Elderly Homes Health Tracking System

ECE 445 Design Document - Spring 2023

Team 6

Sanjana Pingali (pingali4)

Aishee Mondal (aisheem2)

Jeep Kaewla (ckaewla2)

TA: Akshatkumar Sanatbhai Sanghvi

Professor: Olga Mironenko

Date: 02-21-2023

Table of Contents

Design Document

1. Introduction

1.1 Problem

It is very common for many elderly people nowadays to live in elderly homes after retirement or other such assisted living facilities. It is also a common occurrence among them to experience multiple and varied health problems, many of which are chronic. Even if these homes and living centers are equipped with good healthcare facilities, round-the-clock vitals tracking and a centralized, continuous monitoring system are very lacking. It is also difficult to monitor multiple people and keep track of their health, in addition to reaching them in time in case assistance is required. Due to difficulty communicating with the nurses or staff, and the absence of such monitoring systems, the nurses would only be made aware of medical problems after the situation becomes worse. Sometimes, that can be too late.

In certain cases, it may take some time to travel from the nursing station to where the person requiring assistance is present, and being able to reach them quickly without losing precious time searching is of the utmost importance. Innovative technologies like Fitbits or Apple Watches also do not work in these situations, as it is not intuitive for older people to use and would be too expensive to equip each person in an elderly home. In addition, it is not customized to their specific needs and provides no such space for the centralized monitoring of multiple people. It would not be feasible to ask the staff to monitor each individual Fitbit and track it separately every day without a convenient and intuitive centralized monitoring system.

1.2 Solution

In this situation, a cost-effective solution would be a device that keeps track of various health data and vitals, such as heart rate and blood oxygen levels. It would also be able to detect emergency situations like fall detection, where assistance would most likely be needed. In order to reach them as fast as possible, it would employ the use of GPS tracking to figure out the location of the person and assist them as needed. To save as much time as possible and also to make nearby people aware of an emergency situation, an alarm system would be triggered as the sound emitted from a beeper.

The connecting web application would allow the elderly home caretakers to monitor multiple elderly people at once, as well as track individual health irregularities and communicate them to the doctors. A notification would be sent on the app when an irregular critical heart rate or breathing activity for a particular person is observed, and an alarm on the person would be triggered. We could also store past health data points in a database and monitor for any irregularities, or doctors can use this during checkups in order to correlate trends with persisting health issues.

The device consists of components positioned in a way as follows:

A person wears a belt with a main PCB attached to the control system, a beeper alarm, and sensors such as GPS and fall detection using an accelerometer and gyroscope. In addition, the heart rate and blood oxygen will be measured using sensors that are attached to a sticky pad placed on the chest connected to the wearable belt system. All of these sensors will measure data and send it to the microcontroller, which will then use a Wi-Fi module to update a database and reflect changes in our web application. If any data from these sensors is outside normal parameters, then the microcontroller will send an alert to a beeper and a notification to the web application.

1.3 Visual Aid

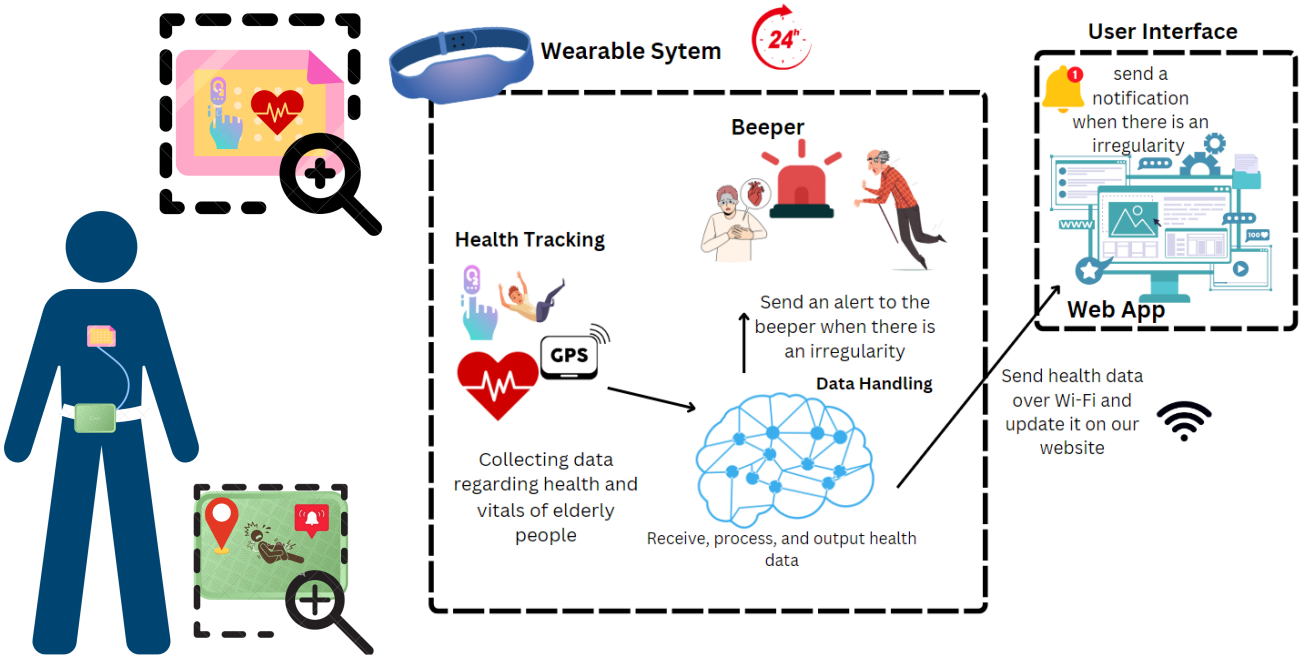

Figure 1: A visual representation of our project

1.4 High-level requirements list:

1. A notification is sent to the web application within 90 seconds ± 30 seconds when an irregularity is observed for a specific elderly person. These irregularities include the following:
   The heart rate is less than 60 BPM or greater than 135 BPM, the blood oxygen level is less than 96%, or a fall is detected, and the person is immobile for more than 60 seconds.
2. The staff can monitor heart rate, blood oxygen level, GPS location, and fall detection of the person through the web application within 90 seconds ± 30 seconds of measurement on the device.
3. The beeper on the belt emits an alert to attract immediate attention within 30 seconds ± 10 seconds of an irregularity (as defined above in the first high-level requirement) being detected.

2. Design

2.1 Block Diagram:

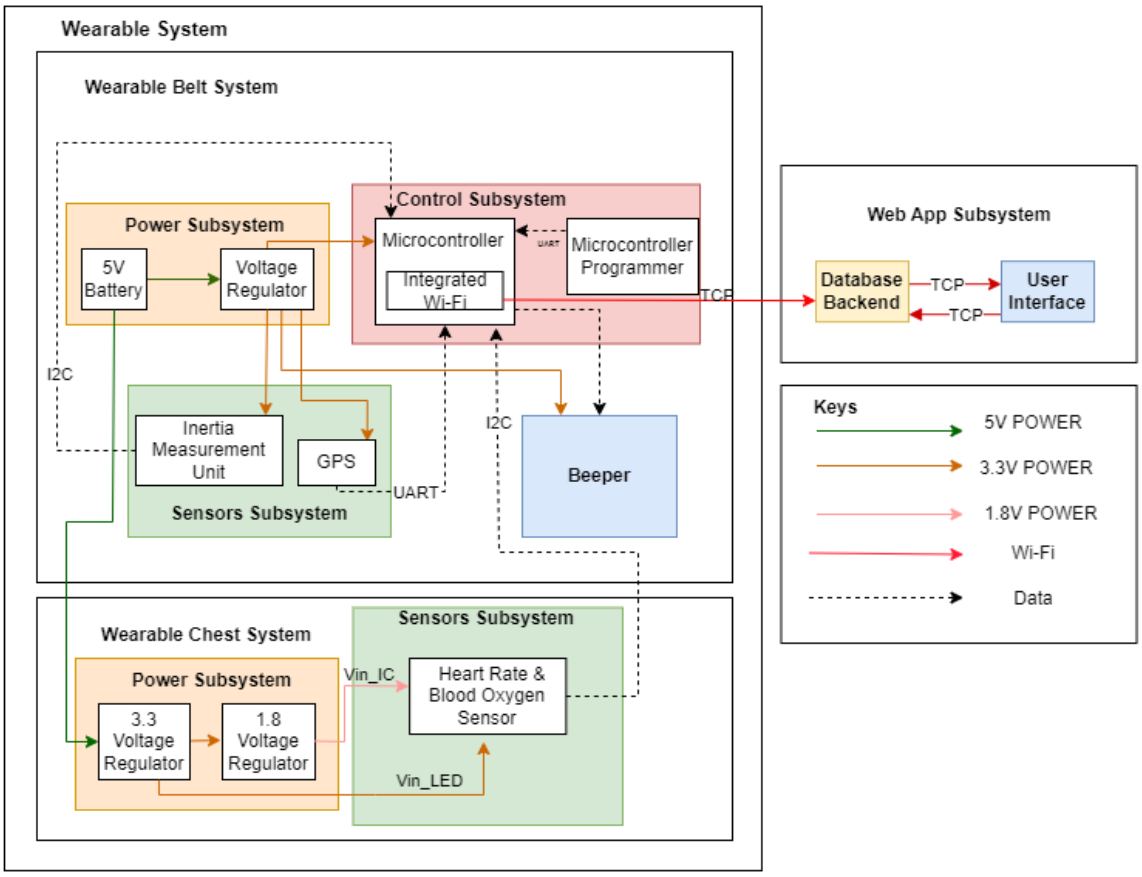

Figure 2. The Block Diagram of our project detailing the subsystems and connections.

There are two main systems in our design, which are the wearable and web application system. The wearable systems are divided into two smaller systems, the wearable belt and the sticky chest pad system. We decided to put the accelerometer and GPS on the belt system so that our device would not be too heavy and bulky on the elderly's wrist. Placing the accelerometer on the belt would also increase fall detection accuracy compared to placing the sensor elsewhere[1]. The heart rate and blood oxygen sensors are placed on the wearable chest system so that our device would have accurate readings of the heart rate and blood oxygen levels since the sensors require contact with the skin for accuracy. We would ensure that all the wires connecting the belt and chest are encased so that our product is safe to wear. Note that the pulse and blood oxygen sensors require both 3.3V and 1.8V supplies to be used for their pulse measurement and blood oxygen measurement, which is further detailed in the Heart Rate and Blood oxygen section.

The sensors collect the health data of interest and then transfer these data to the microcontroller. All our sensors, except GPS, use the I2C protocol to communicate with the microcontroller module. GPS sensor, on the other hand, uses UART protocol to communicate with the microcontroller. After the microcontroller receives the data, we would transfer the data to store in our database using Wi-Fi over a TCP connection. The web application subsystem would then display the health system obtained from HTTP requests from the database. The microcontroller programmer used the UART protocol to communicate with the microcontroller.

2.2 Physical Design (if applicable):

The figures below are a representation of the physical design of our proposed system. In this design, we can see a plastic box that houses the accelerometer and the GPS sensor modules, separately connected to the printed circuit board. This box has a USB slot to allow our rechargeable battery to be charged externally. There are holes at the bottom of the box to allow the sound from the beeper to be more audible. There is also a button at the top to reset the beeper and a switch to switch on the device. Besides this, there is a wire that connects this box to a heart rate sensor and blood oxygen concentration measurer. We use webbing to cover these sensors that can then be placed on the chest.

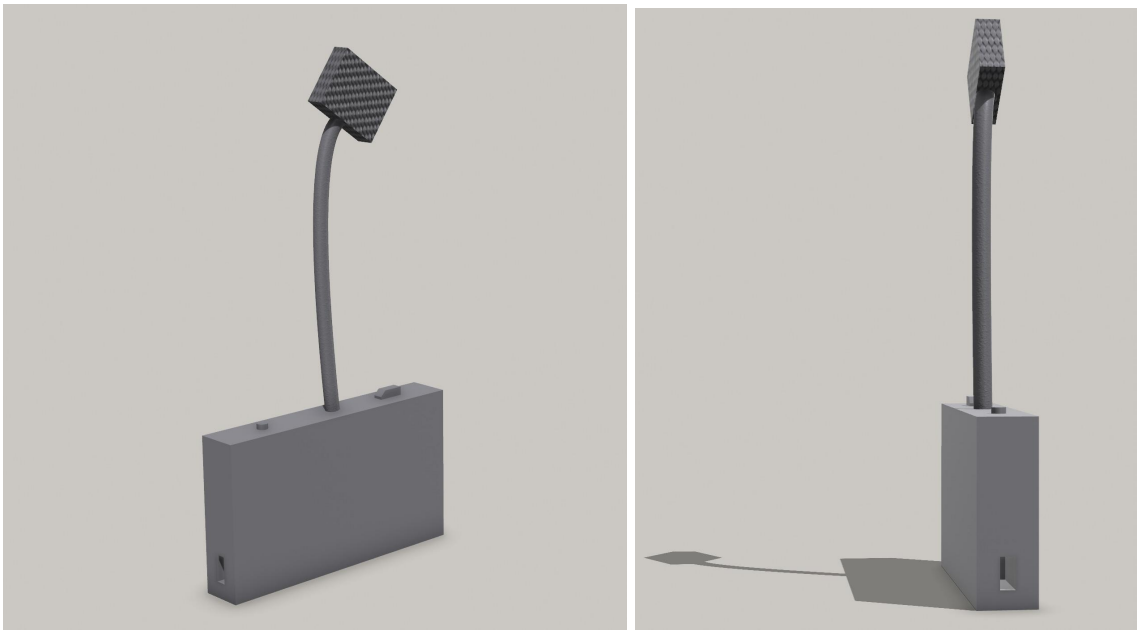

Figure 3-4. The front and side view of the physical design of our wearable system, respectively.

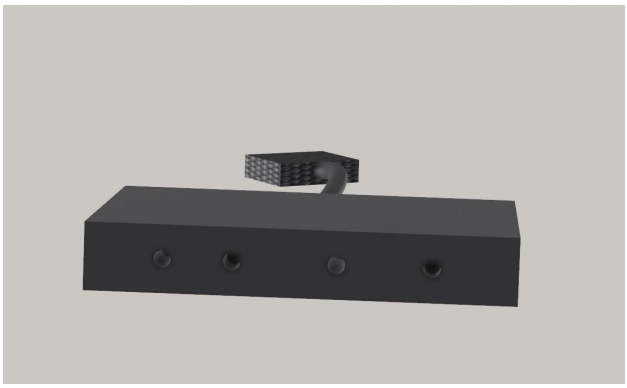

Figure 5: Bottom view of the physical design of our wearable system

2.3 Subsystems

2.3.1 Wearables System

2.3.1.1 Sensor Subsystem

The sensor subsystem collects various health parameters and indicators. These sensors will be placed in a box on a wearable belt with a system attached to the chest to measure pulse and blood oxygen levels. We are planning on implementing the following sensors:

- **Heart Rate**: The heart rate - how fast heart beats - is measured using the MAX30102 chip. The MAX30102 chip has two LEDs, one that is Infrared and the other that is a red light. [2]

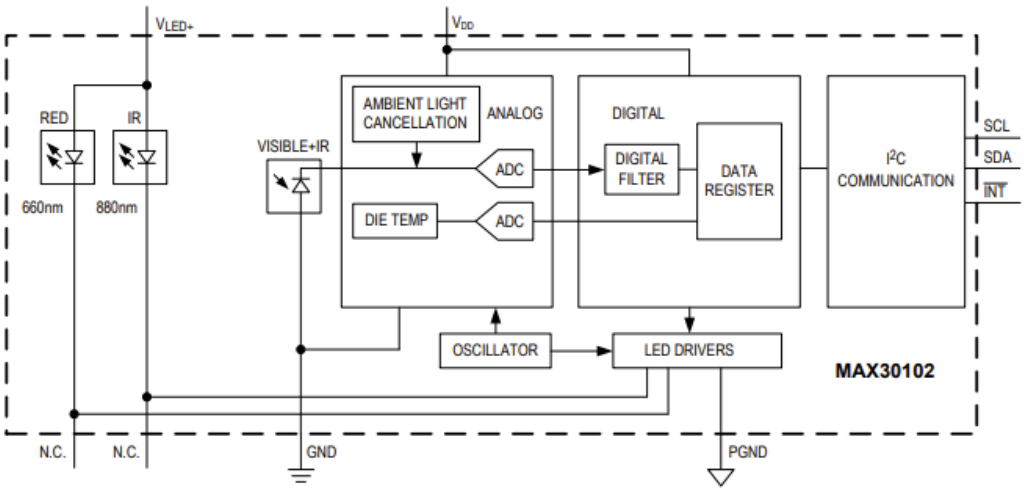

Figure 6: Inner View [3]

Oxygen-rich hemoglobin tends to absorb Infrared light, and it absorbs more if the blood is richer in oxygen. There will be more oxygen in the blood when the heart pumps and less when it relaxes. Therefore, the amount of light that is reflected back is a good indicator of Heart rate. This reflected light is measured using a photodetector, where the amount of current that is produced helps determine the heart rate, and a graph can be produced similar to the one below, which helps with heart rate readings.

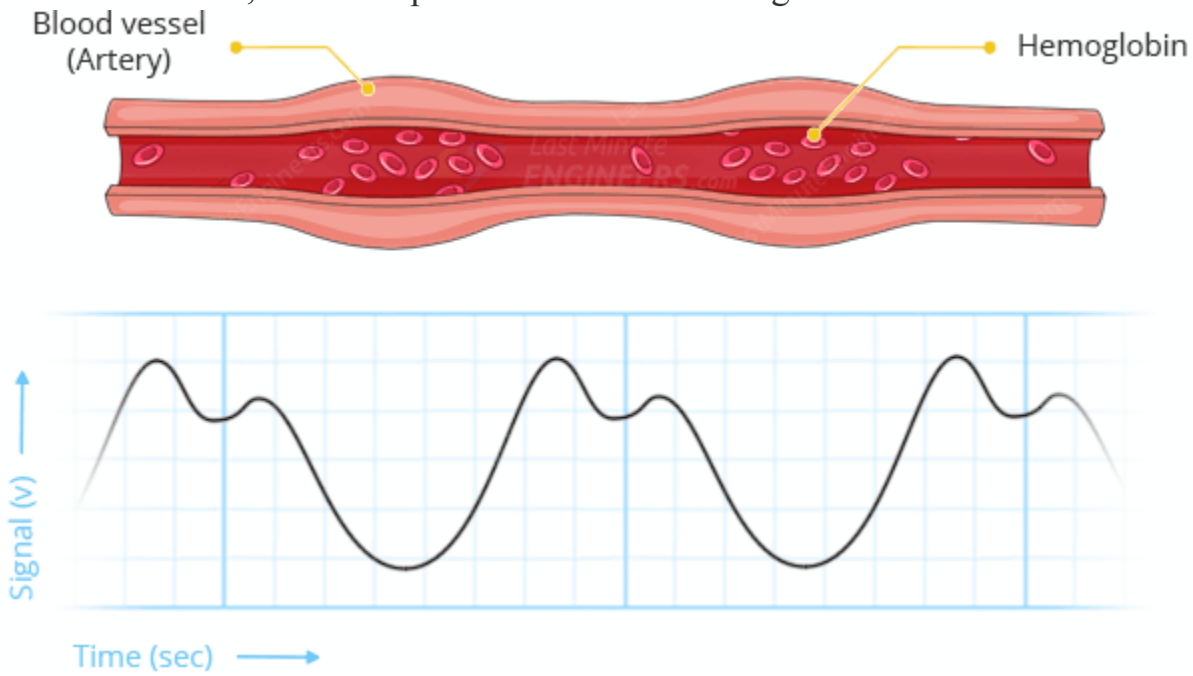

Figure 7. A visual Representation of how the oxygen content in the blood produces a time-varying signal to represent Heart Rate [2]

As seen in the image below, we are able to determine that the output is due to blood because this output produces an AC signal as compared to DC from other parts of the body. [4]

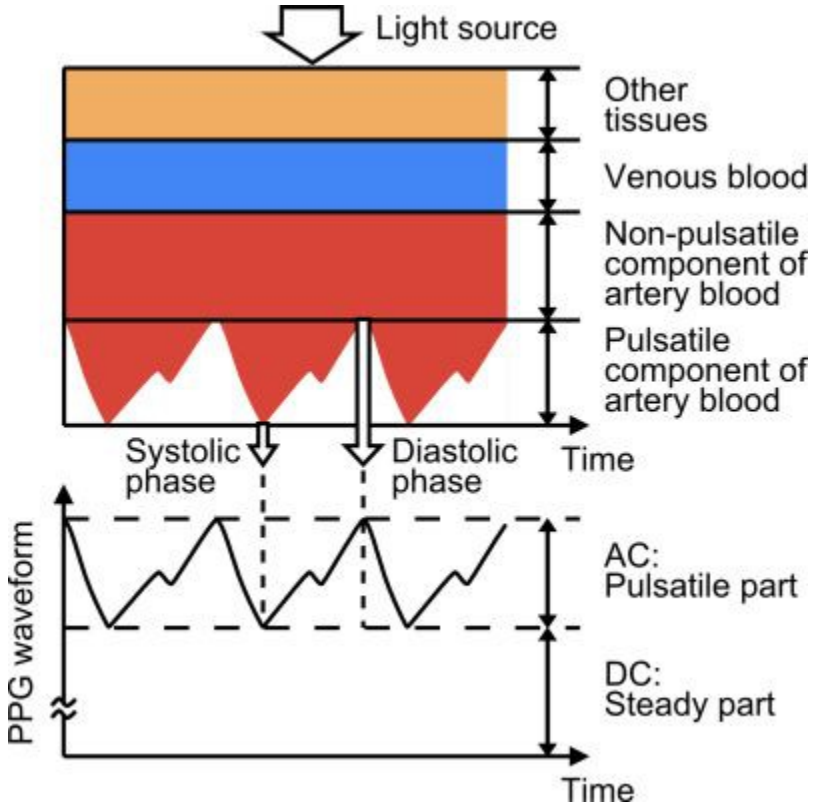

Figure 8. The Analog signal generated from the Heart Rate [4]

This sensor will be placed on the person's chest, with the LEDs making direct contact with the skin through a glass casing. The accuracy of the heart rate measurer is estimated at around 97% [5].

The operating voltage of the device is 3.3-5V. The communication protocol for the MAX30102 is I2C. This chip has a FIFO buffer that stores 32 samples for the heart rate. When the FIFO data register is read, the IR data value (which measures Infrared data) is cleared using an Interrupt [3].

Below is the event sequence that the buffer takes when a heart rate reading is taken:

EVENT	DESCRIPTION	COMMENTS
1	Enter into Mode	I²C Write Command sets MODE[2:0] = 0x02. Mask the A_FULL_EN Interrupt.
2	FIFO is Almost Full, Interrupt Generated	Interrupt is generated when the FIFO has only one empty space left.
3	FIFO Data is Read, Interrupt Cleared	
4	Next Sample is Stored	New sample is stored at the new read pointer location. Effectively, it is now the first sample in the FIFO.

Figure 9: Event Sequence for Reading Heart Rate using Interrupts [3]

Requirement	Verification
Provide an accurate ± 5 beats per minute (BPM) compared to a calibrated Apple Watch measurement	1. One of our members wears our device with the heart rate sensor on their chest and the box on a belt on their waist in addition to an Apple Watch. 2. We monitor the heart rate from our device at a specific time stamp and compare it to the heart rate measured by our Apple Watch at the same timestamp. This timestamp will be determined by sending a digital high to light up an LED when the heart rate is measured. We record the heart rate measurement from the Apple Watch at the time the LED lights up to synchronize the comparison of heart rate measurement. 3. We repeat step two three more times and compare the average of the differences between the heart rate readings of our device and the heart rate readings from the Apple Watch in a table. We will then check if this average is less than 5 BPM.

Table 1: Requirement and Verification table for Heart Rate Sensor

- **Blood Oxygen:** The Blood Oxygen Levels in the body can also be measured using the MAX30102. As mentioned above, the more oxygenated blood is, the more Infrared light is absorbed. The less oxygenated blood is, the more red LED light is absorbed. Based on the amount of red and infrared light that is absorbed, the blood oxygen levels can be determined [6].

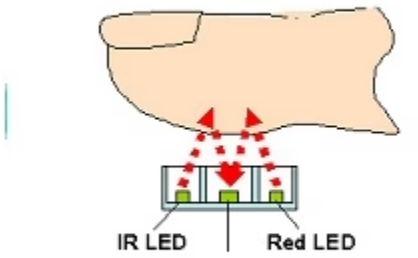

Figure 10. The LEDs interact with the skin to measure blood oxygen levels [6].

The accuracy of the oxygen saturation measurement is estimated at around 98%. This module contains a temperature sensor that is used to calibrate the device for blood oxygen measurements.

This chip has a FIFO buffer that stores 32 samples for blood oxygen measurement. The temperature sensor is not needed for every sample taken.

Below is the event sequence that the buffer takes when blood oxygen level readings are taken:

EVENT	DESCRIPTION	COMMENTS
1	Enter into SpO_2 Mode. Initiate a Temperature measurement.	I^2C Write Command sets MODE[2:0] = 0x03 and A_FULL_EN. Then to enable and initiate a single temperature measurement, set TEMP_EN and DIE_TEMP_RDY_EN.
2	Temperature Measurement Complete, Interrupt Generated	DIE_TEMP_RDY interrupt triggers, alerting the central processor to read the data.
3	Temp Data is Read, Interrupt Cleared	
4	FIFO is Almost Full, Interrupt Generated	Interrupt is generated when the FIFO almost full threshold is reached.
5	FIFO Data is Read, Interrupt Cleared	
6	Next Sample is Stored	New Sample is stored at the new read pointer location. Effectively, it is now the first sample in the FIFO.

Figure 11: Event Sequence for Reading Blood Oxygen Levels using Interrupts [3]

Requirement	Verification
Provide an accurate ± 2% oxygen level compared to a calibrated Apple Watch measurement.	1. One of our members wears our device with the oximeter sensor on their chest and the box on their waist in addition to a smartwatch. 2. We monitor the oxygen level from our device at a specific time stamp and compare it to the oxygen level measured by our Apple Watch at the same timestamp. This timestamp will be determined by sending a digital high to light up an LED when the oxygen level is measured. We record the oxygen level measurement from the Apple Watch at the time the LED lights up to synchronize the comparison of oxygen level measurement. 3. We repeat step two three more times and compare the average of the differences between the oxygen level readings of our

	device and the oxygen level readings from the Apple Watch in a table. We will then check if this average is less than 2%.

Table 2: Requirement and Verification table for Blood Oxygen Sensor

Below is a schematic for the Heart Rate sensor and oxygen blood level monitoring subsystem.

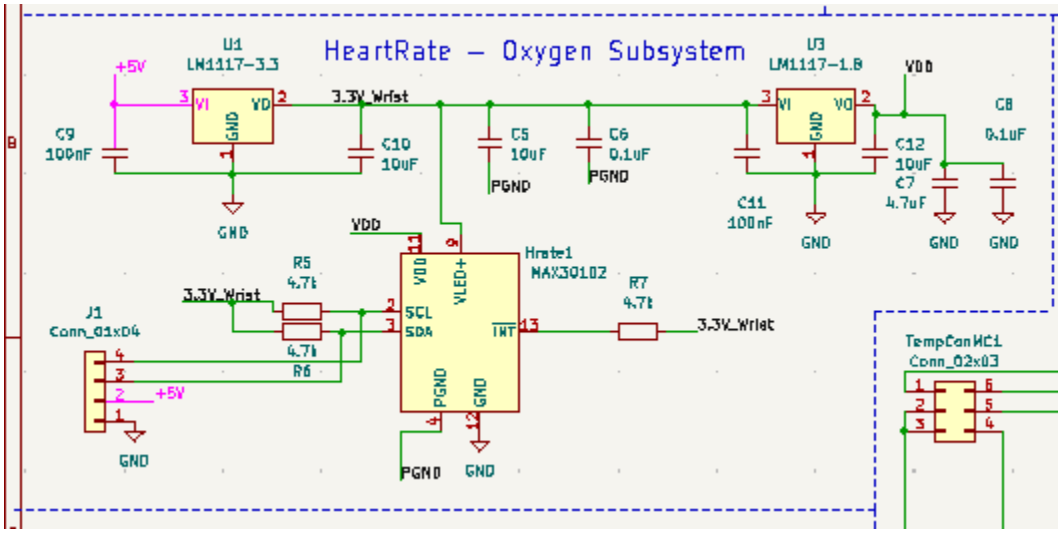

Figure 12: Schematic for Heart Rate and Blood Oxygen Sensor

- **Inertia Measurement Unit Subsystem**: This sensor (**MPU-6050**) is used for fall detection with acceleration across three axes and a 3-axis gyroscope [7]. The accelerometer measures motion in three dimensions as well as orientation. When a person falls, there is a sudden increase in the magnitude of acceleration with a change of orientation and then a decrease in the magnitude of acceleration since the person is no longer moving [8]. Using the data front the 3-axis gyroscope and 3-axis acceleration, we can calculate the orientation - roll, pitch, yaw (see figure 13)- of the elderly using either the complementary filter or digital motion processor.

Therefore, this sensor will help us observe such drastic changes in motion and orientation of the elderly's body. Therefore, we would be able to detect if a person has fallen. Additionally, placing the accelerometer on the belt would also increase fall detection accuracy compared to placing the sensor on the wrist, as seen in figure 14 below [1].Additional Feature: If time permits, we are also thinking of detecting the amount of exercise using the accelerometer to keep track of the step count.

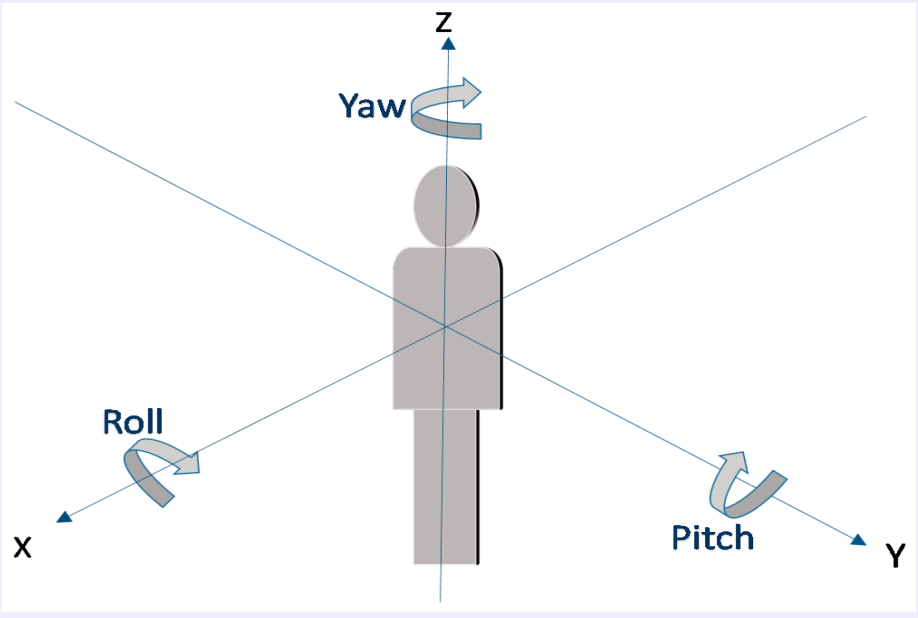

Figure 13: Yaw, Pitch, and Roll angle of the human body [1]

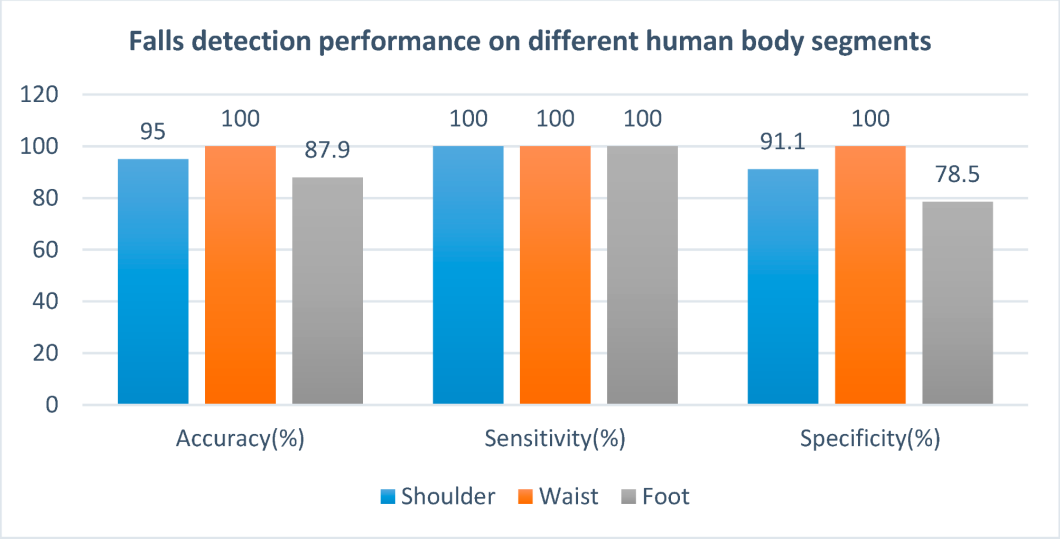

Figure 14: Fall detection accuracy, sensitivity, and specificity performance on different human body parts [1]

The accelerometer uses I2C to communicate with the microcontroller. The SDA and SCL pins will be connected to the I2C pins of the microcontroller for the transfer of data and clock. The VCC pin is connected to 3.3V, and the GND pin is grounded. We also have an alert pin that will be connected to a digital input pin of the microcontroller. The pin we are not using is the ADD0 pin, which is also connected to GND, as we do not use more than one MPU 6050 chip and thus do not need to overwrite the default address for write operations. We could enable an interrupt from MPU 6050 by connecting the interrupt pins to the microcontroller too, or we could choose to do a polling method instead. An interrupt is generated when the Digital Motion Processor FIFO buffer is full.

Pin Number	Pin Name	Description
1	Vcc	Provides power for the module, can be +3V to +5V. Typically +5V is used
2	Ground	Connected to Ground of system
3	Serial Clock (SCL)	Used for providing clock pulse for I2C Communication
4	Serial Data (SDA)	Used for transferring Data through I2C communication
5	Auxiliary Serial Data (XDA)	Can be used to interface other I2C modules with MPU6050. It is optional
6	Auxiliary Serial Clock (XCL)	Can be used to interface other I2C modules with MPU6050. It is optional
7	AD0	If more than one MPU6050 is used a single MCU, then this pin can be used to vary the address
8	Interrupt (INT)	Interrupt pin to indicate that data is available for MCU to read.

Figure 15: MPU 6050 to Microcontroller Pin Connections [9]

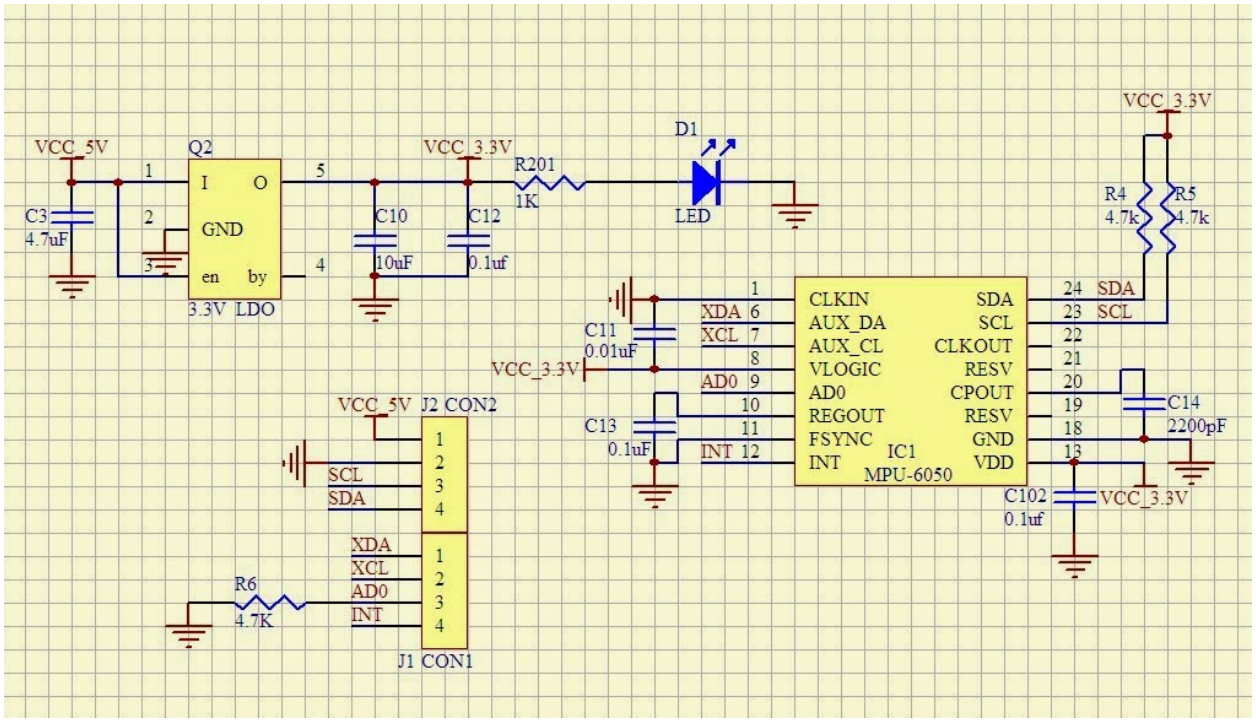

Figure 16: Schematic for MPU6050 chip [10]

Requirement	**Verification**
Be able to detect if a person has fallen with 70% accuracy. This is detected when a person changes their orientation to a different plane rapidly with high acceleration and is motionless for 60 seconds.	1. One of our members wears the device on their waist (Member A). Ensure that our device is turned on. Member B has the IDE terminal open. 2. As member A pretends to fall down on a soft surface, sofa, or bed and remains immobile for 60 seconds, member B starts monitoring the terminal IDE. 3. Member B checks if the “fall detected” prints out on the IDE after member A has fallen. 4. Repeat this experiment ten times to check the accuracy of our fall detection.

Table 3: Requirement and Verification table for Accelerometer

- **GPS**: We will be using the **NEO-6MV2** GPS sensor, which will provide us with the latitude and longitudinal coordinates to pinpoint the individual's exact location. It has precise accuracy updating a location five times a second and tracking a person with a 2.5m distance precision. [11]

  The operating voltage of the device is 2.7V-3.6V. The module we will be using comes with an antenna that receives data from GPS Satellites. This is a patch antenna that is ceramic and must be placed parallel to the horizon. [12]

  The breakout board we will be using is based on the schematic below.

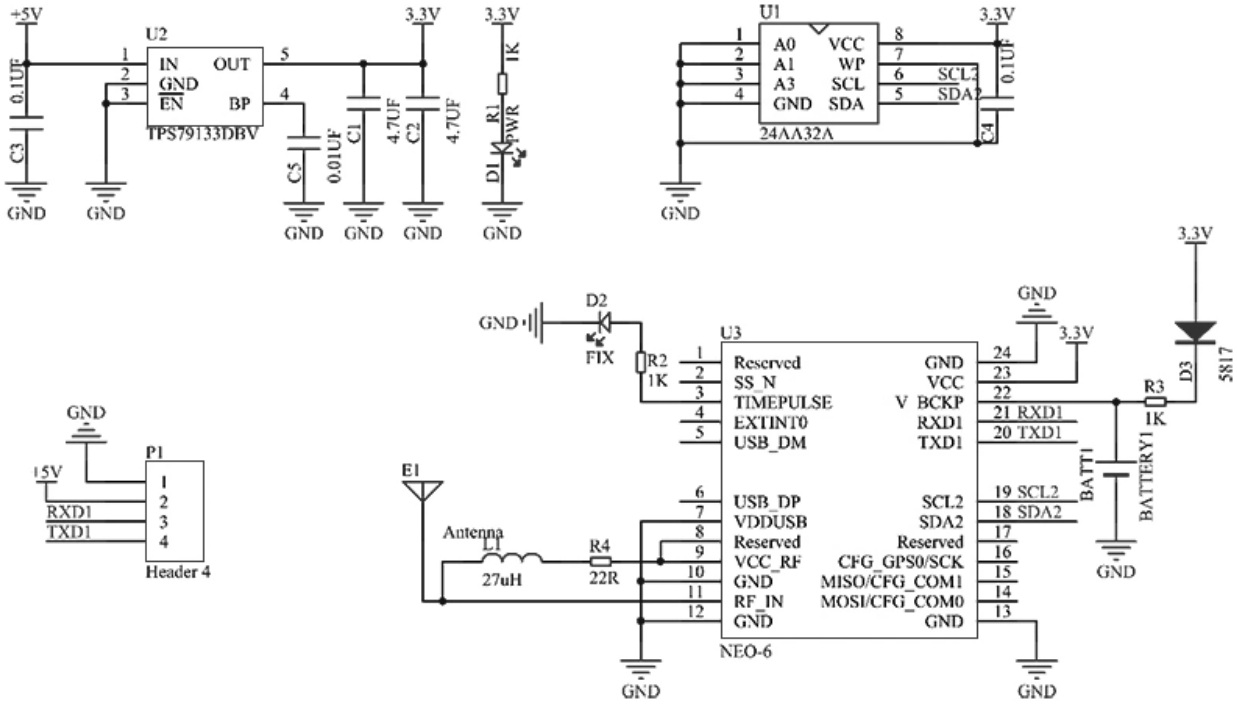

Figure 17: Schematic for the NEO-6MV2 Breakout Board [12]

UART can be used for communication with a default baud rate of 9600. The TXD1 and RXD1 UART pins will be connected to our microcontroller through the UART communication protocol.

A green light indicates that the module is functioning. We will be using the latitude and longitude values obtained from this sensor.

Requirement	Verification

Provide the location of where the elderly person is within an accuracy of ±50 meters from the location measured using Google Maps.	1. One of our members wears our device containing our GPS on their waist and turns it on. The member also opens Google Maps on their phone. 2. Calculate the distance between the coordinates measured by our GPS in the device and the coordinates obtained from Google Maps using Movable Type Scripts. 3. Repeat step two at three different locations. 4. Check if the average of these three distances calculated in step 3 is less than 50 meters.

Table 4: Requirement and Verification table for GPS

2.3.1.2 Control Subsystem

The control subsystem processes the collected health data. With the integrated Wi-Fi, we would send POST requests to our backend API to store this health information in our database.

Microcontroller (ESP32) - The microcontroller processes data from the different sensors, which are connected to the various pins of the ESP32 microcontroller. [13]

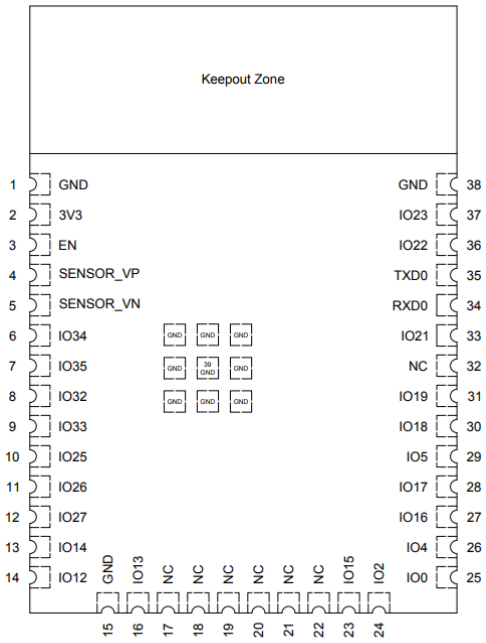

Figure 18: Pinout of the ESP32-WROOM032E chip [13]

Below is the schematic diagram of the ESP32 microcontroller chip. As can be seen from the schematic diagram, the various connectors to various pins of the microcontroller. The button to stop the beeper is connected to pin26(GPIO4) of the microcontroller, which goes to digital high when the button is pressed, thus in turn writing low to the Pin 10 (GPIO25) to which the beeper is connected to stop it from beeping.

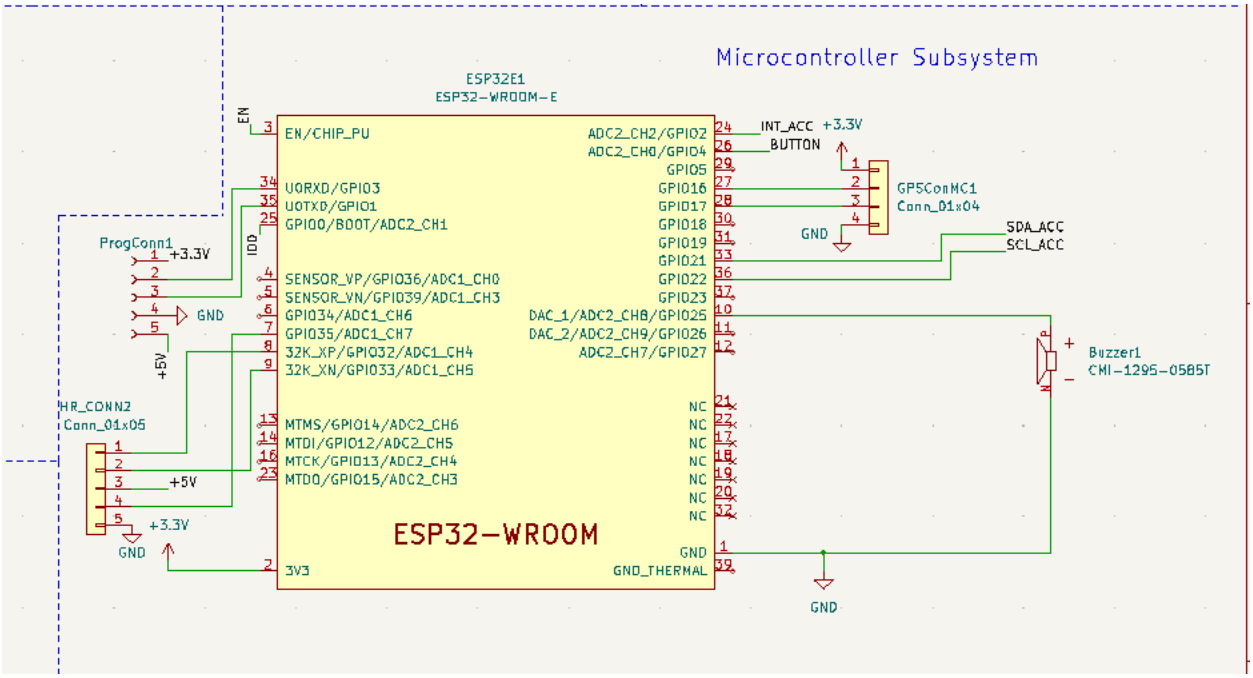

Figure 19: Schematic diagram of the microcontroller pins

The heart rate and blood oxygen subsystem is connected to the microcontroller using the Connector J2, which uses pins 8 and 9 on the microcontroller for SCL and SDA in I2C communication and 5V(from the battery) for VCC supply into the subsystem.

Since the GPS uses UART communication, we connect the GPS breakout board from the TXDO (transmitter pin) to pin 27 (GPIO16) of the microcontroller and RXDO on the GPS Board (receiver pin) to pin 26 (GPIO17) of the microcontroller.

The accelerometer MPU650 (abstracted) can be connected to the microcontroller using the I2C serial communication. The MPU6050 uses SCL and SDA for communication which is connected to pins 33 (GPIO21) and 36 (GPIO22), respectively, of the microcontroller. In addition, the accelerometer is connected to 3.3V through the voltage regulator, and every other pin in the connector is grounded.

The microcontroller sends a signal to the beeper to sound an alarm when there is an irregular heartbeat, blood oxygen levels, or the person has fallen. This microcontroller already has Wi-Fi integrated, so we do not need an additional Wi-Fi module. The data from the microcontroller, after being processed, will be sent over Wi-Fi to the database and then thus displayed on the web application.

Requirement	Verification
Process and format the heart rate reading (BPM), oxygen saturation (percentage), acceleration and orientation in 3 axes (x,y,z), GPS coordinates (latitude, longitude, timestamp, and ESP32 MAC address into JSON format and send this formatted health data to be stored to the MongoDB within 60 seconds ± 20 seconds.	1. One of our members wears our device on their waist (Member A). Ensure that our device is turned on. Member B has the IDE open to monitor the terminal. 2. Member B checks the output received from the various sensors and then ensures that it is accurately processed and stored in the formats specified in the requirement by printing out each reading in the IDE terminal. Member B then checks if this processed data is formatted properly in JSON, which includes sensor data along with the timestamp and the MAC address that uniquely identifies each ESP32 device. 3. Member C then opens up the database and checks if the data collected has reached MongoDB over Wi-Fi. 4. Repeat this experiment three times and check if the JSON formatted data are reaching the database within an average time of 60 seconds ± 20 seconds.

Table 5: Requirement and Verification table for Microcontroller

2.3.1.3 Power Subsystem

The power subsystem powers our entire circuit. This subsystem consists of the following components:

- 5V rechargeable battery, which can be easily recharged connected to the circuit via a Micro-USB B connector header
- The voltage regulator (**LD11173.3V**) ensures that we supply a steady, constant voltage supply of 3.3V through all the components on the box on the belt [14].
- The voltage regulator **(LM111733V**) ensures that we have a supply of 3.3V to the LEDs on the heart rate and blood oxygen sensor worn on the chest and to the **LM111718V** voltage regulator
- The voltage regulator **(LM111718V)** ensures that we have a supply of 1.8 to the IC on the heart rate and blood oxygen sensor worn on the chest.

Below is a schematic diagram of the Power subsystem and the product

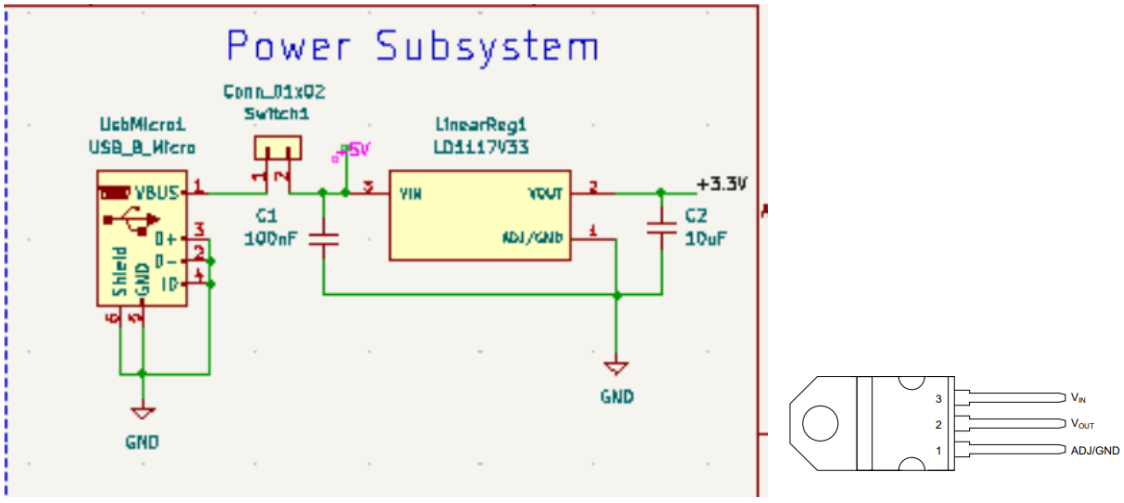

Figure 20: The schematic diagram for the power subsystem and the LD1117v33 [15]

As per the figure above, the 5V battery would be connected to the micro-b USB header, from which we will have a 5V voltage supplied to the entire circuit. We also have a 2-to-1 connector for a switch which could be turned off when the device is not in use to avoid wasting power. This 5V current goes into two-voltage regulators - one is the LD1117V33 which brings it down to 3.3V and is used in the box on the person's waist to power the microcontroller, accelerometer, GPS, and beeper. The other 5V is connected to the voltage regulator (LM1117-3.3), which will bring it down to 3.3V to power the Heart Rate/Blood Oxygen subsystem connected to the sticky pad on the person's chest.

The LD1117V33 is used as it can supply up to a high amount of current - 800mA because that circuit needs around 564 mA of current to power the microcontroller, beeper, accelerometer, and GPS.

The ID pin, Data +VE, and Data -ve pins of the Micro-USB-b are grounded as they do not need to be connected for only a power supply from the connecting header. If this usb-b Micro header were to power a microcontroller, these pins would need to be connected.

Requirement	Verification
1. To provide 3.3V ± 0.2V to the belt wearable subsystem from a 3.3V regulator. 2. To provide 3.3V± 0.2V for the LEDs in the wearable chest subsystem (Heart Rate and Blood Oxygen) using a 3.3V regulator. 3. To provide 1.8V± 0.1V for the IC in the Heart Rate and Blood Oxygen sensor using a 1.8 V regulator.	1. One of our members wears our device on their waist (Member A). Ensure that our device is turned on. Another member (Member B) probes a pin across the LDO voltage out pin and ground using a multimeter. 2. Member B checks the reading on the multimer and ensures that 3.3V± 0.2V is reaching the wearable chest subsystem and the Heart Rate and Blood Oxygen LED from the two different voltage regulators **(Requirements 1 & 2).** 3. Member B checks the reading on the multimer and ensures

	that 1.8V± 0.1V is reaching the heart rate and blood oxygen IC **(Requirement 3)**.

Table 6: Requirement and Verification table for Power Subsystem

2.3.1.4 Beeper Subsystem

The beeper subsystem serves to sound an alarm to call for assistance to the person. This component will play the sound when the microcontroller sends a signal in case of irregularity with the heartbeat and a person falling. The component used in this subsystem is a beeper (**CMI-1295IC-0585T)**.

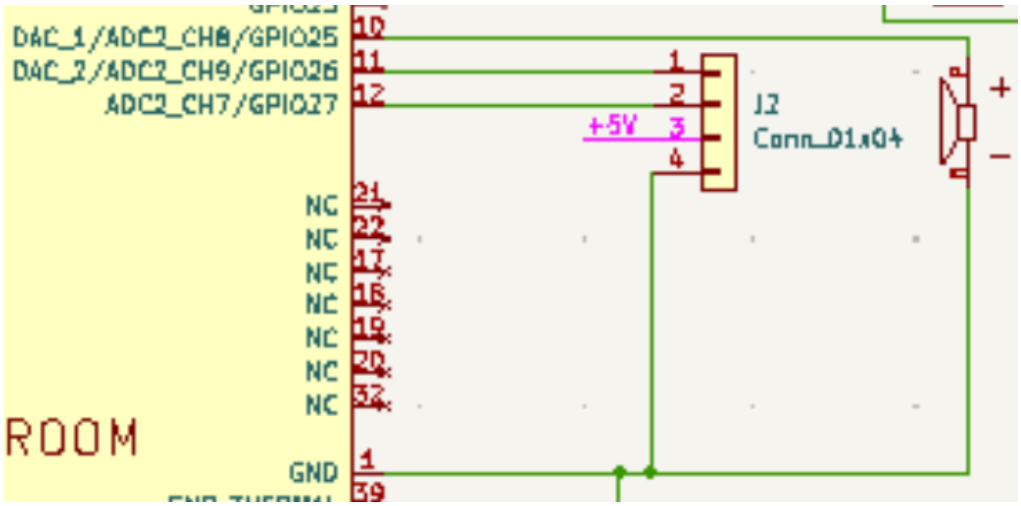

Figure 21: Schematic for Beeper Subsystem

As seen above, the beeper is a component that is connected to a digital logic pin that is set to high when there is an irregularity detected and is otherwise set to digital low. There is a button that ensures that the beeper sound stops when the emergency is acknowledged.

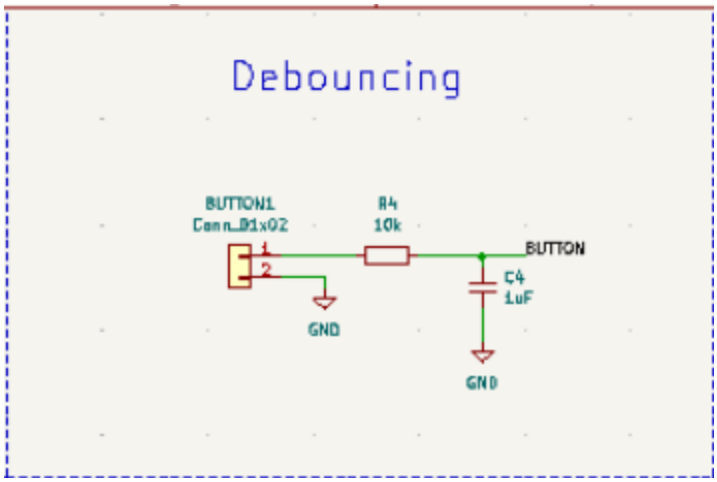

Figure 22: Button Debouncing Circuit

Requirement	Verification
1. Be able to make a sound when a fall is detected with a latency of 30 seconds ± 10 seconds.	1. One of our members wears the device on their waist (Member A). Ensure that our device is turned on. Another member has a phone stopwatch in their hand (member B). 2. As member A pretends to fall down on a soft surface, sofa, or bed and remains immobile for 60 seconds, member B starts the timer. 3. Member B stops the timer when the beeper goes off. We record the duration between when member A falls and is immobile for 60 seconds (the irregularity) and the beeper alert in a table. 4. Repeat the experiment three times and compare if the average time is less than 30

	seconds ± 10 seconds. We repeat the same process by simulating an irregular heart rate.
2. Be able to make a sound when the heart rate is less than 60 BPM and or greater than 135 BPM within 30 seconds ± 10 seconds after the measurement.	1. One of our members wears the device on their waist (Member A) and the sticky pad on the chest subsystem. Ensure that our device is turned on. Member B opens a time display with seconds accuracy on their laptop. 2. We hardcode the heart rate value to 136 BPM on the microcontroller. 3. Print out the current timestamp right after the previous hard code line. Member B presses the print screen button on their laptop as soon as they hear the beeper goes off. 4. Find the difference between the time displayed in the screenshot when the beeper goes off and the time displayed in the terminal IDE when the heart rate is out of threshold 5. Repeat the experiment three times and compare if the average time difference is less than 30 seconds ± 10 seconds. 6. We repeat steps 1- 6 for another value out of threshold - 59 BPM.

3. Be able to make a sound when the blood oxygen level is below 96%.	1. One of our members wears the device on their waist (Member A) and the sticky pad on the chest subsystem. Ensure that our device is turned on. Member B opens a time display with seconds accuracy on their laptop. 2. We hardcode the blood oxygen percentage to 95% on the microcontroller. 3. Print out the current timestamp right after the previous hard code line. 4. Member B presses the print screen button on their laptop as soon as they hear the beeper goes off. 5. Find the difference between the time displayed in the screenshot and the time displayed in the terminal IDE 6. Repeat the experiment three times and compare if the average time is less than 30 seconds ± 10 seconds. We repeat the same process by simulating an irregular oxygen level.

Table 7: Requirement and Verification table for Beeper

2.3.2 Web Application System

2.3.2.1 Database & Backend Subsystem

The database subsystem stores the processed health information we collect from the wearable sensors. We intend to use MongoDB as it is more scalable and allows more flexibility to the structure of the stored data compared to MySQL. The requirement for this subsystem is to store the heart rate information of the last week ± one day, last fall detected for one week ± one day, and blood oxygen levels for a week ± one day. To fulfill this requirement, we plan to use the TTL indexes on MongoDB, automatically removing documents from the collection after a certain period [16].

Our design has two main collections: the person collection and the sensor data collection. The person collection holds the information about the person's name and associates a person with the MAC address of ESP32 to identify an elderly person with a device uniquely. The MAC address is stored in the person_id field, as shown in figure 23. The sensor data collection holds all the sensors' data with their measurements time. We also include the personId field in the sensor data collection as a reference to the person collection.


```
{
  "_id":"507f1f77bcf86cd799439011",
  "person_id":"30:AE:A4:07:0D:64",
  "firstName":"Matty",
  "lastName":"Healy"
}
```


Figure 23: Sample Document in the Person Collection


```
{
  "_id":"507f1f77bcf86cd799439011",
  "personId":"30:AE:A4:07:0D:64",
  "timeStamp":"2023-02-21T01:11:18.965Z",
  "HeartRate":80,
  "BloodOxygen":95,
  "Acceleration":[
    3,
    4,
    5
  ],
```

```
  "Gyroscope":[
    20,
    23,
    44
  ],
  "YawPitchRoll":[
    20,
    22,
    32
  ],
  "Notification":[
    0,
    0,
    0,
    1
  ],
  "GpsLat":"32.18N",
  "GpsLong":"52.5W",
  "ExpireAfterSeconds":604800
}
```


Figure 24: Sample Document in the Sensor Data Collection

Requirements	Verifications
1. Storage of heart rate information of the last week ± one day, blood oxygen levels for a week ± one day, and fall history of the last week ± one day	1. One member put the device on. 2. Turn on the device for at least 5 minutes. 3. Check if the newly recorded heart rate and blood oxygen levels, and the last fall are stored on the database with the correct TTL index.

	4. Turn off the device 5. Seven days later, at the same exact time, query the database using the recorded time attribute and see if last week's health data has been removed. 6. Record the trial in a table format. (Date Recorded, Database Revisit Date, Data Status - Active/Deleted)

Table 8: Requirement and Verification table for Database System

The backend subsystem supports GET and POST requests. The GET requests are used for obtaining data from documents from the MongoDB collection to be displayed on the frontend of our web application. The POST request is used for sending data from the microcontroller to the database.

Requirements	**Verifications**
1. Able to send GET requests to obtain the heart rate, blood oxygen, fall history, and GPS coordinates from the database to display on the user interface.	1. Open the MongoDB collection and select a specific document. 2. Formulate the request in Postman, using the GET by id method to find the specified document. 3. Send the GET request to Postman and see if the same document is returned.
2. Able to send POST requests to enter the heart rate, blood oxygen, fall history, and GPS coordinates into the database from the microcontroller in the JSON format.	1. Create a POST request containing sensor data in JSON format using C on the Arduino IDE. 2. Open MongoDB and check if this data entry is added to the database. 3. In order to see the formatted output, we can query this

	entry from the database using the Postman application, sorting by the latest timestamp.

2.3.2.2 User Interface Subsystem

This web interface allows the staff to detect everyone's vitals, and the staff receives a notification when a person's vitals are outside the normal range. We will be using React and Javascript to create this frontend web platform. The frontend web page would display close to real-time heart rate and blood oxygen levels of multiple elderly as well as the measurement time by sending HTTPs requests to query the database.

Figure 25: Landing page/home page design of our web application

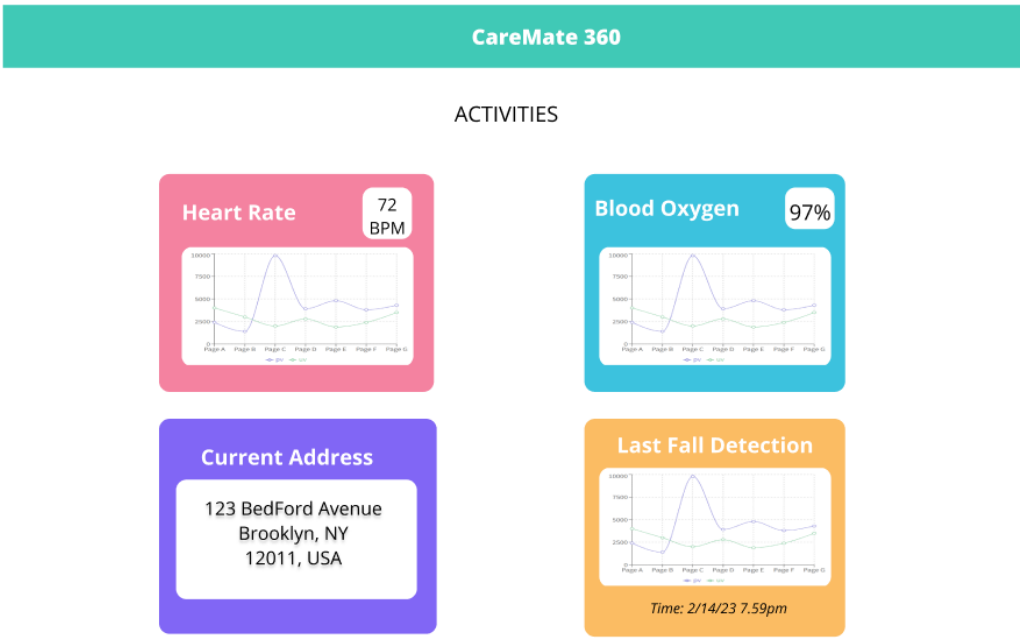

Figure 26: Detail page design of our web application

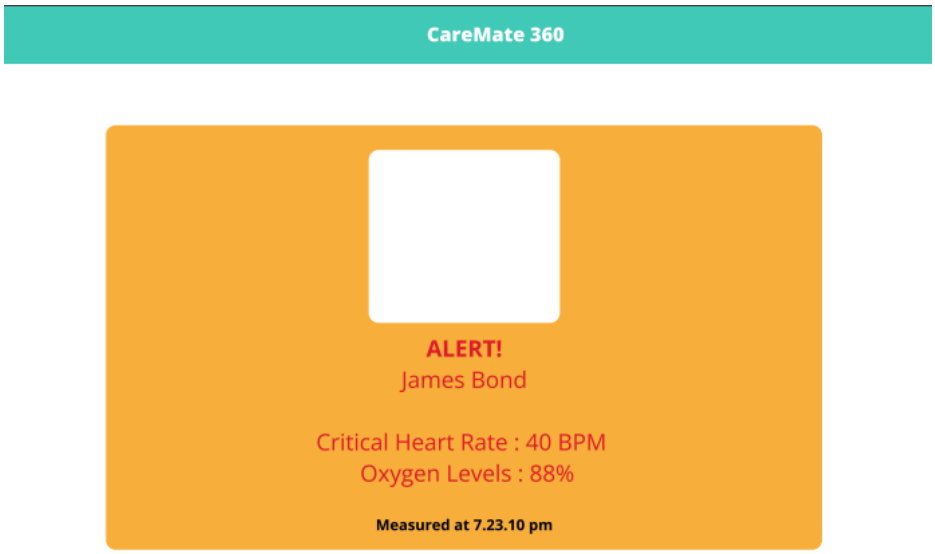

Figure 27: Alert page design of our web application

Requirements	Verifications
1. Display heart rate, fall detection, blood oxygen levels, and GPS coordinates within 90 seconds ± 30 seconds after measurements.	1. One member puts the device on (member A) and makes sure it is turned on. 2. One member opens our web application page and a clock up to seconds (member B). Member C opens the Arduino IDE terminal console and prints out the timestamp when sensors have measured the data (time X) . 3. Member B records the current time with accuracy up to seconds (time Y) using the clock at which this time X shows up on the web application. 4. Calculate the duration taken for the measurement to show up on the web application using the formula time Y - time X. 5. Record the time taken in a table format (Record Time X, Time Y, Time of measurement) 6. Turn off the device.
2. Send out an alert when an irregularity(as defined in the first high-level requirement) occurs within 90 seconds ± 30 seconds.	1. One of our members wears the device on their waist (Member A) and the sticky pad on the chest subsystem. Ensure that our device is turned on. Member B opens a time display with seconds accuracy on their laptop. 2. We hardcode the heart rate value to 136 BPM on the microcontroller. Print out the current timestamp right after the previous hard code line. Member B presses the print screen button on their laptop as soon as

	they see the notification pop up on the website with this irregular heart rate to capture the time the alert appears. 3. Find the difference between the time displayed in the screenshot when the website gets the notification and the time displayed in the terminal IDE when the heart rate is detected to be out of threshold.ll 4. Repeat the experiment three times and compare if the average time difference is less than 90 seconds ± 30 seconds. 5. We repeat steps 1- 6 for other irregularities.

Table 9: Requirement and Verification table for User Interface Subsystem

2.4 Tolerance Analysis

When using different voltage regulators to provide the different subsystems with the required amount of constant voltage, it is possible that the regulator can overheat based on the power used at the regulators. Therefore, we will calculate and analyze the amount of power used by each of these regulators based on the different components each regulator supplies voltage to and the regulators' dropout voltage. Based on how high it heats up, we can check whether this is below the maximum temperature range for the device, and this method can prevent overheating.

As seen below, we will be using three different voltage regulators. Two of them are used to step down the voltage to a steady 3.3V, and one voltage regulator steps down the voltage from 3.3V to 1.8V. The figure below illustrates the circuit that steps down from 5V supplied by the battery to 3.3V, which is needed by devices present in a box on a wearable belt. The accelerometer, GPS, microcontroller, and Beeper are all supplied with 3.3V.

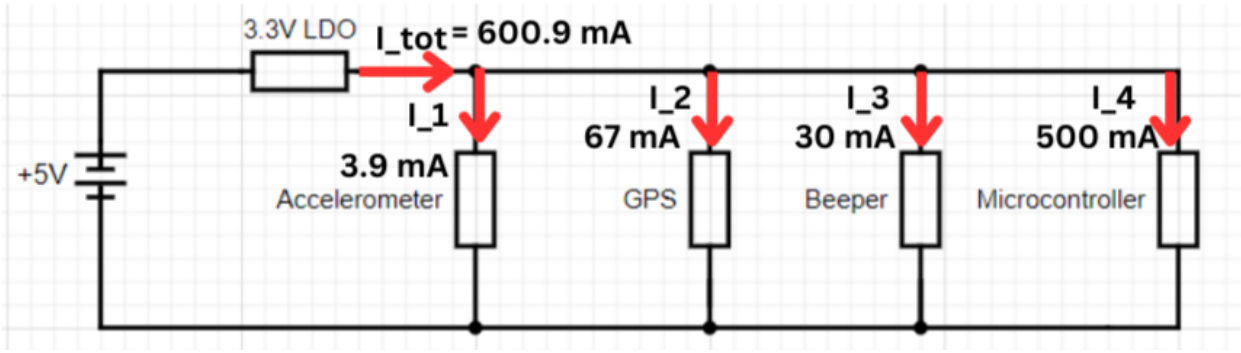

Figure 28: Abstract circuit for tolerance analysis calculation for wearable belt

For the wearable belt system
The thermal resistance of the voltage regulator **(LD1117v33-3pin-TO220)**, $\theta_{Ja} = 50$ degree C/W
For the sensors which are connected to a 5V battery and the corresponding voltage regulator using 3.3V within the **Wearable Device on Belt**: GPS, Beeper, Accelerometer and Microcontroller draw the current as following:
Accelerometer Sensor: At 3.3V, for resistance of $10k\Omega$, current supplied, $I_1 = 3.9mA$
GPS At 3.3V, current supplied, $I_2 = 67mA$
Beeper At 3.3V, current supplied, $I_3 = 30mA$
Microcontroller: At 3.3V, current supplied, $I_4 = 500mA$

Adding up all the current, we get total current from the voltage regulator at 3.3V, $I_{total} = I_1 + I_2 + I_3 + I_4$
$= 3.9 + 30 + 500 + 67\text{mA} = 600.9$ mA
Voltage drop, $V = V_{in} - V_{out} = 5$ - 3.3 V = 1.7V
Power consumed, P $= V * I_{total} = 1.7 * 600.9 * 10^{-3} = 1.02153W$
Temperature change, $\Delta T = P * \theta_{Ja} = 50 * 1.02153 = 51.0765$ degree Celsius.

If we add the temperature difference to the room temperature, max temperature achieved = 25 degree C + 51.0765 degree C = **76.0765 degree Celsius.**

Since this is well within the maximum temperature of 125 degree C of the voltage regulator, the circuit is safe for operation within the wearable device of the belt and the voltage regulator will not blow up.

Figure 29: Mathematical verification of tolerance of 3.3V LDO on wearable belt

A different voltage regulator is used to step down the voltage from 5V to 3.3V for all components that require 3.3V on the wearable system attached to the chest. A voltage of 1.8V is also required for the MAX30102 chip, so the voltage of 3.3V that is supplied should be stepped down to 1.8V using a 1.8V regulator.

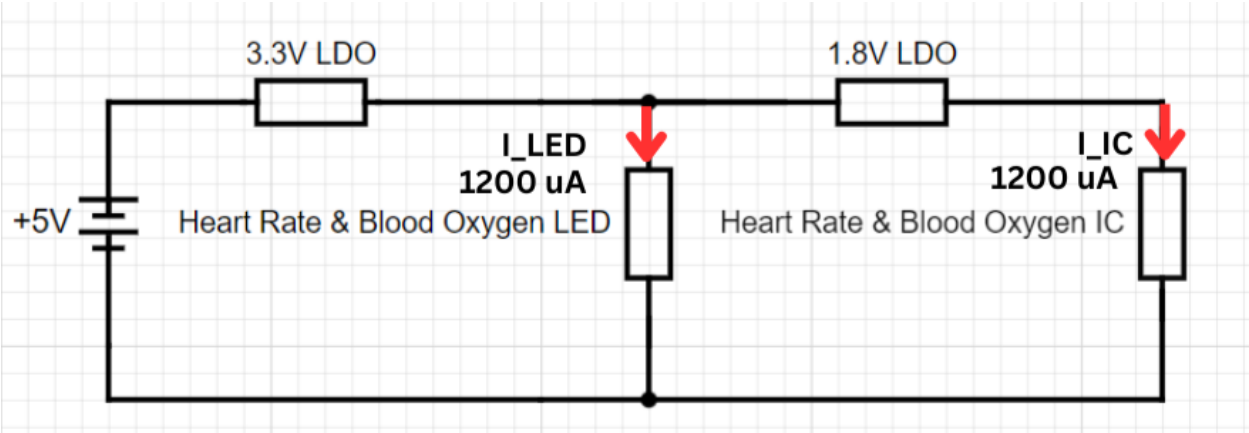

Figure 30: Abstract circuit for tolerance analysis calculation for wearable chest strap

For the sticky chest pad
3.3V LDO
The thermal resistance of the voltage regulator **(LM1117v33-4pin-SOT223)**, $\theta_{Ja} = 61.6$ degree C/W

For the sensors which are connected to a 5V battery and the corresponding voltage regulator using 3.3V within the **Wearable Device on Chest**: Heart Rate and Blood Oxygen Sensor LEDs
Heart Rate Sensor LED: At 3.3V,current supplied, $I_{LED} = 1200uA$
Voltage drop, $V = V_{in} - V_{out} = 5$ - 3.3 V = 1.7V
Power consumed, P $= V * I_{LED} = 1.7 * 1.2 * 10^{-3} = 0.00204W$
Temperature change, $\Delta T = P * \theta_{Ja} = 61.6 * 0.00204 = 0.125664$ degree Celsius.

If we add the temperature difference to the room temperature, max temperature achieved = 25 degree C + 0.125664 degree C = **25.1256 degree Celsius**

Since this is well within the maximum temperature of 125 degree C of the voltage regulator and has negligible temperature difference with the environment, the circuit is safe for operation within the wearable device (even in contact with skin) and the voltage regulator will not blow up.

Figure 31: Mathematical verification of tolerance of 3.3V LDO on chest pad

1.8V LDO

The thermal resistance of the voltage regulator **(LM1117v18-4pin-SOT223)**, $\theta_{Ja} = 61.6$ degree C/W

For the sensors which are connected to a 3.3V source and the corresponding voltage regulator at 1.8V within the **Wearable Device on Chest**: Heart Rate and Blood Oxygen IC

Heart Rate Sensor IC: At 3.3V,current supplied, $I_{IC} = 1200uA$
Voltage drop, $V = V_{in} - V_{out}$ = 3.3 - 1.8 V = 1.5V
Power consumed, $P = V * I_{IC} = 1.5 * 1.2 * 10^{-3} = 0.0018W$
Temperature change, $\Delta T = P * \theta_{Ja} = 61.6 * 0.0018 = 0.11088$ degree Celsius.

If we add the temperature difference to the room temperature, max temperature achieved = 25 degree C + 0.11088 degree C = **25.11088 degree Celsius**

Since this is well within the maximum temperature of 125 degree C of the voltage regulator and has negligible temperature difference with the environment, the circuit is safe for operation within the wearable device (even in contact with skin) and the voltage regulator will not blow up.

Figure 32: Mathematical verification of tolerance of 1.8V LDO on chest pad

3. Cost and Schedule

3.1 Cost Analysis

Parts

Subsystem	Part Number	Description	Manufacturer	Quantity	Unit Price	Total Price	Link
	ESP32-DevKitC-32E	Microcontroller Devboard for Prototyping ESP32	Mouser	2	10	20	DevBoard Link
	ESP32-WROOM-32E CHIP	ESP32-WROOM-32E Microcontroller module	Digi-Key	2	2.7	5.4	ESP32 Module
Control Subsystem	CP2102 USB 2.0	CP2102 Microcontroller USB-to-UARTProgrammer	SparkFun	2	0	0	Programmer Link
GPS Subsystem	NEOGY6MV2 GPS	NEO6M GPS module with Antenna	Amazon	2	8.99	17.98	GPS Link
	MPU-6050	MPU6050 Accelerometer Integrated Chip	Mouser	2	8	16	Accelerometer Chip Link
Accelerometer Subsystem	GY-521 MPU-6050	MPU6050 Accelerometer GY521 Breakout Board	Amazon	3	3.296666667	9.89	Accelerometer Link
	RA11131121	2 connector Toggle Switch	Digi-Key	2	0.6	1.2	Switch Link
	LD1117V33	Voltage Regulator LD1117V33, 800mA, 3.3 V	Digi-Key	3	0.76	2.28	LDO 3.3 TO-220 Link
	USB Battery Pack - 2200 mAh Capacity	5V Rechargeable Battery with USB Cable	Adafruit	1	14.95	14.95	Battery Link
Power Subsystem	USB3076-30-A	Micro-USB-b 5-pin Connector Header	Digi-Key	2	0.7	1.4	Micro USB B Header Link
Temperature Sensor	TMP102	TMP102 Digital Temperature Sensor	Digi-Key	3	1.87	5.61	Temperature Sensor Link
	Sensor Module MAX30102	Heart Rate and Blood Oxygen Level Pulse Sensor	Amazon	3	2.996666667	8.99	HeartRate Sensor Link
	MAX30102EFD+T	MAX30102 Heart Rate and Blood Oxygen IC	Digi-Key	2	10.77	21.54	MAX30102 Chip Link
	LM1117MP-1.8/NOPB	1.8V Voltage Regulator	Digi-Key	2	1.7	3.4	LDO 1.8V Link
Heart Rate Sensor	LM1117MP-3.3/NOPB	3.3V Voltage Regulator	Digi-Key	4	0	0	LM1117MP-3.3 Link
Alarm	CMI-9605-0580T	Beeper	Digi-Key	3	1.22	3.66	Beeper Link
Total Budget Spent						132.3	

Figure 33: Breakdown of cost analysis for non-standard parts

** In the budget, a temperature sensor is included because we initially planned to do temperature measurement, but the parts included only measure die temperature and are not accurate enough. In the revised design, this temperature sensor is not included but took up someplace in our budget since the part was already ordered.*

Total cost from parts, including shipping and taxes = $132.3

Labor

Typically, as a Computer Engineering graduate from ECE at Illinois, the hourly rate of the Engineer would be $50/hour. Since all three of us are Computer Engineering majors, we estimate the Cost per hour for each member would be $50/hour. We then estimate that each of us would be working 25 hours per week for the 11 weeks of the semester as defined in the schedule.

Thus the salary for each member would be ($50/hour) x 2.5 x 275 hours of work = **$34,375**

Thus total salary for all 3 of us would be **$103,125, which is Total Labor Cost.**

Labour Cost	Total Number of Hours	Cost ($)/ Hour	Total Cost
Per Member Cost	275	50	34375
3 Member cost			103125

Figure 34: Labor cost included for three members with a reasonable salary

SUM OF COSTS as a GRAND TOTAL = $103,125 + $132.30 = $103,257.30

3.2 Schedule:

Week	Task	Assignee
	Finish CAD design	Everyone
	Finish Design Document	Everyone
	Frontend Design	Jeep
	Database Design	Aishee
	Finish PCB Design	Everyone
02/20 - 02/26	Start Prototyping Heart Rate Sensor and start data collection	Sanjana
	Start writing Code for Blood Oxygen	Sanjana
	Start Writing Code for Accelerometer	Jeep
	Start Writing Code for GPS	Aishee
	Obtaining and Processing data from Sensors	Jeep, Sanjana
02/27 - 03/05	Fall Detection and orientation code and testing	Aishee, Jeep, Sanjana
	Finalize protoype with multiple sensors and data collection	Aishee, Jeep
	Finish code writing for ALL sensors	Everyone
	PCB Revision and Finalize Design	Sanjana, Jeep
	Create Database on MongoDB	Aishee, Sanjana
03/06 - 03/12	Writing code for sending health data to store in the database	Jeep
03/13 - 03/19	Debugging/Finishing all sensor related codes	Everyone
	Finish Solder PCB	Sanjana
	Program Microcontroller	Aishee
	Test Soldered PCB	Jeep
03/20 -03/26	Debugging of PCB and revise design	Everyone
	Debugging PCB and Software	Everyone
	Ensure data goes into database in correct format	Aishee, Sanjana
03/27 - 04/02	Encasing the product	Sanjana
	Debugging PCB	Everyone
	Structure for Web Architecture and Database	Everyone
	Start Coding Frontend	Jeep
04/03 - 04/09	Work on Backend	Aishee, Sanjana
	Integrate everything in web application	Aishee, Jeep
	Testing product and notification on web app	Everyone
04/10 - 04/16	Prepare for Mock Demo	Everyone
	Final Testing	Aishee, Sanjana
	Finishing touches on frontend	Jeep
04/17 - 04/23	Prepare for Final Demo	Everyone
04/24 - 04/30	Final Demo and prepare for presentation	Everyone
	Final Presentation	Everyone
04/30 - 05/04	Finishing Up Final Paper	Everyone

Figure 35: Schedule breakdown of the project over the next weeks of semester

4. Ethics and Safety

1) Healthcare information of patients should not be disclosed and should be protected under secure databases. We would not collect any data or healthcare information of the patient without consent (under the Privacy Rule and Security Rule of HIPAA). This is also closely related to the code IEEE 7.8.I.1, wherein the patient's health and privacy should be the foremost concern, and our design will ensure in no way are these harmed or violated [17]. We would avoid this ethical breach by ensuring that the elderly are aware of the health activities collected before putting the device on. We would also ensure that our database is secured before releasing this innovation to the public to prevent any protected health information leaks.

2) Related to IEEE code 7.8.II.9, we would ensure that our product does not physically harm any individual [17]. We will avoid this ethical breach by ensuring that covering and material in contact with the patient are not toxic. This includes any potential material used for webbing and adhesion to the users' skin. Additionally, we will ensure that radiation levels from any sensors are not harmful and that all parts are encased in a protective covering so that there are no live wires or power connections in contact with the person wearing the wearable device. Lastly, we will ensure that any component that touches the users' skin is not too hot. One of the design decisions we made to ensure the users' safety was to separate the PCB into two boards. This ensures that the board with sensors (heart rate and oxygen sensor) that require skin contact will not heat up too much. According to our tolerance analysis, the PCB with the heart rate and oxygen sensor would only heat up to 25.13C, using the maximum possible current drawn from the heart rate sensor.

3) According to the ACM Code of Ethics 2.9, we should design and build robust and usable systems [18]. To ensure that we can follow this ethical code, we would robustly test our software to ensure that our web application works and can handle displaying close to real-time health information. We aim to make this close to a real-time system by creating a backend that can access readings collected from all the sensors at a time period close to the last sensor's picked-up reading and simultaneously make these changes to our database, which will then be reflected on our frontend design. This will ensure that the software aspect of our project is able to carry out its desired function in a timely yet accurate manner by ensuring that the health staff can track the health data of an individual in close to real time.

5. Citations:

[1] A. Mao, X. Ma, Y. He, and J. Luo, “Highly portable, sensor-based system for Human Fall Monitoring,” *MDPI*, 13-Sep-2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/9/2096. [Accessed: 21-Feb-2023].

[2] Last Minute Engineers, “Interfacing MAX30102 pulse oximeter and heart rate sensor with Arduino,” *Last Minute Engineers*, 06-Feb-2022. [Online]. Available: https://lastminuteengineers.com/max30102-pulse-oximeter-heart-rate-sensor-arduino-tutorial/. [Accessed: 21-Feb-2023]

[3] “High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health.” [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/MAX30102.pdf. [Accessed: 22-Feb-2023].

[4] “Max30102-- how does it work?: Details,” *Hackaday.io*. [Online]. Available: https://hackaday.io/project/26103-preemiealert/log/64833-max30102-how-does-it-work. [Accessed: 21-Feb-2023].

[5] “How max30102 pulse oximeter and Heart Rate Sensor Works and how to interface it with Arduino?,” *How Does the MAX30102 Pulse Oximeter and Heart Rate Sensor Work and how to Interface it with Arduino?* [Online]. Available: https://circuitdigest.com/microcontroller-projects/how-max30102-pulse-oximeter-and-heart-rate-sensor-works-and-how-to-interface-with-arduino#:~:text=How%20accurate%20is%20MAX30102%3F,saturation%20(SpO2)%2C%20respectively. [Accessed: 21-Feb-2023]

[6] “Keyestudio MAX30102 Heart Rate Sensor Oxygen pulse breakout for Arduino,” *Future Electronics Egypt*. [Online]. Available: https://store.fut-electronics.com/products/pulse-oximeter-spo2-heart-rate-sensor-max30100#:~:text=The%20MAX30102%20can%20be%20used,the%20infrared%20light%20is%20needed. [Accessed: 21-Feb-2023].

[7] InvenseSense, “MPU-6000 and MPU-6050 Product Specification Revision 3.4”, PS-MPU-6000A-00 datasheet, Aug. 2013.

[8] D. Patel, “How to build a fall detector with Arduino: Arduino,” *Maker Pro*, 07-Feb-2023. [Online]. Available: https://maker.pro/arduino/tutorial/how-to-build-a-fall-detector-with-arduino. [Accessed: 07-Feb-2023].

[9] “MPU6050 accelerometer and Gyroscope Module,” *Components101*. [Online]. Available: https://components101.com/sensors/mpu6050-module. [Accessed: 21-Feb-2023].

[10] “Haoyu Electronics: Make engineers job easier.” [Online]. Available: https://www.haoyuelectronics.com/Attachment/GY-521/GY-521-SCH.jpg. [Accessed: 22-Feb-2023].

[11] Last Minute Engineers, “In-depth: Interface ublox NEO-6M GPS module with Arduino,” *Last Minute Engineers*, 26-Jun-2022. [Online]. Available: https://lastminuteengineers.com/neo6m-gps-arduino-tutorial/. [Accessed: 21-Feb-2023].

[12] T. K. Hareendran, “NEO-6M GPS module: Setup & Introduction: Electroschematics,” ElectroSchematics.com, 06-Feb-2023. [Online]. Available: https://www.electroschematics.com/neo-6m-gps-module/. [Accessed: 21-Feb-2023].

[13]“Espressif ESP32-WROOM-32E datasheet.” [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf. [Accessed: 22-Feb-2023].

[14] M. Staff, “What is a voltage regulator?,” *Digi*, 01-Jul-2020. [Online]. Available: https://www.digikey.com/en/maker/blogs/2020/what-is-a-voltage-regulator. [Accessed: 07-Feb-2023].

[15] “LD1117V33: Digi-key electronics,” *Digi*. [Online]. Available: https://www.digikey.com/en/products/detail/stmicroelectronics/LD1117V33/586012?utm_adgroup=Integrated+Circuits+%28ics%29&utm_source=google&utm_medium=cpc&utm_campaign=Shopping_Supplier_STMicroelectronics&utm_term=&utm_content=Integrated+Circuits+%28ics%29&gclid=CjwKCAiAuaKfBhBtEiwAht6H75JyZpzM_EroEn7Scb0qmD7dckJhzxS9mqZbZrp3e-5RfxQbMpnH_BoCpS8QAvD_BwE. [Accessed: 21-Feb-2023].

[16] MongoDB, “TTL Indexes,” *TTL Indexes - MongoDB Manual*. [Online]. Available: https://www.mongodb.com/docs/manual/core/index-ttl/. [Accessed: 21-Feb-2023].

[17] “IEEE Code of Ethics”, IEEE. [Online]. Available:https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed: 06-Feb-2023].

[18] “ACM Code of Ethics”, ACM. [Online]. Available: https://www.acm.org/code-of-ethics [Accessed: 06-Feb-2023].

[19] E. I. T. S. Services, “Course overview,” *Course Overview :: ECE 445 - Senior Design Laboratory*. [Online]. Available: https://courses.engr.illinois.edu/ece445/. [Accessed: 21-Feb-2023].

Appendix A

Lab Safety Manual

1. All members should read the battery safety manual provided by ECE 445 course, noting all emergency contacts we might need.
2. All members should get fire safety and fire extinguisher training.
3. Before connecting power to our PCB, we will get our circuit verified by a power-centric TA.
4. Remove all jewelry before going to the lab to avoid short-circuiting the terminals [19].
5. According to the ECE 445 battery safety manual, in the case that our battery's physical appearance changes, we would contact Casey Smith and TA in order to safely dispose of it.[19]