final PCB design. A layout of this design is seen below in figures 4.1.2. These show the basic breadboard implementation that was used to test connections and basic performance of the PPG sensor.

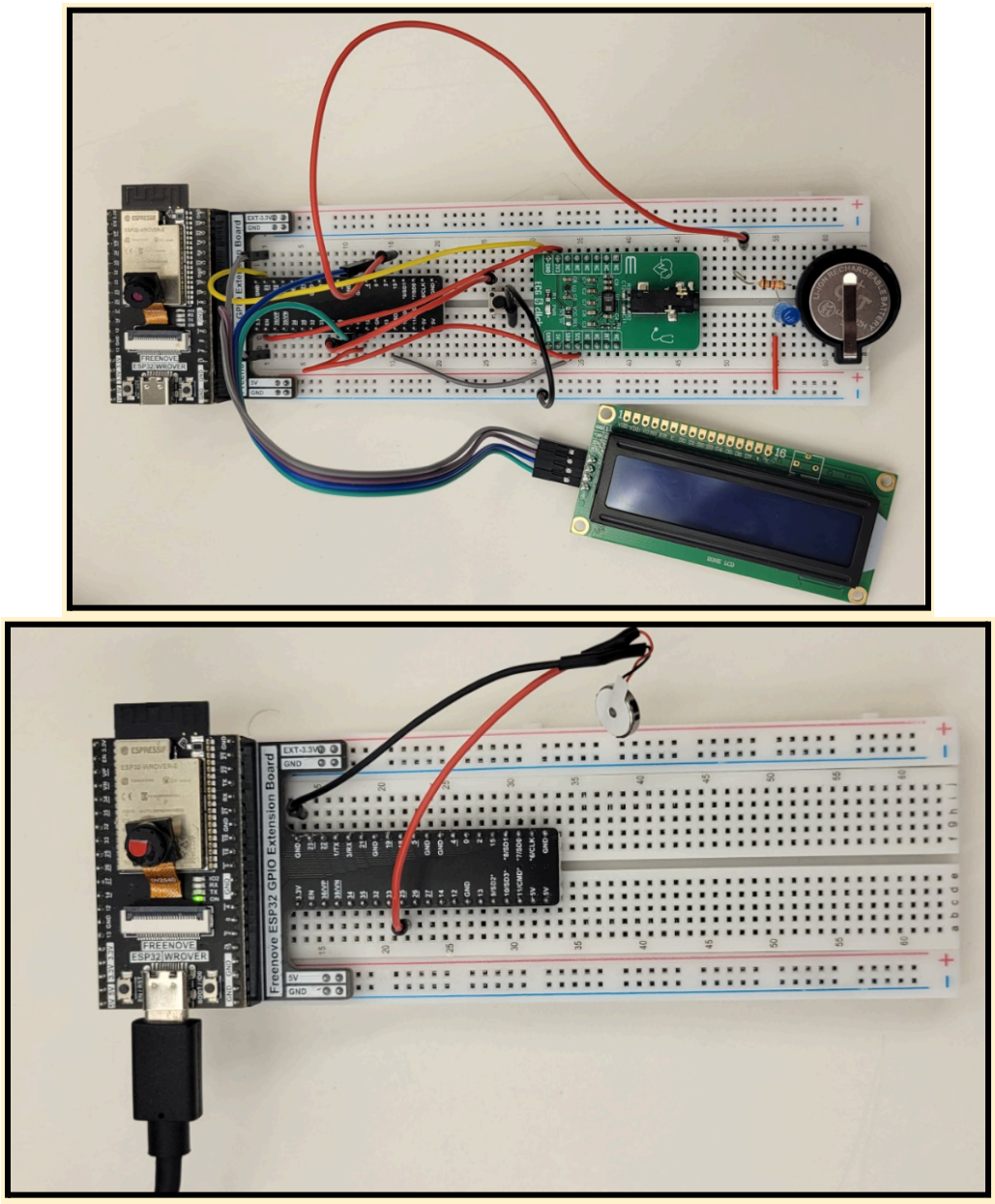


Figure 5: Previous Senior Design Team Implementation

4.1.3 Technical Complexity

Our project included technical complexity through the integration of multiple subsystems between the two devices and the programs that were run on them. The user side devices incorporated a wearable PCB featuring an ESP32-S3 microcontroller and a MAX86150 sensor that is capable of photoplethysmography (PPG) and electrocardiogram (ECG) measurements. This device also included multiple voltage regulation and battery charging circuits. The service animal device receives bluetooth low energy signals and in turn drives ERM vibration motors via discrete MOSFET switching. These two systems are capable of iterating completely wirelessly.

The design apple embedded systems engineering, real time signal processing, power management, and wireless communication protocols. Data sampled from the PPG/ECG sensor is sampled and interpreted using algorithms that extract heart rate and estimate blood pressure. This project aligns with IEEE standards for wearable medical devices and low power wireless communications.

In all this project included multiple challenging requirements that match and exceed current solutions or industry standards.

4.2 Design Exploration

4.2.1 Design Decisions

ESP32 -S3

This is a microcontroller with integrated 2.4 GHz Wi-Fi and Bluetooth capabilities. It is being used for the user wearable subsystem. It will serve as the central control for our project, collecting and processing data from our selected sensors. The decision on this specific microcontroller was based on its Bluetooth capabilities. This wireless communication is vital for our project to function as proposed because there needs to be a method of dialogue between microcontrollers to exchange the information that the user is in distress and thus needs comfort from their service animal. Without this wireless capability, the service animal microcontroller would need to be connected through wire to the user's wearable. This creates inconvenience and becomes an obstruction to lifestyle.

DC ERM Motor Vibration

This vibration device will signal to the service animal that their owner is in distress. Many different design options came up for this decision. Between DC and AC power, as well as the mechanical design choices. This is a human and discrete method of signaling to the service animal that their owner needs comfort.

MAX86150 - Integrated Photoplethysmogram and Electrocardiogram

This is dual-integrated with a PPG and EKG sensor. The PPG detects blood pulses, and the EKG detects electrical pulses corresponding to heartbeats. This sensor has lots of built in configurability including pulse frequency scaling, LED bandwidth adjustment, LED pulse strength, and continuous sampling.

4.2.2 Ideation

When deciding on a microcontroller for the project we considered the following options

- ESP32-WROOM: Good Bluetooth, cheaper, but larger and older model.
- ESP32-S3-WROOM-N4-1: Smaller, BLE 5.0 capable, better for wearable use (Chosen).
- Arduino Nano 33 BLE Sense: Easy programming but poor processing power.
- Raspberry Pi Zero W: Powerful, but large and power-hungry for wearable.
- Nordic Semiconductor nRF52840: Great BLE support but more expensive and harder to prototype with.

We selected the ESP32-S3 for its balance between size, cost, and Bluetooth capabilities.

4.2.3 Decision-Making and Trade-Off

When selecting the microcontroller that would be best for out design we considered the following criteria:

- Cost
- Bluetooth capability
- Power consumption
- Size/Form Factor
- Ease of Development

Each category was weighted based on importance:

- Cost:20%
- Bluetooth capability: 25%
- Power consumption: 20%
- Size/form factor: 20%
- Ease of Development: 15%

Option	Cost	Bluetooth	Power Consumption	Size	Ease of Development	Total Score
ESP32-WROVER	8	7	6	6	8	7.05
ESP32-S3-WROOM-N4-1	7	9	7	8	9	8.15
Arduino Nano 33 BLE	6	8	8	7	8	7.45

Raspberry Pi Zero	5	9	4	3	7	5.7
Nordic nRF5280	5	10	9	8	5	7.2

Figure 6: Weighted Decision Matrix

Based on the results of our decision matrix we chose to use the ESP32-S3-WROOM-N4-1.

4.3 Final Design

4.3.1 Overview

Our prototype went with our own physical design components that would work together with our software for PTSD detection and machine learning algorithms to accomplish our task of comforting the user. Our group went with from the ground up PCB designs that share lots of common pieces to streamline our purchasing and development process. While other components are unique to their specific devices. For bluetooth we would use the on board antenna for each microcontroller to send data to one another. We used industry approved heart rate and blood pressure algorithms to increase our accuracy in PTSD detection. We then developed an algorithm that takes the PPG-based heart rate and blood pressure values and decides if the user is experiencing a PTSD attack. Our design is relatively complicated and will be discussed in the following sections.

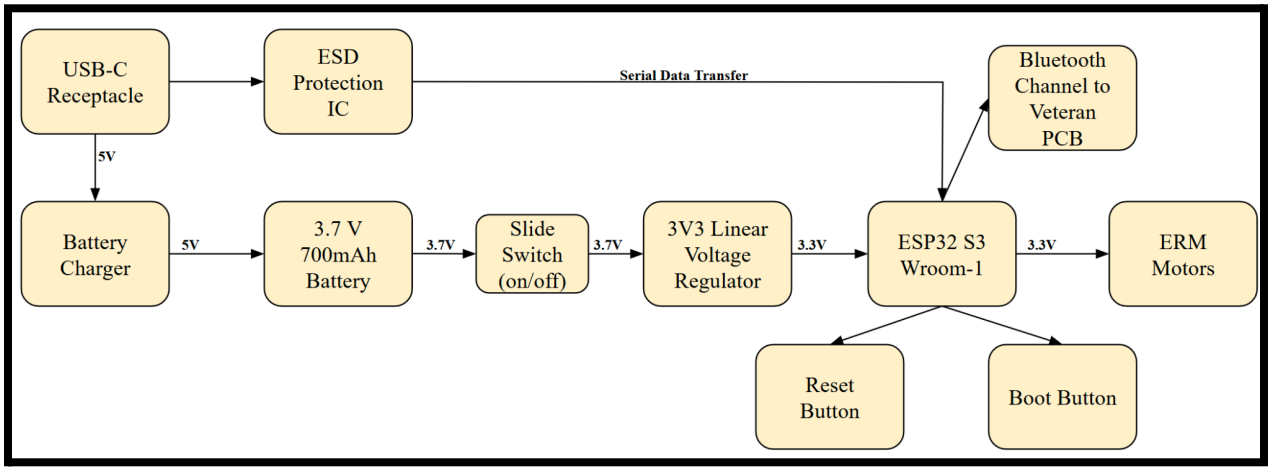


Figure 7: Block Diagram for Service Animal

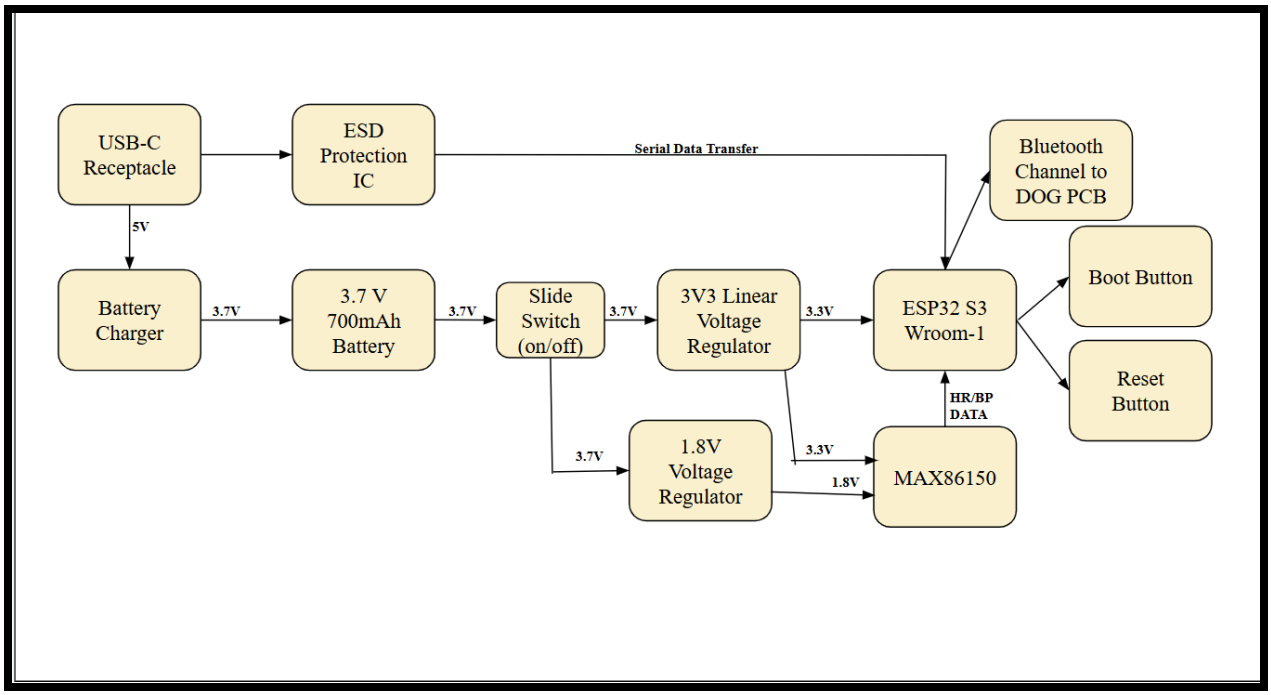


Figure 8: Block Diagram for User

4.3.2 Detailed Design and Visuals

Joint Design Components for PCBs

Both PCBs in our design use many of the same components in our basic designs from the battery to the microcontroller. In the high level block diagrams in the overview section you can see the similarities and differences between both our devices. Our main design decision was to pick the ESP32-S3-Wroom-N4-1 as it operated almost identically to our first semester design, but had a few more perks for ease of design. A major factor in this decision was the fact that the new microcontroller has built in data lines for our USB type C choice in programming. That choice would remove the need for a separate USB to UART bridge that would be required for the ESP32-Wrover-E. The USB type C was another easy decision as it would then combine perfectly with our current microcontroller. We used an ESD protection IC to make sure our data lines have smooth communication to the microcontroller. We would also use the 5V line to charge our battery when the user plugs in a charging USB cord. Our 3.7V battery requires a compatible charging IC that has built-in temperature sensing to cut off charging to the battery when it overheats. The battery charging IC has a built-in charging LED that shows the user when the battery is charging when to unplug when the battery is sufficiently charged. We chose a smaller base quick charge value as that would ensure that the battery would not overheat. Since the microcontroller requires 3.3V for nominal operation, that would require our design to incorporate a 3.3V linear voltage regulator. This would step our battery voltage down for usable levels on the microcontrollers and the PPG sensor located only on the veteran pcb. The PCB will have an on and off switch located between the battery and regulator. Our group recommends that the device

be turned off while charging. Each microcontroller has a Boot and Reset button to allow for smooth testing when we upload our software. Each microcontroller has proper strapping pins which are properly pulled high or low. The PCBs are also equipped with extra GPIO output pins, if we were to connect an LCD display or another device to our microcontroller.

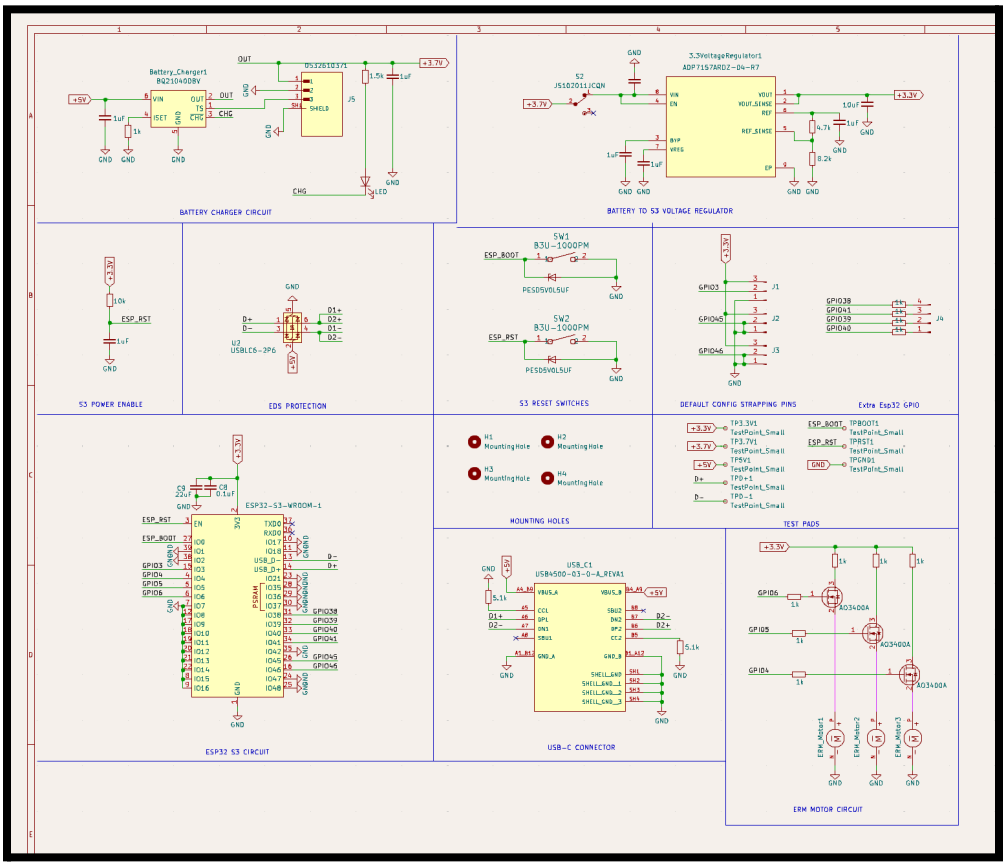


Figure 9: Schematic of Service Animal Device

Design for Service Animal

For the service animal the unique components are clear as the erm motors used to alert the service animal are required. The circuitry used to turn on the motors is very simple as we would use a mosfet as a gate that would turn on the motor and off. This would be a big change from our original first semester prototype as we would directly turn the motor on and off via a pin on the microcontroller which would send too much current through that pin. This would potentially damage the microcontroller. The schematic above shows all of the parts associated with the joint components between our devices. One major difference between our devices is the amount of test points that would be required.

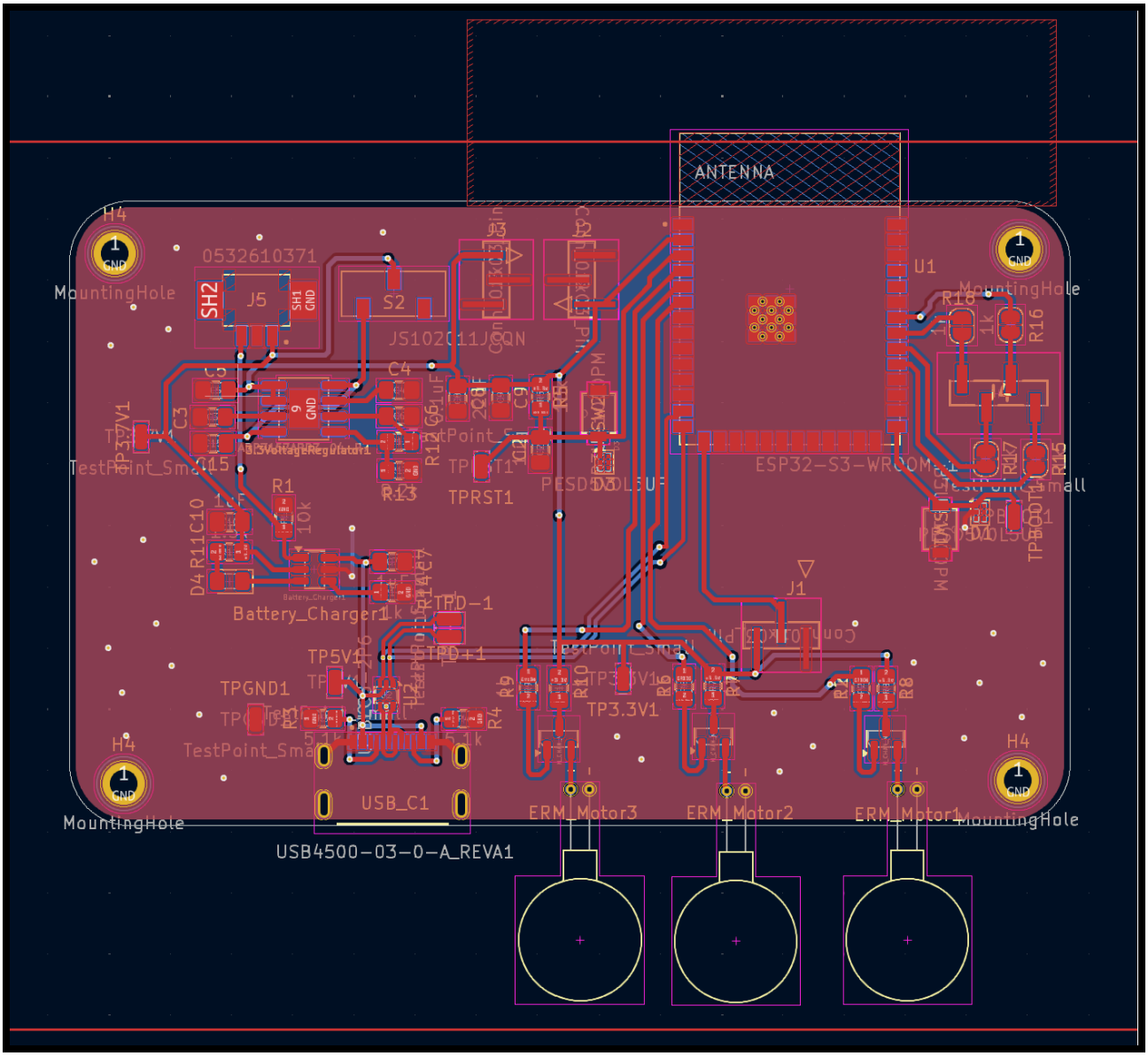


Figure 10: PCB for Service Animal

PCB for Service Animal:

This custom PCB seen in **Figure 10** is designed to support the wireless control of ERM vibration motors. This PCB will work as the client side of the Bluetooth communication, receiving a signal and turning off or on the ERM vibration motors. The board includes power regulation to supply a consistent 3.3V to the ESP32S3 and peripherals.

Power is delivered via an external 5V battery source and then stepped down using a voltage regulator. Decoupling capacitors are placed near the microcontroller’s power pins to suppress voltage fluctuations and maintain stable operation. Protection components such as reverse polarity diodes and capacitors are also included to increase power reliability.

To ensure strong and consistent BLE performance the antenna area of the microcontroller is left off the edge of the board ensuring that there are no copper and signal traces within the keepout region. This helps minimise electromagnetic interference with the signal propagation. The board layout also includes LEDs to indicate power on, BLE connection and charging status. The PCB is laid out as a two-layer board, this is best for cost efficiency while maintaining signal integrity. These layers consist of a ground plane and signal plane.

Mounting holes are included and grounded so it can be securely mounted inside of a case that will be located in the service dogs vest. The dimensions of the board are approximately 3 inches by 2 inches. This allows the device to be secured in all types of service animal vests.

Also included in the board is additional connections that give the opportunity for the addition of more ERM vibration motors. This gives the user flexibility in how strong of a vibration is needed to alert the service animal.

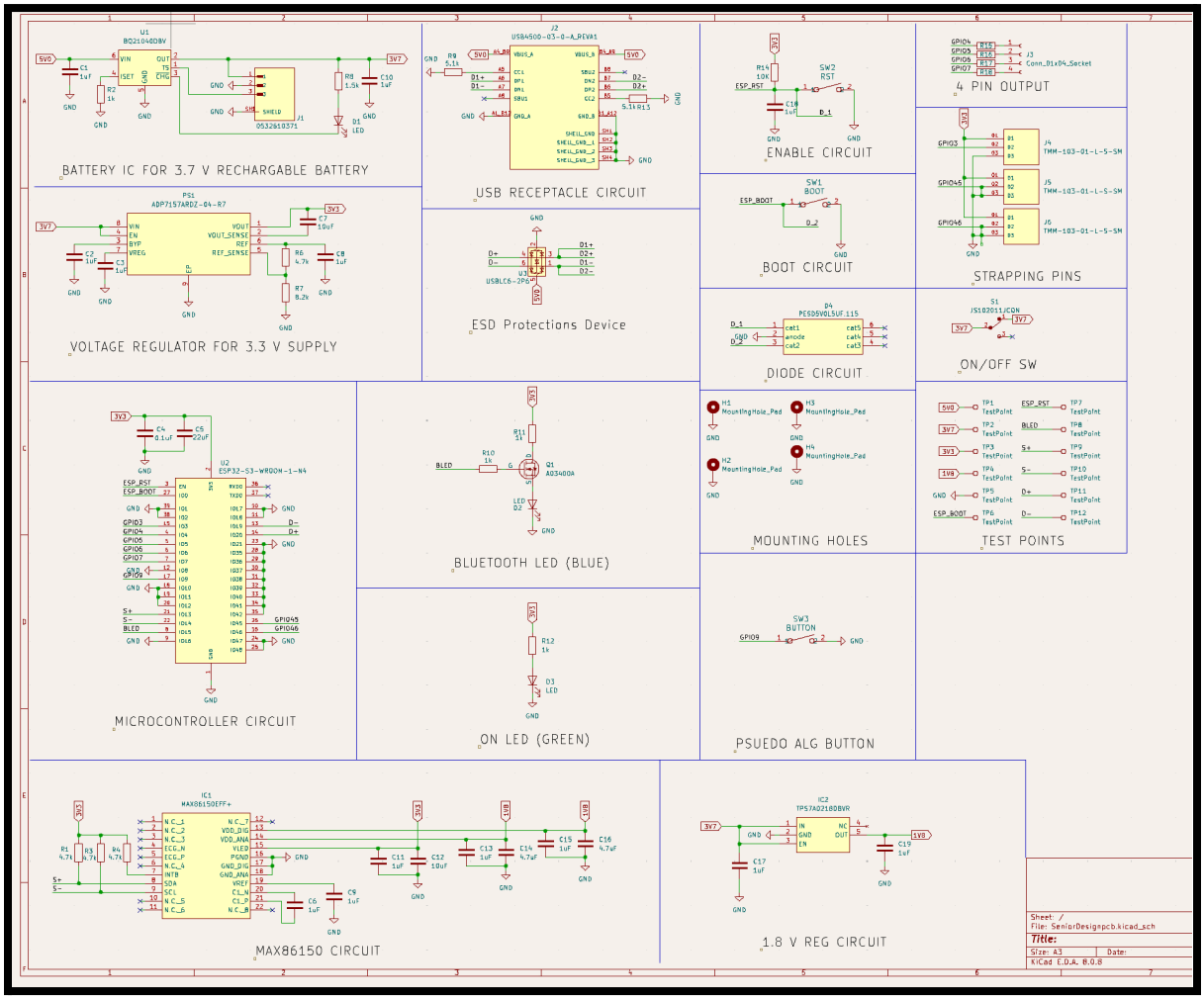


Figure 11: Schematic of User Device

Design for User

The schematic diagram for the user device is much more complicated compared to the service animal's device. The following will discuss the differing parts compared to the service animal's design.

The schematic diagram integrates the MAX86150 sensor into the design with that required many different design requirements. The first is the I2C setup that requires the data lines of the sensor to be pulled up to 3.3 V by a few 4.7 k Ohm resistors. The sensor also required a 1.8V supply for some of the pins. To cover the 1.8V we used a 1.8 V regulator that steps the battery voltage from 3.7V to 1.8V to be used by the sensor. Many of the supply pins on the sensor required decoupling capacitors of varying capacitances.

The design would incorporate a few different LEDs to give some varying ways the user can interact with the device. The first LED is a simple on and off LED that is connected to the 3.3V supply after the slide switch. The second LED is a blue Bluetooth LED to tell the user that the devices are properly connected. The circuit for the blue LED operates quite similarly to the ERM motors on the dog in which the blue LED uses the mosfet transistor as a gate to then turn the LED on.

The user device would also incorporate the pseudo button for detecting the PTSD attack that would be directly to ground so the microcontroller can detect when the button is pressed. The button pressed action will be discussed in the bluetooth connection section.

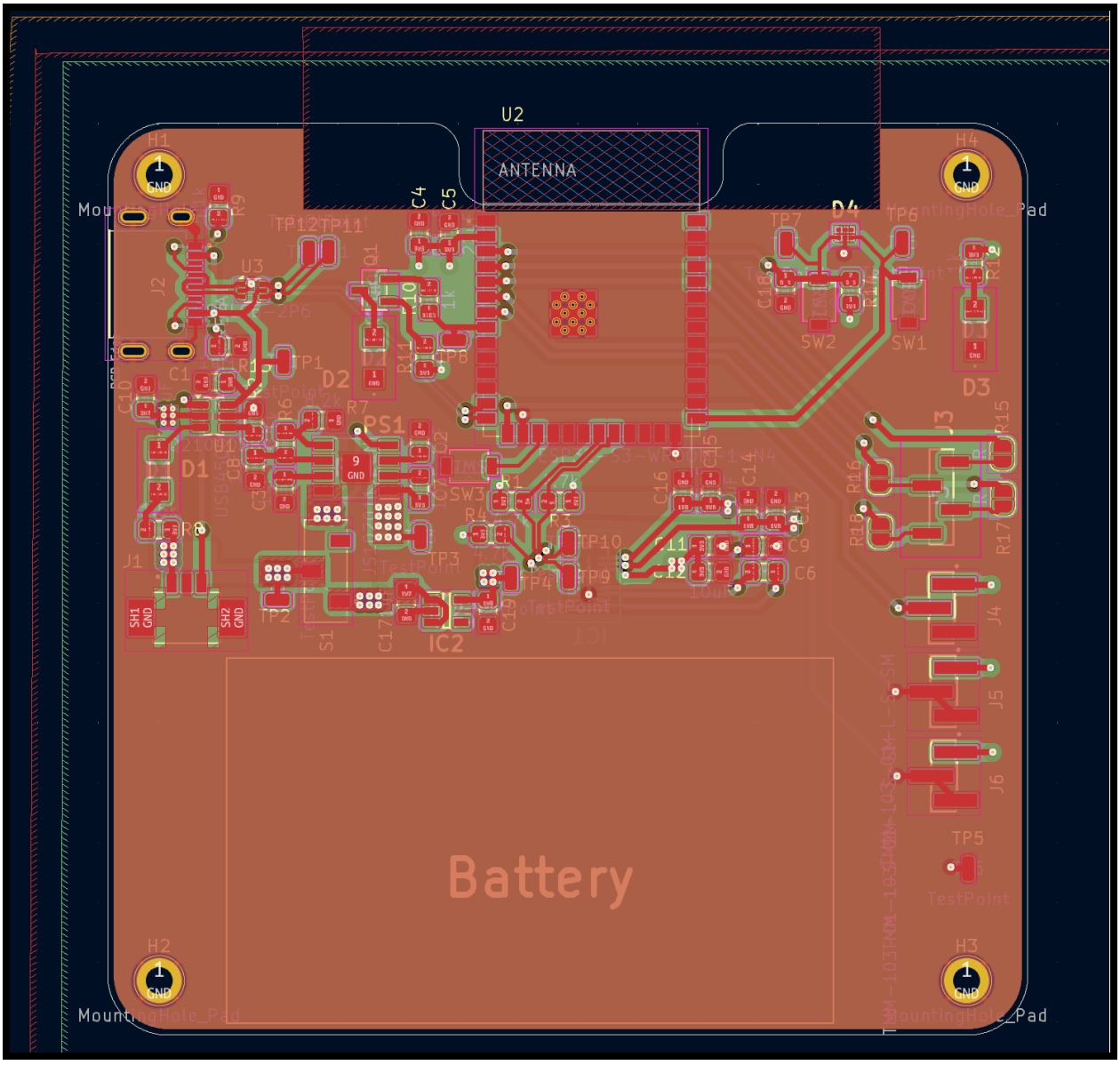


Figure 12: PCB for User

PCB for User

The PCB for the user was far more complicated than the service animal pcb as it would require 4 layers to make designing the board far easier. The layers go as follows: first was signal ground, second was 3.3V, third was ground, and the bottom layer would also be signal ground. The first layer would be used for most component mounting from the microcontroller to the battery. The bottom layer would be used for mounting the PPG sensor on the board. The 3.3V layer would be used to give 3.3V power to all the components that require that voltage. The grounding layer is used to have a grounding plane that can be used at any time and make sure whether there is a component on the top or bottom of the PCB. The PCB was designed for 3in. x 3in. because this was our initial PCB design and the group wanted a proof of concept.

The PCB has four mounting holes for easy mounting into a case if we had more time to develop. THe mounting holes are appropriately grounded to prevent short circuiting. The PCB also comes equipped with testing points for all signals that would require them.

The battery will be mounted on the surface of the PCB and would be directly plugged into the receptacle female header that is soldered directly on the board. The boards overall design attempts to have most of the parts near the USB type C and would cascade towards the PPG sensor as the voltage magnitudes would decrease. Each of the main power sources that are voltage outputs have multiple vias that help disperse the potential changes in temperature.

Both the I2C and Data lines for the PPG and USB type C require them to be differential pairs to make sure there are no disruptions or differences in the length of the traces. All traces for the PCB bottom out at 0.3 mm in tighter locations and maxed out at 0.5 mm. Many of the power traces were planes of differing sizes.

Bluetooth Implementation

Our devices use a simple program to implement a Bluetooth Low Energy (BLE) communication system between two ESP32S3 modules, one acting as a server and the other as a client. The client continuously scans for nearby devices advertising a specific UUID. Once a matching UUID is found the client establishes a connection with the server allowing notifications from a designated characteristic.

The server monitors a digital input (a push button or PTSD symptom) and sends a notification with a predefined message, “ALERT” or “No Alert”, based on the status of the digital input. One the client side, the notifications are handled using a callback function. If the message is “ALERT”, the client activates a set of GPIO pins to power ERM(Eccentric Rotating Mass) vibration motors. IF the message is “No Alert”, those GPIO pins will remain low and the ERMs are turned off.

The program features serial monitoring for real-time debugging and status reporting. BLE operations such as scanning, connection handling, and characteristic identification are managed through the ESP32 Arduino BLE library. Additionally connection status is monitored through LEDs identifying whether a connection has been made.

PPG Heart Rate and Blood Pressure Detection Algorithms

Our heart rate monitoring uses raw PPG data samples in 6 second windows from the MAX86150. The data is sampled at 100hz, therefore producing 600 19 bit ADC values corresponding to the pulse strength returned to the MAX86150’s IR sensor. By analyzing the raw waveform, one can observe that peaks correspond to the user's heart beats.

Detecting beats in a PPG waveform can be a challenging task, especially on microcontrollers. To combat this issue, we utilized the publicly distributed MSPTD algorithm, which is capable of

accurately detecting peaks and valleys in a PPG waveform. In order to use this algorithm on the ESP32 microcontroller, we needed to convert from Matlab to C code, and perform some optimizations that reduced the memory footprint of the algorithm. This included removing the ability to detect valleys in the waveform (as these are not needed for heart beat detection), and changing any dynamic memory allocation to be static.

Once peaks have been detected, we check to see if the number of peaks falls into a reasonable range. This is because the algorithm can occasionally miss peaks corresponding to heart beats, and missing too many of these peaks can lead to unreasonable heart rate values. If it is determined that the number of peaks falls into a reasonable range, heart rate is calculated by finding the average time between beats, taking the inverse, and multiplying by 60 to get the beats per minute.

Detecting blood pressure purely from PPG data is known to be a difficult task, and is still a research area in its early stages. This problem is even more difficult when considering the limited compute resources available on an ESP32 microcontroller, thus eliminating many of the most common solutions involving very large machine learning models and deep neural networks. To solve these problems, our team designed a novel 1D Convolutional Neural Network (CNN) architecture, which is able to estimate a patient's systolic blood pressure based on a 2.1 second sample of data.

Our neural network architecture was trained on the PPG-BP Database, which contains PPG and corresponding blood pressure samples from 210 patients. A significant amount of preprocessing needed to be done for this database to train a model that was compatible with our microcontroller and PPG sensor.

We started off by downsampling the database samples from 1KHz to 100Hz. Next, we normalize the values by subtracting the mean of each sample and dividing by the standard deviation. This was necessary because the datasets use 12 bit PPG values, but our device uses 19 bit values. Additionally, this normalization helps the training pipeline extract features relevant to the patient, rather than qualities of the PPG signal such as how close it was to the user at the time of recording. Lastly, the preprocessing finds the first and second derivatives of the PPG signal. This is done to add more information to the training pipeline, and allows for better feature extraction, as was verified through training and testing.

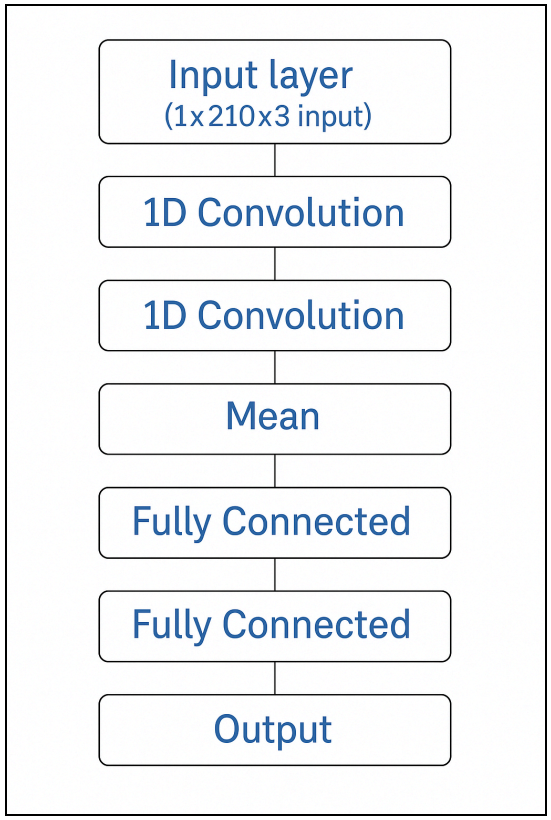


Figure 13: Neural Network Architecture

The neural network's architecture is pictured in figure 13. This architecture implements a classic 1D CNN, involving three convolutional layers, a global mean layer, and two fully connected layers, with the second fully connected layer yielding the output of the model, which is the patient's predicted blood pressure.

For training, the dataset was split into train and test data, with 80% of the original dataset being used for training. The network was trained using a Huber loss function, and was trained for 50 epochs. When evaluating the model on the testing dataset, we found the mean absolute error to be about 12mmHg, which we determined was sufficiently low for our use case.

PTSD Detection Algorithm

For PTSD detection, we wrote an algorithm that tracks heart rate variability in real time and detects when there is an abnormal change. We do this by calculating the inter-beat intervals and keeping a buffer of the latest 300 values in a buffer. These values would be calculated from the BPM measurement taken each second by the PPG sensor and its heartrate algorithm. Each time a new value is added to the buffer, if it's full, the oldest value kept track of by an index is deleted and replaced by the new value. The baseline values are then updated using the new values which are used to calculate average BPM, a standard

deviation of IBIs and a root mean square of differences in successive IBIs. Once enough data is in the buffer, the latest data is analyzed and compared to the overall baseline data. If there is an abnormally high heart rate or a drop in the root mean square of differences or the RMSSD/SDNN ratio, the alert is triggered and sent to the dog device.

4.3.3 Functionality

DOG PCB
The service animal side of the PCB implementation works with the real world user by having a rechargeable battery and ERM vibration motors that work as a notification to the service animal. The notification is received from the USER PCB.

USER PCB
The USER PCB works with the real world user by having connections that can be used for battery recharging. The PPG sensor interacts with the user by taking samples of heart rate and an estimated blood pressure. The device then sends a bluetooth signal to the service animal PCB.

Heart Rate and Blood Pressure
The heart rate and blood pressure data is gathered by a physical contact with the user by a PPG sensor. This sensor is connected physically to the PCB design on the user side PCB.

PTSD Detection
The PTSD detection algorithm uses heart rate data collected by the PPG sensor. This data is then alalyzed on the ESP32 on the user side PCB.

4.3.4 Areas of Challenge

DOG Design
For the design of the device that would be located in the dog vests some of the challenges that we faced when designing were size constraints as well as battery life. We were able to successfully develop a PCB that will be small enough to fit in a small case inside of the standard America's Vetdogs dog vest. Battery life was addressed by adding a larger lithium battery to our design that allows for recharging and has a much longer battery life. Another design obstacle when creating this device was making sure we had the correct number of ERM vibration devices in order to create a pulse that is able to be felt by the dogs. To make sure that we had done this properly we set up additional pins that would make it easy for the addition of more motors.

USER Design
The main constraint when creating the wearable device for the user was making sure that the device was small enough to be able to be worn as well as ensuring that the PPG sensor would make contact with the user in order to extract data. Similar to the design of the dog device we were able to develop PCBs so that they were reasonably sized. The intention is that next iterations of design can use the same components at

a smaller scale to create a watch sized wearable device. This would satisfy the user requirements to be small enough to be discrete.

HR and BP

For heart rate detection, the largest challenge faced was the ability to run the MSPTD beat detection algorithm on the ESP32. This algorithm (and many similar algorithms) have a high memory overhead, sometimes requiring over 1MB of dynamically allocated memory. The original version of MSPTD we implemented required an amount of memory larger than could be reliably allocated on our microcontroller, thus leading to frequent code crashes. To combat this, several techniques needed to be employed, which included downsampling our signal, removing valley detection, and creating a statically allocated array for processing, whose size could be fixed given our 6 second sample.

Creating a blood pressure detection algorithm from solely PPG data posed a challenging task, especially given the resource constraints of our microcontroller. There are a few main approaches that have been published on solving this problem, but our research found that none of these approaches met both our accuracy standards while also being suitable for running on a microcontroller. Our solution to this problem was to develop a new approach for recognizing blood pressure from PPG data by using a 1D Convolutional Neural Network. These networks have been shown to be effective for recognizing patterns in time series data, and are generally very lightweight - making them an ideal candidate for our use case. We were able to use this approach to create a custom network that met both our accuracy and resource constraints.

PTSD Detection

Developing an algorithm for PTSD detection had many challenges and brought up many questions. The largest problem was that everyone has a different body that comes with a different resting heart rate as well as different levels of cardiovascular health. This meant the algorithm had to be able to adapt in real time to different users which brought up the second challenge which was ensuring that the microcontroller could handle all the real time processes. Memory also had to be managed so the program could continue to run as long as the user was wearing the device without crashing due to memory leaks or filling up the memory of the ESP32.

4.4 Technology Considerations

DOG Design

The service animal side of the system was designed to prioritize reliability, low power operation, and mechanical simplicity. The core microcontroller, the ESP32-S3, was chosen for its integrated Bluetooth Low Energy (BLE) stack, sufficient processing capability, and compact footprint. The ERM motors used for notification are controlled through MOSFETs.

The design offers modularity by including multiple motor drive outputs to accommodate dogs with varying fur thickness or size. The PCB layout also includes a dedicated BLE antenna keep-out region to

minimize electromagnetic interference and maintain signal integrity.The board is powered via a single-cell lithium ion battery, stepped down through a voltageregulator to provide a stable 3.3V rail for the microcontroller. Charging is handled by an on board charging circuit that includes a charging LED indicator.

One limitation is the lack of sensory feedback on the dog-side device, which currently only functions as a passive receiver. Additionally, while the system performs reliably at the targeted BLE range of 15 feet, maintaining consistent connection in high-interference environments remains an area for future development.

USER Design

The user side PCB includes several systems that work together to collect and respond to data in real time. The MAX86150 sensor which measures heart rate and ECG signals communicates with the ESP32-S3 microcontroller. The system includes both 3.3V and 1.8V power rails for the sensor. These voltages are provided by a 3.7V lithium ion battery and a set of voltage regulators. Charging is handled by an on board charging circuit that includes a charging LED indicator.

The PCB also includes LEDs used to show that the device is powered on and another to show that connection to the dog side devices has been established. In the current design there is a button included to send test signals to the dog side device. This can be used for training as well as testing to make sure that a connection is properly established. The ESP32-S3 supports USB-C programming which allows for simple testing and code updates.

One of the main challenges is estimating blood pressure from the PPG sensor without the use of a cuff. Another consideration that could be seen as a challenge is power, since the device is continuously sensing and communicating via bluetooth this can drain the battery quickly. Reducing how often data is collected and sent can help balance battery life.

HR and BP

For heart rate detection, we chose to use a PPG based approach as opposed to an ECG. The main advantage of this approach was that acquiring data from the user was less invasive as compared to an ECG. In order to get live sampled ECG data from our patient, two distinct points of contact would be needed. As our device is designed to be wrist wearable, this becomes an impractical constraint to satisfy while keeping the device small and practical.

The weakness of relying solely on PPG data is that PPG waveforms are often less accurate for detecting heart rate and irregular heart beats compared to an ECG. As a consequence, our beat detection algorithm can occasionally miss heartbeats, leading to some inaccuracies in heart rate detection. However, some simple logic can be added to recognize when beats have been missed, and adjust heart rate predictions accordingly.

PTSD Detection

For PTSD detection, we used a buffer containing the latest heart rate data coming from the PPG sensor. This allowed for real time processing and the ability to adapt to each user. The data from each person allows the algorithm to establish a baseline that changes over time as new data enters the buffer. One

problem with this algorithm is that it is not very efficient as it is doing new calculations for every new piece of data it gets. The speed at which new data is acquired is also something that could be improved by the heart rate detection algorithm. The PTSD detection algorithm also lacks evaluation of blood pressure data due to time constraints which could make our algorithm more accurate.

Security

In regards to security, we have based our decisions on the NIST SP 800-121, Guide to Bluetooth Security publication along with the controls recommendations in NIST SP 800-53, Security and Privacy Controls for Information Systems and Organizations. We took a top-down approach when considering security for our system, identifying the most likely attack vectors first, analyzing the existing capabilities of our hardware next, reviewing relevant standards, and then finally reviewing our code. With the fastly evolving nature of our project development, we have fully fleshed out our security design expectations and theoretical design but found actual testing difficult.

We have identified these items as standards for our device and its future development:

High Importance

- Unneeded Bluetooth services and profiles should be disabled
- The device should be undiscoverable until needed for pairing

Medium Importance

- Default Bluetooth settings should reflect the security policy of the company or organization.
- The device should be set to the lowest necessary and sufficient power level in order to keep access to transmissions limited to the intended user(s). This was determined to be of medium importance because of the nature of Bluetooth LE and our device being body-worn. The size limitations will significantly hinder the possibility of this being a worry in public. Our testing figures explained in section 5.8 confirm this to a point that we find acceptable. .

Low Importance

- Encryption keys should be configured to be as large as possible. We see this as low importance because of the nature of the data being transmitted. We are more worried about the integrity and availability of the data, with confidentiality being tertiary.

5. Testing

Testing was a critical part of our development process to ensure that both the user and dog devices functioned reliably and met the design requirements. Our goal was to validate each subsystem individually before combining them for full integration. Testing covered hardware functionality, software behavior, Bluetooth communication, and sensor accuracy.

We began by performing unit tests on individual components such as the ESP32-S3, MAX86150 sensor, and ERM motors. Interface testing was used to confirm reliable communication between sensors and the microcontroller. Integration testing ensured that the user device could collect biometric data and transmit alerts to the dog device. We also evaluated Bluetooth performance for range and signal stability. Power consumption and usability were tested under realistic use scenarios to ensure that the devices could operate throughout a typical day on a single charge. Results from each test phase helped us identify and resolve design issues early, improving system performance and reliability. The following sections detail the specific tests we performed and the outcomes.

5.1 Unit Testing

ESP32 S3 Wroom-1

- Used to test the Bluetooth code
    - Originally set up a phone Bluetooth connection to see if alert messages are sent successfully.
    - Added a blue LED to decipher if the microcontroller has successfully connected to the other microcontroller.

Printed Circuit Board

- As risk mitigation, a testing prototype was built on a breadboard with the equivalent components to those on the PCB.
- Dog side of the PCB, built in sections.
    - Starting with the microcontroller and testing the base wiring.
        - Spotted at issue when a previously tested code would not load to the microcontroller. The problem was that the Tx and Rx pins were grounded versus left floating.
    - Testing the boot and reset pins.
        - Once buttons are implemented, it should boot and reset the microcontroller as expected.
            - Testing by loaded code and examining the serial console.
- Veteran Side of the PCB, built in sections.
    - Testing the bluetooth connection between modules with the bluetooth LED indicator
    - Testing the battery charger IC for:
        - Quick Charge current is below the maximum limit
        - Charging LED functions properly
    - Testing the pseudo button to ensure the microcontroller detects a pressed state.
    - Testing the PPG and USB type C so the raw data is reaching the microcontroller properly

    - Testing both voltage regulators (3.3V and 1.8) to confirm they operate at the correct voltage

MAX 86150

- Heart Rate Acquisition
    - Algorithm ran on a microcontroller was compared to an apple watch heart rate and medically certified blood pressure cuff heart rate.
    - Testing showed that heart rate was within 2 BPM of both devices.
- Blood pressure
    - PPG-BP dataset was split into training and testing data, neural network accuracy was verified on testing data.
        - Accuracy for this testing data, as measured by mean absolute error, was 12mmHg
    - Live inferences were also tested against a medically certified blood pressure cuff.
        - Accuracy for live sampling was shown to be within 8mmHg.

Bluetooth LE

- What is the range of the Bluetooth in feet? Tested by distance test between devices.
- Does the Bluetooth connection continue through walls? Tested by putting a wall between the client and server devices.
- Does the LED for Bluetooth turn on properly for an active secure connection of the microcontrollers? Visual test for LED turning on.

Power Consumption

- The length of time it takes to deplete the batteries. Tested from taking current from a multimeter and calculating the amount of time it would take to deplete the battery.
- The devices were left on for a duration of 8 hours and checked throughout the day to make sure that connection was maintained.

5.2 Interface Testing

MAX86150 PPG Sensor to ESP32 Microcontroller

- Utilizes I2C to communicate between the modules
- Wire library in Arduino supports simple and reliable interface to I2C peripherals
- Data acquisition was tested through examining consistency in RAW ppg data to microcontroller
- Interface utilizes acknowledge (ACK) and not-acknowledge (NACK) bits to ensure each byte was received correctly

Bluetooth to interface between ESP32

- Uses Bluetooth low energy library in arduino
- Testing connection with phone as well as other microcontroller
- Successfully sending messages (characters) over Bluetooth

PTSD Detection Algorithm

- A stream of heart rate data is fed to the algorithm from
- It successfully made the correct calculations needed to detect a change in heart rate.
- Use with a test stream of data shows correct flags being thrown at the appropriate times.

5.3 Integration Testing

MAX 86150 PPG sensor to ESP32 WROVER E microcontroller

- Does the sensor send accurate data to the microcontroller? Tested by using a wristwatch with its own sensor to test accuracy.
- Does the microcontroller display the data properly on a serial monitor or LCD display? Simple visual test with debugging if it does not work properly.

ESP32 Wrover E Bluetooth communication with each other

- What can cause the Bluetooth connection to disconnect:
- Do the microcontrollers communicate properly over long distances? Can be tested by walking far distances between microcontrollers.
- How many walls can the connection be active for. Testing connection between walls in case our service animal is in another room without the veteran.

5.4 System Testing

- Our microcontroller processes the data from the PPG sensor accurately in real time for heart rate. The microcontroller displays that heart rate data properly onto the LCD display.
- Our microcontrollers connect via bluetooth LE without being plugged directly into a laptop. The microcontroller sends the command for the vibration device to turn on over the established Bluetooth connection.

- Main program with integrated BP and HR monitoring is able to run on the patient wearable microcontroller, with long running testing showing that crashes/unexpected behavior are avoided.

5.5 Regression Testing

Bluetooth Code

- We changed our microcontroller from the Wrover-E to the S3-Wroom-1 model.

Adding BP monitoring to HR monitoring

- We reduced dynamic memory allocation to increase the probability that two algorithms do not fight for limited ESP32 memory.
- We tested the integrated code base for long durations to ensure program crashes did not occur.
- Set up memory monitoring to ensure stack/heap collisions never occurred.

5.6 Acceptance Testing

How did you demonstrate that the design requirements, both functional and non-functional were being met? How did you involve your client in the acceptance testing?

Patient Vital Signs Monitoring

Ensuring heart rate met requirements

- Set goal to recognize heart rate within 5 bmp of true value
    - This constraint supports accurate detection of heart rate spikes.
- Tested heart rate monitoring was shown to be within this range.
- Accurate and reliable heart rate monitoring was shown to be supported by our system.

Ensuring BP met requirements

- Set goal to recognize systolic blood pressure within 15mmHg
    - This constraint allows better verification of PTSD attacks when heart rate spikes occur.
- Tested blood pressure monitoring was within this range, both on testing data set and live user data.
- Reasonably accurate blood pressure monitoring was shown to be supported by our system.

5.7 User Testing

How did you test whether your design addresses user needs? How did you involve your users? What were

their reactions? What were your observations of users interacting with your design?

Wearability

The device will not reach the goal of being wrist worn but will be worn on the bicep as the PCB for the user. We tested the PCB on the bicep and it would be relatively comfortable for long time use.

Non-disruptive

The device is still not disruptive as we have not incorporated any disruptive elements to our newest prototype as we still use the ERM motors to communicate with the dog. The PCB for the dog also fits comfortably into the dog vest pocket.

User Interaction

The parts of the device the user interacts with is plugging the device in to charge, turning the device on and off, placing the device on their arm, and reading the LED indicators for proper function.

LED indicators

In order for the user to interact with the device properly they must know if the device is working as expected. The finished prototype has 3 LEDs. A power led to show when the battery dies. A bluetooth LED to let the user know the device is working properly. A charging LED to make sure the user knows when the battery is sufficiently charged. All LED circuits were tested for proper operation.

5.8 Results

Bluetooth Connection Results

- The BLE connection successfully maintained stability for up to 15 feet including connection through interior walls.
- Initial pairing takes about 6 seconds on average and reconnection after temporary loss is consistent.
- The dog device properly received alert messages with a minimal delay after triggering from the user device

Reading patient vital signs

- Our testing on real users showed that heart rate monitoring could achieve similar results to state of the art devices such as an Apple Watch and medically certified blood pressure cuff.

- Device supported a minimally invasive user interface, relying only on PPG to acquire data.
- Our testing showed that our device could be used on real patients to gather meaningful data that could be used for detecting the occurrence of a PTSD attack?

Battery Life Testing Results

- Under normal operation both devices remained powered on for approximately 12 hours with the lithium ion battery.
- Continuous bluetooth data transmission results in a reduced battery life.
- Charging circuits functioned correctly with visual indication confirming charging status allowing the battery to be recharged safely..

6. Implementation

6.1 Design Analysis

Discuss how well your implemented design works? Describe what works well. Why does it work well? What evidence do you have to demonstrate that?

Describe what does not work well or work as expected. Why? What could you have done differently?

DOG Design

The service dog-side device met its primary design goals: reliable wireless communication, consistent haptic feedback through ERM motors, and compact integration for mounting in a dog vest. The BLE connection remained stable during real-world testing within the specified 15-foot range, and the antenna placement (extending off the PCB edge) helped minimize signal degradation.

Power regulation using a linear 3.3V LDO was effective but will benefit from a more efficient regulator in future iterations to extend battery life. Overall, the dog-side design functioned reliably, met key requirements, and is ready for refinement based on feedback from real service dog testing

USER Design

The user-side design was more complex due to its role in data acquisition, processing, and communication. The ESP32-S3 successfully handled Bluetooth transmission and sensor interfacing. Both 3.3V and 1.8V rails were delivered reliably using onboard regulators, and the MAX86150 consistently returned clean heart rate and PPG waveform data. However, real-time blood pressure estimation remains a limitation and will require further algorithm development and filtering techniques

The design met dimensional constraints for wearability, though future revisions could further reduce the footprint and improve ergonomics. Battery life was acceptable for initial testing, but continued optimization of sensor polling rates and BLE behavior will help ensure full-day operation.

HR and BP

The heart rate and blood pressure monitoring scheme has been shown to effectively provide an estimate of important vital signs for detecting the occurrence of a PTSD attack. In particular, the heart rate monitor has been shown to be very accurate when measured against an ECG, as well as a blood pressure cuff. This is a consequence of using what is the consensus most accurate beat detection scheme for PPG signals, rivaling the accuracy of state of the art devices such as an apple watch.

While the blood pressure monitoring still provides useful data for detecting a PTSD attack, its accuracy is still some ways away from that of a blood pressure cuff, or even some of the state of the art algorithms for extracting blood pressure data from a PPG signal. This can be mainly attributed to our severe resource limitation running code on an ESP32. The best BP detection algorithms published use advanced strategies such as deep neural networks with many hidden layers and weights. Not only do these models exceed the ESP32 memory capacity by orders of magnitude, but the time it would take to calculate a blood pressure using these techniques would be insufficient for detecting a PTSD attack quickly. As such, our algorithm aims to hit a middle ground between accurate BP monitoring and reasonable resource requirements, but this of course means that calculated blood pressure can be somewhat inaccurate.

PTSD Detection
The PTSD detection algorithm had shown successful testing in development, and was able to get input from the PPG sensor. The heart rate data was able to be used to establish an individual baseline and detect a change in real time. The PTSD detection algorithm unfortunately doesn't include evaluation of blood pressure data due to time constraints which would make it more reliable. Once blood pressure evaluation is added, the algorithm would be ready for real world testing. Overall, the heart rate detection is successful and is proficient in flagging the appropriate changes needed to alert the service dog of a possible PTSD attack.

7. Ethics and Professional Responsibility

7.1 Areas of Professional Responsibility/Codes of Ethics

Area Of Responsibility	Definition in our own words	Relevant item from Code of Ethics	Description of how our team has adhered to the code
Work Competence	Deliver accurate work in a professional way	ACM 2.1	Our team has strived to achieve high quality in both our design process and professional work
Financial	Deliver a product that	ACM 1.1	Our team is

Responsibility	is within the allotted budget		knowledgeable and respectful of the monetary design constraints given to us by our client.
Communication Honesty	Present work in an honest and transparent way	ACM 1.3	Our team has been honest and trustworthy throughout our design process
Health, Safety, Well Being	Reduce the risks to the safety and health of individuals using the product	ACM 1.7	Our team has been diligent in designing our product in a way that does not compromise users' health data.
Property	Honor and safeguard the property, ideas, and information of clients and others	ACM 1.6	Our team prevents the leakage of our users data to people that would seek to harm them
Sustainability	Preserve the environment and conserve natural resources on both local and global scales	ACM 1.2	Our team will not harm the environment that the device will be interacting with.
Social Responsibility	Create Product and Services that positively impact society and strengthen communities	ACM 3.1	Our team upholds the code as our product aims to positively impact the people we are developing our device for.

Figure 14: Table of seven areas of professional responsibility

One area that our team is doing well is communication honesty. We have been doing a good job of prioritizing communication in our assignments and delegating work throughout the project. This good communication helps us develop trust and reduces misunderstandings. However one area where we can seek improvement is in sustainability. Moving forward we should keep track

of our materials and work to incorporate more sustainable materials. This will ensure that our design aligns with the best practices that support sustainability.

7.2 Four Principles

7.2 Four Principles

	Beneficence	Nonmaleficence	Respect of Autonomy	Justice
Public health, safety, and welfare	Helps the quality of life of the handler. Can also mitigate public disturbances to do outbursts	The product is non-disruptive to the user's environment	The product is non-disruptive to the user's environment. And customizable	Promotes general well-being for a selective group
Global, Cultural, and social	Can improve the social life of the handler.	Does not harm any group or animal	Does not disturb cultural or religious practices	Create different wrist sizes wearables
Environmental	The product can be recyclable and made with renewable resources	Possible pollution from the development of products and materials	Create an eco-friendly option. Creating alternatives and choices	Will not disrupt the environment
Economic	Is an affordable solution	More affordable cost compared to current market alternatives	can create different variations for different costs	Will not break the beak in cost

Figure 15: Four Principles Table

Public Health, Safety, and Welfare are important principles for our project. Our project directly affects the welfare of veterans as this device is made to alleviate possible PTSD episodes. A group indirectly affected is bystanders around the veteran, since episodes can be abrupt, disruptive, or worrying to those unfamiliar with the disorder. We ensure these principles by giving veterans a device they can reliably trust to alleviate their symptoms. Extensive reliability testing and a design that keeps uptime and reliability in mind is key to our success in achieving these principles.

Respect for autonomy is a principle that is somewhat difficult to completely achieve given the nature of the project. Because the design requires that veterans are helped by a service dog that is in turn helped by the device itself, there is an implicit loss of autonomy when a veteran decides to use the device. However, we plan to alleviate this loss of autonomy by giving the user choices and control over the device. Also, despite the inherent loss of autonomy, this project should restore autonomy in other parts of the user's life by allowing them to do activities that PTSD has prohibited in the past.

7.3 Virtues

Three virtues that are especially important to our team are compassion, justice, and diligence.

As a team, compassion has to be a base for all of the work that is done on design and implementation. With a medical device such as this, there is always a human using and relying on our design for their well-being and safety. Specific to our project, a dog is also an integral user. To keep compassion at the forefront of our design, we have done extensive research on our users to anticipate their needs and potential problems with the device.

Justice is also an important virtue for this project. A goal of the project is to allow veterans with PTSD to be able to live lives equal to those who do not suffer from it. Achieving this goal to the best of our ability ensures justice by promoting equality and fairness despite differences in medical status.

Finally, diligence is important because it helps ensure tasks are completed thoroughly, on time, and to an acceptable standard. This helps to focus on the challenge at hand and pay necessary attention to detail. Doing so reduces the likelihood of big mistakes.We have shown this by constant communication of when assignments should be completed by and what is expected as to information that should be included. As well as we have many folders and sub folders helping aid in organization so files are easily accessible.

Katerina Zubic

Virtue Demonstrated: Friendliness

- **Why is it important to you?**
    - Friendliness is important to me because it helps promote communication, collaboration, and trust. This makes it easier for our team to feel comfortable to share ideas. Additionally, it helps create a friendly environment that can boost productivity and the comfortability to support each other. In all, it is a vital virtue for a successful team.
- **How have you demonstrated it?**

    - I have shown my support and encouragement to my teammates by encouraging them to share ideas and keeping a positive attitude towards their thoughts. Even when challenges arose, I approached it with a constructive attitude. Additionally, if a team member did something well, I shared joy in their success.

Virtue Not Demonstrated: Clear and Thorough Documentation

- **Why is it important to you?**
    - This virtue is incredibly important and useful because it helps to ensure that all details, processes, and decisions are well recorded and easy to find. It helps to maintain consistency and allow for all team members to stay informed and on track. Additionally it makes it easy to compile final documents if organization is thorough.
- **What might you do to demonstrate that virtue?**
    - I have done some to demonstrate this virtue, but I believe it still needs work. For example, better organizing all the schematics and prototype photos into one folder. As well as if we update the schematic, ensure proper labels are given to the different iterations. Finally, organizing our code and making sure all revisions are properly pushed to gitlab would help.

Justin Jaeckel

Virtue Demonstrated: Patience

- **Why is it important to you?**
    - Patience is very important as an engineer as most problems we face aren't easy and don't have simple solutions. Patience is also important when projects require many different parts and the team has to rely on each other to get the work done.
- **How have you demonstrated it?**
    - This semester, I was working with embedded systems for the first time in awhile and taking deep dives into datasheets for the first time since CPRE 288. I thought I understood the material better than I actually did, so it took a long time for me to catch myself up to where I needed to be.

Virtue Not Demonstrated: Discipline

- **Why is it important to you?**
    - Discipline is an extremely important part of engineering and it is essential for teams to work together and accomplish their goals. Discipline is needed to make sure tasks are completed quickly and efficiently. Discipline also leads to a better quality product, which is what we're trying to produce.
- **What might you do to demonstrate that virtue?**
    - This semester, I have felt that my work has been inconsistent and I could do better. I have spent a lot of time on this project just catching myself up to where I

felt I should have been already and the amount of accomplishment hasn't been as much as I would have liked. I have had to rely on my team to help me with tasks I felt I should have been able to complete myself and want to do more next semester.

Ty Decker

Virtue Demonstrated: Humility

- **Why is it important to you?**
  - Humility is important as a base for good teamwork and cooperation with others. Being humble allows a group member to have realistic expectations for themselves and their teammates and keeps emotions in an appropriate and professional place when decisions are made.
- **How have you demonstrated it?**
  - As a Cyber Security Engineering student, large parts of the project have felt quite foreign to me. I like to be helpful and have input in design, but for many parts of the project I have had to realize that other group members are much more experienced. This has kept me humble and thankful for each opportunity I really do have to make a difference, especially as the design becomes sophisticated enough to become more security-involved.

Virtue Not Demonstrated: Discipline

- **Why is it important to you?**
  - Discipline is needed for any task to be completed, especially in the setting of CPRE 4910. Classes and meetings require discipline to be physically and mentally present for, and class assignments and design implementation require the same. Without discipline, nothing gets done.
- **What might you do to demonstrate that virtue?**
  - I have demonstrated this virtue by attending all group meetings and fulfilling the work that is required of me, however I was not always disciplined in fulfilling personal goals for project research, design, etc. I simply wish I did more work than I did this semester, and that requires discipline.

Neil Prange

Virtue Demonstrated: Discipline

- **Why is it important to you?**
  - Discipline is important in any engineering project because most technical challenges faced by engineers will take extraordinary amounts of time doing tasks an individual might not want to do. Being able to push through the most difficult and challenging tasks is necessary in order to produce useful results. An engineer

who is able to use discipline is well-suited to tackle the most challenging but exciting problems.

- **How have you demonstrated it?**
    - I have demonstrated discipline by pushing through some of the more technical tasks I have been working on, despite some major setbacks that were very discouraging. When writing the code to detect heart beats from PPG data, various issues arrose from compatibility with our microcontroller to runtime requirements and resource availability. Despite these challenges, I was able to keep pushing and eventually produce code that could effectively detect heart beats.

Virtue Not Demonstrated: Team Orientation

- **Why is it important to you?**
    - When working with a team on an engineering project, the ability of each individual to put aside their own goal in favor of team goals is critical for success. When members are able to all strive towards a common goal by helping each other out whenever and however necessary, the team is far more effective than the individuals who compose of it.
- **What might you do to demonstrate that virtue?**
    - Going forward, I should improve on helping out more with team assignments and group tasks rather than focusing solely on my individual tasks. When I don't do my part on these group tasks, other members are forced to take time off of their own tasks to pick up my slack. If I took more initiative on tackling these team assignments and organizational tasks, the team as a whole would be better off.

Justin Scherrman

Virtue Demonstrated: Integrity

- **Why is it important to you?**
    - Integrity is crucial when working with a group because it helps with trust and accountability. Integrity ensures honesty in the group which in turn helps with collaboration. It allows for me to take responsibility for my actions during the design process.
- **How have you demonstrated it?**
    - Throughout the project thus far I have demonstrated integrity by making sure that my contributions to the design were accurate. This meant revisiting my work and redoing it in order to make sure I am contributing the best possible design. I also made sure to communicate openly with the team.

Virtue Not Demonstrated: Perseverance

- **Why is it important to you?**

    - Perseverance is important to me because it allows me to push through challenges and possible setbacks during our project. For a team perseverance makes sure that we remain focused on completing our goals regardless of difficulties.
- **What might you do to demonstrate that virtue?**
    - At times I have demonstrated perseverance but there have been moments where I could have pushed harder to work on our project. I plan on demonstrating perseverance by staying focused on problems during team meetings and in class. Another way I plan on maintaining perseverance is by keeping a positive attitude when progress seems slow.

Aidan Klimczak

Virtue Demonstrated: Patience

- **Why is it important to you?**
    - Patience is important especially in a group environment because not all things can be completed in a short amount of time. Having patience builds trust in a team setting allowing me to focus on the tasks that I need to complete instead of tasks I should not be worried about.
- **How have you demonstrated it?**
    - The functionality of our device requires many moving parts to work properly. Although I am not working on some of those parts I had patience that we could get our prototype functionality working. Also demonstrating patience when waiting for an ordered part to arrive so I can test it.

Virtue Not Demonstrated: Discipline

- **Why is it important to you?**
    - This virtue is important to me because group settings require discipline and focus. Without discipline little to nothing would get done in a group setting. This virtue handles distractions which can prevent you from doing things you shouldn't be doing when you are asked to do a task.
- **What might you do to demonstrate that virtue?**
    - I have demonstrated this virtue regularly throughout this semester although at some points I could have been more disciplined and worked harder on the project. I have been to all team meetings and done almost everything I had goals of completing by the end of the semester.

8. Conclusions

8.1 Summary of Progress

Over the course of this project we successfully progressed from initial research and breadboard prototyping to the development of two PCBs. We implemented reliable bluetooth communication between the user and dog devices, integrated biometric sensing using the MAX86150. Developed a functional testing evaluation criteria to validate features of our devices. Key milestones throughout our project were transitioning to the ESP32-S3 microcontroller, designing and assembling custom PCBs and verifying our system through testing. While real time blood pressure detection remains an area for improvement our current design meets the primary goals of usability and responsive alert functionality.

8.2 Value Provided

Our design provides a proactive and accessible solution for individuals with PTSD by enabling early detection of physiological distress and alerting their service animal before symptoms escalate. Online current market alternatives which are often reactive, expensive or intrusive our system emphasizes wearability, affordability and simplicity. By using bluetooth communication, real-time biometric monitoring and haptic feedback the device can help strengthen the bond between the user and their service animal while promoting greeting independence and safety in daily activity. This project helps directly support the mental health and well-being of veterans and others affected by trauma.

8.3 Next Steps

The next steps for this project are to refine the progress we made throughout senior design. We developed one iteration for our PCBs and were not able to get them properly functioning. This would require a group in the next few years to optimize our current boards and get them working properly. Those designs hit all client goals as a proof of concept on the breadboards we developed. Another step for the next iteration is working on increasing the accuracy of our blood pressure learning model as it is not entirely perfect compared to a cuff. A good step would be implementing other ways for the user to interact with the device like a low power LED or a LED to indicate to the user that the battery on the service animal side has died. The final step of making housings for our designs in 3D modeling software would be a final step to finishing our product.

9. References

[1] Affanni, Antonio, *Wireless Sensors System for Stress Detection by Means of ECG and EDA Acquisition,* MDPI, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/7/2026.

[2] Castaneda, D. Esparza, Aibhlin. Ghamari, Mohammad. Soltanpur, C. Nazeran, H., *A Review on Wearable Photoplethysmography sensors and their potential future applications in health care,* PubMed Central (PMC), 2019. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC6426305/.

[3] Espressif Systems, *ESP32 Technical Reference Manual*, Espressif Systems, 2024. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf.

[4] Espressif Systems, *ESP32-WROVER-E and ESP32-WROVER-IE Datasheet*, Espressif Systems, 2023. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wrover-e_esp32-wrover-ie_datasheet_en.pdf.

[5] Espressif Systems, *ESP32 Bluetooth Networking User Guide*, Espressif Systems, 2024. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_bluetooth_networking_user_guide_en.pdf.

[6] Maxim Integrated, *MAX86150: Integrated Optical Front-End for Heart Rate and SpO2 Monitoring*, Analog Devices, 2021. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/max86150.pdf.

[7] Park, J. Seok Seok, H. Kim, S. Shin, H., *Photoplethysmogram Analysis and Applications: An integrative Review,* 2022. [Online] Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC8920970/.

[8] PUI Audio Inc., *HD-EMC1002-LW20-R Data Sheet,* 2021. [Online]. Available: https://docs.google.com/gview?url=https://api.puiaudio.com/filename/HD-EMC1002-LW20-R.pdf&embedded=true.

[9] Ryan C., Richard, C., *Detection of Malingered PTSD: An Overview of Clinical Pyschometric and Physiological Assessment,* 2007. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1556-4029.2007.00434.x.

[10] Stephane, M. Mouchabac, W. Florian, F., *An Overview On How New Technologies Can Improve Prediction and Assessment of Posttraumaic Stress Disorder (PTSD),* 2018. [Online] Available: https://www.tandfonline.com/doi/full/10.1080/20008198.2018.1424448#abstract.

[11] P. H. Charlton, J. Mant, and P. A. Kyriacou, *MSPTDfast: An Efficient Photoplethysmography Beat Detection Algorithm,* "2024.07.18.24310627v1," *medRxiv*, Jul. 18, 2024. [Online]. Available: https://www.medrxiv.org/content/10.1101/2024.07.18.24310627v1.

[12] Liu, C., Zhang, Z., & Wei, C. (2018). *PPG-BP Database [Data set]*. Figshare. https://doi.org/10.6084/m9.figshare.5459299.v2

[13] S. Shaffer and J. P. Ginsberg, "An Overview of Heart Rate Variability Metrics and Norms," *Frontiers in Public Health*, vol. 5, p. 258, Sep. 2017. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5624990/

[14] Y. Kim, H. Kim, and K. Kim, "Stress and Heart Rate Variability: A Meta-Analysis and Review of the Literature," *Psychiatry Investigation*, vol. 15, no. 3, pp. 235–245, Mar. 2018. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5900369/

10. Appendices

Appendix 1 - Operation Manual

System Overview

The PTSD detection system consists of two devices:

- User Device: A wearable unit that monitors heart rate and estimates blood pressure.
- Dog Device: A vest-mounted unit that receives Bluetooth alerts and activates ERM motors to notify the service dog.

Startup Procedure

1. Power On:
    - Slide the power switch on both the user and dog devices to the ON position.
    - The red LED on each device will illuminate to indicate power is on.

2. Bluetooth Connection:

    - When the blue LED remains solid, the devices are successfully paired.

Normal Operation

- The user device continuously monitors biometric signals.
- In the current version, pressing the test button on the user device simulates a PTSD episode.
- When triggered, the device sends an “ALERT” message via Bluetooth.
- The dog device receives the message and activates the ERM vibration motors to alert the service animal.

Charging

- Plug a USB-C cable into the port on each device to charge.
- While charging, the onboard charging LED will illuminate.
- Once fully charged, the LED will turn off
- It is recommended to power off the device while charging.

Power Off

- Slide the power switch on both devices to the OFF position to shut down.

Appendix 2 - Alternative/Initial Version of Design

APP Development

There was another alternative design the team discussed, and that was interfacing and programming a comercial wrist wearable. This means developing an application for the watch and code it in such a way that the watch specializes in detecting PTSD attacks. However, will this design, we decided that this wouldn’t be in the best interest in the team. This is because we were comprised of mostly electrical and computer engineering major students, we did not have much if any experience with app development, and we would like to continue our hardware development skills. This is why we ended up choosing the route we did.

Initial Design

The initial design was a simple breadboard implementation. We used the development boards for the microcontroller and the photoplethysmogram. This was so we could directly interface with the IDE’s that control these modules. We were able to code a working Bluetooth channel along side a heart rate detection algorithm.

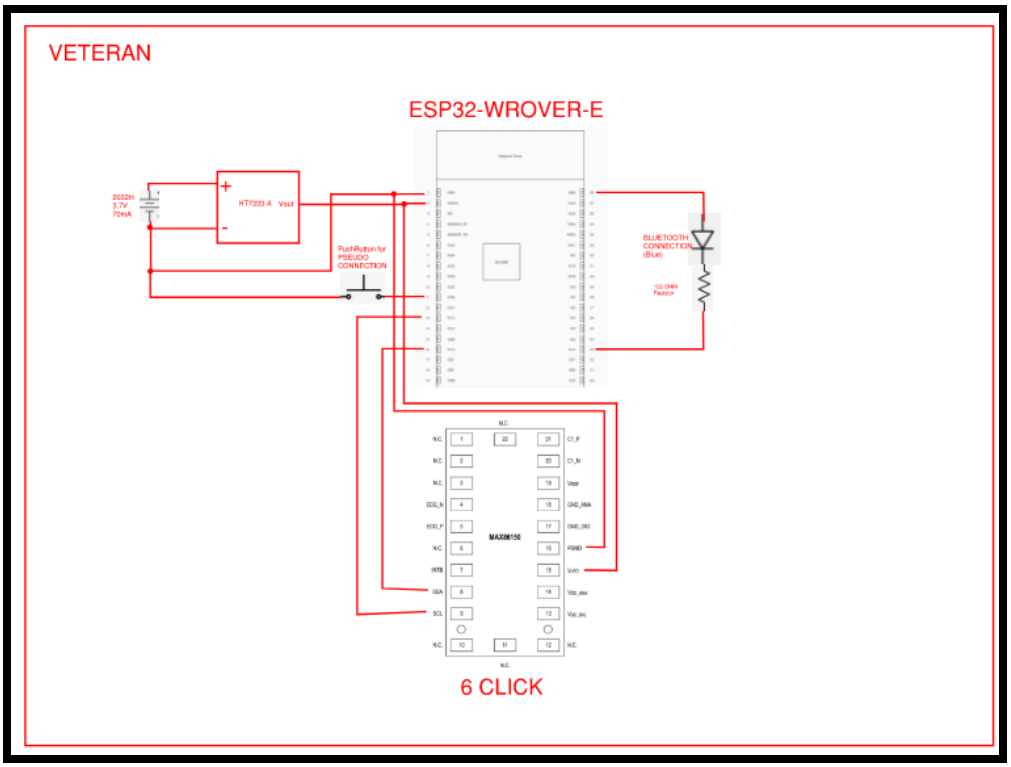


Figure 16: Initial Veteran Schematic

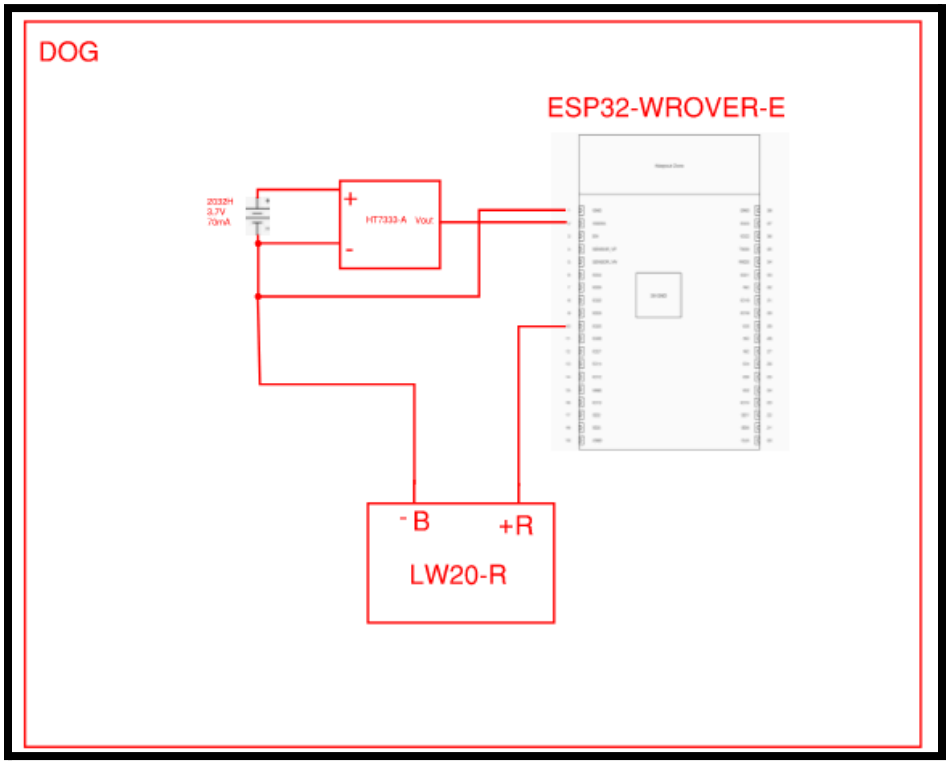


Figure 17: Initial Dog Schematic

The reason for doing this way was to develop a proof of concept first. From here we would work on making our design smaller (wrist wearable size) and have it be one solidified module.

The only design change we included was switching the microcontroller from the Wrover E to the S3 Wroom-1. This was because in the S3 there is an auto integrated serial to uart bridge IC inside. This means it would be one less component to add on the PCB, which means a smaller footprint and less components where things could fail. Otherwise, we continued our original vision of the project, and developed our PCB with the rechargeable battery, sensor, LEDs, button, microcontroller, and USB-C connector.

Appendix 3 - Other Considerations

PCB Parts List - Google Sheets

Appendix 4 - Code

Bluetooth Code

Veteran Side (Notifcation/Server)

```
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>

//wip

BLEServer* pServer = NULL;
BLECharacteristic* pCharacteristic_1 = NULL;
bool deviceConnected = false;
bool oldDeviceConnected = false;
uint32_t value = 0;

#define SERVICE_UUID        "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define CHARACTERISTIC_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"

class MyServerCallbacks: public BLEServerCallbacks {
    void onConnect(BLEServer* pServer) {
      deviceConnected = true;
    };

    void onDisconnect(BLEServer* pServer) {
      deviceConnected = false;
    }
};
//button in 9
//led is 15
const int buttonPin = 9;

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

  pinMode(15, OUTPUT); //LED PIN
  digitalWrite(15,LOW);
  pinMode(buttonPin, INPUT_PULLUP); //button pin
  // Create the BLE Device
  BLEDevice::init("ESP32");

  // Create the BLE Server
  pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());

  // Create the BLE Service
  BLEService *pService = pServer->createService(SERVICE_UUID);

  // Create a BLE Characteristic
  pCharacteristic_1 = pService->createCharacteristic(
                      CHARACTERISTIC_UUID,
                      BLECharacteristic::PROPERTY_READ   |
                      BLECharacteristic::PROPERTY_WRITE  |
                      BLECharacteristic::PROPERTY_NOTIFY |
                      BLECharacteristic::PROPERTY_INDICATE
                    );
```

```

  // Create a BLE Descriptor
  BLEDescriptor *pDescr;
  pDescr = new BLEDescriptor((uint16_t)0x2901); //this is the characteristic user description
  pDescr->setValue("Turn On Motor"); //NOT WORKING??????
  pCharacteristic_1->addDescriptor(pDescr);

  BLE2902 *pBLE2902;  //to send automatically
  pBLE2902 = new BLE2902();
  pBLE2902->setNotifications(true);

  pCharacteristic_1->addDescriptor(pBLE2902); //0x2902 is the client chracteristic configuration

  // Start the service
  pService->start();

  // Start advertising
  BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->addServiceUUID(SERVICE_UUID);
  pAdvertising->setScanResponse(false);
  pAdvertising->setMinPreferred(0x0);  // set value to 0x00 to not advertise this parameter
  BLEDevice::startAdvertising();
  Serial.println("Waiting a client connection to notify...");
}

void loop() {
    // notify changed value
    if (deviceConnected) {
        digitalWrite(15,HIGH);
        int buttonState = digitalRead(buttonPin);
          if (buttonState == 0){
            Serial.println("It's on?.");
            String message = "ALERT";
            pCharacteristic_1->setValue(message); //has a limit to 20 characters
            pCharacteristic_1->notify();
            value++;
            delay(1000); // bluetooth stack will go into congestion, if too many packets are sent
          }
          else{
            Serial.println("It didn't work.");
            Serial.println(buttonState);
            String idle = "No Alert";
            pCharacteristic_1->setValue(idle);
            pCharacteristic_1->notify();
            value++;
            delay(1000);
          }
```

```

    }
    // disconnecting
    if (!deviceConnected && oldDeviceConnected) {
        digitalWrite(15,LOW);
        delay(500); // give the bluetooth stack the chance to get things ready
        pServer->startAdvertising(); // restart advertising
        Serial.println("start advertising");
        oldDeviceConnected = deviceConnected;
    }
    // connecting
    if (deviceConnected && !oldDeviceConnected) {
        // do stuff here on connecting
        oldDeviceConnected = deviceConnected;
    }
}

```

X

Dog Side (Client)

```
#include "BLEDevice.h"
//#include "BLEScan.h"

static BLEUUID serviceUUID("4fafc201-1fb5-459e-8fcc-c5c9c331914b"); //need to have same UUIDs as Server
static BLEUUID charUUID("beb5483e-36e1-4688-b7f5-ea07361b26a8");

static boolean doConnect = false;
static boolean connected = false;
static boolean doScan = false;
static BLERemoteCharacteristic *pRemoteCharacteristic;
static BLEAdvertisedDevice *myDevice;

//called everytime the server sends out a notify to it's client
static void notifyCallback(BLERemoteCharacteristic *pBLERemoteCharacteristic, uint8_t *pData, size_t length, bool isNotify) {
  Serial.print("Notify callback for characteristic ");
  Serial.print(pBLERemoteCharacteristic->getUUID().toString().c_str());
  Serial.print(" of data length ");
  Serial.println(length);
  Serial.print("data: ");
  Serial.write(pData, length);
  Serial.println();

  // creates an empty string that will be populated with the message
  String receivedData = " ";

  for(size_t i = 0; i < length; i++){
    // casts to a character at place i in the received data array and adds it to the string
    receivedData += (char)pData[i];
  }
  Serial.print("Received data: ");
  Serial.println(receivedData);

  // space in front is there because it is adding to received data string that st
  if (receivedData == " ALERT") {
    digitalWrite(5, HIGH);
    digitalWrite(4, HIGH);// Turn on motor
    digitalWrite(6, HIGH);
    Serial.println("Motor ON");
  } else if (receivedData == " No Alert") {
    digitalWrite(5, LOW); // Turn off motor
    digitalWrite(4, LOW);
    digitalWrite(6, LOW);
    Serial.println("Motor OFF");
  }

}
```

```

//to know if it's connected or not connected
class MyClientCallback : public BLEClientCallbacks {
  void onConnect(BLEClient *pclient) {}

  void onDisconnect(BLEClient *pclient) {
    connected = false;
    Serial.println("onDisconnect");
  }
};

bool connectToServer() {
  Serial.print("Forming a connection to ");
  Serial.println(myDevice->getAddress().toString().c_str());

  BLEClient *pClient = BLEDevice::createClient();
  Serial.println(" - Created client");

  pClient->setClientCallbacks(new MyClientCallback());

  // Connect to the remove BLE Server
  // if you pass BLEAdvertisedDevice instead of address, it will be recognized type of peer device address (public or private)
  pClient->connect(myDevice);
  Serial.println(" - Connected to server");
  //set client to request maximum MTU from server (default is 23 otherwise)
  pClient->setMTU(517);

  // Obtain a reference to the service we are after in the remote BLE server.
  BLERemoteService *pRemoteService = pClient->getService(serviceUUID);
  if (pRemoteService == nullptr) {
    Serial.print("Failed to find our service UUID: ");
    Serial.println(serviceUUID.toString().c_str());
    pClient->disconnect();
    return false;
  }
  Serial.println(" - Found our service");

  // Obtain a reference to the characteristic in the service of the remote BLE server.
  pRemoteCharacteristic = pRemoteService->getCharacteristic(charUUID);
  if (pRemoteCharacteristic == nullptr) {
    Serial.print("Failed to find our characteristic UUID: ");
    Serial.println(charUUID.toString().c_str());
    pClient->disconnect();
    return false;
  }
  Serial.println(" - Found our characteristic");
```

```

  // Read the value of the characteristic.
  if (pRemoteCharacteristic->canRead()) {
    //AlertSignal = String("Alert");
    String value = pRemoteCharacteristic->readValue();
    Serial.print("The characteristic value was: ");
    Serial.println(value.c_str());
      //if (value.c_str == AlertSignal){


      //}
  }

  if (pRemoteCharacteristic->canNotify()) {
    pRemoteCharacteristic->registerForNotify(notifyCallback);
  }

  connected = true;
  return true;
}
/**
 * Scan for BLE servers and find the first one that advertises the service we are looking for.
 */

//this scans for a server and once it finds one, it verifies this is the correct server
class MyAdvertisedDeviceCallbacks : public BLEAdvertisedDeviceCallbacks {
  /**
   * Called for each advertising BLE server.
   */
  void onResult(BLEAdvertisedDevice advertisedDevice) {
    Serial.print("BLE Advertised Device found: ");
    Serial.println(advertisedDevice.toString().c_str());

    // We have found a device, let us now see if it contains the service we are looking for.
    if (advertisedDevice.haveServiceUUID() && advertisedDevice.isAdvertisingService(serviceUUID)) {

      BLEDevice::getScan()->stop();
      myDevice = new BLEAdvertisedDevice(advertisedDevice);
      doConnect = true;
      doScan = true;

    } // Found our server
  } // onResult
};  // MyAdvertisedDeviceCallbacks

void setup() {
  Serial.begin(115200);
  Serial.println("Starting Arduino BLE Client application...");
  BLEDevice::init("");
```

```

void setup() {
  Serial.begin(115200);
  Serial.println("Starting Arduino BLE Client application...");
  BLEDevice::init("");

  // Retrieve a Scanner and set the callback we want to use to be informed when we
  // have detected a new device.  Specify that we want active scanning and start the
  // scan to run for 5 seconds.
  BLEScan *pBLEScan = BLEDevice::getScan();
  pBLEScan->setAdvertisedDeviceCallbacks(new MyAdvertisedDeviceCallbacks());
  pBLEScan->setInterval(1349);
  pBLEScan->setWindow(449);
  pBLEScan->setActiveScan(true);
  pBLEScan->start(5, false);

  pinMode(4, OUTPUT); //setting motor pin to output
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
}

void loop() {

  // If the flag "doConnect" is true then we have scanned for and found the desired
  // BLE Server with which we wish to connect.  Now we connect to it.  Once we are
  // connected we set the connected flag to be true.
  if (doConnect == true) {
    if (connectToServer()) {
      Serial.println("We are now connected to the BLE Server.");
      //IMPLEMENT BLUE LED TO SAY IT IS ON AND CONNECTED
      //digitalWrite(pin #, high); // this turns on the blue LED after connected to server.


    }
    else {
      Serial.println("We have failed to connect to the server; there is nothing more we will do.");
    }
    doConnect = false;
  }
```

```

  // if (connected = false){
  //  digitalWrite(pin #, low); // this turns the BLUE LED off when not connected.
  // }

  // If we are connected to a peer BLE Server, update the characteristic each time we are reached
  // with the current time since boot.
  if (connected) {
   // String newValue = "Time since boot: " + String(millis() / 1000);
    // Serial.println("Setting new characteristic value to \"" + newValue + "\"");

    // Set the characteristic's value to be the array of bytes that is actually a string.
    // pRemoteCharacteristic->writeValue(newValue.c_str(), newValue.length());
  } else if (doScan) {
    BLEDevice::getScan()->start(0);  // this is just example to start scan after disconnect, most likely there is better way to do it in arduino
  }

  delay(1000);  // Delay a second between loops.
} // End of loop
```

X

Heart + Blood Pressure Main Program

```
#include <Chirale_TensorFlowLite.h>
#include <Wire.h>
#include <stdlib.h>

#include "model.h"
#include "tensorflow/lite/micro/all_ops_resolver.h"
#include "tensorflow/lite/micro/micro_interpreter.h"
#include "tensorflow/lite/schema/schema_generated.h"

extern "C" float mnptd(float* sig, int32_t len, int32_t freq);
void preprocess_data(const float *raw, size_t length, float *processed);

/* Begin TFLite global variables */
const tflite::Model* model = nullptr;
tflite::MicroInterpreter* interpreter = nullptr;
TfLiteTensor* input = nullptr;
TfLiteTensor* output = nullptr;

constexpr int kTensorArenaSize = 32768;

alignas(16) uint8_t *tensor_arena;

constexpr int kInputWidth = 210;
constexpr int kNumChannels = 3;
constexpr int test_input_length = 630;

float test_input[test_input_length];
/* End TFLite global variables */

/* Begin PPG global variables */
byte status;
byte address = 0x5E;
byte CONFIG_START = 0x09;
uint32_t ppg_data;
byte REGISTER_INIT[] = {0x01, 0x00, 0x00, 0x00, 0x04, 0x03, 0x00, 0x00, 0x12, 0x32, 0x00, 0x00, 0xFF};
```

```
constexpr int SAMPLE_RATE = 100;
constexpr int SECONDS_PER_WINDOW = 6;
constexpr int NUM_SAMPLES = 600;

float samples[NUM_SAMPLES];
/* End PPG global variables */

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

  /* PPG init */
  Wire.begin(GPIO_NUM_13, GPIO_NUM_14);
  Wire.beginTransmission(address);
  Wire.write(CONFIG_START);

  for(int i = 0; i < sizeof(REGISTER_INIT); ++i) {
    Wire.write(REGISTER_INIT[i]);
  }

  Wire.endTransmission();

  /* TFLite init */
  model = tflite::GetModel(hp_model);

  // Ensure TF version used to train model equals inference version
  if (model->version() != TFLITE_SCHEMA_VERSION) {
    Serial.println("Model provided and schema version are not equal!");
    while(true); // stop program here
  }

  static tflite::AllOpsResolver resolver;

  tensor_arena = (uint8_t*)malloc(kTensorArenaSize);
```

```
  // Build an interpreter to run the model with.
  static tflite::MicroInterpreter static_interpreter(
      model, resolver, tensor_arena, kTensorArenaSize);
  interpreter = &static_interpreter;

  // Allocate memory from the tensor_arena for the model's tensors.
  // If an error occurs, stop the program.
  TfLiteStatus allocate_status = interpreter->AllocateTensors();
  if (allocate_status != kTfLiteOk) {
    Serial.println("AllocateTensors() failed");
    while(true); // stop program here
  }

  // Obtain pointers to the model's input and output tensors.
  input = interpreter->input(0);
  output = interpreter->output(0);

  Serial.println("Initialization done.");
  Serial.println("");
}

void loop()
{
  // Begin data collection
  int32_t sample_no = 0;

  while(sample_no < NUM_SAMPLES) {

    // Read register to determine how many values are ready
    Wire.beginTransmission(address);
    Wire.write(0x04);
    Wire.endTransmission();
    Wire.requestFrom(address, 3);
    byte wr_ptr = Wire.read();
    byte ovf_counter = Wire.read();
    byte read_ptr = Wire.read();
```

```
  // If the FIFO has overflown, read all 32 values, otherwise read wrap-around distance between wr_ptr and read_ptr
  int samples_to_read;
  if(ovf_counter == 0) {
    samples_to_read = (wr_ptr >= read_ptr) ? wr_ptr - read_ptr : 32 - read_ptr + wr_ptr;
  } else {
    samples_to_read = 32;
  }

  uint32_t data[32];
  if(samples_to_read) {
    Wire.requestFrom(address, 3 * samples_to_read);
    for(int i = 0; i < samples_to_read; ++i) {
      uint32_t val1 = Wire.read();
      uint32_t val2 = Wire.read();
      uint32_t val3 = Wire.read();
      uint32_t ir_val = (val3 & 0xFF)
                      | ((val2 & 0xFF) << 8)
                      | ((val1 & 0x07) << 16);
      data[i] = ir_val;
    }
  }

  for(int i = 0; ((i < samples_to_read) && (sample_no < 600)); ++i) {
    samples[sample_no++] = (float)data[i];
  }
  delay(5);
}

// Pre-process our raw PPG data into form usable by our model
preprocess_data(samples, 210, test_input);

// Copy input into model
for(int i = 0; i < test_input_length; ++i) {
  input->data.f[i] = test_input[i];
}
```

```
  // Run inference, and report if an error occurs
  TfLiteStatus invoke_status = interpreter->Invoke();
  if (invoke_status != kTfLiteOk) {
    Serial.println("Invoke failed!");
    return;
  }

  TfLiteTensor* output = interpreter->output(0);

  // Process the inference results.
  float bp = output->data.f[0];

  Serial.print("Estimated Systolic BP: ");
  Serial.println(bp);

  // Run our 6 seconds of samples through MSPTD
  int32_t len = NUM_SAMPLES;
  int32_t freq = SAMPLE_RATE;
  //float heart_rate = mnptd(samples, len, freq);
  float heart_rate = 0;

  if(heart_rate < 0) {
    Serial.print("Error: MSPTD returned with exit code: ");
    Serial.println(heart_rate);
  } else if(heart_rate > 0) {
    Serial.print("Heart Rate: ");
    Serial.println(heart_rate);
    Serial.println("\n");
  }
```

```
  }

  delay(1000);
}

/*
 * Pre-processes raw PPG data into a form compatible with our TFLite model.
 * Normalizes the samples to match the magnitude and range of samples used in training pipeline,
 * and adds input channels for the first and second derivative of the waveform. Allocates data at
 * processed for processed data, including additional added channels.
 *
 * @param *raw Pointer to raw PPG data.
 * @param length Length in samples of raw data.
 * @param processed Pre-allocated array of processed data, must be >= 3 * length.
 */
void preprocess_data(const float *raw, size_t length, float *processed)
{
  float sum, sum_sq, mean, diff, stddev;
  size_t i;

  // Get the mean of the signal
  sum = 0.0f;
  for(i = 0; i < length; ++i) {
    sum += raw[i];
  }
  mean = sum / length;

  // Get the standard deviation
  sum_sq = 0.0f;
  for(i = 0; i < length; ++i) {
    diff = raw[i] - mean;
    sum_sq += diff * diff;
  }
  stddev = sqrt(sum_sq / length);
```

```
  stddev = sqrt(sum_sq / length);
  if(stddev < 1e-8f) { stddev = 1e-8f; }

  // Write normalized signal to output array, depth-major indexing
  for(i = 0; i < length; ++i) {
    processed[i * 3] = (raw[i] - mean) / stddev;
  }

  // Write first and second derivative to other input channels
  processed[1] = 0.0f;
  processed[2] = 0.0f;
  for(i = 1; i < length; ++i) {
    processed[i * 3 + 1] = processed[i * 3] - processed[(i - 1) * 3];
    processed[i * 3 + 2] = processed[i * 3 + 1] - processed[(i - 1) * 3 + 1];
  }
}
```

MSPTD Arduino Code Implementation

```c
#include <math.h>
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

static int detect_peaks_and_onsets_using_msptd(float* x, int32_t len, int32_t** peaks, int32_t* len_peaks);
static void detrend(float* vals, int32_t len);
static int32_t remove_duplicates(int32_t* arr, int n);
int compare(const void* a, const void* b);

/*
 * Calculates the average peak rate based on number of peaks in a PPG signal found using the MSPTD algorithm.
 * For accuracy, only call function for windows of 6 seconds or more.
 *
 * @param int32_t** sig Array of raw PPG values.
 * @param int32_t len Length of sig array.
 * @param int32_t freq Frequency of PPG recording (Hz).
 * @return The number of peaks detected.
 *
 *   # License - MIT
 *     Copyright (c) 2022 Peter H. Charlton
 *     Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"),
 *     The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
 *     THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY
 */
float msptd(float* sig, int32_t len, int32_t freq) {
  int32_t win_len = 6;
  float overlap = 0.2;
  int32_t num_samples_in_window = win_len * freq;
  int32_t* win_starts;   // Array of indices for beggining of 6 second windows
  int32_t* win_ends;     // Array of indices for end of 6 second windows
  int32_t num_windows;   // Number of partitioning windows
  int32_t win_offset;
  int32_t final_peaks[200];
  int32_t num_final_peaks = 0;
```

```c
  int32_t num_final_peaks = 0;
  if (len <= num_samples_in_window) {
    num_windows = 1;

    win_starts = (int32_t*)malloc(sizeof(int32_t));
    win_starts[0] = 0;

    win_ends = (int32_t*)malloc(sizeof(int32_t));
    win_ends[0] = len - 1;
  }
  else {
    win_offset = (int32_t)round(num_samples_in_window * (1 - overlap));
    num_windows = (int32_t)((len / win_offset) + (len % win_offset != 0));    // Round up
    win_starts = (int32_t*)malloc(num_windows * sizeof(int32_t));
    win_ends = (int32_t*)malloc(num_windows * sizeof(int32_t));
    if (!(win_starts) || !(win_ends)) return -1;
    for (int32_t i = 0; i < num_windows; ++i) {
      win_starts[i] = i * win_offset;
      win_ends[i] = win_starts[i] + num_samples_in_window - 1;
    }
    // Last window extends beyond the length of the signal, push back last window
    // TODO: Might need to update if parameters like overlap are changed.
    if (win_ends[num_windows - 1] > len - 1) {
      win_starts[num_windows - 1] = len - num_samples_in_window - 1;
      win_ends[num_windows - 1] = len - 1;
    }
  }

  // Detect peaks and onnsets for each window
  for (int32_t win_no = 0; win_no < num_windows; ++win_no) {
    float* win_sig = (float*)malloc(num_samples_in_window * sizeof(float)); // 4.8KB of heap
    if (!win_sig) {
      return -1;
    }
    memcpy(win_sig, &sig[win_starts[win_no]], num_samples_in_window * sizeof(float));
```

```c
    int32_t* peaks = NULL;
    int32_t len_peaks;

    int status = detect_peaks_and_onsets_using_msptd(win_sig, num_samples_in_window, &peaks, &len_peaks);
    if(status != 0) {
      return (float)status;
    }

    for (int32_t i = num_final_peaks; i < num_final_peaks + len_peaks; ++i) {
      final_peaks[i] = peaks[i - num_final_peaks] + win_starts[win_no];
    }
    num_final_peaks += len_peaks;
    free(peaks);
    free(win_sig);
  }

  // Remove duplicate peaks
  qsort(final_peaks, num_final_peaks, sizeof(int32_t), compare);    // Begin by sorting the array of indices
  num_final_peaks = remove_duplicates(final_peaks, num_final_peaks);

  // If we've only detected 1 or less peaks, we cannot make heart rate estimation.
  if (num_final_peaks < 2) return 0;

  // Calculate BPM based on average time between peaks
    // TODO: Remove peak deltas that are significantly outside expected range
  int32_t delta_sum = 0;
  for (int32_t i = 0; i < num_final_peaks - 1; ++i) {
    delta_sum += (final_peaks[i + 1] - final_peaks[i]);
  }
  float delta_avg = (1.0 * delta_sum) / (num_final_peaks - 1);
  float rate = 60 * freq / delta_avg;

  free(win_starts);
  free(win_ends);

  return rate;
}
```

```c
/*
 * Takes a signal, and finds the indices of peaks within the signal as well as the number of peaks detected.
 * Removes linear trend, and finds values that are a local maximum compared to all values in a window that
 * grows out from the index of the peak.
 *
 * Note: calls malloc on peaks and assumes caller function calls free().
 *
 * @param int32_t* x Windowed signal pointer
 * @param int32_t len Length of windowed signal
 * @param int32_t** peaks Pointer to array of peaks detected, dynamically allocated
 * @param int32_t* len_peaks Pointer to the number of peaks detected.
 *
 */
static int detect_peaks_and_onsets_using_msptd(float* x, int32_t len, int32_t** peaks, int32_t* len_peaks) {
  //int32_t l = ((len + 1) / 2) - 1;  // Max window length
  int32_t l = ((len + 3) / 4) - 1;

  detrend(x, len);

  // LMS matrices, initialized to 0s, with dimensions [l, len]
  // l - #rows, len - #columns
  int8_t** m_max = (int8_t**)malloc(l * len * sizeof(int8_t));  // 180 KB heap space
  if (!m_max) return -2;

  // Populate with 0's
  for (int32_t i = 0; i < l; ++i) {
    for (int32_t j = 0; j < len; ++j) {
      m_max[i * len + j] = 0;
    }
  }

  // Populate matrices
  for (int32_t i = 1; i <= l; ++i) {
    for (int32_t j = i + 2; j < len - i + 1; ++j) {
      if ((x[j - 1] > x[j - i - 1]) && (x[j - 1] > x[j + i - 1])) {
        m_max[(i - 1) * len + (j - 1)] = 1;
```

```c
      }
    }
  }

  // Find what scale value produces the most local extrema
  int32_t* gamma_max = (int32_t*)malloc(l * sizeof(int32_t));   // 1.2 KB heap space
  if (!gamma_max) return -1;

  for (int32_t i = 0; i < l; ++i) {
    for (int32_t j = 0; j < len; ++j) {
      gamma_max[i] += (int32_t)m_max[i * len + j];
    }
  }

  int32_t lambda_max = 0;
  int32_t max_max = gamma_max[0];
  for (int32_t i = 0; i < l; ++i) {
    if (gamma_max[i] > max_max) {
      max_max = gamma_max[i];
      lambda_max = i;
    }
  }

  // Remove any elements of matrices in which k > lambda
  // Each row corresponds to a k value
  // Value of -1 indicates a value has been removed
  // TODO: Should be able to remove this -> don't check m_max for rows >= lambda_max
  for (int32_t i = lambda_max * len; i < l * len; ++i) {
    m_max[i] = -1;
  }

  // Find peaks peaks and onnsets
  // Sum the number scales that did not have a max at a given position
  int32_t* m_max_sum = (int32_t*)malloc(len * sizeof(int32_t)); // 2.4 KB Heap space
  if (!m_max_sum) return -1;
```

```c
  int32_t num_peaks = 0;

  for (int32_t i = 0; i < len; ++i) {
    m_max_sum[i] = 0;
    for (int32_t j = 0; j < lambda_max; ++j) {
      if (m_max[j * len + i] == 0) {
        m_max_sum[i]++;
      }
    }
    if (m_max_sum[i] == 0) {
      num_peaks++;
    }
  }

  // Return the indices of m_max_sum equal to 0
  *(len_peaks) = num_peaks;
  *peaks = (int32_t*)malloc(num_peaks * sizeof(int32_t));
  if (!(*peaks)) return -1;
  int32_t i = 0;
  for (int32_t j = 0; j < len; ++j) {
    if (m_max_sum[j] == 0) {
      (*peaks)[i++] = j;
    }
  }

  // Free allocated buffers, except for peaks which needs to be deallocated by caller.
  free(m_max);
  free(gamma_max);
  free(m_max_sum);
  return 0;
}
```

```c
/*
 * Implements the detrend(x) function from Matlab which given an input vector x, removes the best straight-fit
 * line from the data in x and returns the remaining data. This is used in order to calculate peaks and onsets
 * from the data without interference of any overall trend (e.g. the PPG sensor changing distance from the user).
 *
 * In this implementation, the algorithm assumes int32_t input array and simply removes the linear trend from
 * the input. It does not return any value.
 *
 * @param int32_t* vals Pointer to array to be detrended.
 * @return void
 */
static void detrend(float* vals, int32_t len) {
  float y_sum = 0;
  float y_avg;
  float x_avg;

  float slope_num = 0;
  float slope_denom = 0;

  float slope;
  float intercept;

  // Get the averages
  for (int32_t i = 0; i < len; ++i) {
    y_sum += vals[i];
  }
  y_avg = 1.0 * (y_sum / len);
  x_avg = (len - 1) / 2;

  // Get the numerator and denominator of the best-fit line slope
  for (int32_t i = 0; i < len; ++i) {
    slope_num += (i - x_avg) * (vals[i] - y_avg);
    slope_denom += (i - x_avg) * (i - x_avg);
  }
  slope = slope_num / slope_denom;
  intercept = y_avg - (slope * x_avg);
```

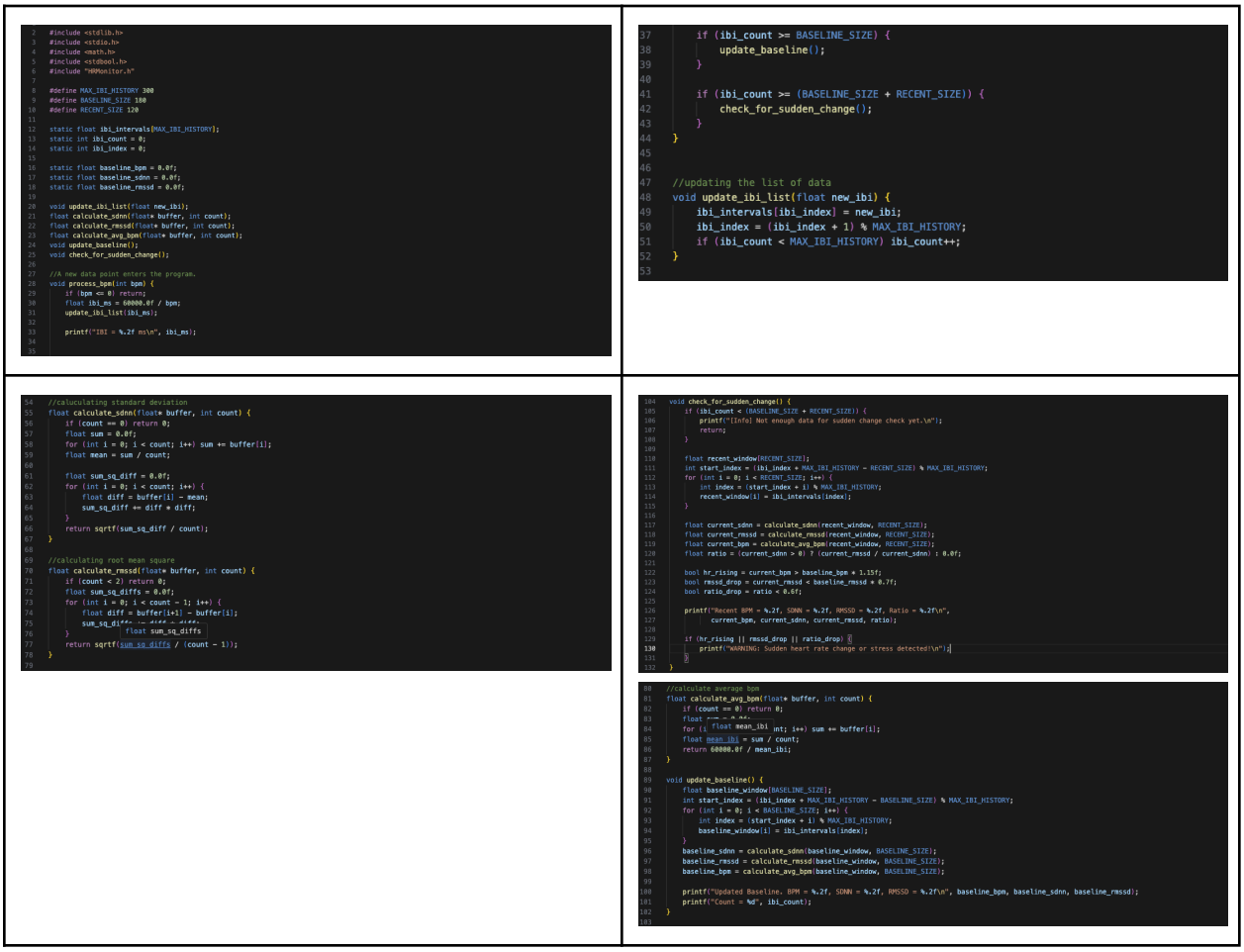
```c
  // Remove best-fit linear trend line value from each value
  for (int32_t i = 0; i < len; ++i) {
    vals[i] = (vals[i] - (slope * i + intercept));
  }
}
/*
 * Remove duplicate values from an array
 */
static int32_t remove_duplicates(int32_t* arr, int n) {
  if (n == 0) return 0;

  int32_t j = 0;
  for (int32_t i = 1; i < n - 1; ++i) {
    if (arr[i] != arr[j]) {
      arr[++j] = arr[i];
    }
  }

  return j + 1;
}

int compare(const void* a, const void* b) {
  return (*(int*)a - *(int*)b);
}
```

Heart Rate Algorithm



```c
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <stdbool.h>
#include "HRMonitor.h"

#define MAX_IBI_HISTORY 300
#define BASELINE_SIZE 180
#define RECENT_SIZE 120

static float ibi_intervals[MAX_IBI_HISTORY];
static int ibi_count = 0;
static int ibi_index = 0;

static float baseline_bpm = 0.0f;
static float baseline_sdnn = 0.0f;
static float baseline_rmssd = 0.0f;

void update_ibi_list(float new_ibi);
float calculate_sdnn(float* buffer, int count);
float calculate_rmssd(float* buffer, int count);
float calculate_avg_bpm(float* buffer, int count);
void update_baseline();
void check_for_sudden_change();

//A new data point enters the program.
void process_bpm(int bpm) {
    if (bpm <= 0) return;
    float ibi_ms = 60000.0f / bpm;
    update_ibi_list(ibi_ms);

    printf("IBI = %.2f ms\n", ibi_ms);

}
```

```c
    if (ibi_count >= BASELINE_SIZE) {
        update_baseline();
    }

    if (ibi_count >= (BASELINE_SIZE + RECENT_SIZE)) {
        check_for_sudden_change();
    }
}


//updating the list of data
void update_ibi_list(float new_ibi) {
    ibi_intervals[ibi_index] = new_ibi;
    ibi_index = (ibi_index + 1) % MAX_IBI_HISTORY;
    if (ibi_count < MAX_IBI_HISTORY) ibi_count++;
}
```

```c
//caluculating standard deviation
float calculate_sdnn(float* buffer, int count) {
    if (count == 0) return 0;
    float sum = 0.0f;
    for (int i = 0; i < count; i++) sum += buffer[i];
    float mean = sum / count;

    float sum_sq_diff = 0.0f;
    for (int i = 0; i < count; i++) {
        float diff = buffer[i] - mean;
        sum_sq_diff += diff * diff;
    }
    return sqrtf(sum_sq_diff / count);
}

//calculating root mean square
float calculate_rmssd(float* buffer, int count) {
    if (count < 2) return 0;
    float sum_sq_diffs = 0.0f;
    for (int i = 0; i < count - 1; i++) {
        float diff = buffer[i+1] - buffer[i];
        sum_sq_diffs += diff * diff;
    }
    return sqrtf(sum_sq_diffs / (count - 1));
}
```

```c
void check_for_sudden_change() {
    if (ibi_count < (BASELINE_SIZE + RECENT_SIZE)) {
        printf("[info] Not enough data for sudden change check yet.\n");
        return;
    }

    float recent_window[RECENT_SIZE];
    int start_index = (ibi_index + MAX_IBI_HISTORY - RECENT_SIZE) % MAX_IBI_HISTORY;
    for (int i = 0; i < RECENT_SIZE; i++) {
        int index = (start_index + i) % MAX_IBI_HISTORY;
        recent_window[i] = ibi_intervals[index];
    }

    float current_sdnn = calculate_sdnn(recent_window, RECENT_SIZE);
    float current_rmssd = calculate_rmssd(recent_window, RECENT_SIZE);
    float current_bpm = calculate_avg_bpm(recent_window, RECENT_SIZE);
    float ratio = (current_sdnn > 0) ? (current_rmssd / current_sdnn) : 0.0f;

    bool hr_rising = current_bpm > baseline_bpm + 1.15f;
    bool rmssd_drop = current_rmssd < baseline_rmssd * 0.7f;
    bool ratio_drop = ratio < 0.6f;

    printf("Recent BPM = %.2f, SDNN = %.2f, RMSSD = %.2f, Ratio = %.2f\n",
           current_bpm, current_sdnn, current_rmssd, ratio);

    if (hr_rising || rmssd_drop || ratio_drop) {
        printf("WARNING: Sudden heart rate change or stress detected!\n");
    }
}
```

```c
//calculate average bpm
float calculate_avg_bpm(float* buffer, int count) {
    if (count == 0) return 0;
    float sum = 0.0f;
    for (i float mean_ibi int; i++) sum += buffer[i];
    float mean_ibi = sum / count;
    return 60000.0f / mean_ibi;
}

void update_baseline() {
    float baseline_window[BASELINE_SIZE];
    int start_index = (ibi_index + MAX_IBI_HISTORY - BASELINE_SIZE) % MAX_IBI_HISTORY;
    for (int i = 0; i < BASELINE_SIZE; i++) {
        int index = (start_index + i) % MAX_IBI_HISTORY;
        baseline_window[i] = ibi_intervals[index];
    }
    baseline_sdnn = calculate_sdnn(baseline_window, BASELINE_SIZE);
    baseline_rmssd = calculate_rmssd(baseline_window, BASELINE_SIZE);
    baseline_bpm = calculate_avg_bpm(baseline_window, BASELINE_SIZE);

    printf("Updated Baseline, BPM = %.2f, SDNN = %.2f, RMSSD = %.2f\n", baseline_bpm, baseline_sdnn, baseline_rmssd);
    printf("Count = %d", ibi_count);
}
```

Appendix 5 - Team Contract

Team Name: sdmay25-13

Team Members:

1) Aidan Klimczak
2) Ty Decker
3) Justin Scherrman
4) Katerina Zubic

5) Justin Jaeckel
6) Neil Prange

Team Procedures

Day, time, and location (face-to-face or virtual) for regular team meetings:
Our team meetings are set to occur every Wednesday at 2:15 in the Student Innovation Center. To maintain a private and productive work environment, we will reserve a room in SICTR. Booking a room is simple through an online portal, and we can reserve it months in advance, ensuring we always have a space to collaborate.

Additionally, we have established a standard with our faculty advisor to meet biweekly via Zoom.

Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
We have decided to communicate through a Snapchat group chat. All team members are comfortable with the app, and it works across different operating systems. Snapchat's direct messaging also ensures that all members will be updated quickly. For contacting our faculty advisor, we will use email as needed for any inquiries; otherwise, we have our biweekly zoom calls as the primary mode of communication. Finally, we will use email to communicate with our client/BAE mentors.

Decision-making policy (e.g., consensus, majority vote):
We will make decisions based on a consensus. This is because we want all group members to have the opportunity to express their opinions and develop the best decision that incorporates everyone's ideas and thoughts. The goal is to find and create a solution acceptable to everyone. Coming to a consensus can also open the doors to having a deeper conversation about the said problem statement or argument.

Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
We have created and shared a Google Drive folder containing all written documents for easy accessibility.

The entire group will work together to keep meeting minutes. This ensures that everyone is contributing to the discussion at hand. We have a master document that will be updated during our weekly meetings and will be properly labeled with the appropriate dates.

Regarding recording minutes spent working on the project, we have a master Google sheet that holds all the hours, including a small description of what it was for every week. This is expected that every individual updates on their own.

Expected individual attendance, punctuality, and participation at all team meetings:
All team members are expected to attend and participate in all scheduled meetings unless they have a valid excuse or extenuating circumstances, granted that it was communicated to the group prior to said meeting.

It is also expected that all team members contribute something to the discussion during the team meetings.

Punctuality is important for creative and productive work environments. It is encouraged that all members be present for the meetings on time, and if they can't, they should at least communicate how late they may be.

Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
It is encouraged that all team assignments are completed by a soft due date of the morning of the official due date. This enables any final tweaks to be made by the end of the day so that the finished product is satisfactory.

All submitted assignments shall be reviewed by all team members and be held to a high standard and accountability.

Expected level of communication with other team members:
Response to messages are expected within a 24-hour period from the team members. Additionally, if a team member is working on a specific aspect of the project, send an informal message to update the team on your work. This ensures all team members are constantly updated on the project's progress.

Expected level of commitment to team decisions and tasks:
We expect integrity. If a member says they will work on a portion of the project, it is expected they stay true to their word. If problems arise that hinder the ability to achieve certain personal deadlines, communicate them. This is so we know what's going on and can possibly contribute to resolving the problem.

Leadership

Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

It is said that each team member will be responsible for their own individual component designs and testing. This is not to say that we cannot contribute to each other's design ideas and thoughts.

Katerina Zubic - Team organizer; responsible for keeping the group on track for deadlines.

Justin Scherrman - Meeting coordinator; reserving the weekly meeting room.

Anointed that anyone can respond to faculty/advisor emails, but communication of action is required. Additionally, everyone plays the role of the stenographer for weekly meetings.

Other roles will be assigned once the project's scope has been ironed out.

Strategies for supporting and guiding the work of all team members:
Plan to start weekly reports together to ensure every team member is on the same page.

Stay verbally encouraging. Sometimes, we face adversities, but we are a team and are here to support and help one another. As well as give credit when credit is due.

Strategies for recognizing the contributions of all team members:
Each team member will log and discuss their contributions in the weekly report and during weekly meetings with members and faculty.

Collaboration and Inclusion

Describe the skills, expertise, and unique perspectives each team member brings to the team.

Aidan Klimczak- Expertise in design, basic knowledge of embedded systems and coding

Justin Scherrman- Experience with electrical systems design, Desire to learn hardware aspects of our system. Basic understanding of code.

Justin Jaeckel- Experience with embedded systems, software development, as well as other computer engineering disciplines

Ty Decker- Experience with software development and computer engineering concepts. Small amount of medical experience.

Neil Prange- Experience with embedded systems design, microcontroller programming, communication protocols.

*Katerina Zubi*c- Experience with electrical system design, schematics, CAD, communications, and control systems.

Strategies for encouraging and supporting contributions and ideas from all team members:

Verbally recognizing all contributions in team meetings. As well as ensuring that everyone's thoughts and ideas are heard. If you disagree from the get-go, don't tune them out, listen to what everyone has to say. This is also why our method of coming to a consensus is highly stressed.

Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment obstructs their opportunity or ability to contribute?)

It is encouraged to bring up the conflict when it first occurs. This hinders the ability to have the problem snow-ball and become a bigger issue. If extenuating circumstances persist (on a case-to-case basis), contact with the faculty member will be required.

Goal-Setting, Planning, and Execution

Team goals for this semester:

We plan to complete research into any necessary components (sensors, microcontroller, etc) for the product. As well as generate a flow chart showing how we would like everything to communicate.

Additionally, we hope to have preliminary prototype testing by December of at least all the individual components to ensure compatibility.

Strategies for planning and assigning individual and team work:

Assigning individual tasks is to be announced once the scope of the project is better understood as well as understanding everyone's strengths and weaknesses.

Hope to assign individual components to different group members.

Strategies for keeping on task:

During weekly meetings we'll start by stating the goals and deadlines for the next week. This will ensure all members are updated with the current expectations and there are no surprises.

Ensuring that all members are updating the master note sheet as well as filling out their time tables.

Hold one another accountable; integrity.

Consequences for Not Adhering to Team Contract

How will you handle infractions of any of the obligations of this team contract?

Discuss the issue with the group internally first to reach an agreement or create a course of action to solve the problem.

Will also communicate the issue, if necessary, in our weekly report if the circumstance permits.

What will your team do if the infractions continue?

If necessary, communication with the faculty or TA will be required.

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*
b) *I understand that I am obligated to abide by these terms and conditions.*
c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) _____Aidan Klimczak_____ DATE _____9/17/24__
2) _____Neil Prange_____ DATE _____9/17/24__
3) _____Ty Decker_____ DATE _____9/17/24__
4) _____Katerina Zubic_____ DATE _____9/17/24___
5) _____Justin Jaeckel_____ DATE _____9/17/24___
6) _____Justin Scherrman_____ DATE _____9/17/24___