Healthy-Gamer Final Report

Prepared for:
Professor Nathalia Peixoto

Prepared by:
Saad Abdullah
Moneeb Ahmad
Aayush Aryal
Priyam Das
Ryan Gunawan
Muhammed Jamil Rahman

I. Executive Summary

With the advancement of technology, it has become easier to track an individual's physical and mental health. Although it is encouraged to live a healthy and active lifestyle the recent COVID-19 virus has made it harder for people to go outside and live an active lifestyle. As a result many people have been looking for other pastimes during this pandemic in order to entertain themselves. One of the most common hobbies that has been participated in during this pandemic is playing video games. Although indulging in video games is a great pastime, many of today's gamers are spending a huge chunk of their time gaming in order to entertain themselves. According to the Washington Post it has been estimated that video game usage has increased by 45% when compared to the usage of last year [9]. This increase of video game usage may be detrimental as spending a lot of time on gaming can cause potential health hazards. Carpal Tunnel Syndrome (CTS) is one of the main health issues that can arise from a gamer if they were to play a game while putting their wrist at a non neutral angle for a long period of time. In addition to causing physical health issues, video games can also cause psychological health issues. According to HeartMath Institute, a gamer who plays violent video games experiences an increase in physiological arousal, such as heart rate, blood pressure and body temperature [10]. As a result of a person who plays violent video games will produce more stress hormones such as epinephrine and nor-epinephrine [10].

In order to mitigate the health issues that can arise from gaming we developed a portable health monitoring device called "Healthy-Gamer" that will monitor a gamers psychologically and physically while they are gaming. The device contains different sensors such as a pulse rate sensor, temperature sensor and accelerometer as well as a raspberry pi zero in order to monitor and analyze the health of the user while they are gaming. With the sensors, the health information that the device will track are a user's: stress level, heart rate, body temperature and wrist movement. If the Healthy-Gamer detects that a user is experiencing a health hazard while they are playing (such as their stress level being high) the device will alert the user via email detailing them about the health hazard. The Healthy-Gamer also includes an LCD screen where a user can see their health information in real time. The Healthy-Gamer also includes a feature where the user's health information will be stored on ThinkSpeak which is a cloud based storage system. With this feature users can check their past health information while they were gaming.

II. Table of Contents

III. Approach

I. Project Origin

This project originated from our faculty advisor professor Peixoto. For the project specification, professor Peixoto wanted her students to create a health monitoring device that could monitor and record a patient's stress level, heart rate, body temperature, and body movement. The project also required the user's health data to be uploaded to a cloud-based dashboard for off-site monitoring. With these project requirements, the group decided to take these specifications and apply them by making a portable health monitoring device that was targeted towards those who play video games. We decided to make our health monitoring device targeted towards gamers because many of them can be so caught up in this hobby that they fail to take into consideration how long-term gaming can affect their physical and psychological health. As stated earlier one of the main physical health hazards that can occur from a long gaming session is that a user can experience CTS. CTS is an instance of pain, numbness, and/or tingling that occurs as a result of the nerve being pinched, squeezed, or compressed [8]. Most often, CTS will arise due to a series of repetitive movements performed over a long period of time, for example, a gamer typing on a keyboard or button-mashing a controller. CTS can also occur if a user has a wrist angle at a high angle as it puts compression stress on the median nerve. Typically a user can experience CTS if their wrist angle is observed to be at 35° or higher [8]. In addition to causing physical health issues, video games can also cause a person to feel stressed depending on the type of game that they are playing. This is because their body temperature and heart rate is increased causing stress hormones to be released into their body.

II. Solution to the Problem

With the assistance of the Healthy-Gamer device, an individual can be alerted when their average health readings (such as their wrist angle or heart rate) are high. The Health-Gamer device is designed to read a patient's stress level, heart rate, body temperature, and wrist angle movement every second and then send the average value of 15 readings to the ThinkSpeak cloud storage system. The user's health information is obtained using different sensors as well using a

raspberry pi zero in order to analyze the health data. The sensors that are incorporated into the Health-Gamer are a: accelerometer, pulse rate sensor, and a temperature sensor. The accelerometer is used to detect a user's wrist angle and the pulse rate sensor is used to calculate a user's heart rate (BPM) and stress level. If the average of one or more of the health reading is high the Healthy-Gamer device will send an email notification to the user letting them know which health vital is showing hazardous behavior. For the parameters we set the Healthy-Gamer device will send a notification to a user's email if their: BPM is greater than 100, HRV is greater than 130, body temperature is is greater than 100℉, or wrist angle is greater than 35°. If the user were to receive an email notification of their health reading being high they can take a break from their gaming session until their health vitals return to normal. By taking these periodic breaks a user can ensure that they can manage their physical and psychological health while also enjoying their gaming session.

III. Alternative Design

When designing the Health-Gamer device it was important for the team to have an alternative approach in mind on how to incorporate this project in case our current method of carrying out this project failed. One of the alternative aspects that the team wanted to pursue was replacing the raspberry pi zero with an MSP430 as the microcontroller. The reason for this is because we initially had concerns about finding a viable power system for the raspberry pi zero while also keeping the Heath-Gamer device compact. The MSP430 on the other hand doesn't require as much power in order to operate so finding a compact power system would have been easier. Another design change that was incorporated into the Health-Gamer was increasing the size of its PCB. Initially the components placement on the PCB for the Health-Gamer device were very close to one another in order to minimize the overall dimension of the device. However, this ended up providing issues when it came to soldering as we mistakenly solder different components together. In order to avoid this mistake on the next iteration we ended up spacing the competents farther away from each other which ended up making the PCB slightly larger. Another alternative approach that was implemented into the Health-Gamer device is the placement of the pulse rate sensor. Initially, the pulse rate sensor was supposed to be placed on a user's wrist in order to read their heart rate. However, after further testing we decided to place

the pulse rate sensor on the user's fingertip instead. This is due to the fact that the pulse rate sensor was able to more accurately read a user's heart rate if we are collecting the data from a fingerprint rather than their wrist. Because the Health-Gamer device is designed to read a user's health information as accurately as possible we decided that this design change was ultimately better for the project.

IV. Contribution of Each Team Member

This section will highlight what each member did order to implement the Health-Gamer device.

Saad Abdullah: Worked on implementing the cloud storage capabilities of the Health-Gamer device. Created the schematic layout for the PCB of the Health-Gamer device. Soldered the sensor components to the PCB. Worked on implementing the pulse rate sensor in order to read the user's heart rate.

Moneeb Ahmad: Wrote a script that would alert a user via email if their health reading were high. Created and coded a mathematical equation that would calculate a user's HRV level. Worked on implementing the temperature sensor in order to read s user's body temperature.

Aayush Aryal: Created and coded a mathematical equation that would calculate a user's HRV level. Worked on implementing the accelerometer sensor into the Health-Gamer device in order to read a user's wrist angle.

Priyam Das: Worked on implementing the LCD screen in order to read off a user's health data in real time. Worked on implementing the case of the Healthy-Gamer device.

Ryan Gunawan: Worked on creating the schematic layout for the PCB of the Health-Gamer device. Worked on implementing the case of the Healthy-Gamer device.

Muhammed Jamil Rahman: Worked on creating the schematic layout for the PCB of the Health-Gamer device. Worked on implementing the case of the Healthy-Gamer device.

IV. Technical Section

This section covers all the different function levels of the Healthy-Gamer device as well as a quick summary on each level function. It also covers the PCB design as well as the General and System Architecture, State Diagram and Software Design in detail. Each of the components used in the device is also touched upon.

I. System/Design Architecture

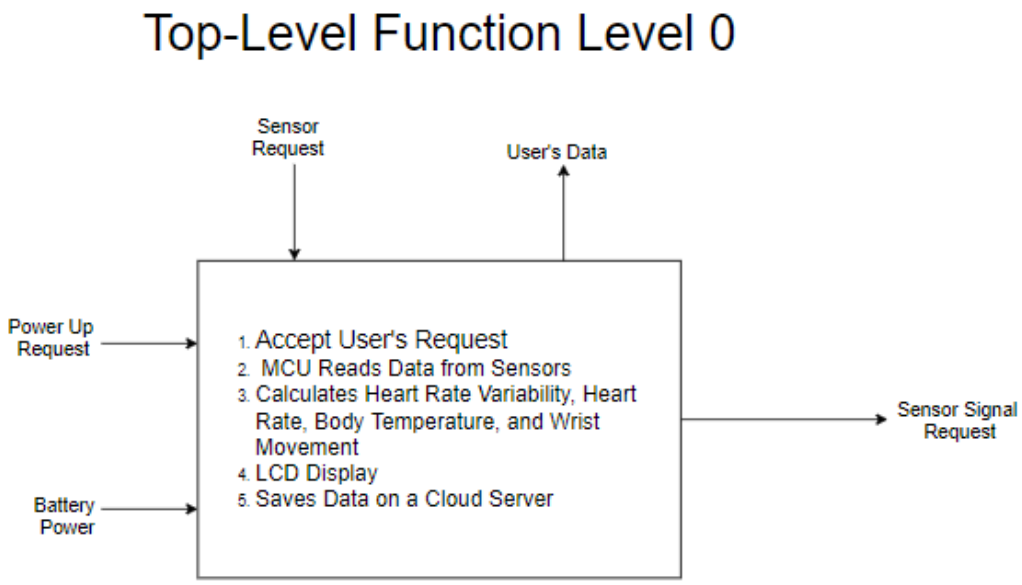


Figure 1: Top Level Function Level 0

This is level 0, a general overview of the system design/architecture of the physiological device. The inputs to the Healthy-Gamer are placed on the left and top side of the block diagram. Above it, we had the user input that was going to interact with the Healthy-Gamer. The input went through the device and was outputted as a sensor signal request which is placed on the right side of the diagram. The main functions of the Healthy-Gamer are as follows: accepts user requests, has data reading capability, calculates Heart rate variability (HRV), calculates heart rate, body temperature using a thermal resistor sensor, and wrist movement using an accelerometer.

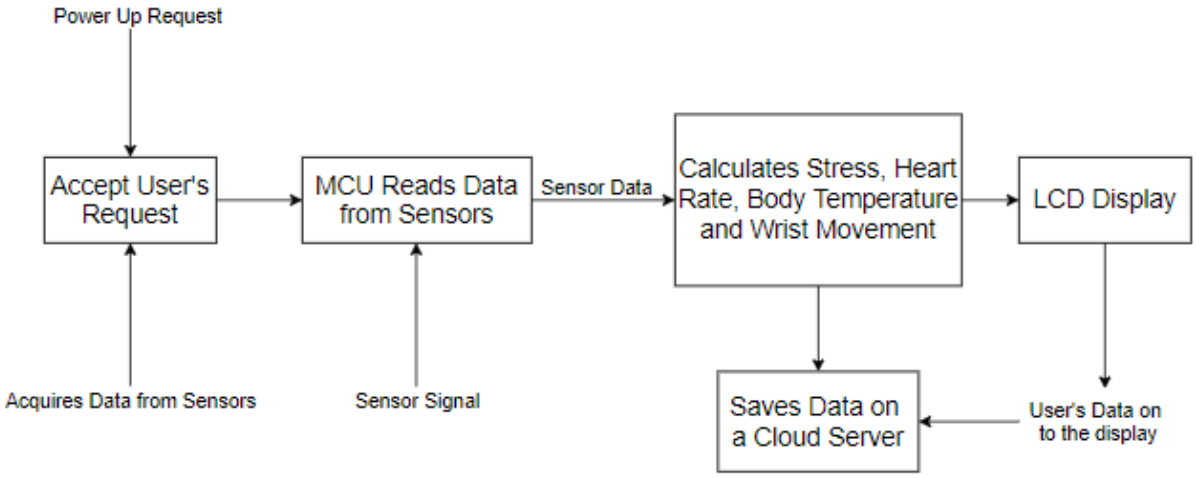


Figure 2: Top Level Function Level 1

A one step down from level 0 functional architecture is level 1 functional architecture. It goes into more detail regarding the top level functions and how they are integrated with each other. The above diagram illustrates how each element communicated with each other and shows the device's operation as it went through its functions. On the top left corner, the device accepted the users' requests. Then, it sends that to the raspberry pi zero which reads data from various sensors. The output from that then goes through the CPU of the device and the various values that were measured based on the inputs. After that, the calculated data goes to the LCD display for the user to see and take all necessary actions based on the values. Along with sending the data to the LCD display, the device also sends the user's health data to a ThinkSpeak cloud storage server where the user can go through his/her health data that has been accumulated over the time.

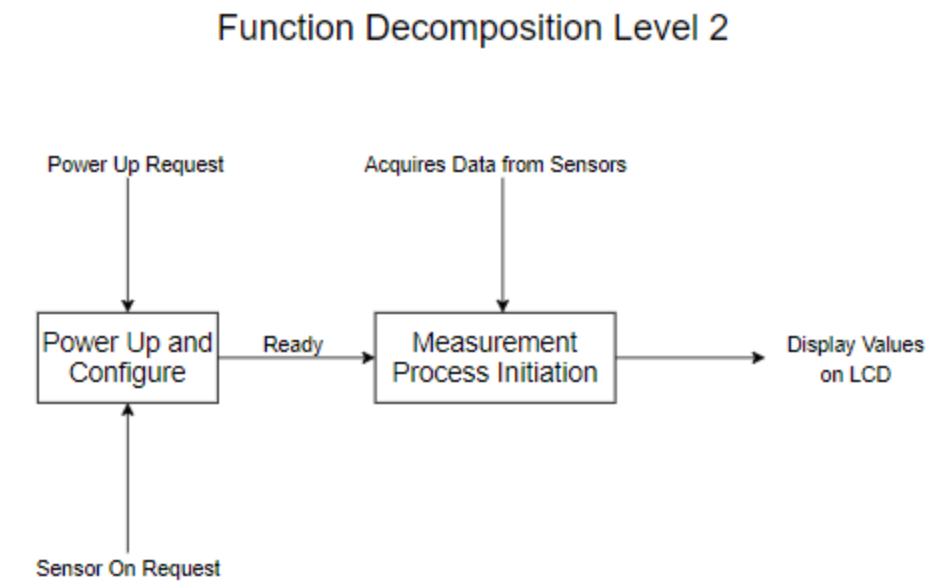


Figure 3: Function Decomposition Level 2

The user had to power on the Healthy-Gamer in order for the device to start reading their vitals. Once the user turns the Healthy-Gamer device on, the sensors begin to read the user's health information.

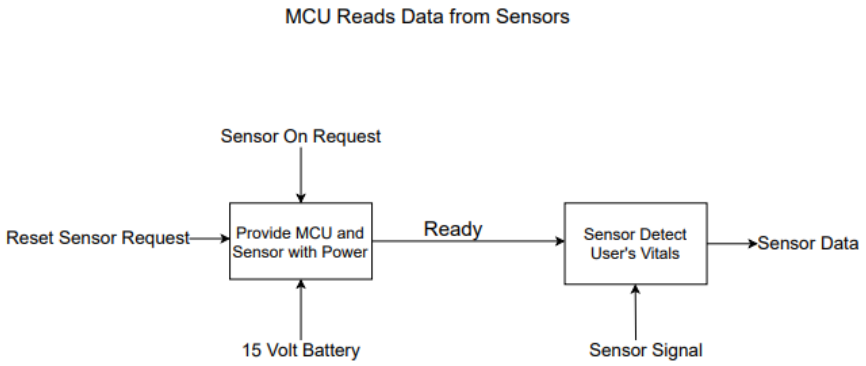


Figure 4: Reading Data from Sensors

The sensor within the Healthy-Gamer sent the user's data over to the MCU. The MCU then begins to process the user's health information.

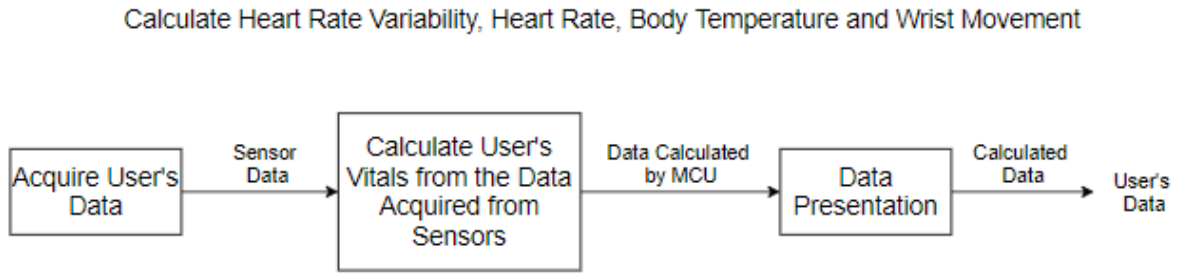


Figure 5: Calculations done by the MCU

The data collected from the user is sent to the Raspberry Pi Zero where data from different sensors such as the Heart Rate-Sensor and Accelerometer calculated the user's vitals. The Raspberry Pi then shows the user's data.

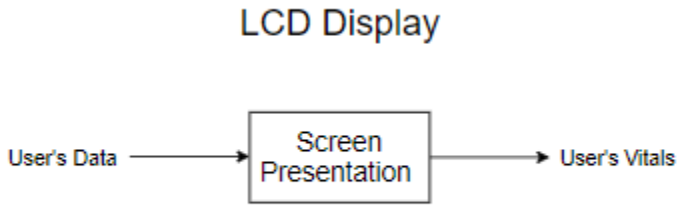


Figure 6: LCD Display

The user's Data is then sent to the LCD Display's screen presentation allowing the user to see the data collected from the sensors..

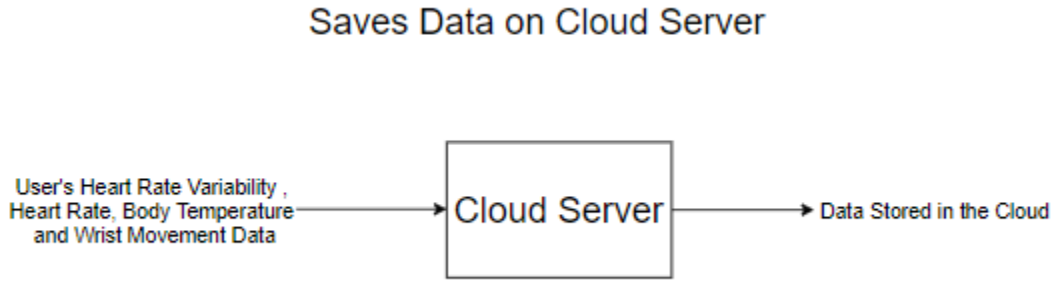


Figure 7: Saving Data on a Cloud Server

Data collected from the sensor is uploaded into the ThinkSpeak Cloud Server to be stored.

II. General Physical Architecture

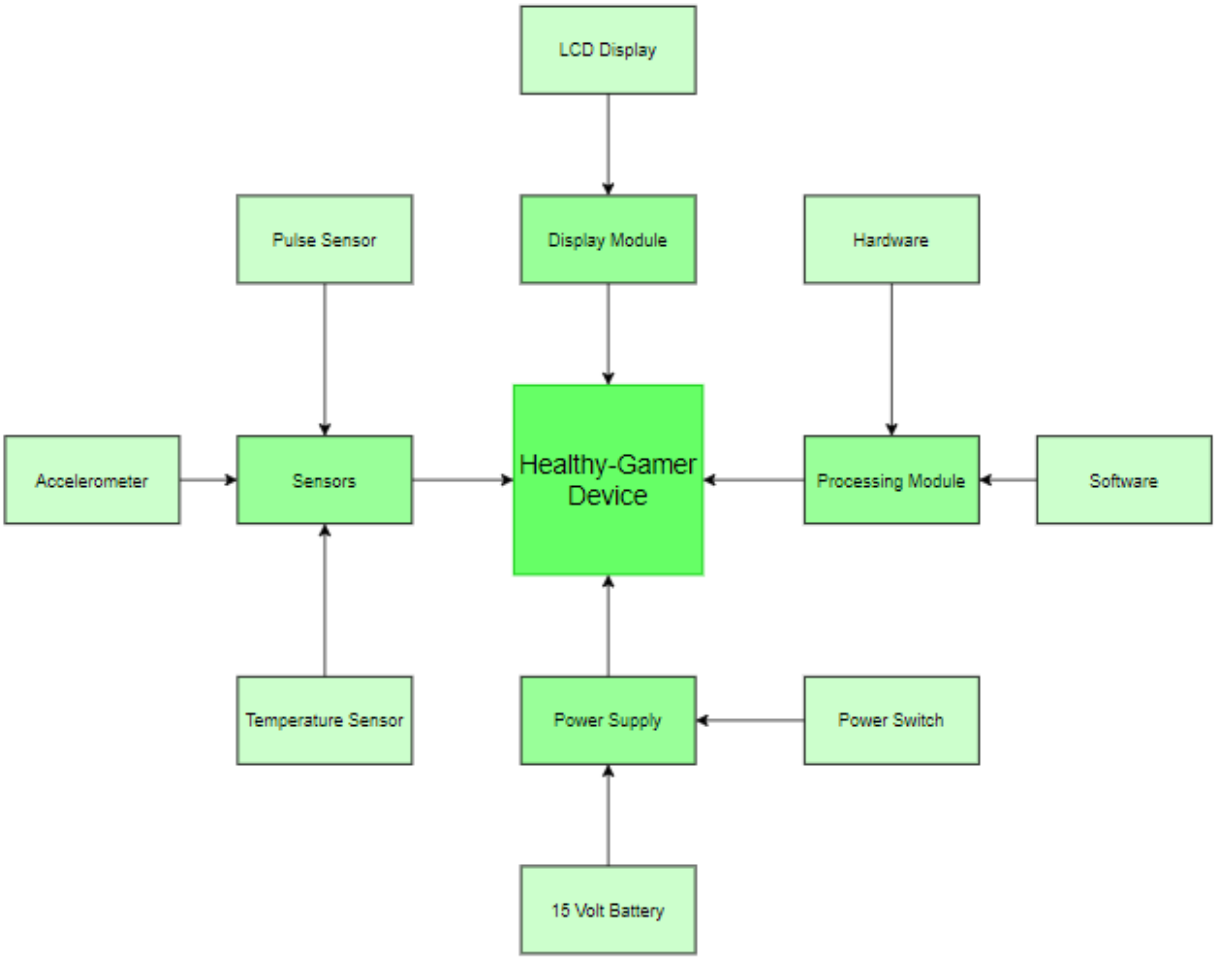


Figure 8: General Physical Architecture

III. System Architecture

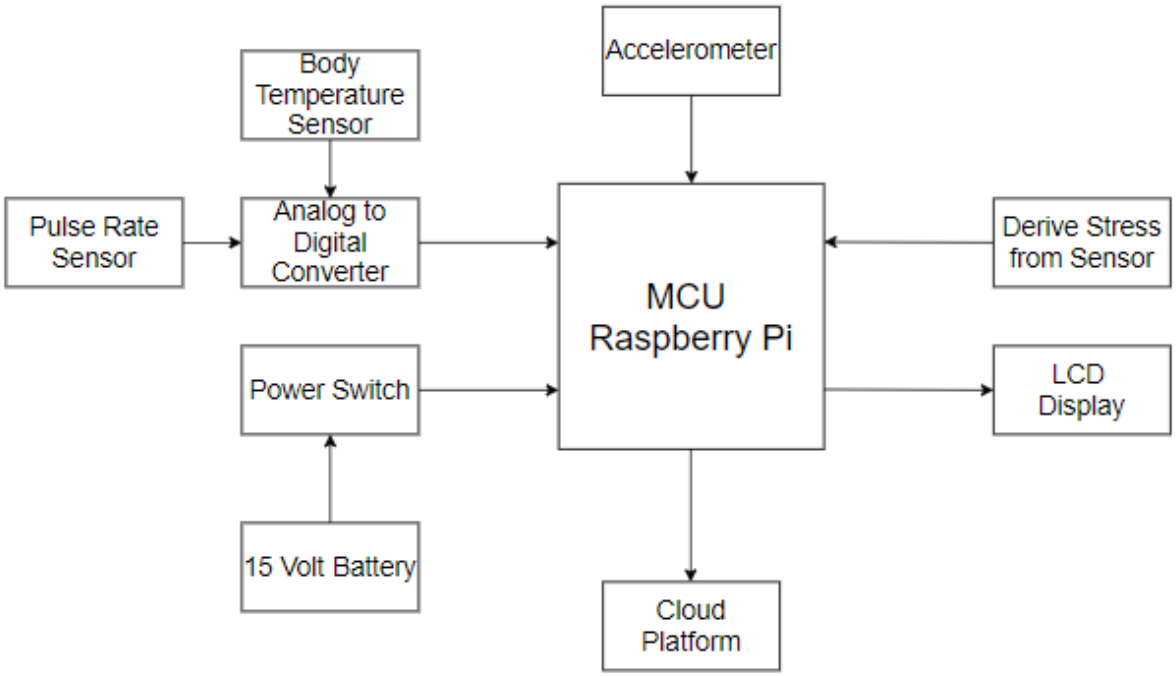


Figure 9: System Architecture

The figure above is a visual representation of all components that are either directly connected or is accessed by the Raspberry Pi. The Raspberry Pi connected to all the sensors through the serial interface and received data from the Analog-to-Digital Converter for both the temperature and Heart Rate sensor. The LCD Display was connected to the Raspberry Pi directly through Serial Peripheral Interface (SPI). The Display allowed the user to view vital information about their health. Also, the data from the Raspberry Pi was then sent to the ThinkSpeak servers for secure storage.

IV. System Model

I. State Diagram

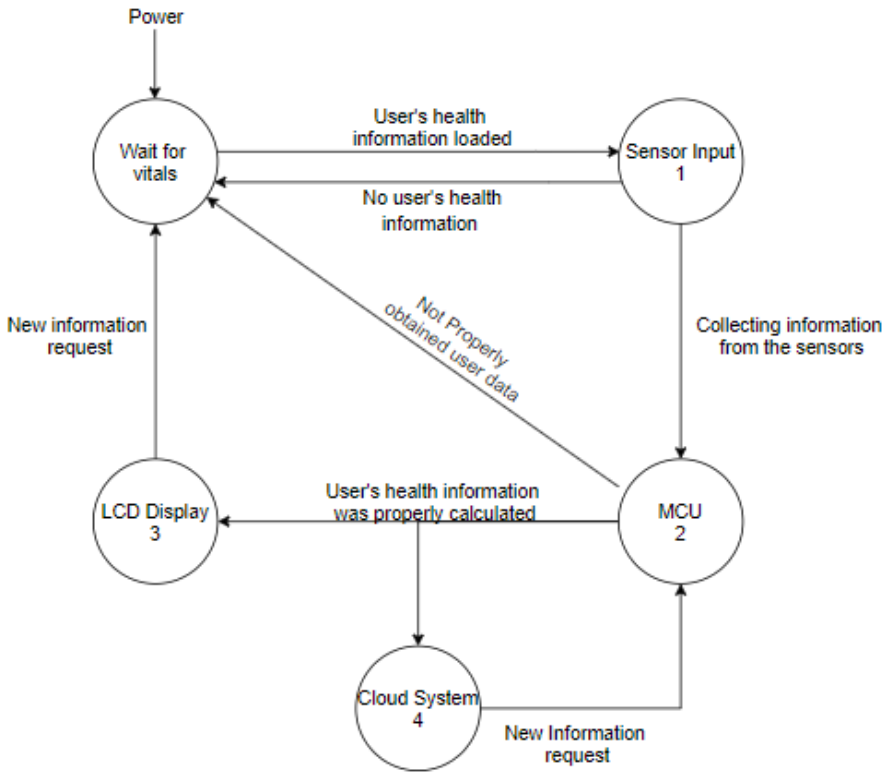


Figure 10: State Diagram

The figure above represents the state diagram for the Healthy-Gamer Device. Once the Healthy-Gamer is powered on, the software requests the sensors to read a user's health information. Once the sensors in the Healthy-Gamer collect the user's vitals it then sends it over to the Raspberry PI where it then begins to convert the data from the sensor into readable information. Once the Raspberry PI finished converting the information received from the sensors, it then sent it to the LCD screen where it displays the user's health information. The Raspberry Pi was also responsible for uploading a user's health data into the cloud. The LCD alerted the user if their health is in a critical state and advised them to take a break from their gaming section.

II. Software Design

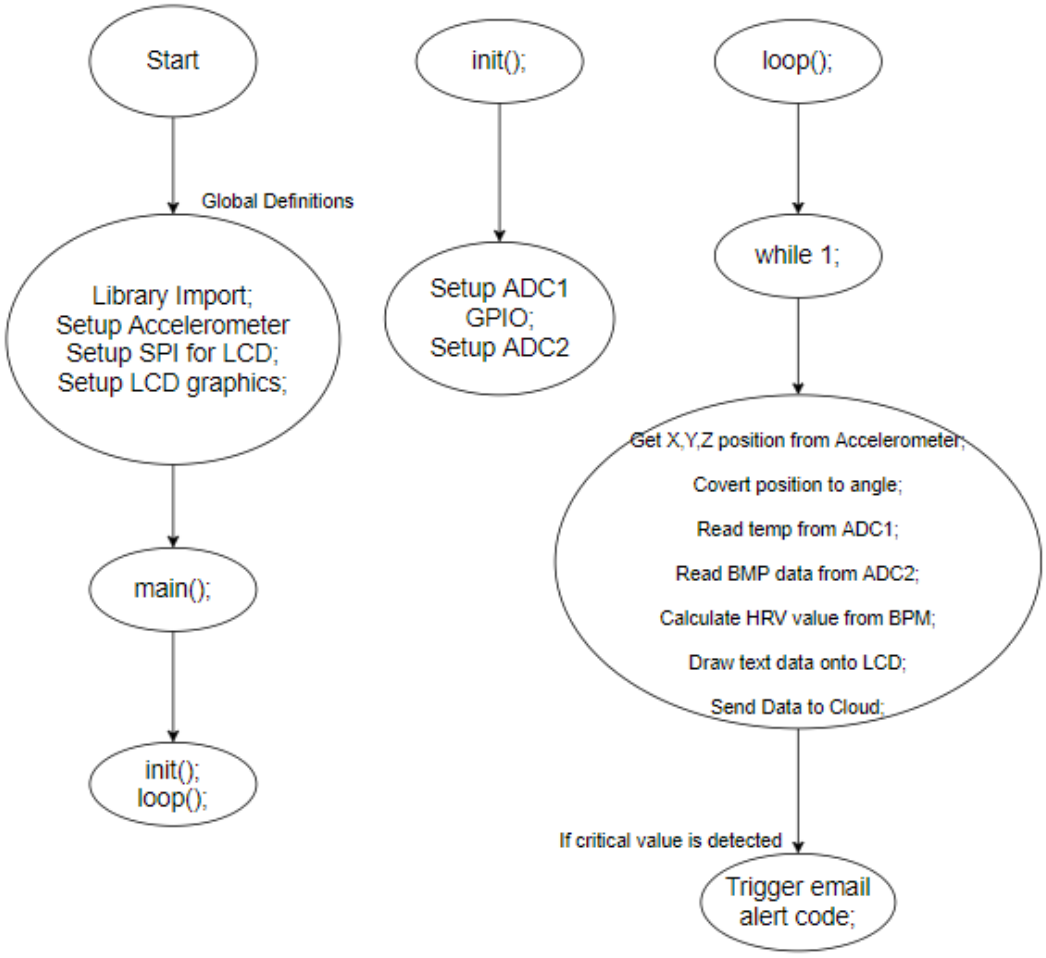


Figure 11: Software Design

The algorithm began by setting up I2C for the accelerometer and SPI for the LCD. The LCD also needed some graphical data to set up ahead time in order for everything to be properly drawn on the display. Then the algorithm setups both ADCs to properly convert the thermistor data into a temperature value and to convert the pulse rate sensor into BPM. After that it enters the main loop where everything was done. First it obtained the x, y and z positions from the accelerometer and then converted it into a wrist angle. Then it read the temperature from one of the ADCs, and got the BPM from the other ADC. With the data from the ADC, the main loop calculated Heart Rate Variability using the BPM. Then this data was sent to the LCD and the Cloud. If any critical values were detected then it would trigger the email alert feature.

III. Elements of the System

1. Raspberry Pi Zero:

   The selected MCU was the raspberry pi zero W. The pi zero is equipped with an ARM11 CPU running at 1 GHZ. A fast CPU was very helpful as the accelerometer and the HRV required intensive mathematical calculations to derive their associated values. The pi zero also was equipped with a built-in wifi module which was critical for uploading user data to the cloud and sending email alerts. The pi also had an I2C bus which the accelerometer required. The Pi Zero also had an SPI bus which was essential for the pulse sensor and the LCD to operate.

2. Reflection-Type Pulse Sensor:

   Reflection-type pulse sensors were used to obtain a user's BPM as well as their HRV values. The pulse sensor emitted an infrared light onto a user fingertip and measured the amount of light reflected off it using a phototransistor. Oxygenated hemoglobin present in the blood of the arteries has the characteristic of absorbing incident light. By sensing the blood flow rate that changes following heart contractions over time, the sensor was able to obtain a pulse wave signal which showed the users heart rate. The sensor output was an analog signal so the pulse sensor needed to be attached to a MCP3008 ADC chip.

3. Accelerometer:

   The accelerometer consists of different parts with piezoelectric effect and the capacitance sensor being the most important. The accelerometer is able to create voltage from stress which is then interpreted as voltage to determine velocity and orientation. The accelerometer would measure the wrist's x, y, and z positions. These x, y and z positions were used to calculate a user's wrist angle. The I2C protocol aided in acquiring the user's velocity and orientation.

4. Thermal Resistor:

   The thermal resistor was used to calculate the user's internal body temperature. When a voltage divider was set up with the thermal resistor, the output voltage varied with a

user's temperature. This output voltage varied because a thermistor's resistance varied with temperature. Since the thermal resistor outputs an analog signal, an AD0832 ADC chip was required to send the data over to the Pi Zero. The ADC was an 8-bit channel so it could read from 0 degrees to 255 degrees which is plenty for human body temperature.

5. LCD display:

   We had integrated an LCD screen for the Healthy-Gamer device to display a user's health data while they were gaming. The LCD screen would display the user's: BPM, stress level, wrist angle, and their body temperature.

6. Power supply:

   We used a Pisugar portable 1200 mAh Lithium battery to power up our device. This battery was able to power the device simultaneously for 5 hours. This battery used a small test-pad to supply power, therefore it did not occupy any GPIO of Raspberry Pi. It could be used in the Raspberry Pi Zero without the use of any GPIO header. The battery was placed under the PCB board to avoid any possible interference.

7. ADC0832:

   We used an 8-Bit, two channel serial I/O, A/D converter for the thermal resistor. It had 0V to 5V input range with a single 5V power supply and had a shunt regulator that allowed operation with high voltage supplies. This turned the analog signal from the thermistor into a digital signal for the Pi Zero.

8. MCP3008:

   The MCP3008 is an SPI ADC chip that is capable of converting a pulse wave into a digital signal. The SPI bus was needed to convert an analog pulse wave into a digital signal therefore the MCP required input from the SPI bus, and it's output was also connected to the SPI bus.

9. ThinkSpeak

One of the features that is implemented with the Healthy-Gamer device is that it has the capability of storing a user's health information into a cloud storage system. The cloud system that the data is being stored in is ThinkSpeak. The Healthy-Gamer device will send the average of 15 health readings of the user to ThinkSpeak.

10. Email Notification

The Healthy-Gamer was designed to send an email notification to a user if their health reading were showing signs of hazardous behavior. An email notification would be sent to a user if their: BPM is greater than 100, HRV is greater than 130, body temperature is is greater than 100℉, or wrist angle is greater than 35°.

IV. PCB Design

I. Circuit Schematic Level

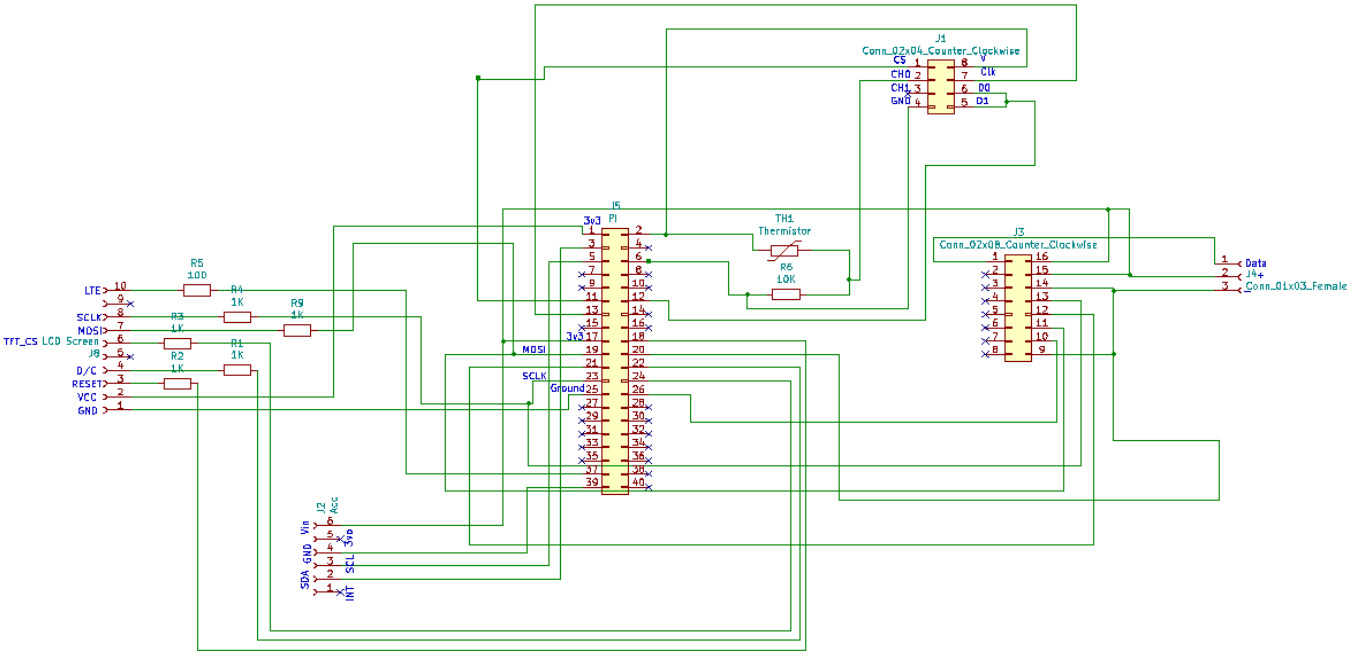


Figure 12: Circuit Schematic

The accelerometer is connected to the Raspberry Pi's SDA and SCL ports. Pin 3 is the SDA port and Pin 5 is the SCL port. As seen in the schematic above, the SDA pin goes to pin 2 of the accelerometer and SCL port goes into pin 3 of the accelerometer. This configuration allows for the I2C protocol to be properly established as well as synchronous communication between the pi and the accelerometer. The thermal resistor is connected to an ADC 0832 and the data from the ADC is read from the pi. The thermal resistor value is interpreted by the ADC chip which is then transferred to Pin 12 of the pi. The Pulse sensor is connected to an ADC MCP 3008 which allows the pulse sensor to communicate to the SPI bus on the pi. Pin 21 of the pi is connected to Pin 12 of MCP3008. Pin 21 is the SPI MISO of the pi. The LCD SCLK connected to the SPI clock which is Pin 23. Pin 1 and 2 of the LCD are the ground and power source. Pin 3 on the LCD is reset and it is in series with a 1k ohms resistor value. For the DC (pin4), we used the 1k ohms resistor for regulating the voltage. Pin 6, 7, and 10 of the LCD regulates the output. Pin 8 is the clock of the LCD. Clock enable pin is the refresh rate that we could set. When the output changed, the LCD display was updated and the update rate was the clock cycle.

II. PCB Design

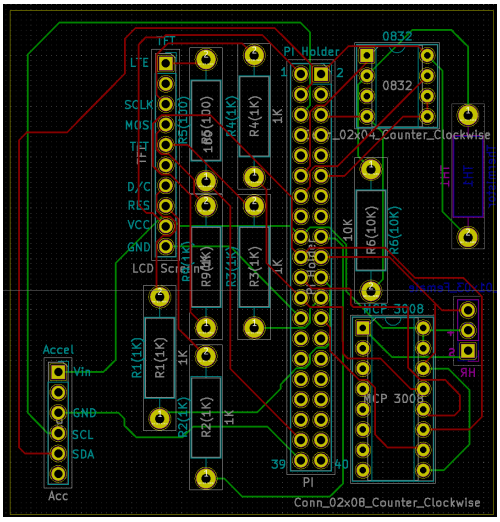


Figure 13: PCB Design Schematic

The picture above details the placement of each component on the PCB design. The top left side of the PCB will contain the LCD screen while the bottom left side of the PCB will contain the

accelerometer. The left side of the PCB also contains the six resistors that are connected to the different pinouts of the LCD. The right side of the PCB implements the thermal resistor and pulse rate sensor of the Healthy-Gamer device. The right side also contains the two ADC chips that are associated with each sensor as well as a resistor that is associated with the thermal resistor. The green wiring indicates the top wiring of the PCB while the red wiring indicates the bottom wiring of the PCB. It was important to have both a top and bottom wiring layer for this PCB as it simplified wiring the entire PCB. If the PCB were to be wired on a single surface we would have to find different methods of wiring each component while making sure none of them intersect with one another. For the PCB design we also decided to flip the thermal resistor and pulse rate sensor so they are facing the back of the PCB. We decided to do this because the sensors will be pushing out of the Health-Gamer box and onto a user's wrist and fingertip.

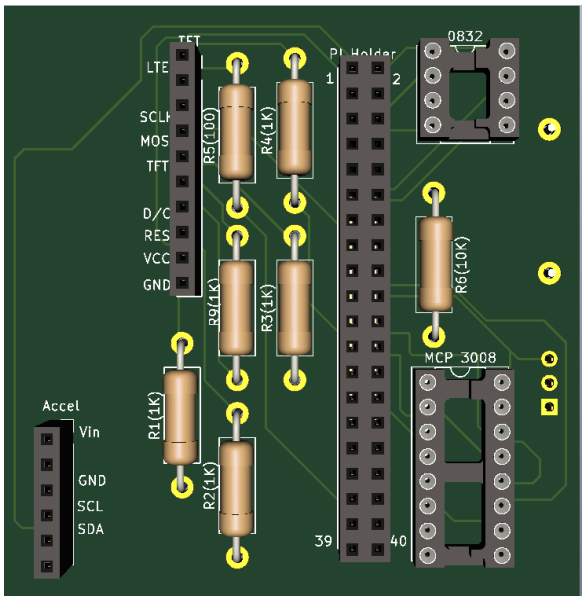


Figure 14: 3D Design of PCB (Front View)

The picture above showcases the 3D of the front side of the PCB. The visible wiring is a result of the green wiring shown in figure 11.

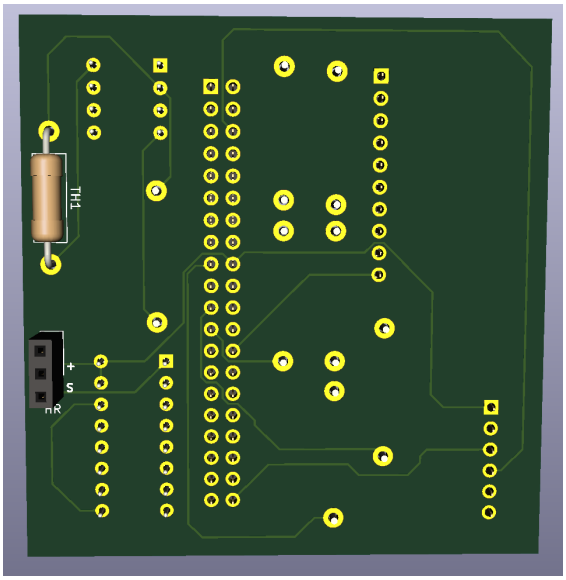


Figure 15: 3D Design of PCB (Back View)

The picture above showcases the 3D of the back side of the PCB. The visible wiring is a result of the red wiring shown in figure 11. The top component on the board incorporates the thermal resistor while the bottom component incorporates the pulse sensor.

III. Pictures of Device

Figure 16: Healthy-Gamer Device Without its Cover.

The picture above shows the Health-Gamer device's without its cover on. In the picture the front side of the PCB is shown as well as the components that are integrated with it. As stated earlier, the LCD screen and the accelerometer is on the left side while the thermal resistor and the pulse sensor are on the right side.

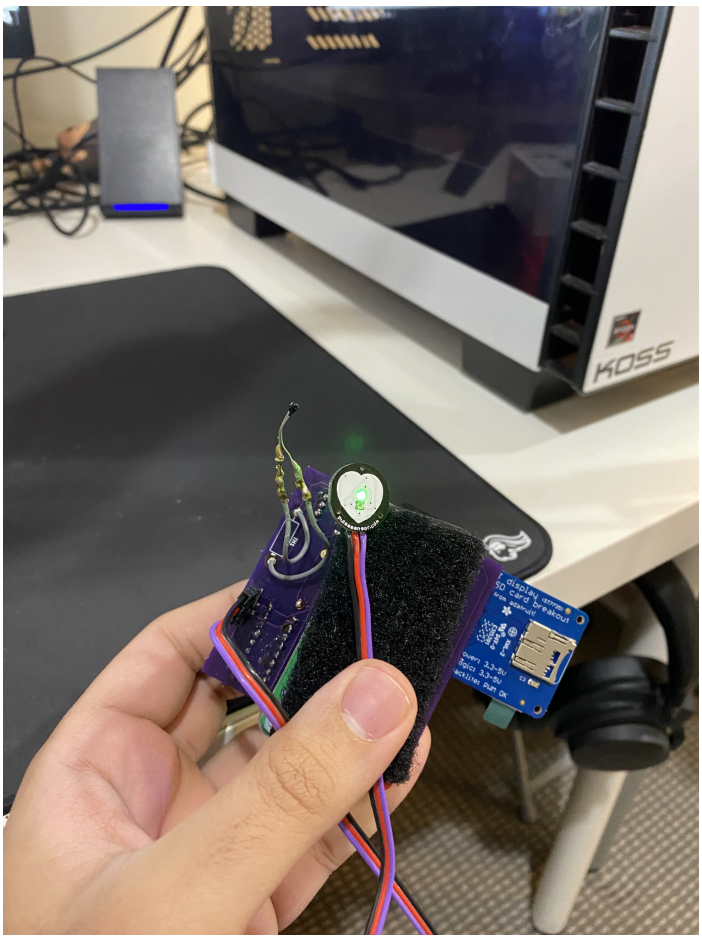


Figure 17: Back of the Healthy-Gamer Device Without its Cover.

The picture above shows the back side of the Healthy-Gamer device. As seen the pulse sensor and the thermal resistor are soldered on the back of the device.

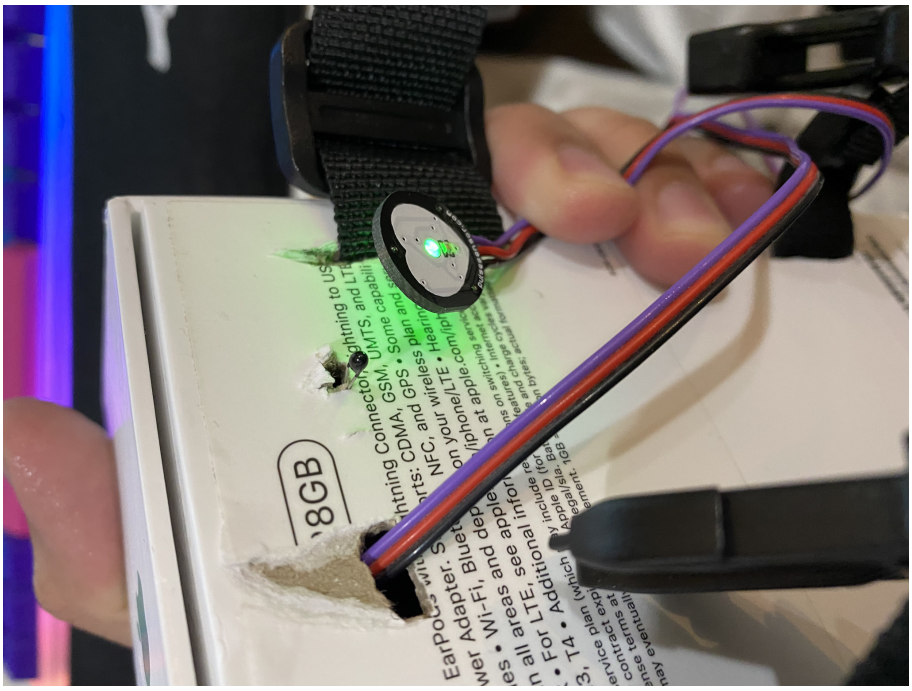


Figure 18: Back of the Healthy-Gamer Device Without its Cover

The picture above shows the thermal resistor and pulse sensor sticking out of the Healthy-Gamer box. These sensors need to stick out in order to read a user's health readings.

Figure 19: Picture of the Health-Gamer Device with its case on

The picture above shows the Health-Gamer device fully integrated with its cover on. The cover has a cutout in order to show the LCD screen. As seen in the picture the user's: BPM, wrist angle, stress level, and body temperature are displayed on the LCD screen. The pulse sensor is also attached to the user's finger tip and is attacked with a velcro piece.

V. Experimentation

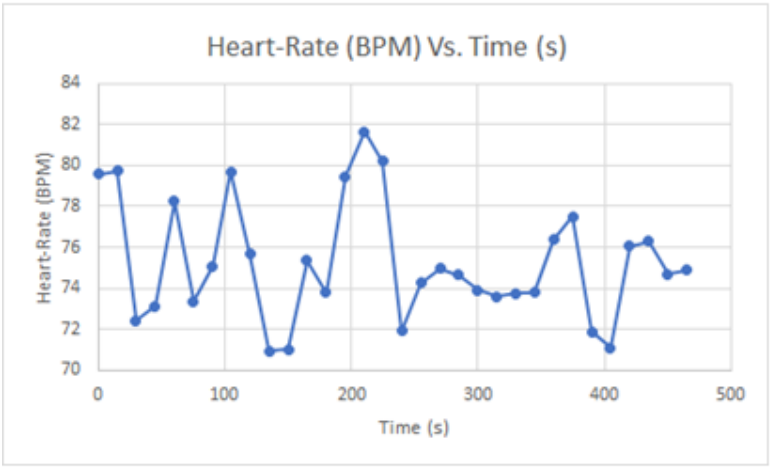
In this section we will be discussing the different experiments that we conducted in order to test the validation and accuracy of the Healthy-Gamer device. The experiment section will show that each sensor works as well showcasing that the cloud feature and notification feature both working as intended.

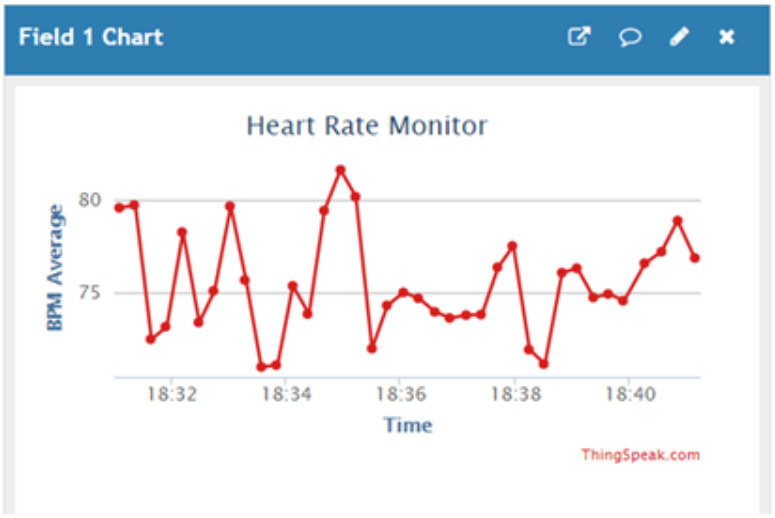
Experimentation:

Sensor data over 8 minutes while sitting [Experiment 1]:

We have both the thingspeak graphs and excel graphs as excel graphs are easier to read, but they are the same exact sets of data. Here we had the healthy-gamer device strapped to our arms and let it take sensor data while we were just sitting. This experiment was to just verify that our system was working as one complete integration. This is not a part of any test case. All the graph points are 15 moving averages. The data can be seen below.

Heart-Rate while sitting:



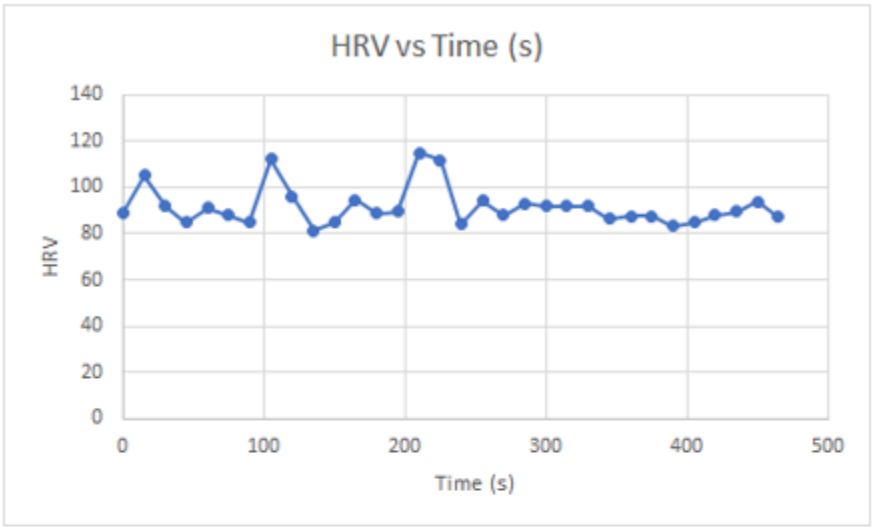


Heart-Rate Data Points

Time (s)	Heart-Rate (BPM)
0	79.58862422
15	79.7397517
30	72.44191913
45	73.12710946
60	78.26615027
75	73.35511311
90	75.07023297
105	79.67205186
120	75.65696528
135	70.9333985
150	71.03661065
165	75.34548773
180	73.81394063
195	79.4377324
210	81.65254143
225	80.19155642
240	71.93292296
255	74.27896645
270	74.98497965
285	74.67022893
300	73.93054186
315	73.59651174
330	73.75233956
345	73.78435302
360	76.35580281
375	77.51898488
390	71.87749611
405	71.09118808
420	76.06004412
435	76.30107182
450	74.71129491
465	74.90234644

Figure 20: Heart Rate While Sitting

HRV data while sitting:

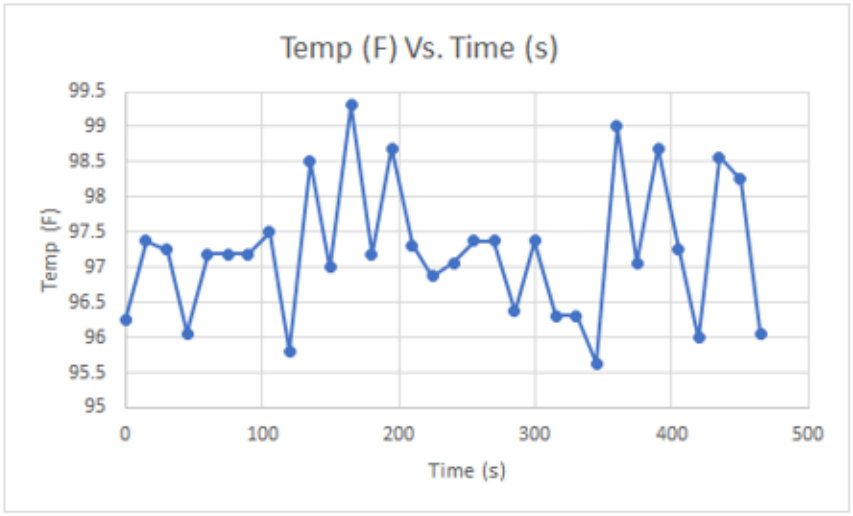


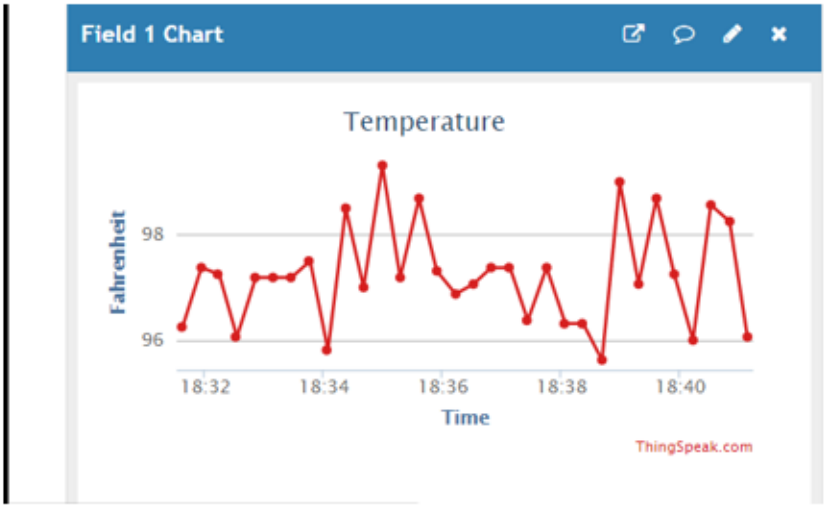
HRV data points

Time (s)	HRV
0	88.787
15	104.9306
30	91.76079
45	84.88482
60	90.91426
75	87.93025
90	84.85838
105	112.2499
120	96.04476
135	81.31924
150	84.69643
165	94.42436
180	88.7514
195	89.58774
210	114.7773
225	111.9291
240	83.98534
255	94.16948
270	87.79375
285	92.8586
300	91.87665
315	91.64409
330	91.84467
345	86.49378
360	87.69873
375	87.42036
390	83.13074
405	84.68253
420	88.10121
435	89.36811
450	93.66251
465	87.00172

Figure 21: Heart Rate while Standing

Internal Body Temperature while sitting:



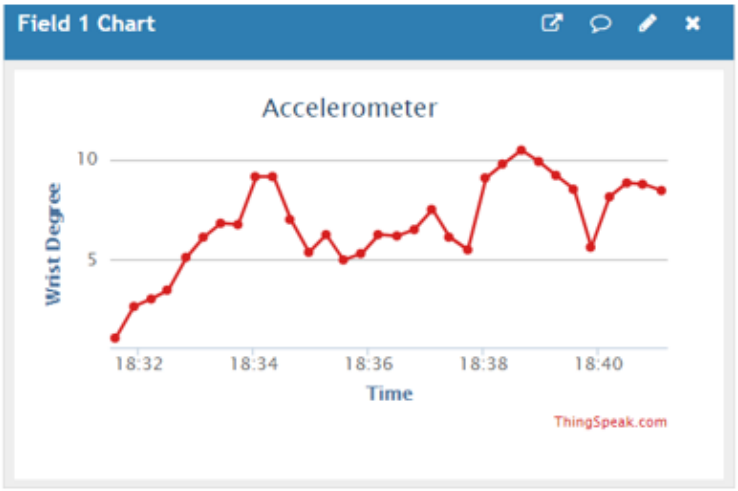


Temp data points

Time (s)	Temp (F)
0	96.25
15	97.375
30	97.25
45	96.0625
60	97.1875
75	97.1875
90	97.1875
105	97.5
120	95.8125
135	98.5
150	97
165	99.3125
180	97.1875
195	98.6875
210	97.3125
225	96.875
240	97.0625
255	97.375
270	97.375
285	96.375
300	97.375
315	96.3125
330	96.3125
345	95.625
360	99
375	97.0625
390	98.6875
405	97.25
420	96
435	98.5625
450	98.25
465	96.0625

Figure 22: Internal Body Temperature while Sitting

Wrist Angle while sitting:



Wrist Angle Data points

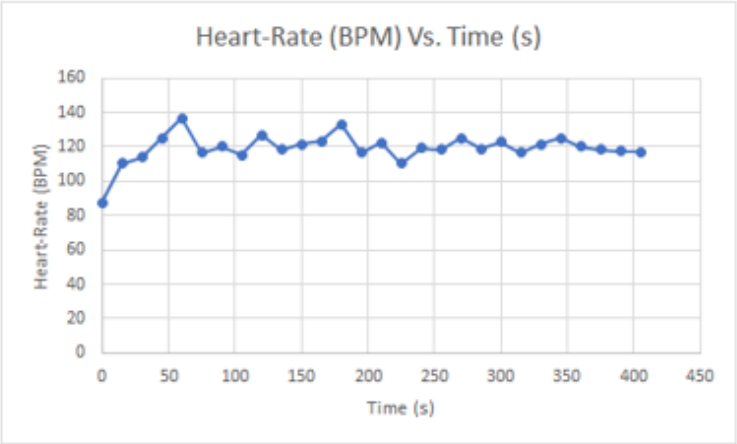
Time (s)	Wrist Angle (deg)
0	1.125
15	2.6875
30	3.0625
45	3.5
60	5.125
75	6.125
90	6.8125
105	6.75
120	9.125
135	9.125
150	7
165	5.375
180	6.25
195	5
210	5.3125
225	6.25
240	6.1875
255	6.5
270	7.5
285	6.125
300	5.5
315	9.0625
330	9.75
345	10.4375
360	9.875
375	9.1875
390	8.5
405	5.625
420	8.125
435	8.8125
450	8.75
465	8.4375

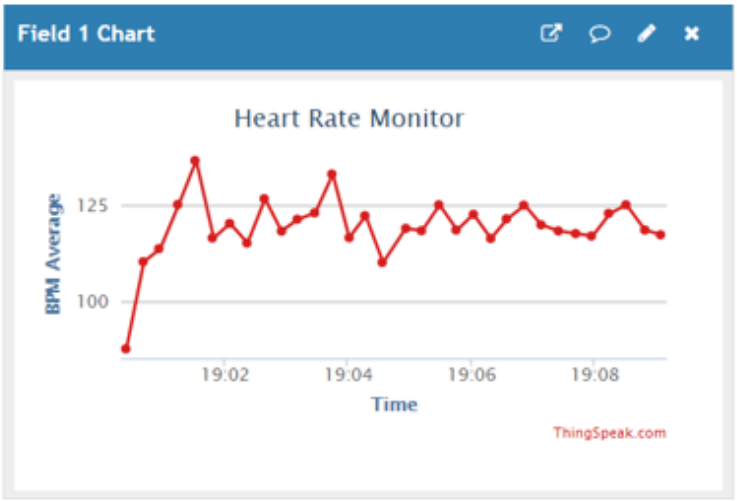
Figure 23: Internal Body Temperature while Standing

Sensor Data over 7 Minutes while Exercising [Experiment 2]:

We have both the thingspeak graphs and excel graphs as excel graphs are easier to read, but they are the same exact sets of data. Here we had the healthy-gamer device strapped to our arms and let it take sensor data while we were exercising. This experiment was to make sure our sensors worked over an extreme set of values. This is not a part of any test case. All the graph points are 15 moving averages. The data can be seen below.

Heart-Rate while Exercising:



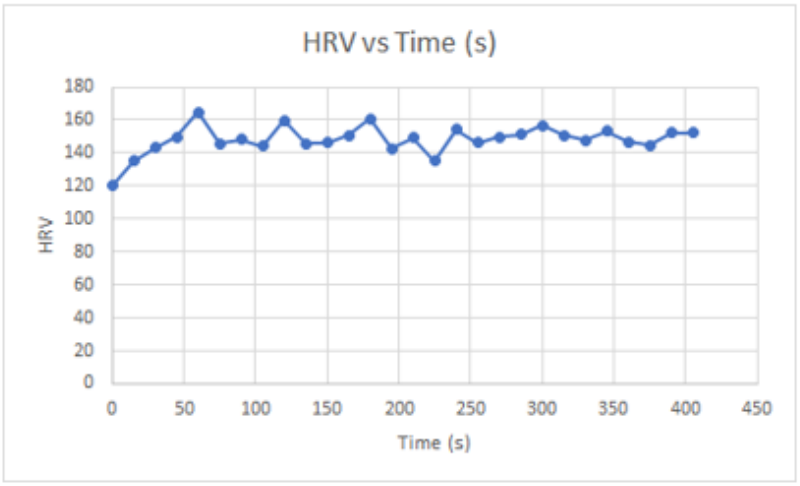


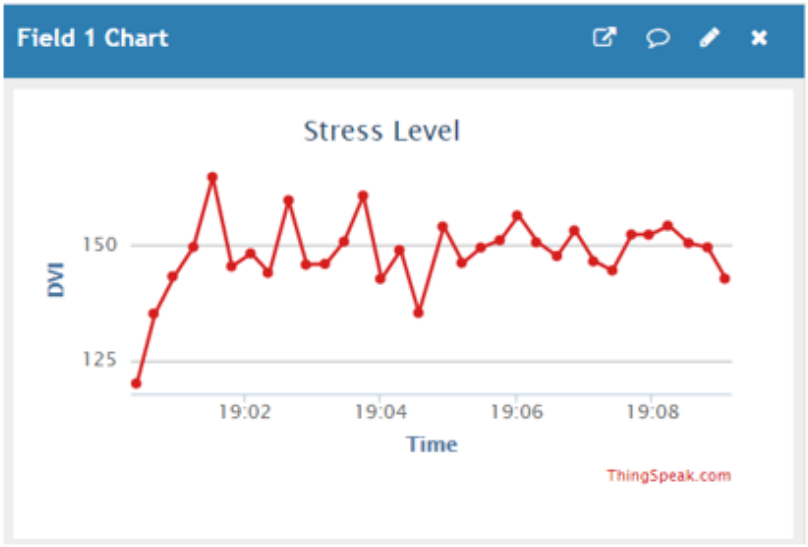
Heart-Rate Data Points

Time (s)	Heart-Rate (BPM)
0	87.69438341
15	110.3440864
30	113.7330289
45	125.2146852
60	136.65017
75	116.5201504
90	120.3019978
105	115.2048596
120	126.7246237
135	118.297251
150	121.3762697
165	123.0593512
180	133.1325288
195	116.6073794
210	122.2956831
225	110.1556703
240	119.012651
255	118.4153429
270	125.1255461
285	118.5896785
300	122.6676403
315	116.3978323
330	121.4986425
345	125.0276916
360	119.9487717
375	118.3562682
390	117.6637504
405	117.0409568

Figure 24: Heart Rate while Exercising

HRV while exercising:



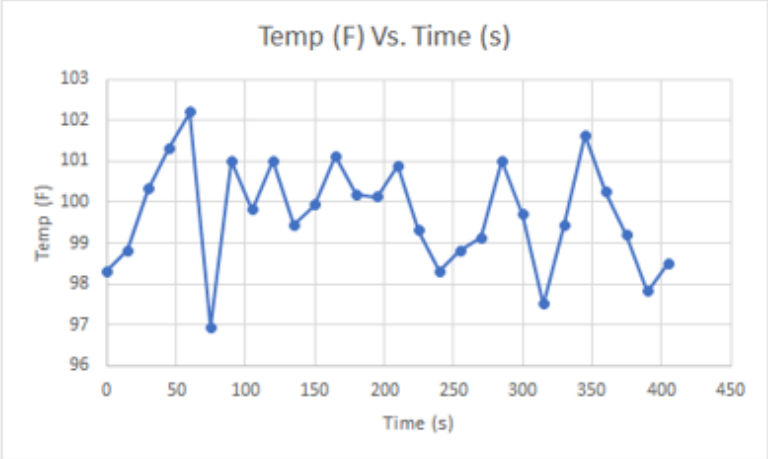


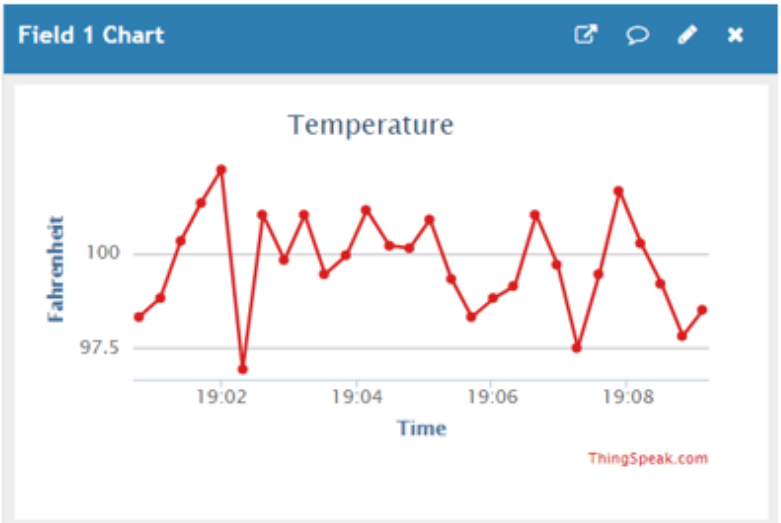
HRV Data Points:

Time (s)	HRV
0	119.9562
15	135.1468
30	143.257
45	149.6528
60	164.8644
75	145.4391
90	148.2462
105	144.0403
120	159.8102
135	145.8176
150	145.9836
165	150.8256
180	160.8184
195	142.7209
210	148.9718
225	135.3241
240	154.094
255	146.2091
270	149.522
285	151.1376
300	156.5051
315	150.6695
330	147.6852
345	153.2361
360	146.536
375	144.5522
390	152.3674
405	152.3317

Figure 25: HRV while Exercising

Internal Body Temperature while Exercising:



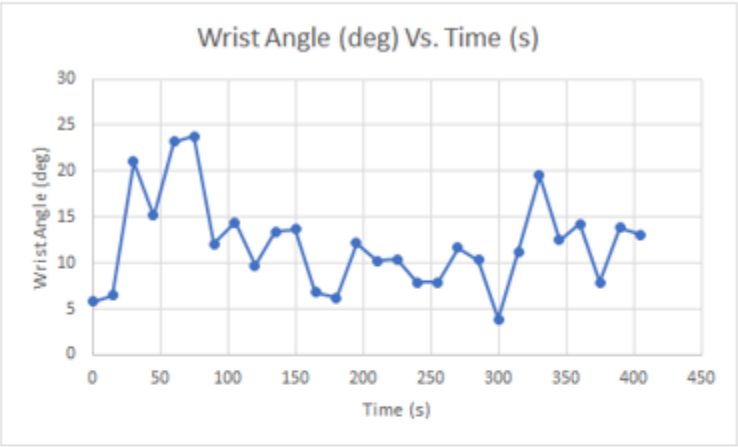


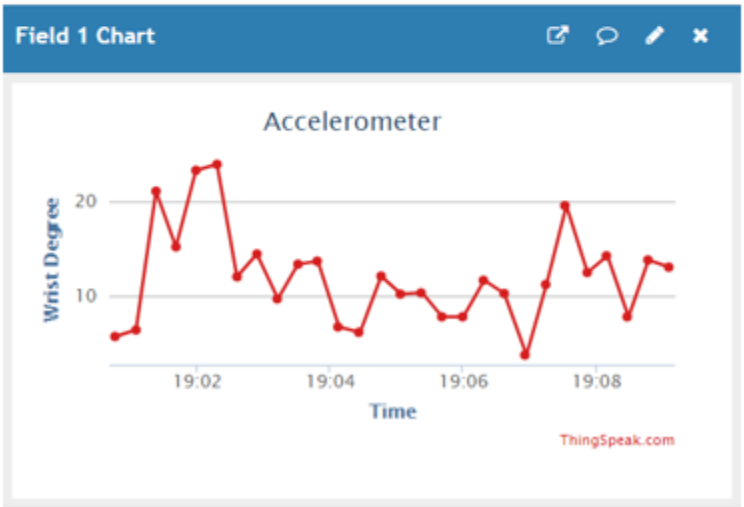
Temp Data Points:

Time (s)	Temp (F)
0	98.3125
15	98.8125
30	100.3125
45	101.3125
60	102.1875
75	96.9375
90	101
105	99.8125
120	101
135	99.4375
150	99.9375
165	101.125
180	100.1875
195	100.125
210	100.875
225	99.3125
240	98.3125
255	98.8125
270	99.125
285	101
300	99.6875
315	97.5
330	99.4375
345	101.625
360	100.25
375	99.1875
390	97.8125
405	98.5

Figure 26: Internal Body Temperature while Exercising

Wrist Angle while exercising:





Wrist Angle Data Points

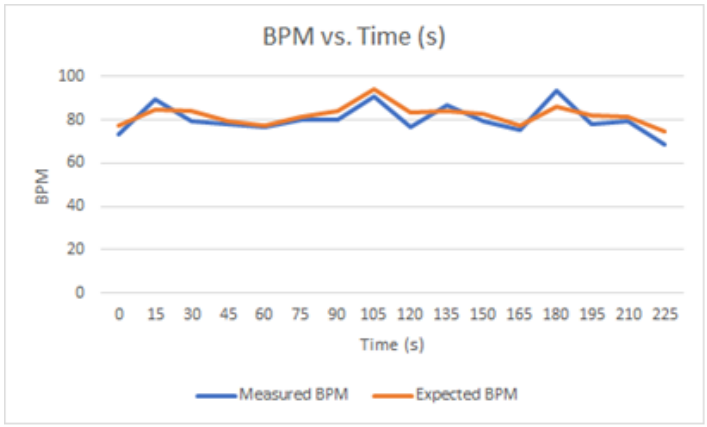
Time (s)	Wrist Angle (deg)
0	5.8125
15	6.5
30	21
45	15.1875
60	23.1875
75	23.8125
90	12.0625
105	14.4375
120	9.75
135	13.375
150	13.6875
165	6.8125
180	6.25
195	12.125
210	10.25
225	10.375
240	7.875
255	7.875
270	11.6875
285	10.3125
300	3.875
315	11.25
330	19.5
345	12.5
360	14.25
375	7.875
390	13.8125
405	13.0625

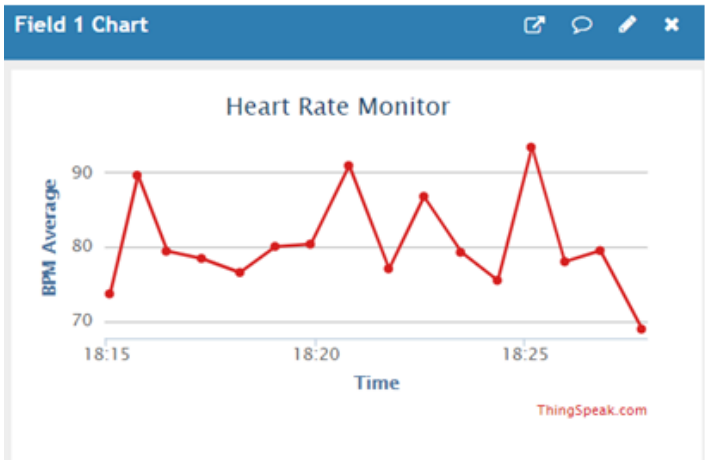
Figure 27: Wrist Angle while Exercising

Sensor Data Over 4 Minutes while Gaming [Experiment 3 [Test case 1]]:

We have both the thingspeak graphs and excel graphs as excel graphs are easier to read. Thingspeak data only shows Measured Data, Expected Data is what we recorded from external devices. Here we wanted to make sure our device worked with the intended use case of gaming. We cross referenced other sensor data to ensure our data wasn't noise and was proper. For assessing data it is important to note that HRV data needs to be examined by its average value over an extended time period to give an assessment of stress. Single data points are not enough as HRV requires multiple data points to give out a single reading for both our device and an apple watch. This is a part of our Test Case 1. All the graph points are 15 moving averages. We only have one picture for Experiment 3 and 4 as there was a part of one test case. The data can be seen below.

Heart Rate While Gaming:

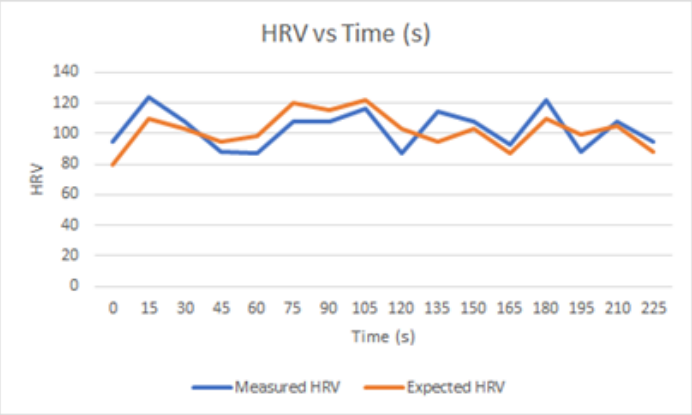




Time (s)	Measured BPM	Expected BPM	% Error
0	73.61825926	77.22	4.664259
15	89.59495475	84.85	5.592168
30	79.3634334	84.08	5.609618
45	78.38812473	79.38	1.249528
60	76.49158593	77.07	0.750505
75	79.99977726	81.61	1.97307
90	80.32106572	83.87	4.23147
105	90.89598336	93.97	3.271274
120	77.00205339	83.5	7.781972
135	86.74844017	84.05	3.210518
150	79.21561481	82.47	3.946144
165	75.43309697	77.35	2.47822
180	93.37767385	86.35	8.138592
195	77.95825277	82.39	5.378987
210	79.44770285	81.44	2.446337
225	68.86584074	74.44	7.488124
Min	68.86584074	74.44	0.750505
Max	93.37767385	93.97	8.138592
Avg	80.42011625	82.1275	4.263174

Figure 28: Heart Rate while Gaming

HRV While Gaming:

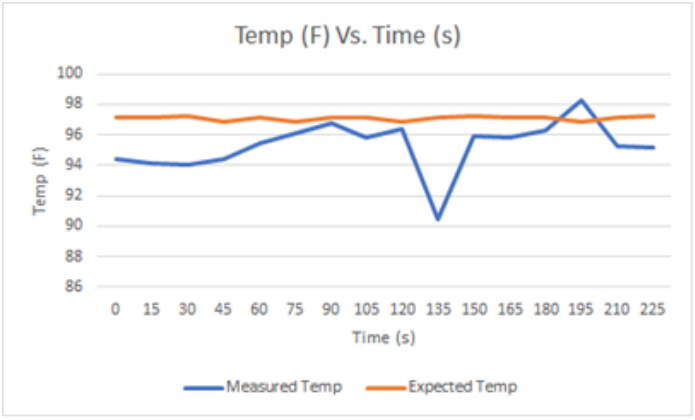


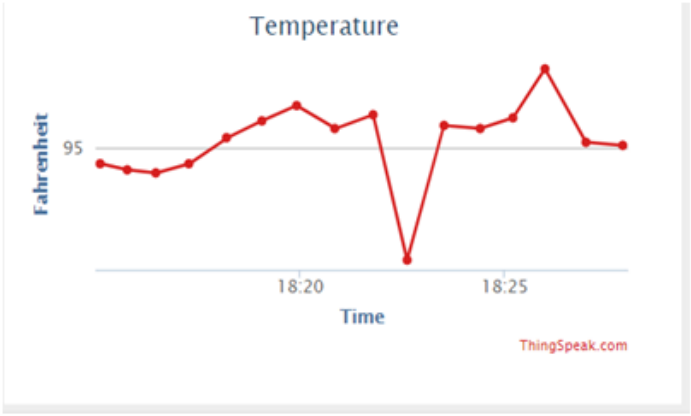
Time (s)	Measured HRV	Expected HRV	% Error
0	94.43435122	80	18.04294
15	124.0361369	110	12.76012
30	107.3611258	103	4.234103
45	88.35243414	95	6.997438
60	87.49158593	98	10.72287
75	107.8397899	120	10.13351
90	107.5101299	115	6.51293
105	115.8350023	122	5.053277
120	87.00205339	103	15.53199
135	114.2207096	95	20.23233
150	107.4616402	103	4.33169
165	92.81992351	87	6.689567
180	122.1710149	110	11.06456
195	87.72344161	99	11.39046
210	107.3294974	105	2.218569
225	94.80076818	88	7.728146
Min	87.00205339	80	2.218569
Max	124.0361369	122	20.23233
Avg	102.8993503	102.0625	9.602781

Figure 29: HRV while Gaming

Internal-Body Temperature While Gaming:

The dip to 90 degrees was a result of the temperature sensor coming out of its fixture, causing the temperature to fall to 0 for a few of the 16 readings

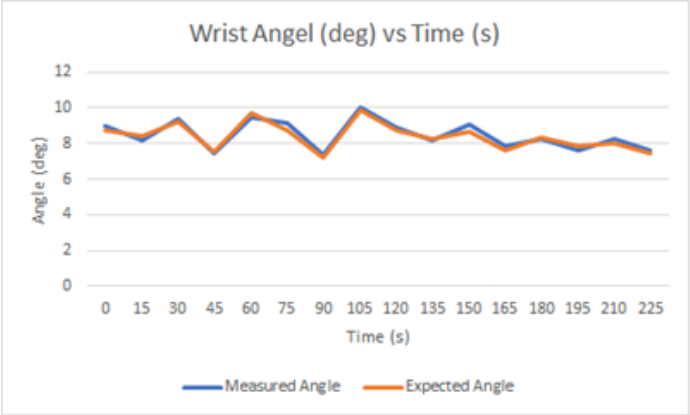


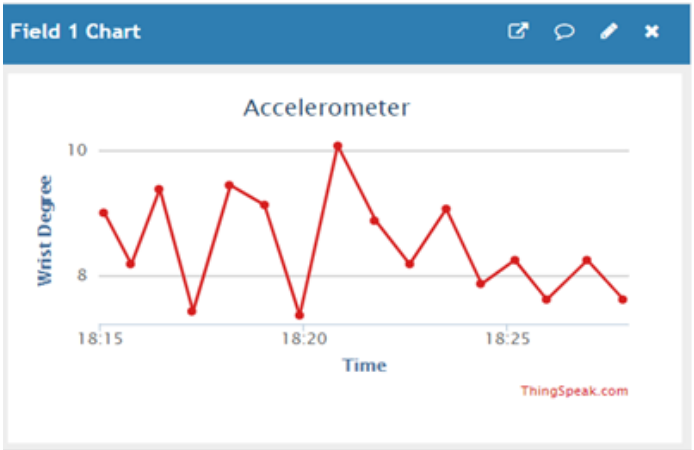


Time (s)	Measured Temp	Expected Temp	% Error
0	94.375	97.1	2.806385
15	94.125	97.1	3.063852
30	94	97.2	3.292181
45	94.375	96.9	2.605779
60	95.4375	97.1	1.712152
75	96.125	96.9	0.799794
90	96.75	97.1	0.360453
105	95.8125	97.1	1.325953
120	96.375	96.9	0.541796
135	90.4375	97.1	6.861483
150	95.9375	97.2	1.298868
165	95.8125	97.1	1.325953
180	96.25	97.1	0.875386
195	98.25	96.9	1.393189
210	95.25	97.1	1.905252
225	95.125	97.2	2.134774
Min	90.4375	96.9	0.360453
Max	98.25	97.2	6.861483
Avg	95.27734375	97.06875	2.018953

Figure 30: Internal-Body Temperature while Gaming

Wrist Angle while Gaming:





Time(s)	Measured Angle	Expected Angle	% Error
0	9	8.73	3.092784
15	8.1875	8.41	2.64566
30	9.375	9.26	1.241901
45	7.4375	7.54	1.359416
60	9.4375	9.74	3.105749
75	9.125	8.76	4.166667
90	7.375	7.22	2.146814
105	10.0625	9.84	2.261179
120	8.875	8.75	1.428571
135	8.1875	8.27	0.997582
150	9.0625	8.68	4.406682
165	7.875	7.64	3.075916
180	8.25	8.34	1.079137
195	7.625	7.86	2.989822
210	8.25	8.05	2.484472
225	7.625	7.43	2.624495
Min	7.375	7.22	0.997582
Max	10.0625	9.84	4.406682
Avg	8.484375	8.4075	2.444178

Figure 31: Wrist Angle while Gaming

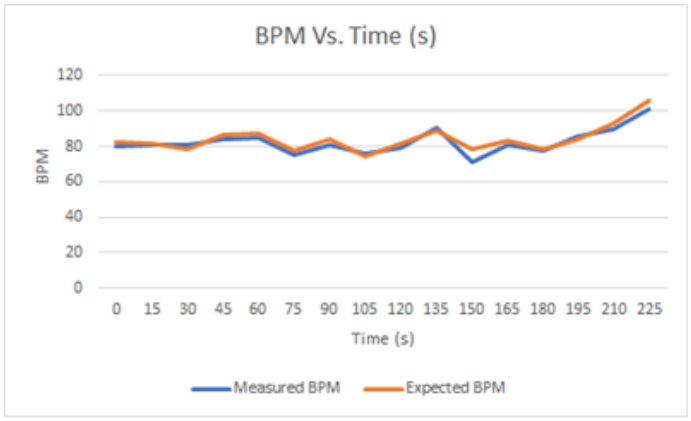
Picture of experiment:

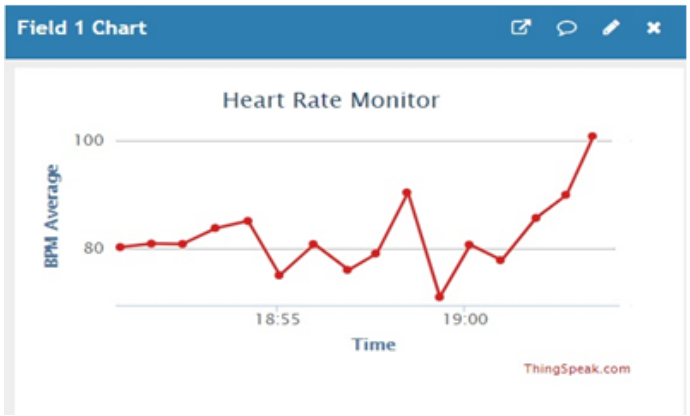
Test case 1 picture is located at the end of experiment 4.

Sensor Data Over 4 Minutes while Gaming 2 [Experiment 4 [Test case 1]]:

We have both the thingspeak graphs and excel graphs as excel graphs are easier to read. Thingspeak data only shows Measured Data, Expected Data is what we recorded from external devices. This is just a repeated experiment of the previous experiment, experiment 3. We just wanted to verify that our sensors are properly working in the intended use case. This is a part of our Test Case 1. All the graph points are 15 moving averages. We only have one picture for Experiment 3 and 4 as there was a part of one test case.

Heart Rate While Gaming 2:

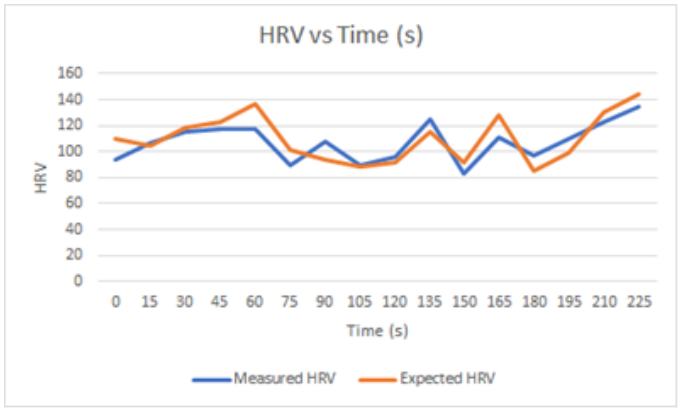




Time (s)	Measured BPM	Expected BPM	% Error
0	80.26852749	82.65	2.881394
15	80.91087992	81.89	1.195653
30	80.84732618	78.68	2.754609
45	83.80736338	86.08	2.640145
60	85.11504052	87.12	2.301377
75	75.02833929	77.72	3.463279
90	80.81896552	83.9	3.67227
105	76.03489129	74.49	2.073958
120	79.08308763	81.2	2.607035
135	90.40598836	88.67	1.957808
150	70.97957061	78.26	9.302874
165	80.70317776	83.58	3.441998
180	77.84348062	78.4	0.709846
195	85.68403068	83.66	2.419353
210	89.9710471	92.59	2.828548
225	100.8442982	105.47	4.385799
Min	70.97957061	74.49	0.709846
Max	100.8442982	105.47	9.302874
Avg	82.39662591	84.0225	3.039747

Figure 32: Heart Rate while Gaming

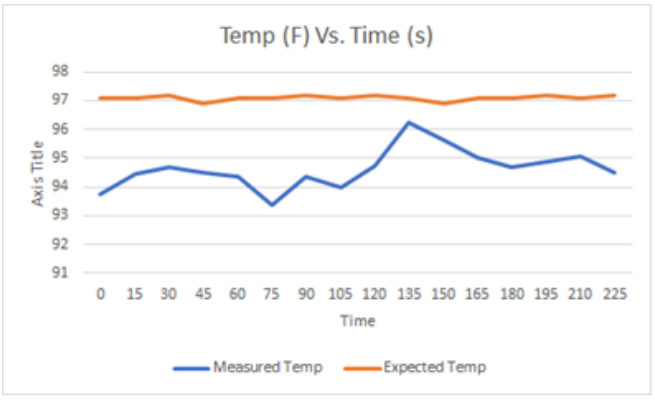
HRV While Gaming 2:

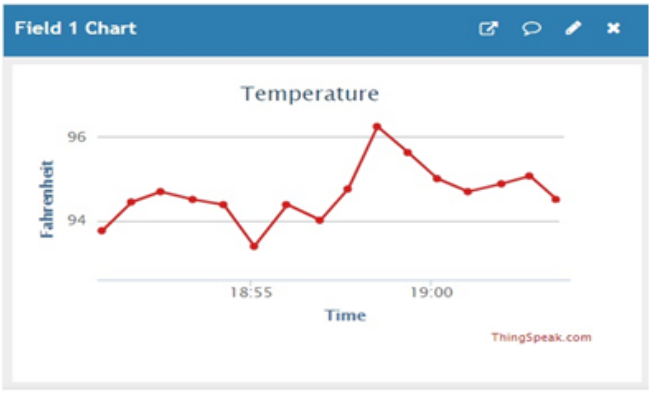


Time (s)	Measured HRV	Expected HRV	% Error
0	93.95917332	110	14.58257
15	106.49076	105	1.419771
30	114.8698842	118	2.65264
45	116.8632848	123	4.989199
60	117.8168468	137	14.0023
75	88.92634228	101	11.95412
90	107.8189655	94	14.70103
105	89.48032866	88	1.682192
120	95.63458956	92	3.950641
135	125.0076103	115	8.70227
150	83.15186139	92	9.617542
165	110.8654546	128	13.38636
180	96.84487728	85	13.93515
195	110.3659079	99	11.48072
210	122.707759	130	5.609416
225	134.7142499	144	6.448438
Min	83.15186139	85	1.419771
Max	134.7142499	144	14.70103
Avg	107.2198685	110.0625	8.694647

Figure 33: HRV while Gaming 2

Internal Body Temperature While Gaming 2:

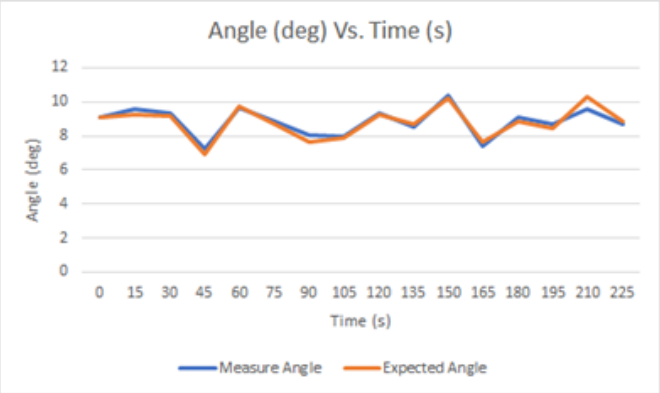


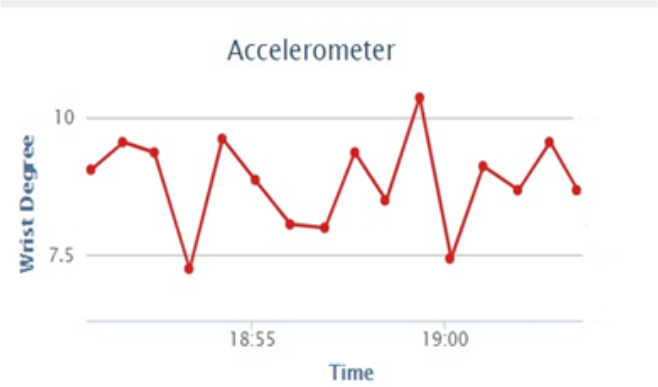


Time (s)	Measured Temp	Expected Temp	% Error
0	93.75	97.1	3.4501
15	94.4375	97.1	2.742
30	94.6875	97.2	2.5849
45	94.5	96.9	2.4768
60	94.375	97.1	2.8064
75	93.375	97.1	3.8363
90	94.375	97.2	2.9064
105	94	97.1	3.1926
120	94.75	97.2	2.5206
135	96.25	97.1	0.8754
150	95.625	96.9	1.3158
165	95	97.1	2.1627
180	94.6875	97.1	2.4846
195	94.875	97.2	2.392
210	95.0625	97.1	2.0984
225	94.5	97.2	2.7778
Min	93.375	96.9	0.8754
Max	96.25	97.2	3.8363
Avg	94.640625	97.10625	2.5389

Figure 34: Internal Body Temperature while Gaming 2

Wrist Angle While Gaming 2:





Time (s)	Measure Angle	Expected Angle	% Error
0	9.0625	9.12	0.630482
15	9.5625	9.23	3.602384
30	9.375	9.17	2.235551
45	7.25	6.95	4.316547
60	9.625	9.77	1.484135
75	8.875	8.73	1.660939
90	8.0625	7.68	4.980469
105	8	7.87	1.651842
120	9.375	9.28	1.023707
135	8.5	8.67	1.960784
150	10.375	10.19	1.815505
165	7.4375	7.63	2.522936
180	9.125	8.84	3.223982
195	8.6875	8.42	3.17696
210	9.5625	10.31	7.250242
225	8.6875	8.87	2.057497
Min	7.25	6.95	0.630482
Max	10.375	10.31	7.250242
Avg	8.84765625	8.795625	2.724623

Figure 35: Wrist Angle while Gaming 2

Picture from Test Case 2:

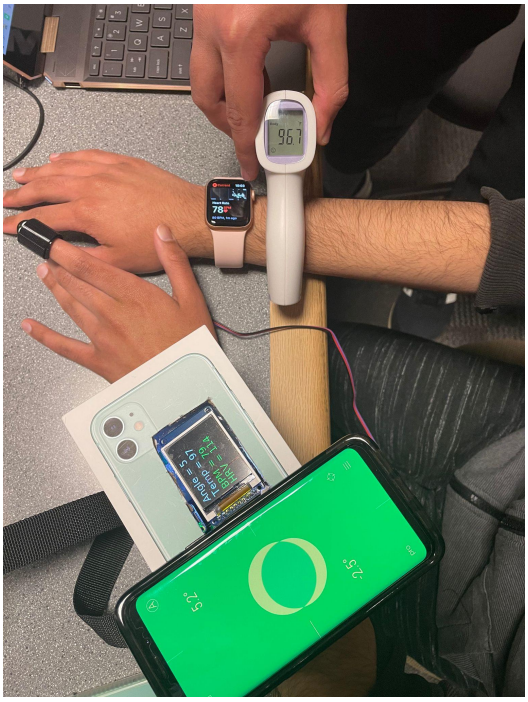


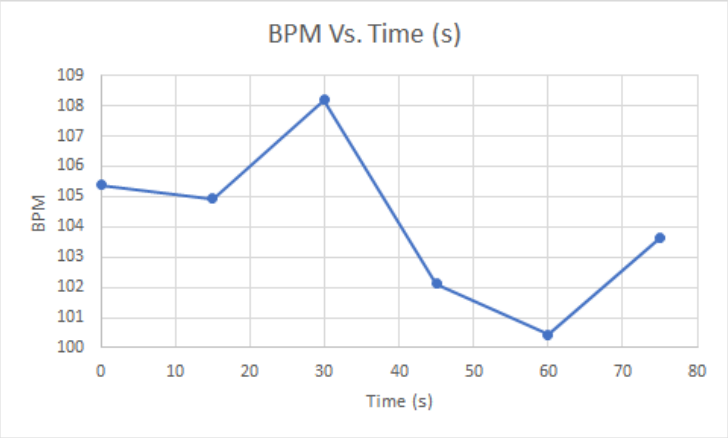
Figure 36: Test Case 2

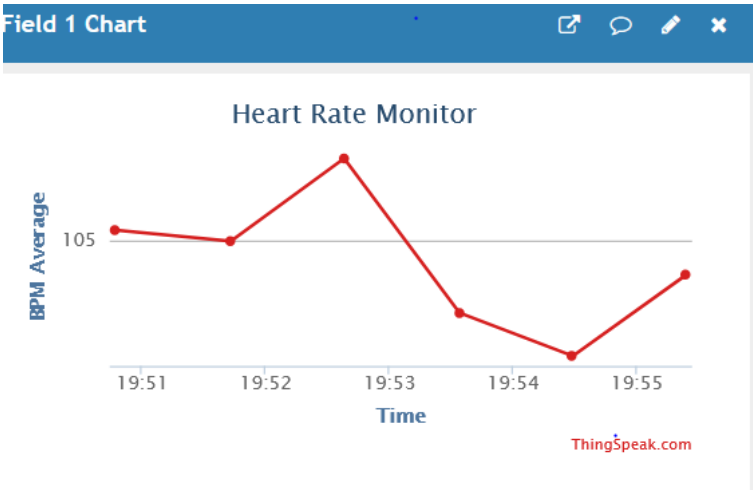
All external devices were managed by other group members while the main member was gaming. Members in picture [Moneeb, Aayush, Jamil]

Sensor Data and Email Status Over 2 Minutes while Exercising [Experiment 5 [Test case 2]]:

We have both the thingspeak graphs and excel graphs as excel graphs are easier to read. Both graphs show the same data. There are also some screenshots from the inbox. Here we were exercising to get an increased heart rate and HRV value to trigger email notifications. This test was to make sure our alert system worked properly and that we didn't miss any alerts or received false positives. This is a part of our Test Case 2. All the graph points are 15 moving averages. We only have one picture for Experiment 5 and 6 as there was a part of one test case.

Heart-Rate Email Status While Exercising:

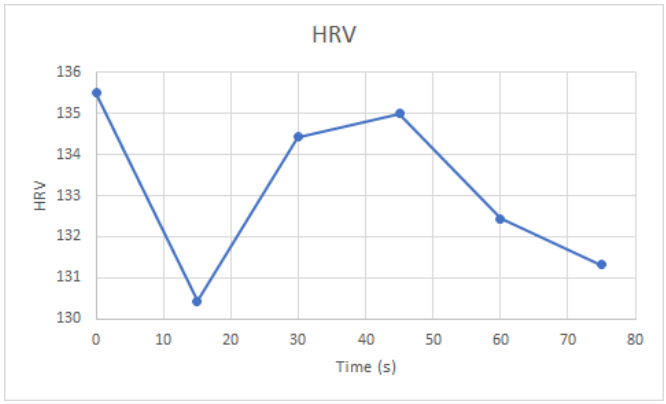


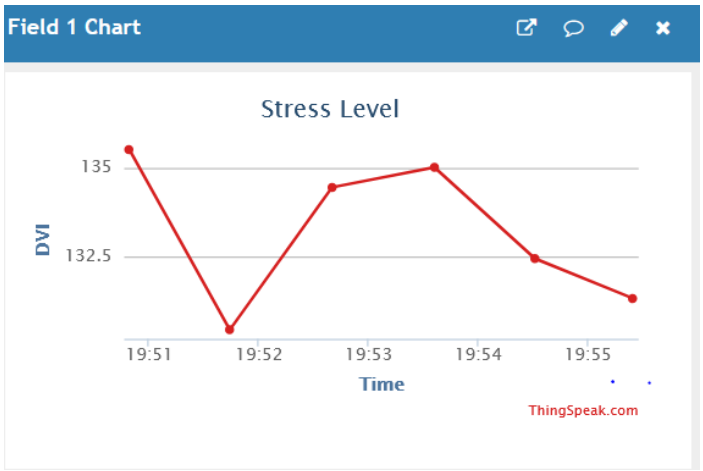


Time (s)	BPM	Actual Notification	Expected Notification	Mismatch
0	105.375	1	1	0
15	104.9375	1	1	0
30	108.1875	1	1	0
45	102.125	1	1	0
60	100.4375	1	1	0
75	103.625	1	1	0

Figure 37: Heart Rate Email Status while Exercising

HRV Email Status While Exercising:

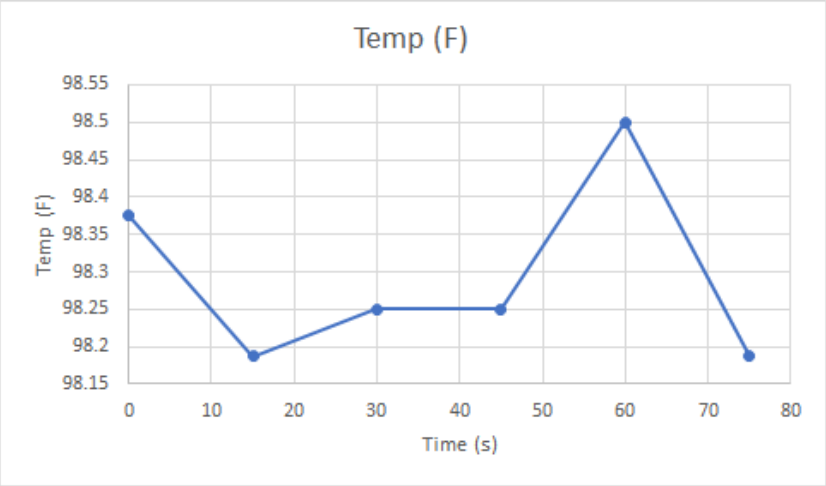


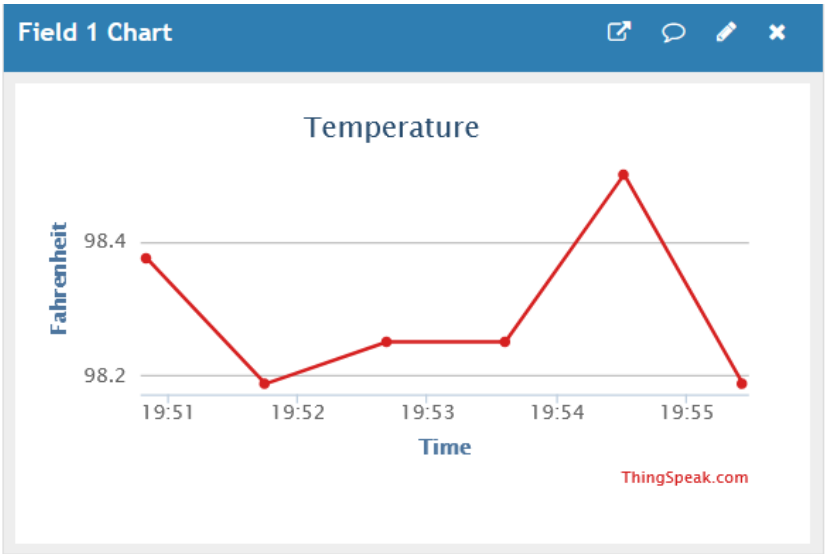


Time (s)	HRV	Actual Notification	Expected Notificaiton	Mismatch
0	135.5	1	1	0
15	130.4375	1	1	0
30	134.4375	1	1	0
45	135	1	1	0
60	132.4375	1	1	0
75	131.3125	1	1	0

Figure 38: HRV Email Status while Exercising

Internal Body Temperature Email Status While Exercising:

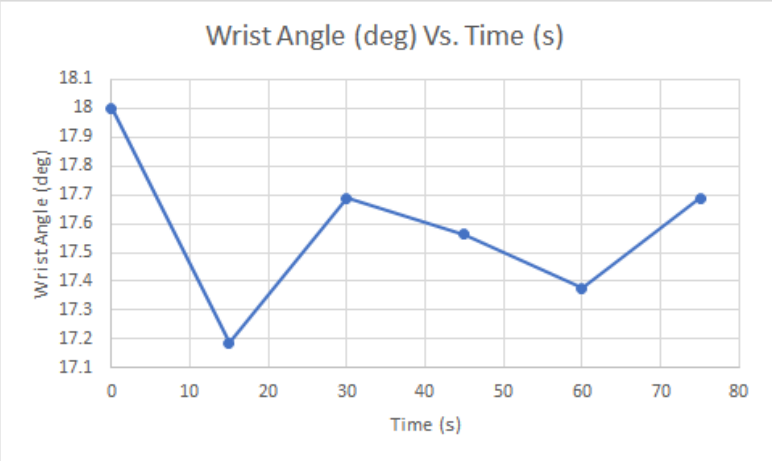


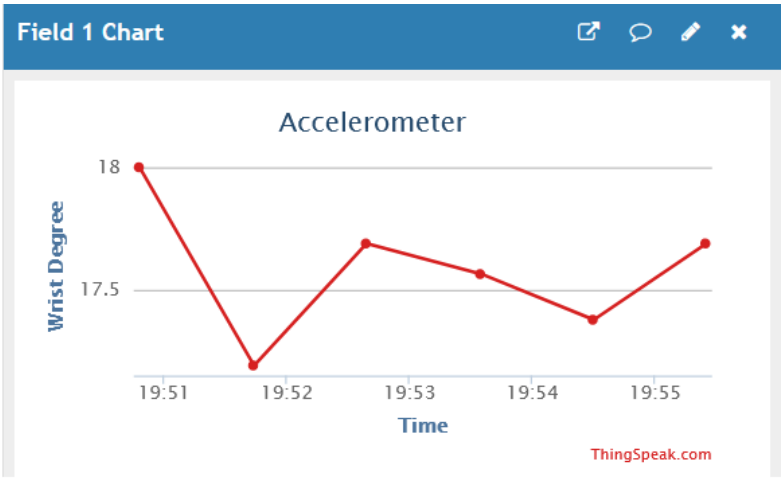


Time (s)	Temp (F)	Actual Notification	Expected Notificaiton	Mismatch
0	98.375	0	0	0
15	98.1875	0	0	0
30	98.25	0	0	0
45	98.25	0	0	0
60	98.5	0	0	0
75	98.1875	0	0	0

Figure 39: Internal Body Temperature Email Status while Exercising

Wrist Angle Email Status While Exercising:





Time (s)	Wrist Angle (deg)	Measured Notificaiton	Expected Notification	Mismatch
0	18	0	0	0
15	17.1875	0	0	0
30	17.6875	0	0	0
45	17.5625	0	0	0
60	17.375	0	0	0
75	17.6875	0	0	0

Figure 40: Wrist Angle Email Status while Exercising

Email Screen Shots for this section:

Figure 41:Email Screenshot

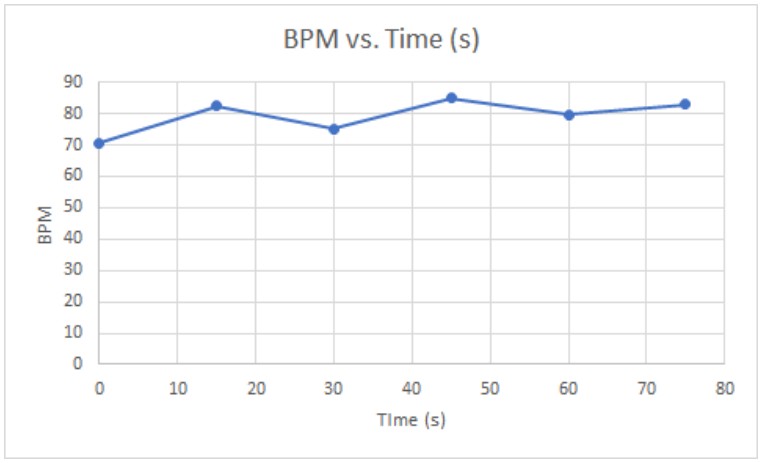
Picture from Experiment:

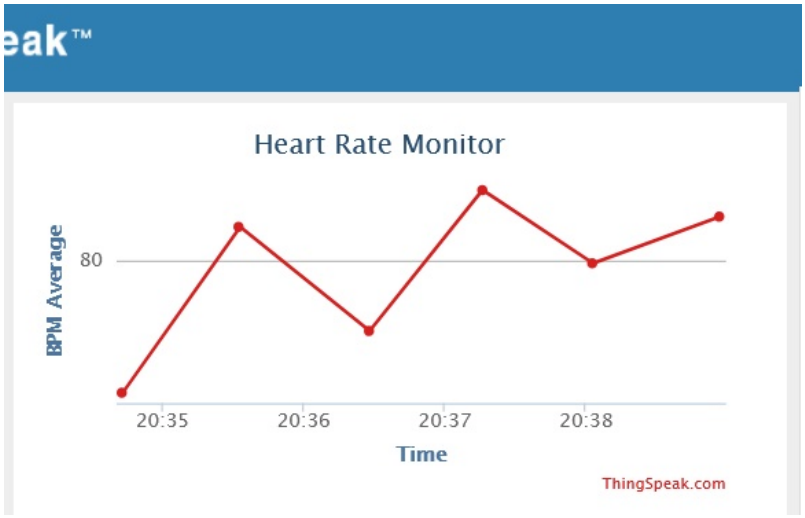
Test case 2 picture is located at the end of experiment 6.

Sensor Data and Email Status Over 2 Minutes while resting [Experiment 6 [Test case 2]]:

We have both the thingspeak graphs and excel graphs as excel graphs are easier to read. Both graphs show the same data. There are also some screenshots from the inbox. Here we were resting and exaggerating our wrist to trigger an email notification. This test was to make sure our alert system worked properly and that we didn't miss any alerts or received false positives for our wrist angle. This is a part of our Test Case 2. All the graph points are 15 moving averages. We only have one picture for Experiment 5 and 6 as there was a part of one test case.

Heart-Rate Email Status While Resting:

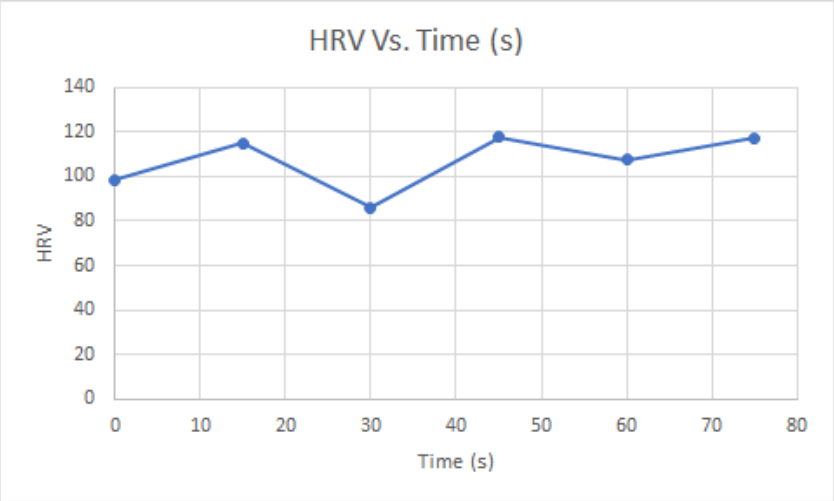




Time (s)	BPM	Actual Notification	Expected Notification	Mismatch
0	70.72061	0	0	0
15	82.31448	0	0	0
30	75.0469	0	0	0
45	84.89653	0	0	0
60	79.77162	0	0	0
75	83.03349	0	0	0

Figure 43: Heart Rate Email Status while Resting

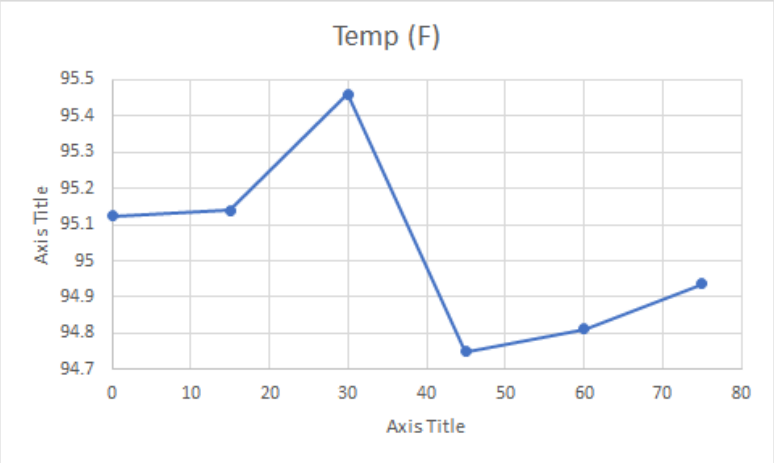
HRV Email Status While Resting:

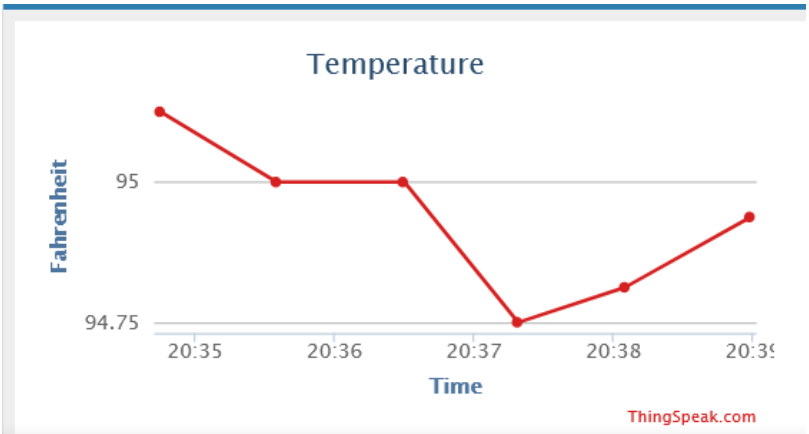


Time (s)	HRV	Actual Notification	Expected Notification	Mismatch
0	98.58011	0	0	0
15	114.7646	0	0	0
30	86.0469	0	0	0
45	117.754	0	0	0
60	107.415	0	0	0
75	117.0335	0	0	0

Figure 44: HRV Email Status while Resting

Internal Body Temperature Email Status While Resting:

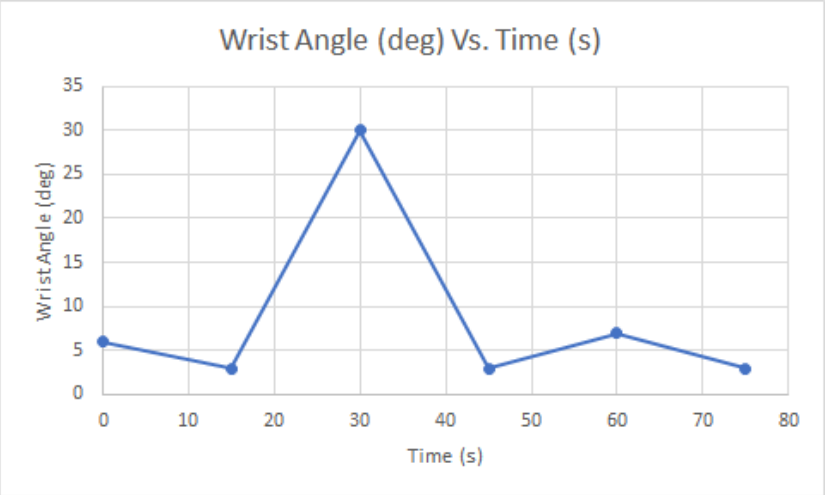


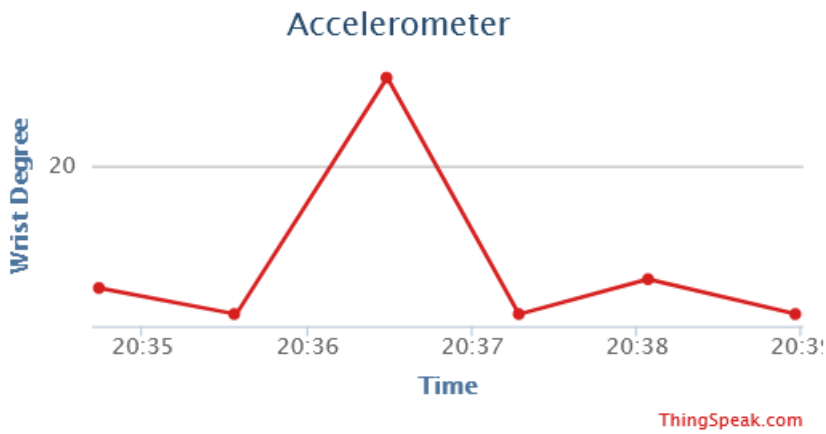


Time (s)	Temp (F)	Actual Notification	Expected Notification	Mismatch
0	95.125	0	0	0
15	95.14	0	0	0
30	95.46	0	0	0
45	94.75	0	0	0
60	94.8125	0	0	0
75	94.9375	0	0	0

Figure 45: Internal Body Temperature Email Status while Resting

Wrist Angle Email Status While Resting:





Time (s)	Wrist Angle (deg)	Actual Notification	Expected Notification	Mismatch
0	6	0	0	0
15	3	0	0	0
30	30	1	1	0
45	3	0	0	0
60	7	0	0	0
75	3	0	0	0

Figure 46: Wrist Angle Email Status while Resting

Email Screen Shots for this section:

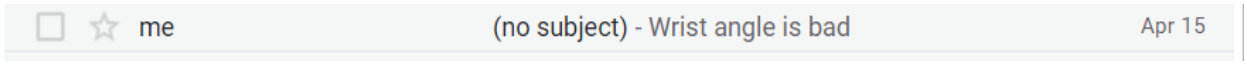


Figure 47: Email Screenshot

Picture from Test Case 2:

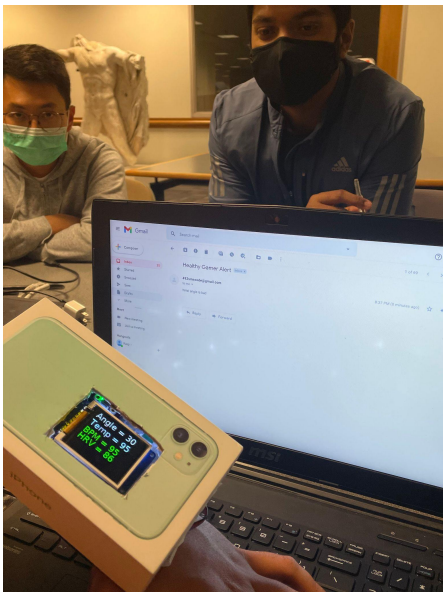


Figure 48: Test Case 2 Picture

Healthy-Gamer Alert working with an exaggerated wrist angle. Members in picture [Saad, Ryan, and Priyam]

VI. Experiment validation using evaluation criteria

Proof of Success:

This section highlights that the different sensors for the Healthy-Gamer device work as intended. The pictures and video below show proof of success for these sensors.

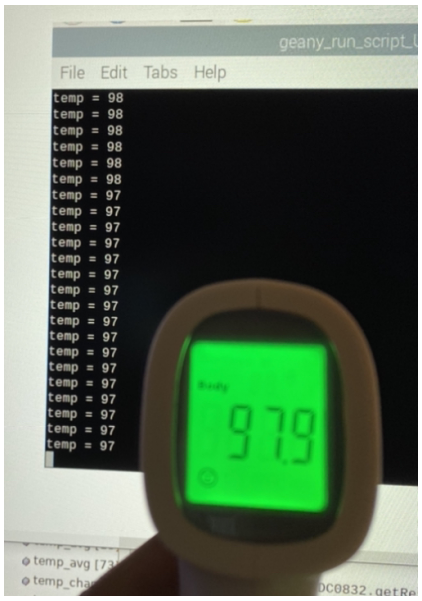


Figure 49: Proof of Success Thermometer

The picture above shows the stand alone temperature reading from the Healthy-Gamer device vs an external device.

Figure 50: Proof of Success BPM

The picture above shows the stand alone pulse sensor reading from the Healthy-Gamer Device vs an external device

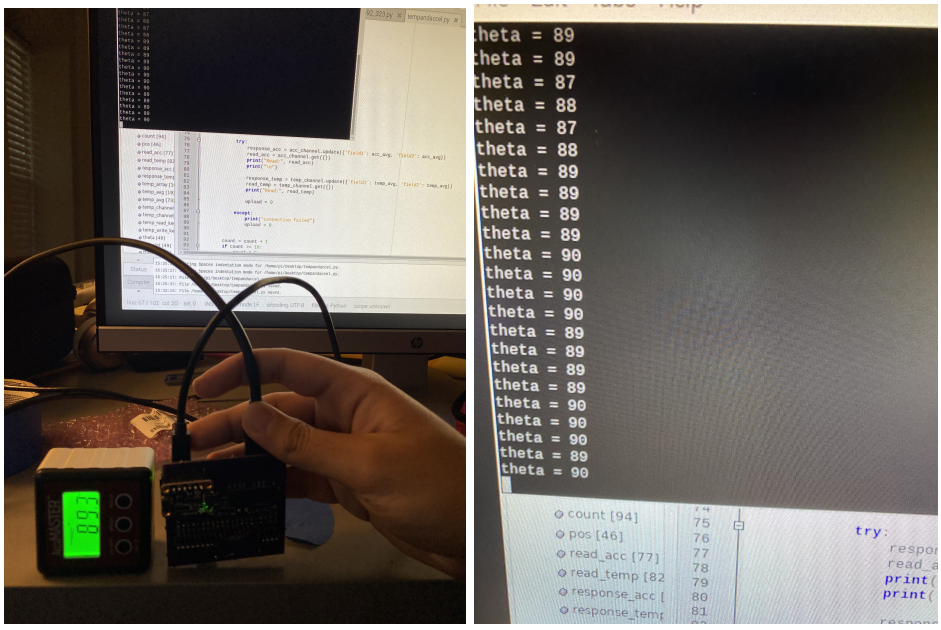


Figure 51: Proof of Success Accelerometer

The picture above shows the stand alone accelerometer reading from the Healthy-Gamer Device vs an external device.

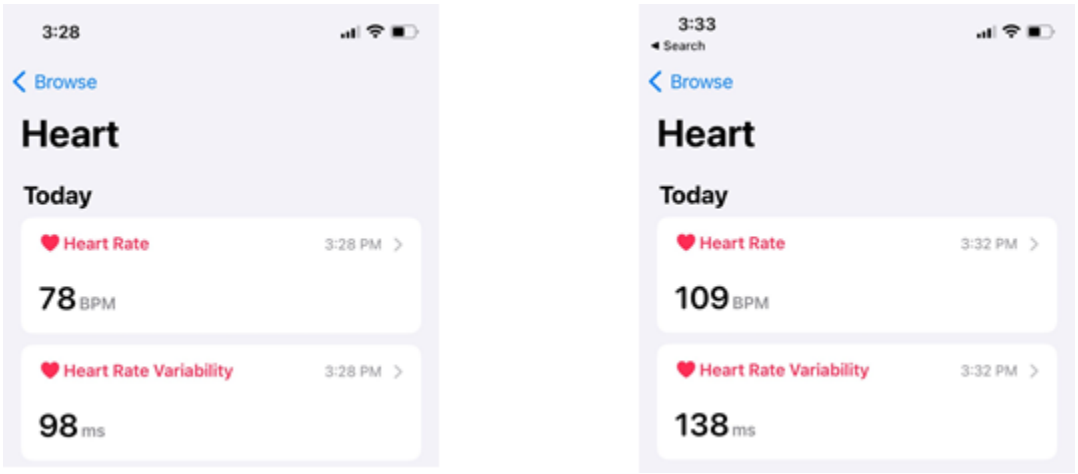


Figure 52: Proof of Success HRV

The picture above shows resting and active data from the Apple Watch Health App for iPhone

bpm	hrv	
84	93	
75	116	
82	98	
82	100	
87	129	
80	111	
86	123	
82	50	Dad
76	80	Mom

Figure 53: Proof of Success HRV 2

Extended BPM-HRV values from the App from different points and people to show that an average over time is needed to properly assess HRV stress level.

Here is a sample demonstration of the Healthy-Gamer Device working as intended. This is not a part of any experimentation or test case.
https://www.youtube.com/watch?v=r3XQp6NNOVc&ab_channel=MoneebAhmad

Processed Data used for conclusion:

Test Case 1 Gaming Test 1 [Experiment 3]:

Heart-Rate while gaming 1

Average percent error was about 4% so this was successful.

	Measured BPM	Expected BPM	% Error
Min	68.86584074	74.44	0.750505
Max	93.37767385	93.97	8.138592
Avg	80.42011625	82.1275	4.263174

HRV while gaming 1

Average percent error was about 10%, but the average HRV values were close together. HRV needs to be assessed over time so since these values were very close we can say this was a success.

	Measured HRV	Expected HRV	% Error
Min	87.00205339	80	2.218569
Max	124.0361369	122	20.23233
Avg	102.8993503	102.0625	9.602781

Temperature while gaming 1

Average percent error was about 2% so this was successful.

	Measured Temp	Expected Temp	% Error
Min	90.4375	96.9	0.360453
Max	98.25	97.2	6.861483
Avg	95.27734375	97.06875	2.018953

Wrist angle while gaming 1

Average percent error was about 2% so this was successful.

	Measured Angle	Expected Angle	% Error
Min	7.375	7.22	0.997582
Max	10.0625	9.84	4.406682
Avg	8.484375	8.4075	2.444178

Test Case 1 Gaming Test 2 [Experiment 4]:

Heart-Rate while gaming 2

Average percent error was about 3% so this was successful.

	Measured BPM	Expected BPM	% Error
Min	70.97957061	74.49	0.709846
Max	100.8442982	105.47	9.302874
Avg	82.39662591	84.0225	3.039747

HRV while gaming 2

Average percent error was about 8%, but the average HRV values were close together. HRV needs to be assessed over time so since these values were very close we can say this was a success.

	Measured HRV	Expected HRV	% Error
Min	83.15186139	85	1.419771
Max	134.7142499	144	14.70103
Avg	107.2198685	110.0625	8.694647

Temperature while gaming 2

Average percent error was about 2% so this was successful.

	Measured Temp	Expected Temp	% Error
Min	93.375	96.9	0.8754
Max	96.25	97.2	3.8363
Avg	94.640625	97.10625	2.5389

Wrist angle while gaming 2

Average percent error was about 3% so this was successful.

	Measure Angle	Expected Angle	% Error
Min	7.25	6.95	0.630482
Max	10.375	10.31	7.250242
Avg	8.84765625	8.795625	2.724623

Test Case 2 Email Test 1 [Experiment 5]:

Heart-Rate while Exercising

No missed alerts so this was successful.

Time (s)	BPM	Actual Notification	Expected Notification	Mismatch
0	105.375	1	1	0
15	104.9375	1	1	0
30	108.1875	1	1	0
45	102.125	1	1	0
60	100.4375	1	1	0
75	103.625	1	1	0

HRV while Exercising

No missed alerts so this was successful.

Time (s)	HRV	Actual Notification	Expected Notificaiton	Mismatch
0	135.5	1	1	0
15	130.4375	1	1	0
30	134.4375	1	1	0
45	135	1	1	0
60	132.4375	1	1	0
75	131.3125	1	1	0

Temperature while Exercising

No false positive alerts were sent so this was successful.

Time (s)	Temp (F)	Actual Notification	Expected Notificaiton	Mismatch
0	98.375	0	0	0
15	98.1875	0	0	0
30	98.25	0	0	0
45	98.25	0	0	0
60	98.5	0	0	0
75	98.1875	0	0	0

Wrist angle while Exercising

No false positive alerts were sent so this was successful.

Time (s)	Temp (F)	Actual Notification	Expected Notificaiton	Mismatch
0	98.375	0	0	0
15	98.1875	0	0	0
30	98.25	0	0	0
45	98.25	0	0	0
60	98.5	0	0	0
75	98.1875	0	0	0

Test Case 2 Email Test 2 [Experiment 6]:

Heart-Rate while resting:

No false positive alerts were sent so this was successful

Time (s)	BPM	Actual Notification	Expected Notification	Mismatch
0	70.72061	0	0	0
15	82.31448	0	0	0
30	75.0469	0	0	0
45	84.89653	0	0	0
60	79.77162	0	0	0
75	83.03349	0	0	0

HRV while resting:

No false positive alerts were sent so this was successful

Time (s)	HRV	Actual Notification	Expected Notification	Mismatch
0	98.58011	0	0	0
15	114.7646	0	0	0
30	86.0469	0	0	0
45	117.754	0	0	0
60	107.415	0	0	0
75	117.0335	0	0	0

Temperature while resting:

No false positive alerts were sent so this was successful

Time (s)	Temp (F)	Actual Notification	Expected Notification	Mismatch
0	95.125	0	0	0
15	95.14	0	0	0
30	95.46	0	0	0
45	94.75	0	0	0
60	94.8125	0	0	0
75	94.9375	0	0	0

Wrist angle while resting with one exaggerated position:

No false positive alerts were sent so this was and only the correct alert was sent

Time (s)	Wrist Angle (deg)	Actual Notification	Expected Notification	Mismatch
0	6	0	0	0
15	3	0	0	0
30	30	1	1	0
45	3	0	0	0
60	7	0	0	0
75	3	0	0	0

Answer to requirement analysis:

Test Case 1 Questions:

1) How much does heartbeat data deviate from the Apple Watch?

BPM had a worst case percent error Minimum of 0.75% with an expected value of 77.07 and a measured value of 76.49. A worst case Maximum of 9.30% with an expected value of 78.26 and a measured value 70.97. A worst case average percent of 4.26%. Since the BPM deviated on average by about 5% the BPM was a success.

2) How much does body temperature data deviate from the Infrared Temperature Gun?

Temperature had a worst case percent error Minimum of 0.88% with an expected value of 97.1℉ with a measured value of 96.25℉. A worst case Maximum of 6.86% with an expected value of 97.1℉ and a measured value of 90.44℉. This error occurred in our testing because the thermal resistor was pushed back into the box for a brief moment before being pushed back out of the box and onto a user's wrist. A worst case average percent of 2.54%. Since the temperature deviated on average by about 3% the temperature was a success.

3) How much does wrist angle data deviate from the Digital Leveler?

The wrist angle had a worst case percent error Minimum of 1.00% with an expected value of 8.27 and a measured value of 8.19. A worst case Maximum of 9.30% with an expected value of 10.31 and a measured value of 9.56. A worst case average percent of 7.25%. Since the wrist angle deviated on average by about 3% the wrist angle was a success

4) How much does HRV data deviate from the Apple Watch App for Iphone?

HRV had a worst case percent error Minimum of 2.22% with an expected value of 105 and a measured value of 107. A worst case Maximum of 20.23% with an expected value of 95 and a measured value of 114. A worst case average percent of 9.61%. Average values recorded from the Apple watch were 110.63 and 102.06. The average values from the device were 107.22 and 102.06. With close average values the HRV was a success.

5) Was data collected in a consistent amount?

Yes the data was collected every second from each component of the Health-Gamer device.

Test Case 2 Questions:

1) Is an alert that is being sent reliable if the user's health is showing signs of hazardous behaviors, any outliers?

   Yes the email system was reliable. It would alert the user if their health reading were high and would also detail which health reading was high. The alert system worked with no outliers.

2) Is the alert being sent within a reasonable time frame?

   The alerts were received on average of about 10 seconds after the cloud data was updated.

3) How many incorrect alerts have been sent?

   Zero incorrect alerts were sent.

4) If an alert has not been sent what is causing the issue?

   No problematic alerts.

5) What is the percent value of correct alerts?

   The alerts had a 100% success rate.

VII. Other issues

I. Reason for the Project

The reason why we created the Healthy-Gamer device is because we have realized how playing video games for an extended period of time can cause issues to a user's physical and psychological health. Indulging in such activity may make one's mind forget about the necessary breaks that he/she needs; however, our body never stops from giving us clues about the necessary breaks that it requires. The Healthy-Gamer is such a product that picks up on those clues and makes it easier for us to take decisions based on data. This product is mainly intended for people who not only enjoy playing video games but also enjoy having a healthy body. It alerts them of any hazardous behavior that can prove to be dangerous if persisted for a long time and implies them to take necessary actions based on that.

II. Potential Use Cases of the Project

The main function of the Health-Gamer device is to track and analyze a user's health while they are gaming and alert them if they show any hazardous health behavior. We believe that this product will be a huge success in today's market as the number of individuals who play video games grow more and more each day. This Seamless Physiological Monitoring device will be marketed towards individuals who want to keep track of their mental and physical health while they game over an extended period. Additionally, because of the type of customers that our product caters to, this has the potential to be sold globally.

III. Cost Figure

We have spent a total of $363.7 inorder to build the prototype of the Healthy-Gamer device. The cost would have been lower if the PCB boards were wired correctly in its initial version. Nevertheless, the total cost was well within our expectation budget for this project. We spent a total of $50 on buying two Raspberry Pi Zeros, one of them as a backup. We spent around $25 on the pulse sensor, $20 on the Temperature gun, and $32 on the Digital Leveler. The

MCP3008 ADC and ADC0832 cost us $12.50 and $2 each respectively. As for the lithium-ion rechargeable battery, we purchased it from Amazon for $40. A table is included in the next section where we have listed out how the funds have been spent on each item. Another table in the same section also describes how much man hours we have spent on each task in order to successfully build the prototype. The total man hour utilized was roughly 85 hours to complete the whole project.

IV. Alternative designs

To reduce the ecological footprint of our product, we have already chosen an alternative to the proposed design solution for the product. In the beginning, we decided to encase everything into a 3D case using the biodegradable printing material called PLA; however, we chose to not persist with it for the prototype for now. We are recycling an old iphone box to use it as our device case. And as far as the size of the device is concerned, maybe it could have been made less bulkier had we chosen a thinner battery and nanoPi NEO instead of Pi Zero. Also, choosing a smaller sized LCD would have made the device smaller but the overall cost to go higher as we are using an old LCD from our ECE 447 lab kit. Our initial plan to implement the server was to use AWS. We had difficulties getting AWS to work with our system, so ultimately we decided to use ThingSpeak for our cloud service. Originally, we wanted ThinkSpeak to upload the user's data every second. However, this led to a bottleneck in performance for the raspberry pi zero. Because of this we ended up sending the average value of the user's health reading after 15 consecutive measurements. An alternative MCU that we contemplated on using was the MSP430 as it was smaller and drew less power when compared to a raspberry pi zero. Since the MSP430 was weaker computationally we ultimately decided to use a Raspberry pi zero instead as the MCU for the Healthy-Gamer device.

V. Maintenance

There are some steps we ought to take in order to maintain the software and hardware of our functionalities of the Healthy-Gamer device. We will constantly keep our eyes on the code that we have written to see if there are any other ways we can improve it to make it run faster

and more efficiently. We also used external libraries so if any of these libraries get updated we would need to update the syntax of our code. And for the hardware, our goal is to keep the rechargeable lithium-ion battery and Pi Zero undamaged for as long as possible. Here are some measures that we are taking to keep the prototype safe and working:

1) Switching off the power when not in use
2) Keeping a backup copy of all of the files and folders
3) Keeping a log so that anyone can see what changes have been made up to that point
4) Using partial-discharge cycle for the battery as using only 20% or 30% of the battery capacity before recharging extends cycle life considerably
5) Avoiding charging the battery to a 100% at once
6) Avoiding high charge and discharge events.

VI. Retirement

There are many places such as Best Buy, Home Depot, Staples or any other electronic stores that offer options to help people recycle their old electronics. Our plan is to take our prototype to one of these places to dispose of it after its lifetime or if anyone wants to keep the prototype as a souvenir, we will happily give it to that person. However, the damaged battery will be disposed of after its shelf life. As lithium-ion batteries contain life-threatening chemicals, we have to take extra precautions before getting rid of them. Here are some steps that we have decided to take for the battery after its shelf life:

- If the battery is physically damaged, we will store it in an insulated plastic bag to avoid any short-circuiting.
- Keep it in a cool and dry place to avoid combustion.
- Bring it to a certified recycling provider who can safely deal with the hazardous material that it contains.
- Contact their manufacturer to see if there is an industry collection program for them.

VIII. Administrative

1) Project Progress:

Finishing prototyping (All parts functioning)

We had finished prototyping by February 5th. At this point we had individually checked each sensor and element to ensure that they were properly working by themselves.

I. PCB Schematic

We had finished the PCB schematic by February 15th. The PCB schematic detailed how each sensor and component was connected to one another. This also showed how each component was connected to the pi.

II. PCB Design

We had finished the PCB design by February 23rd. The PCB design was the KiCAD wiring that was used for the PCB printing.

III. PCB Printing/Soldering

We had got the second copy of the PCB board back from Oshpark by March 28th. We finished soldering the components to the PCB board by March 31st.

IV. 3D Printed Case

We ended up not going through with a 3D printed model. We had some initial Blender models for the 3D case, but, we ultimately ended up just using a cardboard box to enclose the final product.

V. System Integration

We had a complete product by April 10th. The PCB was fully soldered and functioning while in the box. The PCB was also running off of the pi sugar battery pack.

VI. Testing Case 1

We finished test case 1 by April 13th. The test case was considered successful as stated in the Processed Data section.

VII. Testing Case 2

We finished test case 2 by April 15th. The test case was considered successful as stated in the Processed Data section.

VIII. Reporting

The draft final report was completed by April 22nd.

IX. Final Presentation Preparation

Final Presentation information has yet to be released. Information will be given on April 23rd.

2) Changes in schedule/design/tasks:

There were some extra activities that had to be done during the time of this project. The first being switching over from the MSP430 to the Raspberry Pi Zero. Originally the device was going to use the MSP430 however, as the project progressed, the group was able to notice the significant balance of performance when compared to the Raspberry Pi Zero, inturn changing the MCU used for the device. Another change in the design was with the cloud server. AWS was our first choice to store the data from the device however, the components used in the device did not fully work together with AWS causing the team to find an alternative cloud storage service. Because of this we ended up using ThinkSpeak as the cloud based storage system for the Healthy-Gamer device.

The PCB design and soldering also brought some changes in schedule as well as the design. In our original PCB design, the wires were connected to the wrong components. On top of this some of the wire had sharp turns which caused the current to not flow properly to some of the components. The initial PCB design also had spacing issues between the components, they were too close together which caused us to accidentally solder components together. Due to these issues, a completely new version of the PCB had to be designed with new wirings as well as

more space between the components.This incident set us roughly back 2 weeks due to the wait time for the board to arrive and solder. Originally, the Healthy-Gamer device was to be enclosed in a 3D printed case, however with the PCB full implementation setting the project back by two weeks we were forced to abandon the plan and use a cardboard box to encase the device instead.

Another design change that was incorporated into this project was the temperature reading. In our initial design we implemented the thermal resistor to read a user's surface body temperature. We then decided that we were going to change this feature and instead have the thermal resistor read a user's internal body temperature. In order to implement this design change we had to change the resistor value that was in series with the thermal resistor from 10K ohms to 5.1K ohms.

3) Extra/unplanned activities:

One unplanned activity that was implemented in the Healthy-Gamer project was that it was encapsulated in a cardboard box. Initially, the Healthy-Gamer project was supposed to be encapsulated in a 3D design case that we were going to make ourselves. However, due to setbacks that were caused by the PCB implementation we decided not to fully pursue creating a 3D design as it was not a necessary feature. Instead we focused more of our time in making sure that the Healthy-Gamer device was able to read a user's health information in a more concise and accurate way. We ended up encapsulating the Healthy-Gamer device into a cardboard box. Even though this box made our device slightly bulker, the Healthy-Gamer device still functions properly as intended.

4) Funds spent:

In the beginning of our project, we predicted it would cost us less than $500 to complete it. We successfully achieved our goal and it cost us a little less than $365. The most expensive element of our project was the PCB boards because we had to buy several of them. We also had to buy a couple of Pi's so that we can test multiple experiments with the sensors. Also it made sure we had at least two members who can work on implementing the codes on the Pi. Shipping

also took a big chunk of the fund as some of the parts were coming from other countries and also due to Covid-19 expedited shipping cost extra.

Item	**Cost ($)**	**Vendor**
Pi Zero x2	50.3	Amazon
Pulse Sensor	24.95	SparkFun
LCD	19.95	447 Kit
MCP3008 ADC	12.5	Amazon
ADC0832	$2	350 Kit
Thermistor	1	350 Kit
PCB x9	150	Oshpark
Accelerometer	10	447 Kit
Temp. Tester	20	Amazon
Accel. Tester	32	Amazon
Resistors	1	Previous Lab Kits
Fitbit pre-owned	0	
Battery	40	Amazon
Total	$363.70	

5) Time spent:

Task	**Hours Spent**
Created a feature that allows the Healthy-Gamer device to store the user's health reading into ThinkSpeak.	15
Implementing the pulse rate sensor into the Healthy-Gamer device in order to read a user's BPM.	8
Implementing the temperature sensor into the Healthy-Gamer device in	5

order to read the user's internal temperature.	
Implementing the accelerometer sensor into the Healthy-Gamer device in order to read the user's wrist angle.	4
Wrote a script that would calculate a user's Heart Rate Variability Calculation based upon the reading from the pulse rate sensor.	5
Integrated an LCD Screen into the Healthy-Gamer device in order to show a user their health reading in real time.	8
Connecting and testing all the components together on a breadboard in order to test out the functionality of the Healthy-Gamer device.	6
Initialized a 3D model in order to encase the Healthy-Gamer device.	5
Creating a circuit schematic in order to connect all of the components of the Healthy-Gamer device together.	4
Creating a PCB Design based off the circuit schematic	10
Soldering the components of the Healthy-Gamer device on the PCB	6
Experiment testing	6
Designing a box that encapsulated the Healthy-Gamer device	3
Total hour spent	85

IX. Lessons Learned

This senior design project took almost a year to complete, as a result many of us have learned valuable lessons from this project that we plan on using in our professional careers. One of the most beneficial experiences that we have obtained from this project is learning how to properly implement such a high level of hardware design and software design together. Through most of our college careers, we had projects that required us to work with integrating software and hardware components together on a smaller scale. This project however implemented different sensors and features which resulted in our code to be close to 300 lines. When implementing the code we had to find creative ways to structure our code in order to make sure that our code compiled as efficiently as possible in order to avoid any runtime errors. In order to

do this we had 2 function methods which were then encapsulated into a main method. By having these 2 function methods it greatly reduced the compilation time of our program when compared to having all of our code in the main method alone. Another reason why we decided to make our code as efficient as possible is because we have implemented our project using a raspberry pi zero as our microcontroller. We decided to use the pi zero as it is one of the most portable microcontroller devices that we could find and it's SPI interface made it easy for us to implement our sensors onto it. Although the pi zero is great for portable use it is not so great for performance as the CPU is only a single core. Because of this hardware limitation it was very important that we had our code be efficient enough to run on the pi zero without running into any runtime errors.

Another lesson that we have learned is how to properly research in order to find out how to get the components of the device to work with one another. An example of this is when we initially tried to use the LCD screen in order to show off the user's data in real time. Prior to this project none of us have worked with implementing the LCD screen on a raspberry pi. In order for us to implement the screen into the device we had to research and read documentation associated with the LCD screen. After reading through the documentation we had a better understanding of the schematic for wiring the screen and also figured out how to code the screen in order to display our health readings.

All in all, there are so many lessons that we have learned from this project. We learned how to implement our theoretical knowledge into the actual application and solve a problem as a team. We also learned that proper research could lead to more efficient and less time-consuming problem-solving. The way we present our progress and result is also important since we have to make sure the professor, in this case, our client, is happy with our project. We have to work efficiently since time is limited and we believe the senior design project has helped us grow as engineers.

X. References

Literature References (see proposal doc for references) web urls (github pages/libraries we used)

Literature References:

[1] J. M. von der Heiden, B. Braun, K. W. Müller, and B. Egloff, “The association between video gaming and psychological functioning,” *Frontiers in psychology*. Jul. 2019, DOI: 10.3389/fpsyg.2019.01731.

[2] “Carpal Tunnel Syndrome Fact Sheet,” *National Institute of Neurological Disorders and Stroke*. Apr. 2020, [Online]. Available: https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Fact-Sheets/Carpal-Tunnel-Syndrome-Fact-Sheet

[3] “How to Send Data to ThingSpeak Cloud using Raspberry Pi.” Jan. 2019. https://iotdesignpro.com/projects/how-to-send-data-to-thingspeak-cloud-using-raspberry-pi .

[4] Dibya Chakravorty, “STL File Format (3D Printing) – Simply Explained,” All3DP, Jun. 13, 2017. https://all3dp.com/what-is-stl-file-format-extension-3d-printing/.

[5] L. Haghshenas and M. Pooyan, "Investigating the Effect of Computer Games on the Heart Signal," 2019 27th Iranian Conference on Electrical Engineering (ICEE), Yazd, Iran, 2019, pp. 1775-1778, doi: 10.1109/IranianCEE.2019.8786686.

[6] H.-G. Kim, E.-J. Cheon, D.-S. Bai, Y. H. Lee, and B.-H. Koo, “Stress and Heart Rate Variability: A Meta-Analysis and Review of the Literature,” Psychiatry Investigation, vol. 15, no. 3, pp. 235–245, Mar. 2018, doi: 10.30773/pi.2017.08.17.

[7] Abyarjoo, Fatemeh & Barreto, Armando & Abyarjoo, Somayeh & Ortega, Francisco & Cofino, Jonathan. (2013). Monitoring Human Wrist Rotation in Three Degrees of Freedom. Conference Proceedings - IEEE SOUTHEASTCON. 10.1109/SECON.2013.6567517.

[8] M. Bahrudin, “Carpal Tunnel Syndrome (CTS),” Saintika Medika, vol. 7, no. 1, Oct. 2012, doi: 10.22219/sm.v7i1.1090.

[9] N. Smith, “The giants of the video game industry have thrived in the pandemic. Can the success continue?,” *The Washington Post*, 14-May-2020. [Online]. Available: https://www.washingtonpost.com/video-games/2020/05/12/video-game-industry-coronavirus/. [Accessed: 23-Apr-2021].

[10] “Violent Video Games Stress People Out and Make Them More Aggressive,” *HeartMath Institute*. [Online]. Available: https://www.heartmath.org/research/research-library/educational/violent-video-games-stress-people-out-and-make-them-more-aggressive/. [Accessed: 23-Apr-2021].

Webpages for Code Libraries.

ADC0832 Library:
https://github.com/adeept/Adeept_Ultimate_Starter_Kit_Python_Code_for_RPi/blob/master/ADC0832.py

Accelerometer Library:
https://pypi.org/project/msa301/

LCD Library:
https://pypi.org/project/adafruit-circuitpython-mcp3xxx/
https://learn.adafruit.com/1-8-tft-display/python-wiring-and-setup

Pulsesensor + MCP3008 Library:
https://github.com/WorldFamousElectronics/Raspberry_Pi/blob/master/PulseSensor_Processing_Pi/PulseSensor_Processing_Pi.md

Thingspeak Library:
https://pypi.org/project/thingspeak/

Node.js Email Library:
https://github.com/AhadCove/Pi-SMS

XI. Appendix A:

Cover Page

ECE 492 Seamless Physiological Monitoring Proposal

Prepared for:

Professor Nathalia Peixoto

Prepared by:

Saad Abdullah
Moneeb Ahmad
Aayush Aryal
Priyam Das
Ryan Gunawan
Muhammed Jamil Rahman

I. Executive Summary

With the advancement of technology, the necessity for maintaining a stable psychological health has increased exponentially. One of the most common hobbies that the newer generation participates in is playing video games. Although indulging in a hobby is great, many of today's gamers are spending a huge chunk of their time gaming in order to entertain themselves. This comes at a price as spending a lot of time on gaming can cause potential health hazards. To mitigate these problems, we are developing a device that will help gamers monitor themselves both psychologically and physically while they are gaming. The device will record the user's stress level, heart rate, body temperature and wrist movement. The device will contain sensors as well as a microcontroller in order to monitor and analyze the health of the user and alert them if it notices any hazardous behavior. This data will also be seamlessly uploaded in a cloud based dashboard for off-site monitoring. The user can then go back and check what needs to be done so that the experience of gaming can be a pleasant one. The device will also include an LCD screen where a user can see the information of their health in real time.

II. Problem Statement

Motivations:

It is easy for someone to get hooked onto a certain activity or hobby that they participate in for entertainment or for stress alleviation. Gaming can be a fun and an addictive hobby, however, it can sometimes be hard for a gamer to know when to quit playing. The amount of time a gamer spends playing video games has steadily increased, from 5.1 h/week in 2011 to 6.5 h/week in 2017 (von der Heiden et al., 2019). This extended period of gaming can be detrimental to a gamers psychological well-being as well as causing issues with their health. In the DSM-5,

the American Psychiatric Association defined Internet Gaming Disorder with diagnostic criteria closely related to Gambling Disorder (von der Heiden et al., 2019). As for physical health, extended gaming can cause major repercussions such as Carpal Tunnel Syndrome (CTS). CTS is an instance of pain, numbness, and/or tingling that occurs as a result of this nerve being pinched, squeezed, or compressed (National Institute of Neurological Disorders and Stroke). Most often, CTS will arise due to a series of repetitive movements performed over a long period of time, for example, a gamer typing on a keyboard or button-mashing a controller. As gaming continues to grow in popularity these physical and psychological issues that are associated with long gaming sessions will continue to arise in frequency. In order to mitigate these issues we plan on creating a device that will be used to monitor a gamers psychological and physical health while they are gaming. If the device detects any health issues that are arising in the gamers, then the device will send an alert to the user letting them know to take a break from their gaming session. With the implementation of this device, a gamer can continue to enjoy their hobby while also retaining their psychological and physical health.

Identification of Need:

This Seamless Physiological Monitoring product will constantly monitor and measure a user's mental and physical health while they are gaming for an extended period of time. The device will record a user's stress level, heart rate, body temperature, movement and alert them if their levels are high. The data that is being recorded from a user will also be uploaded to a cloud-based dashboard for off-site monitoring. The data will also be displayed to the user through an LCD screen where they can cycle through their different measurements in real time.

Market/Application Review:

This Seamless Physiological Monitoring device will be marketed towards individuals who want to keep track of their mental and physical health while they game over an extended period. With the assistance of this device, an individual can be alerted when their stress and heart rate levels have risen over an extended period of time. The user can then determine whether they want to take a break from their gaming session. This device will also alert users if their wrist movement is sporadic, suggesting that they lower their wrist movement rate in order to avoid obtaining CTS. Also, this device will measure a user's body temperature and alert them if it goes over a certain threshold. Once the threshold is obtained the device will suggest the user to take a break from their gaming session as well as informing them to rehydrate. By taking constant breaks and rehydrating, a user can uphold their mental and physical health while they game over an extended period of time. Although there are many products that can measure a user's stress level and heart rate; there aren't many devices that are commonly known that can measure all the critreas listed above. By producing this product we are encouraging the gaming community to keep better track of their psychological and physical health while still enjoying their favorite pastime.

III. Approach

Problem Analysis

With an increase in weekly time gaming, a gamer can start to notice that their physical and psychological health are deteriorating over time. In order for them to continue enjoying their hobby we plan on creating a Physiological Monitoring Wrist Device that can measure their health while they are gaming. In order for the user to correctly use this device, they have to

power it up and put it on their wrist.. While the user is in their normal gaming session, the sensors on the device will measure the user's heartbeat, temperature, stress levels and wrist movements. The data measured is then sent back to the device and processed. The processed data is then displayed to the user as well as stored on a cloud based service like Azure.

Approach

The Physiological Monitoring Wrist Device that we are trying to build will require multiple sensors as well as a MCU in order to accurately monitor and record a user's heart rate, stress level, body temperature, and wrist movement. In order to properly measure a user's heart rate a Reflection-Type Pulse sensor will be placed on the bottom of the Physiological Device facing towards a user's wrist. The Reflection-type pulse sensors will emit an infrared light onto a user wrist and measure the amount of light reflected off it using a phototransistor. Oxygenated hemoglobin present in the blood of the arteries has the characteristic of absorbing incident light, so by sensing the blood flow rate that changes following heart contractions over time we are able to obtain a pulse wave signal which shows the users heart rate.

The Reflection-Type Pulse sensor will also be used in order to measure a user's stress level while they are gaming. In order to determine the stress level of the user we will use the Pulse sensor to calculate their Heart Rate Variability (HRV). The HRV is the key in determining the health of the cardiovascular system. A High deviation in a user's HRV is associated with better heart wellness and HRV could also be used as a means to determine moods (Farnsworth). A smaller HRV value could be associated with arousing psychological conditions such as stress (Kim, H., Cheon). A user's HRV can be calculated either in the time domain or the frequency domain. The time domain calculation uses the bpm data that was already taken, and the time difference between heartbeats. Using the formula highlighted in figure 1 the N value depends on

how many samples we take and j is the value we start with. RR is the distance (in milliseconds) between the heartbeats. When stressed, a user's HRV value becomes near 0, due to the heart rate surpassing their normal value (Hoffman [12]). If a user's HRV level is consistently low over an extended period of time the device will alert the user advising them to take a break from their gaming session in order to reduce their stress level.

$$HRV = STD\,RR = \sqrt{1/(N-1)\sum_{j=1}^{N}(RR_j - \overline{RR})^2}$$

Figure 1: The formula used for computing the standard deviation of RR intervals

In order to measure a user's wrist movements an accelerometer will be placed inside of the Physiological Device. The accelerometer will measure the wrist's x,y, and z positions. By finding the differences in position at one instance of time to another we can find velocity and the acceleration of a user's wrist. By obtaining the acceleration we can then see how fast a user is moving their wrist at any given moment. Once the accelerometer detects that a user has passed a certain acceleration rate the device will alert the user to slow down their wrist movement. With the deceleration of their wrist a user can lower their chances of obtaining CTS while gaming.

We will also be using a human body temperature sensor to accurately measure the body temperature of the user while they are gaming. Once their temperature reaches a certain threshold the device will alert the user of their temperature and advise them to take a break from gaming as well as reminding them to rehydrate. The sensor converts temperature measurements to digital form using a high-resolution, sigma-delta, analog-to-digital converter (ADC). The I2C serial interface accepts standard write byte, read byte, send byte, and receive byte commands to read the temperature data and configure the behavior of the open-drain over temperature

shutdown output. The sensor has a 2.7V to 3.3V supply voltage range, low 600µA supply current, and a lockup-protected I2C-compatible interface.

The sensors that are identified above will be connected to and controlled by an MSP430 microcontroller. This board was chosen because it contains an I2C bus which is necessary for the data collection which is obtained from the body temperature sensor as well as the accelerometer. This board is also inexpensive which is very crucial to our project since our budget is set at $600. Another reason this microcontroller was chosen was because it is easy to integrate an LCD screen onto the board which will display a user's health data while they are gaming. Users will have a button input on the device where they can switch between the different modes that displays their psychological and physical health while they are gaming. An Espressif Crowtail Serial WiFi module will also be connected to the MSP430 in order to provide the Physiological Monitoring Wrist Device with wifi capabilities. This is an important feature because the data that will be collected from the user by the device will be offloaded into a cloud based storage system. Storing a user's data in the cloud is a useful feature because it makes it easier for them to keep track and monitor their psychological and physical health over time. By looking at past data a user can determine if they need to take more breaks in their gaming sessions.

Alternative Approaches

The Physiological Monitoring Device that we are trying to build will be a wrist device; however, if the device is too bulky we will move the device up to the user's forearm in order to measure their psychological and physical health while they're gaming. Another alternative approach that we have prepared for is soldering a usb port onto the MSP430 in order to connect a Cudy AC Wifi USB which will provide our device with wifi capabilities. We also have a backup

plan where we will place a scroll wheel onto the device in order for a user to cycle through the different modes in order to see their health while gaming.

Background:

Stress level and emotions influence the activity of the heart. According to Hagshenag [2], the heart activity is influenced by ANS (Autonomic Nervous System) and SNS (Sympathetic Nervous System). Playing games influences the mental state which is enforced by Hagshenag in her research. Hagshenag [2] states, “In order to compare the three states; fear, excitement and calmness, it can be said that in the time of horror and excitement, with the increase of sympathetic nervous activity, the width of the Poincare plot decreases and with the increase of parasympathetic nervous activity in relaxation, the width of the Poincare chart is increased”. In other words, fear and excitement influence the heart rate because the signal that is sampled is fluctuating Hagshenag [2]. The way these emotions influence the heart rate is different. Panic increases the heart rate more than excitement and fear. These emotions increase the SNS and decrease the Lyapunov exponent [2]. In the study done by Porter and Goolkasian [2] it showed that players who felt they were in a threatened or challenging state showed more “sympathetic activity” , SNS, which means a higher heart rate, and a lower HRV value.

During a long gaming session, a user’s heart rate can increase for a long period of time causing them to sweat in order to dissipate some of the heat from their body temperature. With this increase of sweat a gamer can find themselves to be discombobulated due to the fact that they are dehydrated. When a user becomes severely dehydrated, their blood pressure can drop significantly due to them not receiving enough oxygen. In order for a user to avoid this issue our psychological device will read their surface body temperature and once it pases a certain threshold it will remind the user to take a break from their session and to rehydrate.

Project Requirements Specifications

Mission Requirements

- The device shall measure the physical and mental health of gamers.

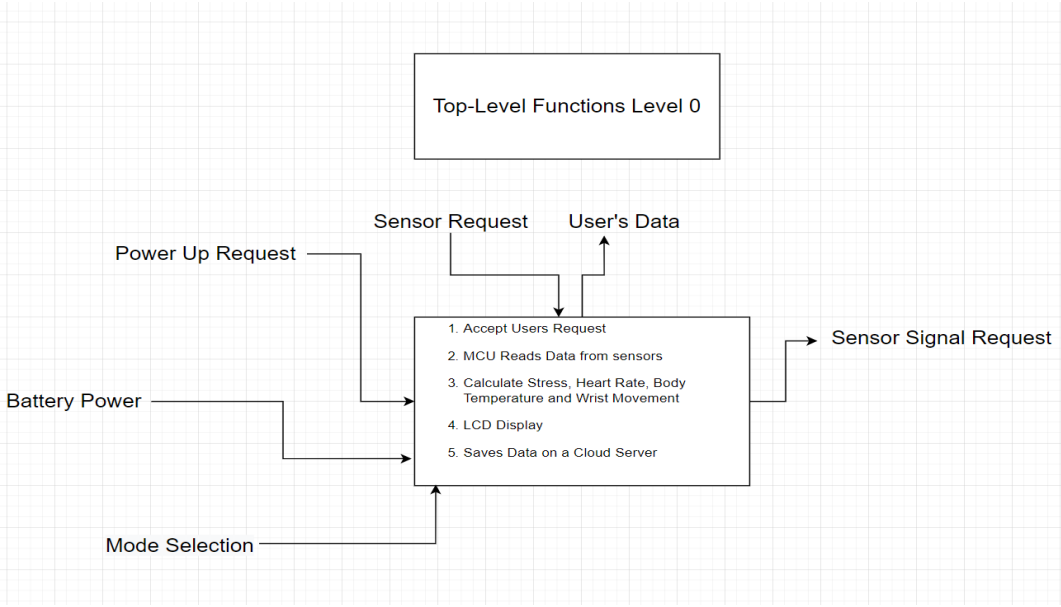
Operational Requirements

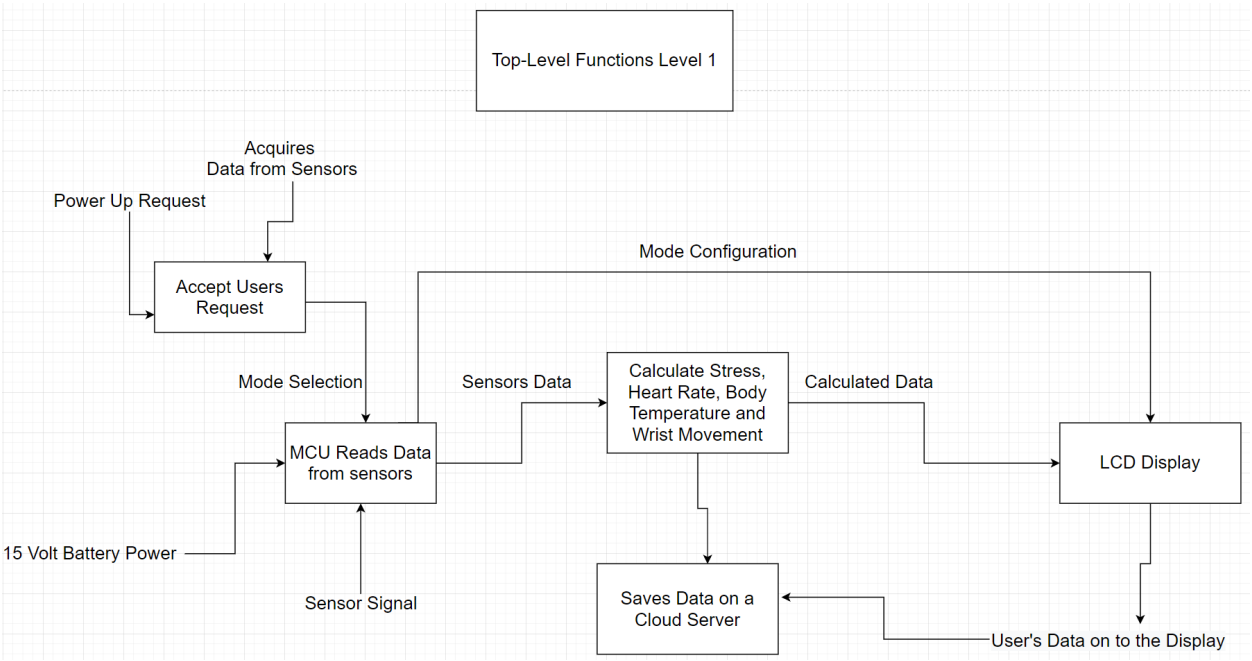
- Input/output requirements
    - The device will have multiple sensors on in order to calculate a user's symptoms and display it onto a simple LCD screen.
    - The device shall accept an input from a user through a push button in order to allow users to cycle between different monitoring sections on the LCD screen.
    - Data will be stored onto a cloud system.
- External Interface requirements
    - The deceive will be battery powered.
- Functional Requirements
    - The device will use the sensors to collect a user's symptoms every 10 milliseconds.
    - The device should detect errors and provide visual notification.
    - The device will alert the user if their stress level, heart rate, body temperature, or wrist movement is at a high rate for an extended period of time.
- Technology and System-wide requirements
    - The cloud storage system that will be used for this project is Azure.
    - The program languages that will be used for this project are C and Python.

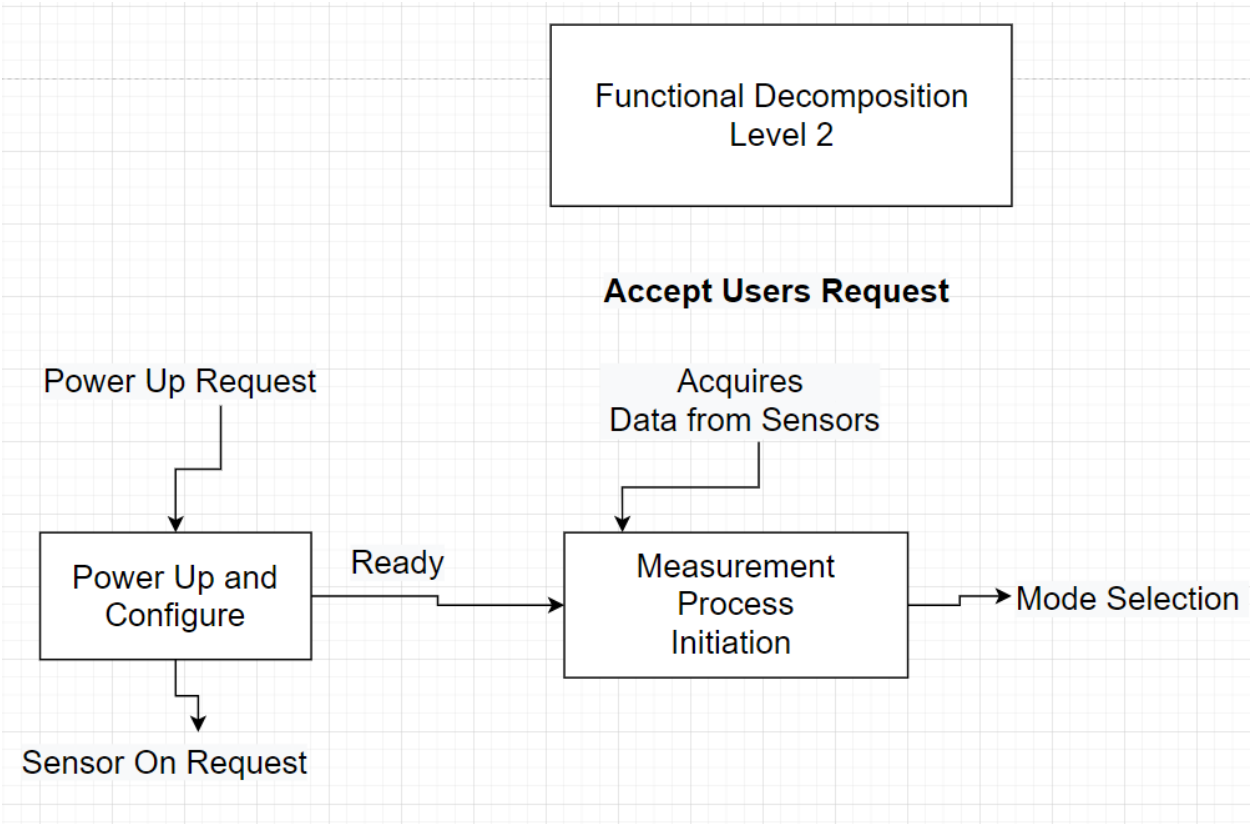
- The sensors that will be used in this project are: Reflection-Type Heart Sensor, Accelerator, Temperature Sensor.
- The Microcontroller that will be used for this project is a MSP430-EXP430G2ET.
- An LCD display as well as a wifi card will be used in this project.
- The buses that will be used in this project will be I2C and UART.
- The cost for this project will be less than $600
- The power requirements for this project will be less than 15 volts.

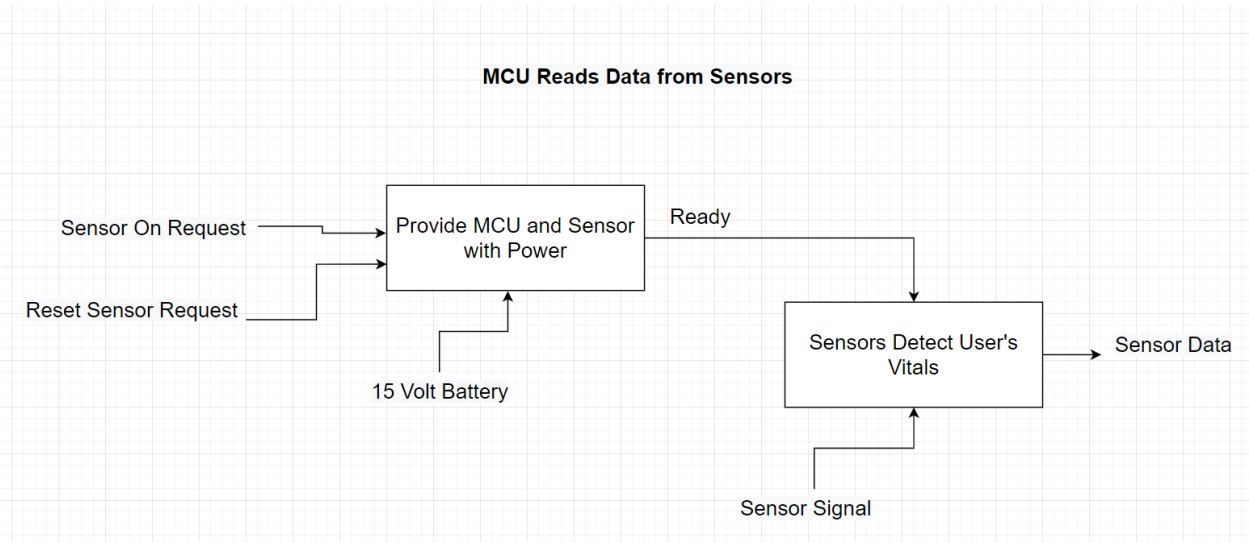
IV. System Design

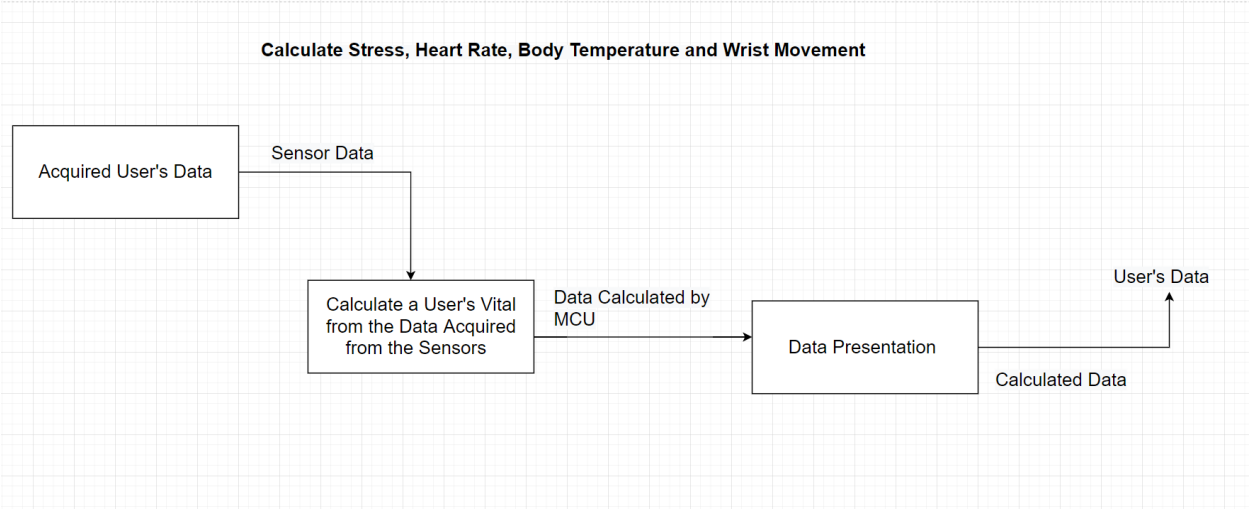
Functional Decomposition

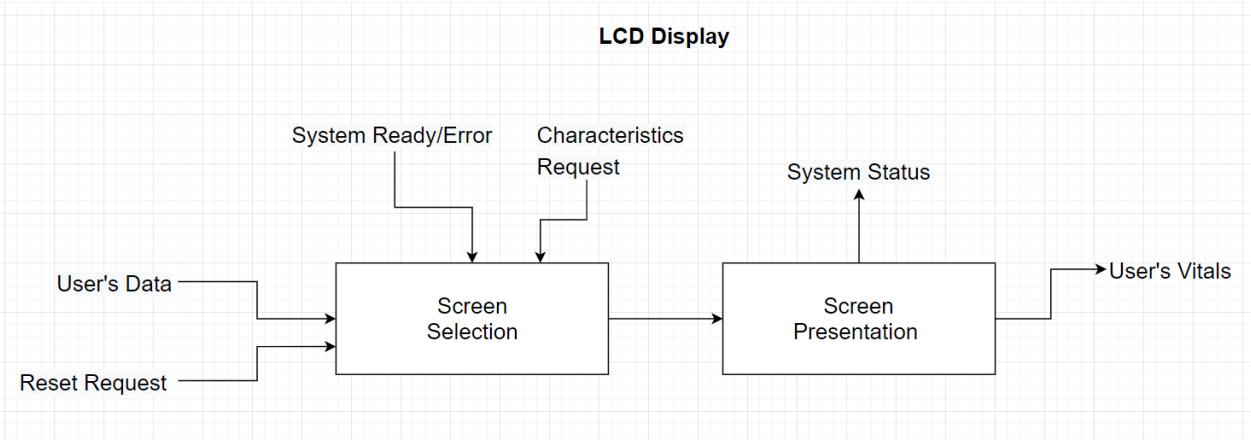


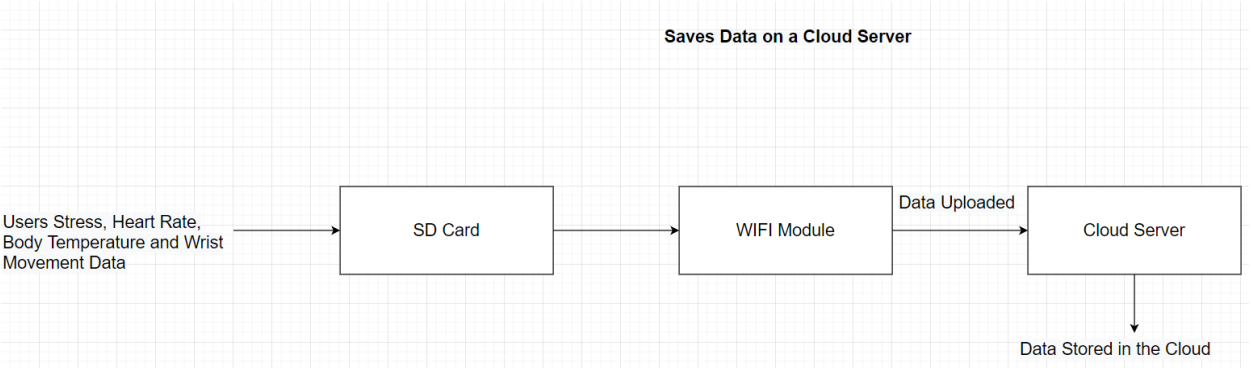

Top-Level Functions Level 1
Acquires
Data from Sensors
Power Up Request
Accept Users
Request
Mode Configuration
Mode Selection
Sensors Data
Calculate Stress,
Heart Rate, Body
Temperature and
Wrist Movement
Calculated Data
MCU Reads Data
from sensors
LCD Display
15 Volt Battery Power
Sensor Signal
Saves Data on a
Cloud Server
User's Data on to the Display


Functional Decomposition
Level 2
Accept Users Request
Power Up Request
Acquires
Data from Sensors
Power Up and
Configure
Ready
Measurement
Process
Initiation
Mode Selection
Sensor On Request

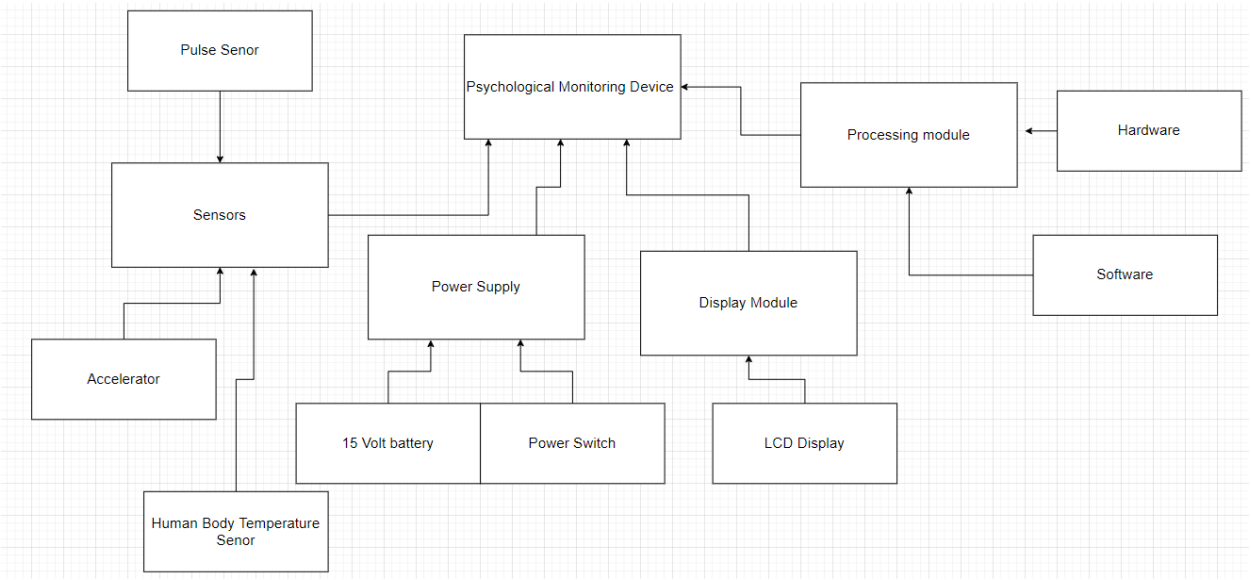

MCU Reads Data from Sensors
Sensor On Request
Reset Sensor Request
Provide MCU and Sensor with Power
15 Volt Battery
Ready
Sensors Detect User's Vitals
Sensor Data
Sensor Signal


Calculate Stress, Heart Rate, Body Temperature and Wrist Movement
Acquired User's Data
Sensor Data
Calculate a User's Vital from the Data Acquired from the Sensors
Data Calculated by MCU
Data Presentation
User's Data
Calculated Data

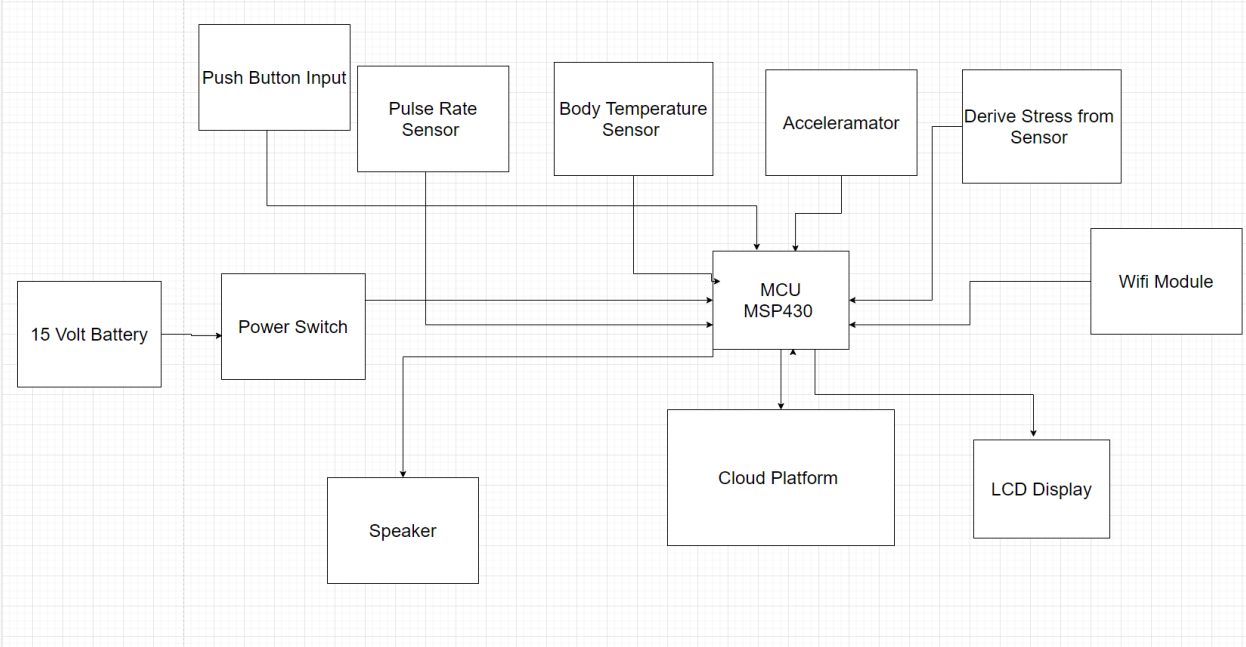




Generic Physical Architecture



System Architecture



V. Preliminary Experimental Plan

This project contains sensors, a power supply, an LCD display and microcontroller. There will be two experiments to test the readiness of the final product: operational requirement evaluation and functional requirement evaluation.

Experiment 1 (Operational Requirement Evaluation):

Goal: To evaluate a user's input and selection using a push button

System Components: LCD, Sensors and Push Button

Testing Process: A user can select through different modes on the Psychological device by clicking on a button in order to see their health data while gaming. The user will be able to click on the button multiple times in order to see the different reading that the device is displaying.

Evaluation: We will verify that once a user pushes the button the LCD on the device it has changed modes in a proper cycle.

Experiment 2 (Functional Requirement Evaluation):

Goal: To evaluate measurements of a user's: stress level, heart rate, body temperature, and wrist movement.

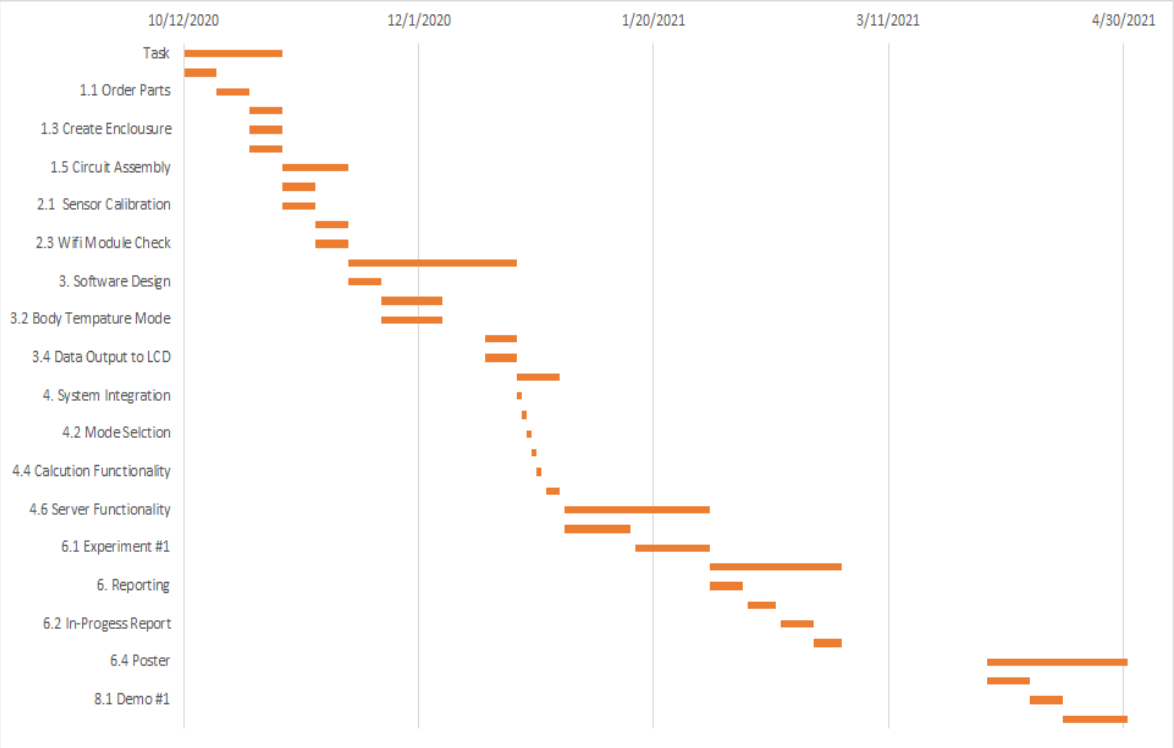
System Components: Sensors, Microcontroller.

Testing Process: We will be using the sensors that are encased in the device in order to measure a users physical and psychological health while they are gaming. The Microcontroller will convert the data that is collected from the sensors into readable data that the user can view.

Evaluation: We will verify that the data that is being collected from the sensor is accurate by using 3rd party health equipment (ie; apple watch) in order to measure a user's health and then compare it to our device reading. This test process will be conducted multiple times in order to assure that the Psychological device that we are creating is consistently outputting the correct information about a user's health.

VI. Preliminary Project Plan

Task	Start Date	End Date	Duration (weeks)
1. Hardware development	10/12/2020	11/2/2020	
1.1 Order Parts	10/12/2020	10/19/2020	1
1.2 Sensor Setup	10/19/2020	10/26/2020	1
1.3 Create Enclousure	10/26/2020	11/2/2020	1
1.4 Circuit Design	10/26/2020	11/2/2020	1
1.5 Circuit Assembly	10/26/2020	11/2/2020	1
2. Software Setup	11/2/2020	11/16/2020	2
2.1 Sensor Calibration	11/2/2020	11/9/2020	1
2.2 LCD Calibration	11/2/2020	11/9/2020	1
2.3 Wifi Module Check	11/9/2020	11/16/2020	1
2.4 Data Server Setup	11/9/2020	11/16/2020	1
3. Software Design	11/16/2020	12/22/2020	5
3.1 Heartrate + HRV Mode	11/16/2020	11/23/2020	1
3.2 Body Tempature Mode	11/23/2020	12/6/2020	1
3.3 Wrist Movement Mode	11/23/2020	12/6/2020	1
3.4 Data Output to LCD	12/15/2020	12/22/2020	1
3.5 Data Output to Server	12/15/2020	12/22/2020	1
4. System Integration	12/22/2020	12/31/2020	1
4.1 initialize Setup	12/22/2020	12/23/2020	1 Day
4.2 Mode Selction	12/23/2020	12/24/2020	1 Day
4.3 Sensor Functionality	12/24/2020	12/25/2020	1 Day
4.4 Calcution Functionality	12/25/2020	12/26/2020	1 Day
4.5 Display Functionality	12/26/2020	12/27/2020	1 Day
4.6 Server Functionality	12/28/2020	12/31/2020	3 Days
5. Testing	1/1/2021	2/1/2021	4
6.1 Experiment #1	1/1/2021	1/15/2021	2
6.2 Experiment #2	1/16/2021	2/1/2021	2
6. Reporting	2/1/2021	3/1/2021	4
6.1 Progess Report	2/1/2021	2/8/2021	1
6.2 In-Progess Report	2/9/2021	2/15/2021	1
6.3 Final Report	2/16/2021	2/23/2021	1
6.4 Poster	2/23/2021	3/1/2021	1
8. Milestones/Demos	4/1/2021	5/1/2021	4
8.1 Demo #1	4/1/2021	4/10/2021	1
8.2 Demo #2	4/10/2021	4/17/2021	1
8.3 Demo #3	4/17/2021	5/1/2021	2



VII. Potential Problems

This project will contain a collection of sensors that will be regulated by an MSP430 microcontroller. The data of the user's vital will be gathered by the sensors and the output readings would be processed in the microcontroller. A potential problem that could occur in this project would be issues that would arise during the programming session. Since we are using an MSP430 microcontroller the main programming languages that will be used in this project are going to be C and Python. We will be mostly relying on program libraries in order to control our microcontroller, however there is a good possibility that we will have to code a lot of the functions ourselves. Because of this there will be a lot of debugging that will need to be done in order for us to control our device the way we intend it to work.

Another potential problem that can occur during this project is issues arising from the wiring and sensor placement on the microcontroller. We have to determine the type of terminal we are using and integrate them inside the MSP430. When the sensors are integrated and the users data has been collected, we will also have to write an algorithm that can calculate the data and produce outputs using the appropriate method that we learned. The final potential problem that can occur during development of this project is that the connection between the sensors and the microcontroller would be terminated if the user's movement makes the sensors fall out of place.

We will need to find a way to properly ensure that all the parts of the device are properly secured in order to assure that this problem doesn’t occur.

VIII. References

[1] J. M. von der Heiden, B. Braun, K. W. Müller, and B. Egloff, “The Association Between Video Gaming and Psychological Functioning,” *Frontiers in psychology*, 26-Jul-2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6676913/.

[2] L. Haghshenas and M. Pooyan, "Investigating the Effect of Computer Games on the Heart Signal," 2019 27th Iranian Conference on Electrical Engineering (ICEE), Yazd, Iran, 2019, pp. 1775-1778, doi: 10.1109/IranianCEE.2019.8786686.

[3] “Heart Rate Variability - How to Analyze ECG Data,” *imotions*. https://imotions.com/blog/heart-rate-variability/.

[4] J. Hughes and F. Iida, “Multi-Functional Soft Strain Sensors for Wearable Physiological Monitoring,” *Sensors (Basel)*, vol. 18, no. 11, Nov. 2018, doi: 10.3390/s18113822.

[5]“Pulse sensor | Electronics Basics | ROHM.” https://www.rohm.com/electronics-basics/sensor/pulse-sensor#:~:text=First%2C%20there%20are%20four%20main,sensors%20use%20the%20photoelectric%20method.

[6]M. Salai, I. Vassányi, and I. Kósa, “Stress Detection Using Low Cost Heart Rate Sensors,” *Journal of Healthcare Engineering*, vol. 2016, p. 5136705, Jun. 2016, doi: 10.1155/2016/5136705.

[7] J. M. von der Heiden, B. Braun, K. W. Müller, and B. Egloff, “The Association Between Video Gaming and Psychological Functioning,” *Front Psychol*, vol. 10, Jul. 2019, doi: 10.3389/fpsyg.2019.01731.

[8] A. M. Porter and P. Goolkasian, “Video Games and Stress: How Stress Appraisals and Game Content Affect Cardiovascular and Emotion Outcomes,” *Front. Psychol.*, vol. 10, 2019, doi: 10.3389/fpsyg.2019.00967.

[9] B. Huang *et al.*, “Wearable Stretch Sensors for Motion Measurement of the Wrist Joint Based on Dielectric Elastomers,” *Sensors (Basel)*, vol. 17, no. 12, Nov. 2017, doi: 10.3390/s17122708.

[10] “Carpal Tunnel Syndrome Fact Sheet,” *National Institute of Neurological Disorders and Stroke*. [Online]. Available: https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Fact-Sheets/Carpal-Tunnel-Syndrome-Fact-Sheet. [Accessed: 02-Oct-2020].

[11] Kim, H., Cheon, E., Bai, D., Lee, Y. and Koo, B., 2018. Stress and Heart Rate Variability: A Meta-Analysis and Review of the Literature. *Psychiatry Investigation*, 15(3), pp.235-245.

[12] T. Hoffman Exercise Physiologist & Master Trainer, “What is Heart Rate Variability (HRV) & why does it matter?: Firstbeat Blog,” *Firstbeat*, 18-Aug-2020. [Online]. Available: https://www.firstbeat.com/en/blog/what-is-heart-rate-variability-hrv/. [Accessed: 02-Oct-2020].

XII. Appendix B

Healthy-Gamer Design Review

Prepared for:
Professor Nathalia Peixoto

Prepared by:
Saad Abdullah
Moneeb Ahmad
Aayush Aryal
Priyam Das
Ryan Gunawan
Muhammed Jamil Rahman

II. Summary of Motivation, Identification of Need and Requirements definition.

Motivations:

It is easy for someone to get hooked onto a certain activity or hobby that they participate in for entertainment or for stress alleviation. Gaming can be a fun and an addictive hobby, however, it can sometimes be hard for a gamer to know when to quit playing. The amount of time a gamer spends playing video games has steadily increased, from 5.1 h/week in 2011 to 6.5 h/week in 2019 [1]. This extended period of gaming can be detrimental to a gamers psychological well-being as well as causing issues with their health. In the DSM-5, the American Psychiatric Association defined Internet Gaming Disorder with diagnostic criteria closely related to Gambling Disorder [1]. As for physical health, extended gaming can cause major repercussions such as Carpal Tunnel Syndrome (CTS). CTS is an instance of pain, numbness, and/or tingling that occurs as a result of this nerve being pinched, squeezed, or compressed [2]. Most often, CTS will arise due to a series of repetitive movements performed over a long period of time, for example, a gamer typing on a keyboard or button-mashing a controller. As gaming continues to grow in popularity these physical and psychological issues that are associated with long gaming sessions will continue to arise in frequency. In order to mitigate these issues we plan on creating a device called the “Healthy-Gamer” that will be used to monitor a gamers psychological and physical health while they are gaming. If the Healthy-Gamer detects any health issues that are arising in the gamers, then the device will send an alert to the user letting them know to take a break from their gaming session. With the implementation of the Healthy-Gamer, a gamer can continue to enjoy their hobby while also retaining their psychological and physical health.

Identification of Need:

This Seamless Physiological Monitoring product will constantly monitor and measure a user’s mental and physical health while they are gaming for an extended period of time. The Healthy-Gamer will record a user’s stress level, heart rate, body temperature, movement and alert them if their levels are high. The data that is being recorded from a user will also be uploaded to a cloud-based dashboard for off-site monitoring. The data will also be displayed to the user through an LCD screen where they can cycle through their different measurements in real time.

Project Requirements Specifications

Mission Requirements

- The Healthy-Gamer shall measure the physical and mental health of gamers.

Operational Requirements

- Input/output requirements
    - The Healthy-Gamer will have multiple sensors on in order to calculate a user's symptoms and display it onto a simple LCD screen.
    - The Healthy-Gamer shall accept an input from a user through a push button in order to allow users to cycle between different monitoring sections on the LCD screen.
    - Data will be stored onto a cloud system.
- External Interface requirements
    - The deceive will be battery powered.
- Functional Requirements
    - The Healthy-Gamer will use the sensors to collect a user's symptoms every 10 milliseconds.
    - The Healthy-Gamer should detect errors and provide visual notification.
    - The Healthy-Gamer will alert the user if their stress level, heart rate, body temperature, or wrist movement is at a high rate for an extended period of time.
- Technology and System-wide requirements
    - The cloud storage system that will be used for this project is Azure.
    - The program languages that will be used for this project are C and Python.
    - The sensors that will be used in this project are: Reflection-Type Heart Sensor, Accelerator, Temperature Sensor.
    - The Microcontroller that will be used for this project is a Raspberry Pi Zero.An LCD display as well as a wifi card will be used in this project.
    - The buses that will be used in this project will be I2C and UART.
    - The cost for this project will be less than $600
    - The power requirements for this project will be less than 15 volts.

Background:

Stress Level:

- Stress can be calculated by finding the user's Heart Rate Variability
- HRV can be calculated using the formula:

$$HRV = STD\,RR = \sqrt{1/(N-1)\sum_{j=1}^{N}(RR_j - \overline{RR})^2}$$

- HRV is the standard deviation of the of all of the RR intervals which is the distance between each heartbeat

- The Variables in the formula itself are as follows:
    - N is the number of heartbeats that will be measured
    - RRj is the starting heartbeat the calculation will go from
    - RR (repeating) signifies that the heart will keep beating

Carpal Tunnel Syndrome:

- Correct Wrist Position and angle can help lessen the chance of developing CTS.
    - By using an accelerometer, it is possible to alert the user if their wrist is positioned at an angle that puts strain on the wrist
- To calculate Rotation Angle of the wrist the following equations are used for the x and y axis:

$$\alpha = tan^{-1}\left(\frac{A_y}{A^2{}_x + A^2{}_z}\right)$$
$$\beta = tan^{-1}\left(\frac{A_x}{A^2{}_y + A^2{}_z}\right)$$

- For the z axis the following equation is used:

$$F = \sqrt{X^2 + Y^2 + Z^2} = \sqrt{H^2 + Z^2}$$

III. System Design/ Architecture - System Decomp and Interfaces Functional Decomposition

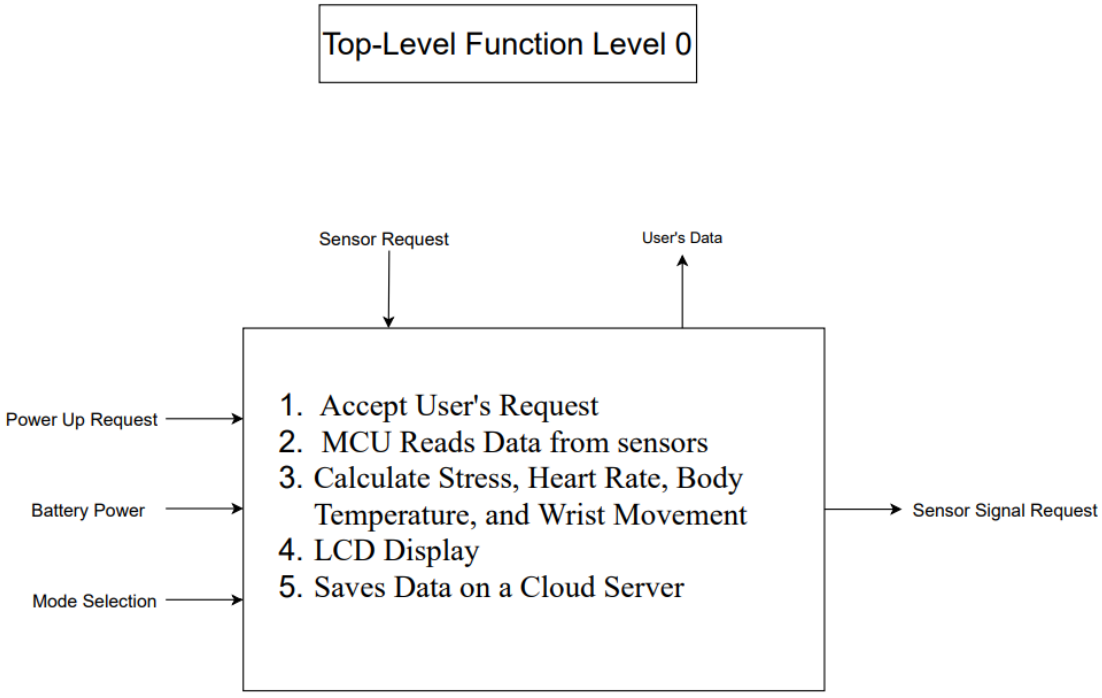


Figure 1: Top Level Function Level 0

This is level 0, a general overview of the system design/architecture of the physiological device. The inputs to the Healthy-Gamer are placed on the left and top side of the block diagram. Above it, we have the user input that is going to interact with the Healthy-Gamer. Users will be able to interact with the Healthy-Gamer through the click of a button. Then, the input will go through the device and be output as sensor signal request which is placed on the right side of the diagram. The main functions of the Healthy-Gamer are as follows: accepts user requests, has data reading capability, calculates stress, calculates heart rate, body temperature using sensors, and wrist movement using accelerometer.

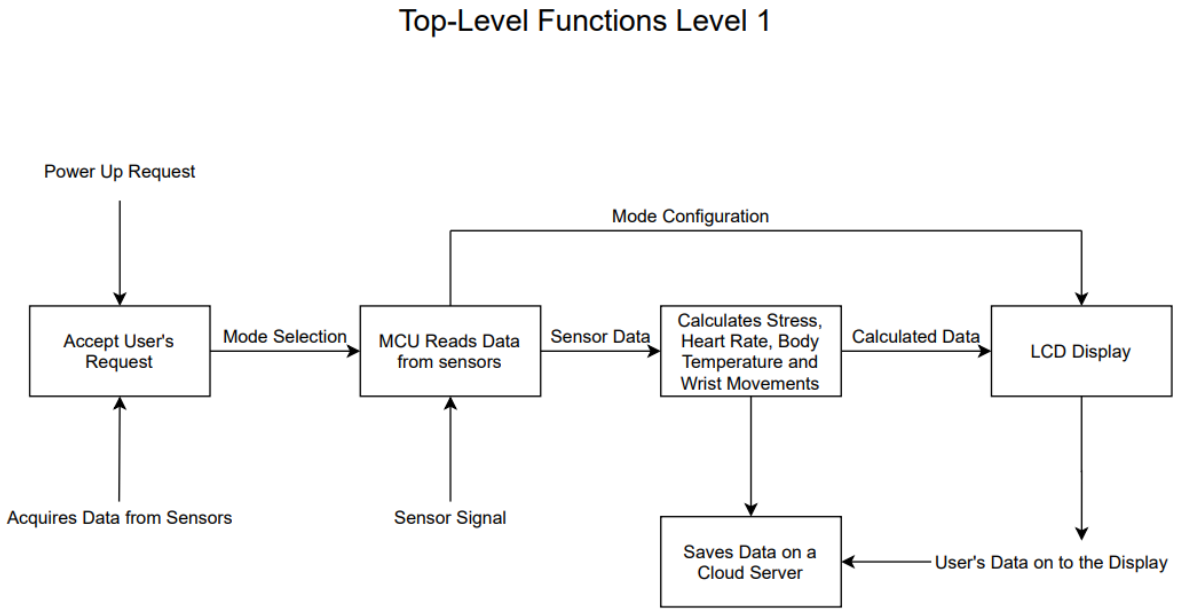


Figure 2: Top Level Function Level 1

A one step down from level 0 functional architecture is level 1 functional architecture. It goes into more detail regarding the top level functions and how they are integrated with each other. The above diagram illustrates how each element communicates with each other and shows the device's operation as it goes through its functions. On the top left corner, we have the device accepting the users' requests. Then, it sends that to the MCU which reads data from various sensors. The output from that then goes through the CPU of the device and various values are measured based on the inputs. After that, the calculated data goes to the LCD display for the user to see and take all necessary actions based on that. Along with sending the data to the LCD display, we also send the data to a cloud server where we will save it and the user can go through his/her data accumulated over the time.

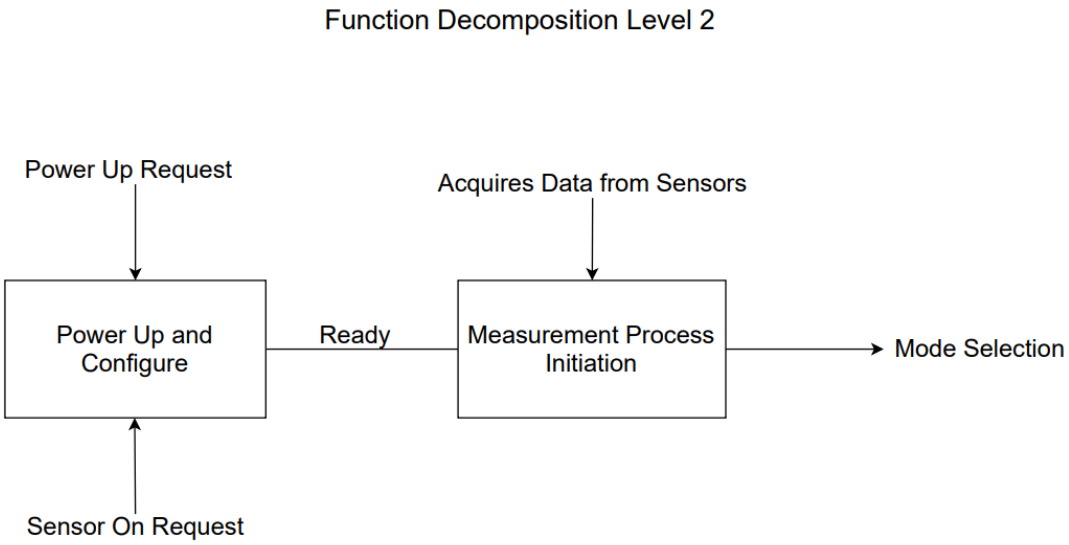


Figure 3: Function Decomposition Level 2

The user will need to power on the Healthy-Gamer in order for the device to start reading their vitals. Once the user turns on the Healthy-Gamer the sensors will begin to read the user's health information. The user will also be able to use the mode selection feature in order to cycle through their different health readings.

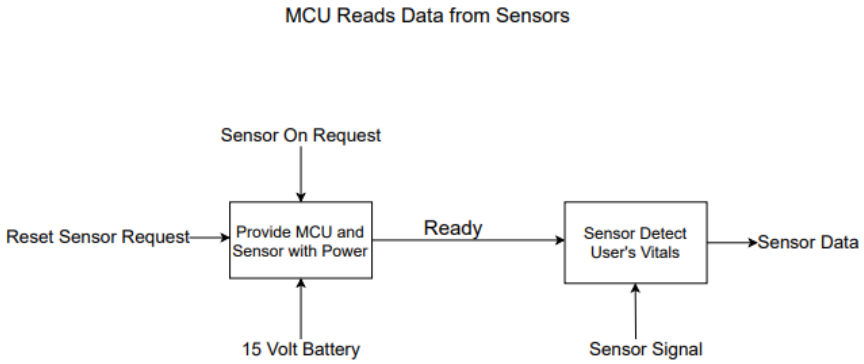


Figure 4: Reading Data from Sensors

The sensor within the Healthy-Gamer will send the user's data over to the MCU. The MCU will then begin to process the user's health information.

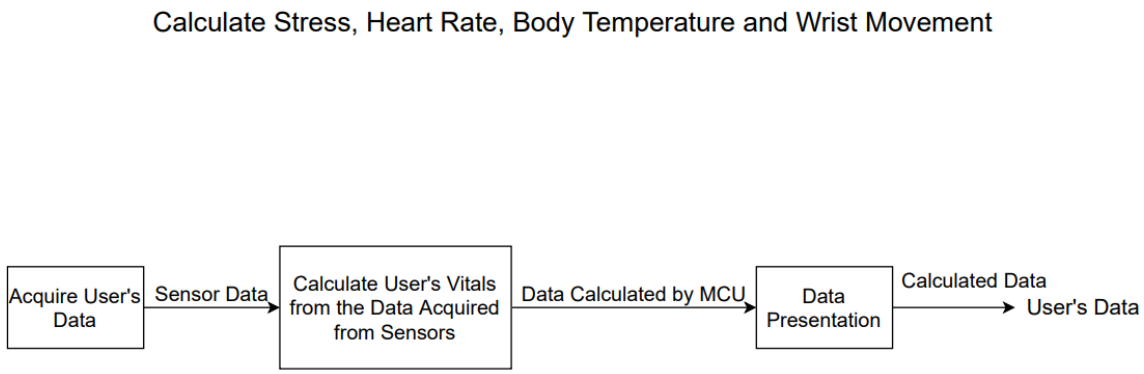


Figure 5: Calculations done by the MCU

The data collected from the user is sent to the Raspberry Pi Zero where data from different sensors such as the Heart Rate-Sensor and Accelerometer calculates the user's vitals. The Raspberry Pi will now begin to show the user's data.

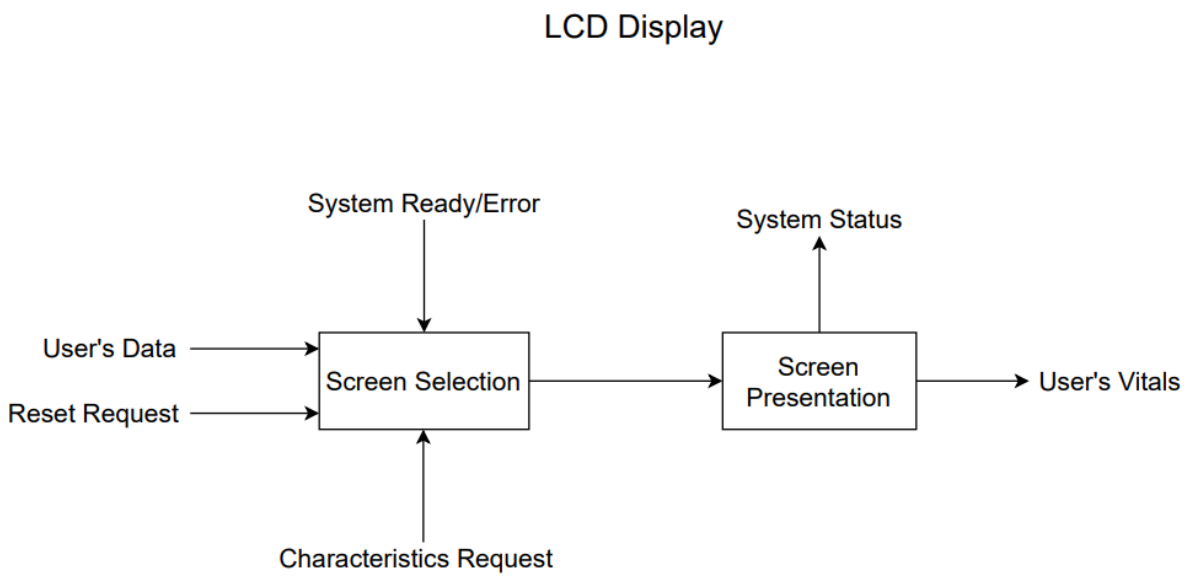


Figure 6: LCD Display

The user's Data is then sent to the LCD Display's screen selection allowing the user to choose what they would like to view. After a selection is made, the data collected from the sensor is shown to the user.

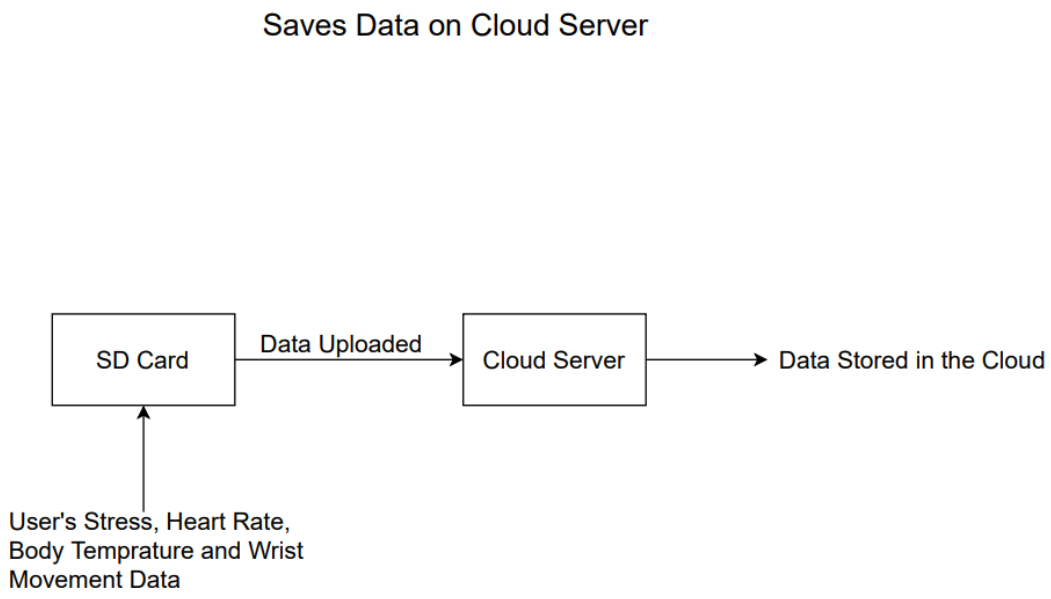


Figure 7: Saving Data on a Cloud Server

A copy of the data collected from the sensor is transported to the SD Card of the Raspberry Pi, it is then uploaded into the ThinkSpeak Cloud Server to be stored.

Generic Physical Architecture

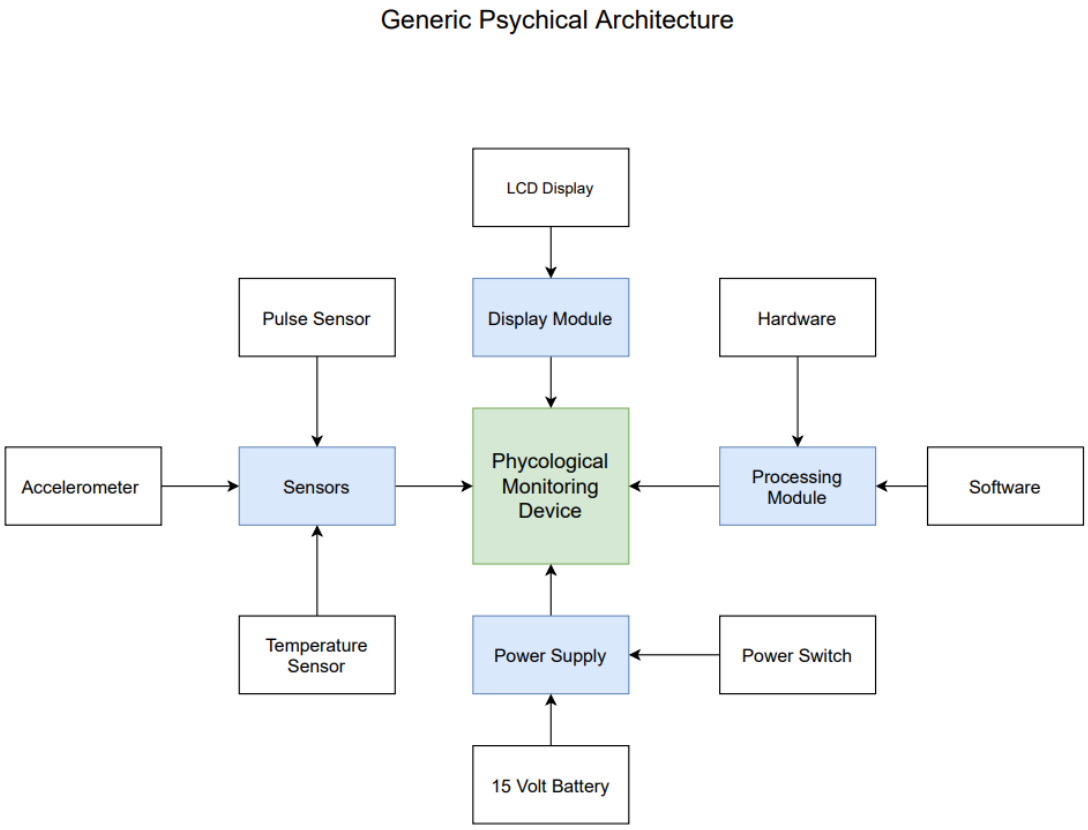


Figure 8: Generic Physical Architecture

The figure above represents the overall physical architecture for our health monitoring device. In the Healthy-Gamer the sensors that we will be using are going to be: an accelerator, heart pulse rate sensor, and thermal resistor sensor. The device will contain a 15 volt battery for its power supply. This battery will be able to provide power to our: sensors, LCD screen, and the MCU. The MCU will be responsible for interpreting the user's health information as well as uploading their information to the cloud.

System Architecture

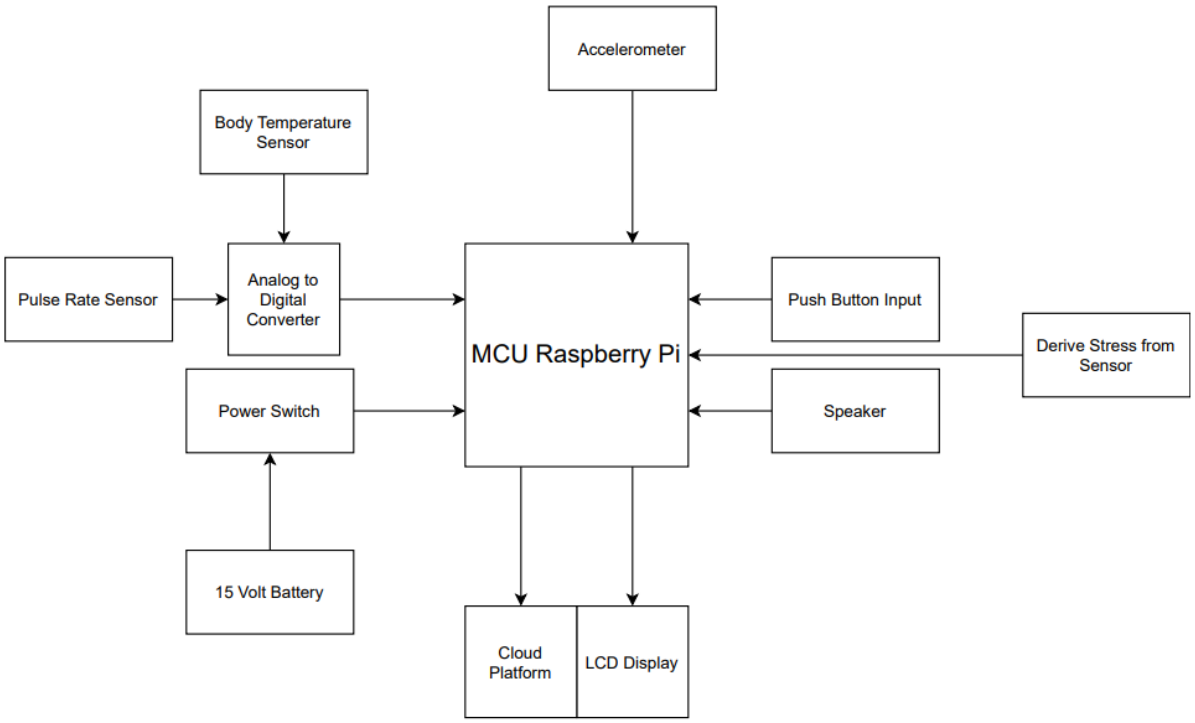


Figure 9: System Architecture

The figure above is a visual representation of all components that are either directly connected or is accessed by the Raspberry Pi. The Raspberry Pi will connect to all the sensors through the serial interface and receive data from the Analog-to-Digital Converter for both the Temperature and Heart Rate sensor. The LCD Display is connected to the Raspberry Pi directly through Serial Peripheral Interface (SPI). The Display will allow the user to view vital information about their health. Also, the Raspberry Pi will store all its data within an SD card - the data is then sent to the ThinkSpeak servers for secure storage.

IV. Detail Design - (a) Circuit Schematic level (b) flow diagrams with identifications of subroutines:

A.) Circuit Schematic Level

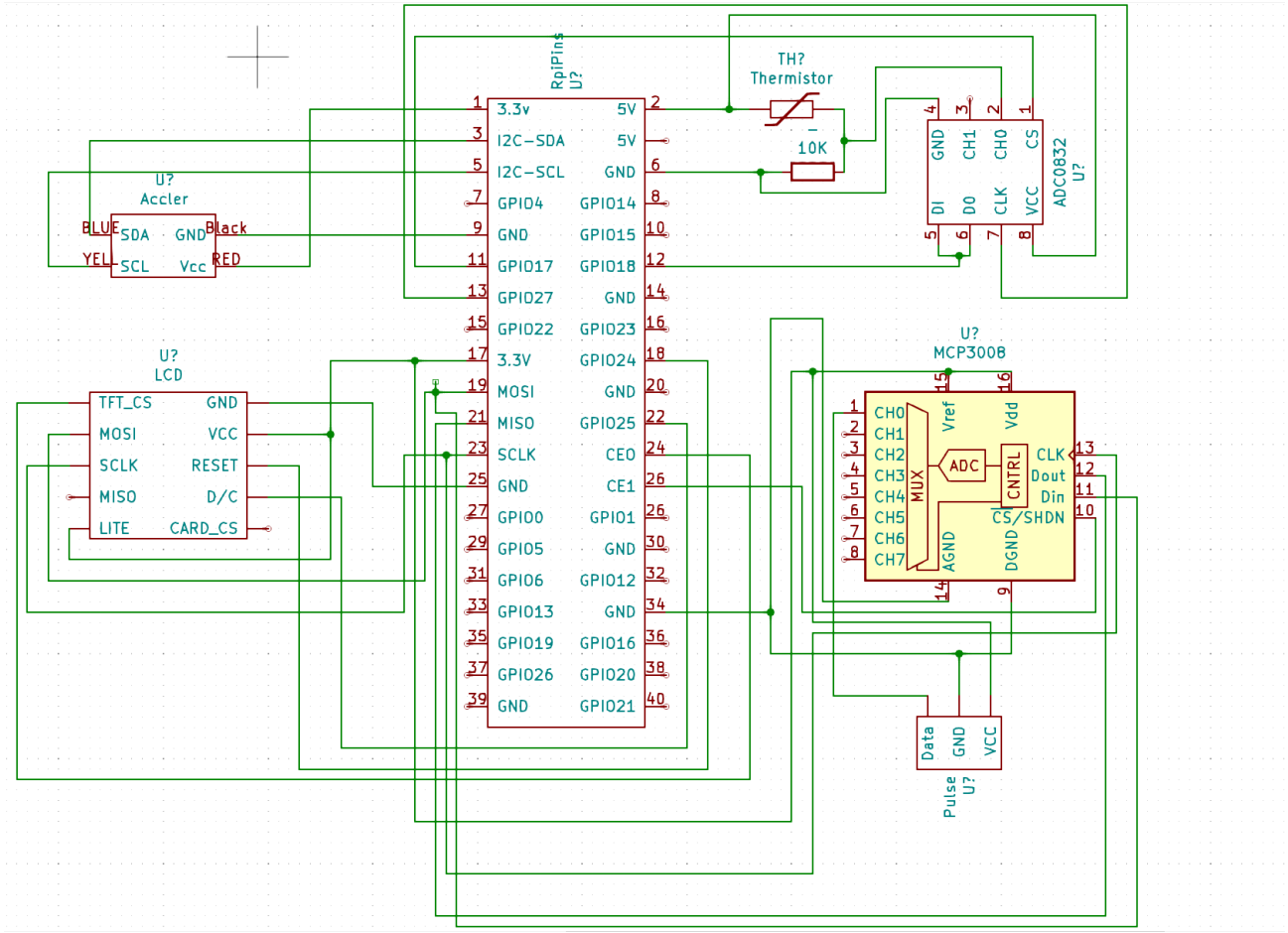


Figure 10: Circuit Schematic

The accelerometer is connected to the Raspberry Pi's SDA and SCL ports. This allows for the I2C protocol to be properly established. This allows synchronous communication between the Pi and the accelerometer . The thermistor is hooked up to an ADC and the ADC data is read from the Pi. The pulse sensor is also hooked up to an ADC which can allow the pulse sensor to speak to the SPI bus on the Pi. The LCD is also hooked up to the SPI bus.

B.) State Diagram

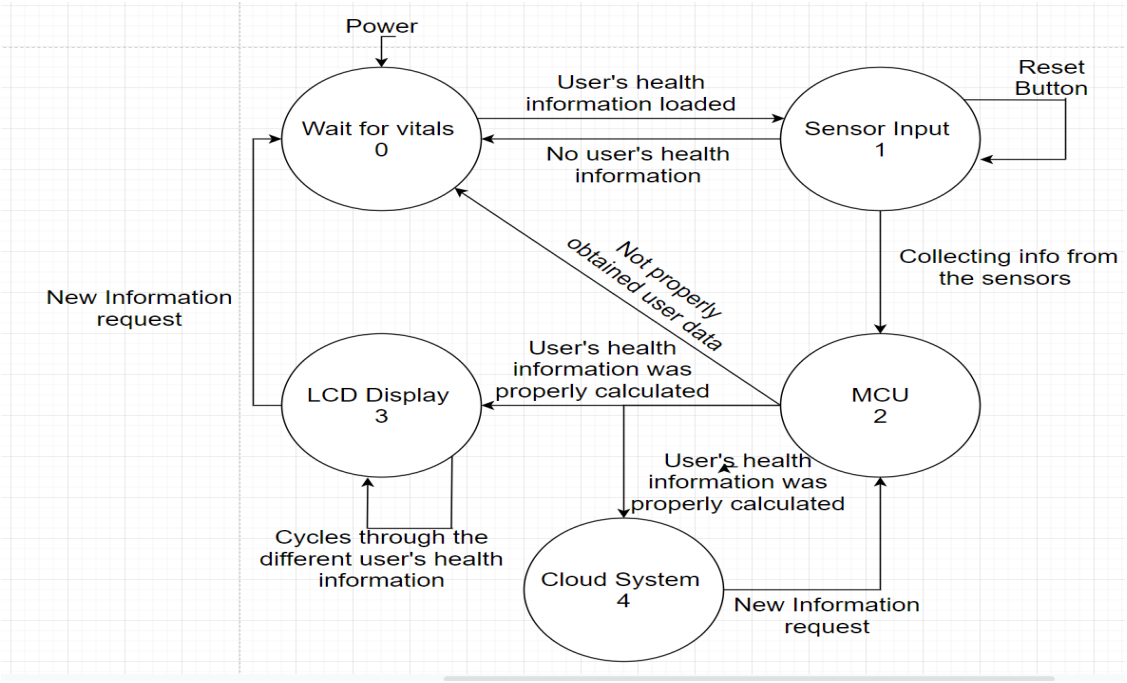


Figure 11: State Diagram

The figure above represents the state diagram for the Health Monitoring device. Once the Healthy-Gamer powers on it the software will request the sensors to read a user's health information. Once the sensors in the Healthy-Gamer collect the user's vitals it will send it over to the Raspberry PI where it will then begin to convert the data from the sensor into readable information. Once the Raspberry PI finishes converting the information received from the sensors, it will then send it to the LCD screen where it will display the user's health information. The Raspberry PI will also be responsible for uploading a user's health data into the cloud. The LCD will then alert the user if their health is in a critical state and advise them to take a break from their gaming section.

C.) Software Design

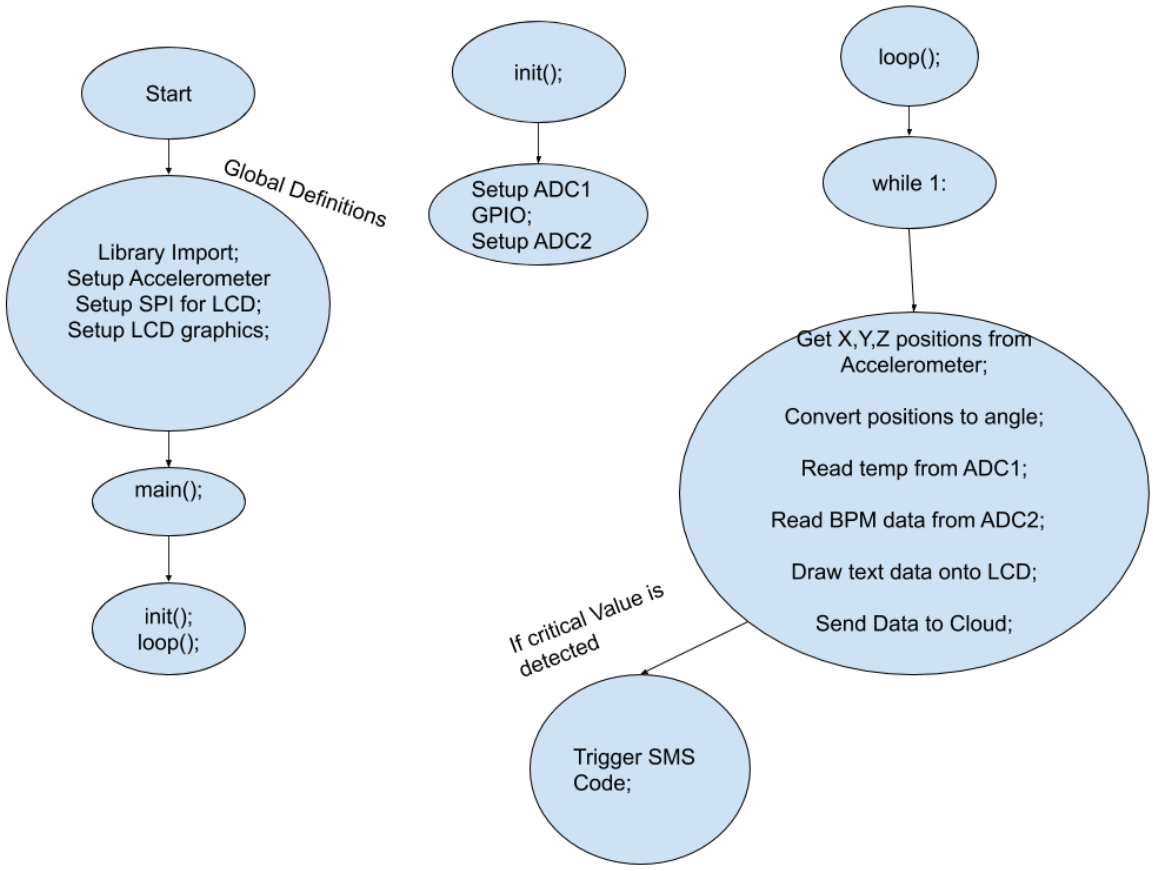


Figure 12: Software Design

The algorithm begins by setting up I2C for the accelerometer and SPI for the LCD. The LCD also needs some graphical data to set up ahead time so everything can be properly drawn on the display. Then the algorithm setups both ADCs to properly convert the thermistor data into a temperature value and to convert the pulse rate sensor into BPM. After that it enters the main loop where everything is done. First it obtains the x,y and z positions from the accelerometer and then converts it into a wrist angle. Then it reads the temp from one of the ADCs, and gets the BPM from the other ADC. Then this data is sent to the LCD and the Cloud. If any critical values are detected then this will trigger the SMS feature.

V. Prototyping progress report - what was built / experimented with, etc.

Prototyping is an important part of this project as it ensures that we have properly designed our portable health monitoring device to work as intended. Although having a slow start we have made a significant amount of progress in creating working portions in our health monitoring device. One of the requirements for this project that we have implemented so far is analyzing a user's body temperature. In order to analyze their temperature we used a thermal resistor which was paired up with an ADC chip which would then send the data of the sensor to the Raspberry Pi. Once the Raspberry Pi received the data we wrote a program that would analyze the data and convert it into a temperature reading. If the sensor detects that a user's body temperature is above 100°F the Health-Gamer will alert the user of their high body temperature and advise them to take a break from their gaming sesion. See figure 13 for a sample screenshot.

Another part of the device that we have successfully implemented is tracking a user's wrist movement using an accelerometer. In order to test our accelerometer we connected it to a breadboard which would then send the information to the Raspberry Pi via the I^2C protocol. The purpose of the accelerometer is to keep track of the user's wrist position and alert them if it is at a strenuous position for a certain period of time. As stated earlier the reason why we decided to focus on keeping track of the user's wrist position is to assure that they don't obtain CTS from their gaming session. See figure 13 for a sample screenshot.

A portion that we have successfully completed is configuring the LCD to display a user's health information. The screen was able to show both the user's wrist angle, heart rate and body temperature in real time. The screen will also alert the user's if their health is in a critical state and will advise them to take a break from gaming. See figure 14 for a sample screenshot.

We have also implemented the pulse rate sensor into our design so it can read a user's heart rate. We were able to design the code which enables the raspberry pi to interpret the information that is produced from the pulse rate sensor. As of right now the pulse rate sensor is able to read a user's heart rate at an accurate but inconsistent level. We believe that the issue arises from the positioning of the sensor. After doing more research we came to the conclusion that the heart rate sensor needs to be elevated slightly above the user's wrist so that way it can read the user's heart rate at a constant value. We plan on adding a thin layer of glass into our Healthy-Gamer device so that the pulse rate sensor can be elevated enough to read the user's heart rate at a constant value. See figure 15 for a sample screenshot.

We have also begun prototyping the different ways that we can offload the user's information to the cloud. We have decided to use ThingSpeak in order to upload all of the user's health information into the cloud. We have decided to go with this cloud platform because it doesn't require much processing power for the raspberry pi to offload the user's information into that

platform. We also plan on integrating different charts into ThingSpeak which will display a user's health information over time. The reason we plan on doing this is because it will group the user's health information in a neat and organized manner.

Another prototyping section that we have begun but have not yet completed is the pcb design and the 3D printed design for our project. We have created a schematic for the Healthy-Gamer device on KiCad which can be found in section IV figure 10. Once we do some more testing with our current prototype we will then be able to figure out how we can design our pcb so it can compactly fit all the electronic components into a small enclosed factor that can fit on a user's wrist. See figure 18 for a screenshot of the model.

The lastest section that we implemented is an SMS feature to alert the user of any critical values. This feature was successfully able to send a text message to a phone whenever the Node.js code was called. See section VI. Test Performed for a sample video this code being run.

VI. Tests Performed.

After finalizing all the designs, we moved on to perform some tests to verify that everything is working as expected. All the members were assigned to work on a particular part of the device and they performed their individual tests solo. After verifying that the individual parts were working, we then moved on to integrate all the parts and checked for the complete functionality of the device.

The first test that we performed measured values obtained by the accelerator from wrist movements, and by the temperature sensor. Then we sent those values to the monitor. The picture below demonstrates the output we obtained after moving our wrists haphazardly. The temp value shows the temperature in the room where we were performing the tests. Theta shows the angle of the wrist. As the device was just laying flat on the table, it was giving us a value close to neutral or 0. As we moved the device around, theta values kept changing however, the temp value stayed the same.

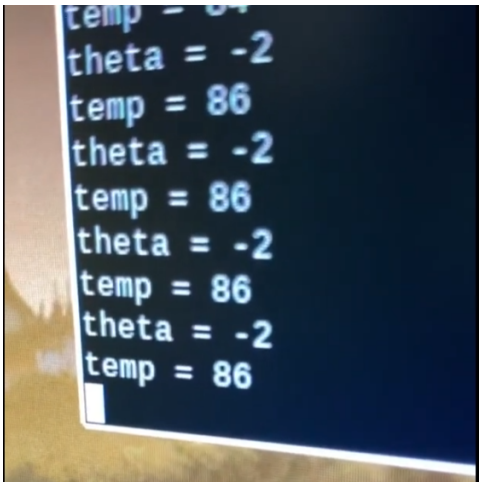


Figure 13: Temperature and Angle Readings

In the second test we performed, we wanted to see whether we could output the value derived from all the sensors to the LCD display. In the first test, we had confirmed that the sensors were working as expected. Now, we wanted to make sure that the LCD display was working in integration with other parts of the device. The figure below was taken during the 2nd test and the video is included in the prototyping video.

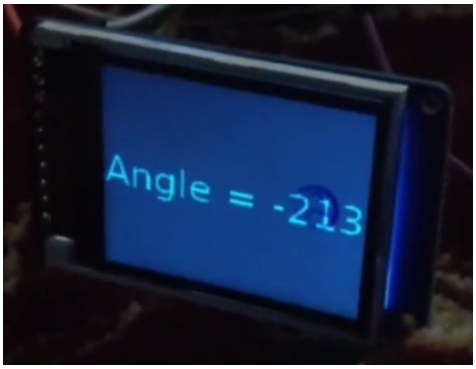


Fig. 14. Screenshot the LCD working and displaying a wrist Angle.

Here is a link to our initial prototyping progress video detailing these early efforts
https://www.youtube.com/watch?v=kHXUFNpBC6k&ab_channel=MoneebAhmad

The figure below shows the heart rate results that are obtained from the pulse rate sensor. If the sensor detects that a user's BPM is above 100 it will send an alert to the user letting them know to take a break from their gaming section.

Fig. 15. Sample output for pulse sensor.

Link to prototyping video:
https://www.youtube.com/watch?v=vG8EGvoUSFI&ab_channel=MoneebAhmad

In our 4th test we wanted to be sure that our SMS code could properly send a message to one of our phones. A screenshot cannot properly convey how the SMS feature works so the below link is a demo video for it.
https://www.youtube.com/watch?v=qaZ_8nI56wY&ab_channel=MoneebAhmad

VII. Experimentation plan.

This project contains sensors, a power supply, an LCD display and microcontroller. There will be two experiments to test the readiness of the final product: operational requirement evaluation and functional requirement evaluation.

Experiment 1 (Operational Requirement Evaluation):

Goal: To evaluate a user's input and selection using a push button

System Components: LCD, Sensors and Push Button

Testing Process: A user can select through different modes on the Psychological device by clicking on a button in order to see their health data while gaming. The user will be able to click on the button multiple times in order to see the different reading that the device is displaying.

Evaluation: We will verify that once a user pushes the button the LCD on the device it has changed modes in a proper cycle.

Experiment 2 (Functional Requirement Evaluation)

Goal: To evaluate measurements of a user's: stress level, heart rate, body temperature, and wrist movement.
System Components: Sensors, Microcontroller.

Testing Process: We will be using the sensors that are encased in the device in order to measure a users physical and psychological health while they are gaming. The Microcontroller will convert the data that is collected from the sensors into readable data that the user can view.

Evaluation: We will verify that the data that is being collected from the sensor is accurate by using 3rd party health equipment (ie; apple watch) in order to measure a user's health and then compare it to our device reading. This test process will be conducted multiple times in order to assure that the Psychological device that we are creating is consistently outputting the correct information about a user's health.

VIII. List of tasks for 493 and their decompositions into subtasks plus individual responsibility:

- **PCB design:** We have used KiCad to design the schematic we will be implementing on a PCB board for our project. We will be hierarchical schematics so that it will enforce organization to our schematics and help us guide through the designing process. To make it aesthetically pleasing, we are going to make it as neat and clean as possible so that whoever tries to follow the pcb design has a good grip on how to integrate individual devices without confusion. Before placing the sensors in the PCB board, we will verify our circuit board layout by running a design rule check (DRC). Another goal for this part of the project is to make design as small as possible. This will help us fit everything into a compact, light-weight watch which will be feasible to wear anywhere.

- **Cloud server:** In order to store the data we collect from our device, we will use ThingSpeak as a cloud server. ThingSpeak is an open IoT platform for monitoring data online. In ThingSpeak channel we can set the data as private or public according to our choice. ThingSpeak generally takes 15 seconds to update the readings [3]. After creating an account, we will open our own channels where the uploaded data will be visible. Each channel will contain different data collected from the sensors. For example, one channel will contain the body temperature information and variation. Another channel will store the body movement information. Our unique code will calculate the stress level from the

sensors and will upload it simultaneously. The users will have access to this information anytime they want to check for it.

- **3d printed case -** after we are done with our exhaustive testing, we will be using a 3D printer to create our own case that will hold our device. We have already designed our case using “Blender” and created the STL file [4]. It will take approximately 3 days to form the 3D printer to build the design.

- **Testing -** We will be conducting extensive testing on our device to make sure the device is performing correctly and efficiently. We will make sure all the sensors are working properly and the connections are in correct order as well as safe. We will monitor how effectively the device is uploading data to the cloud server and if the data sent is accurate. In addition, we will be testing the device on different types of users from different age categories to make sure the device functions properly and is able to differentiate between different users. We will be testing the integrity of the device in various circumstances to see if the device is securely able to stay on the user’s body as well as perform the tasks correctly.

Moneeb is in charge of integrating the individual sensors and devices with the Pi Zero. He was in charge of presenting the State Diagram for the video, and then for our second video, the Milestones section too. He is also in charge of taking the ADC value from the thermistor and turning it into a proper temperature. He will be spending his time debugging and implementing codes using the Pi as it has a co-math processor which can handle floating point with square roots and tangents efficiently. He is implementing the code for the circuit and will be working on rigorous testing so achieve maximum efficiency. Moneeb is also building a SMS notification feature for the system which will allow the Raspberry Pi to send a text message alert whenever a critical value is detected on the system which will help users prevent potential dangers.

Saad is in charge of creating the Cloud server and making sure that the data that has been obtained from the sensors will be able to store it in the server. He has also worked on integrating the heart rate sensor into the Healthy-Gamer device. The goal of the heart rate sensor is to keep track of the user’s heart rate and alert them if it is at a high level over an extended period of time. He has already created the server and made sure the heart sensor works properly. He will be testing the sensor rigorously and create an algorithm so that it can detect any irregularities in the heart which will alert the user.

Jamil and Priyam were assigned to do the software and hardware design, experimentation plan, and Gantt chart. They worked on integrating the LCD with the device and making sure it is displaying the output of the acquired values from the sensors. Priyam has built a 3D model for the physiological device using Blender. They will make sure the device is light and designed

efficiently so that it does not put the user into any discomfort. They will be testing the device thoroughly and make sure the device provides accurate and updated data all the time.

Aayush is in charge of the experimentation plan and to measure stress levels by using his research on Carpal Tunnel Syndrome as well as analyzing Heart Rate Variability. He has also made sure the accelerometer is working accurately and will test the device so that it provides accurate measurement of the position of the user. He is working on trying to make the sensors give real time values on graphs to allow the users understand their data. He will analyze the accelerometer data to determine the range of motion of the user and detect what is a fall or normal user motion. This will minimize the occurrence of any false alerts when the user is active.

Ryan is in charge of prototyping and the prototyping board that will be used to mount the raspberry pi zero. The device will have three layers of PCB. The first layer is the pulse sensor, accelerometer, battery, and raspberry pi mounting hole. The second layer would be the raspberry pi zero. The third layer (upper layer) would be LCD. He will be making sure the device is intact and the connections are well prepared so that any movement of the device and the user does not interfere with the connections and reading of data.

IX. Schedule and Milestones.

Task	Start Date	End Date		Duration(days)
1. System Integration	11/16/2020	12/22/2020		36
1.1 Initialize setup	11/16/2020	11/22/2020		6
1.2 Mode Selection	11/22/2020	11/28/2020		6
1.3 Sensor Functionality	11/28/2020	12/4/2020		6
1.4 Calculation functionality	12/4/2020	12/10/2020		6
1.5 Display Functionality	12/10/2020	12/16/2020		6
1.6 Server Functionality	12/16/2020	12/18/2020		2
2. PCB Design	12/18/2020	12/20/2020		2
3. 3D printed case	12/20/2020	12/24/2020		4
4. Testing	12/24/2020	2/1/2021		39
4.1 Experiment 1	12/25/2020	1/15/2021		21
4.2 Experiemnt 2	1/15/2021	2/1/2021		17
5. Reporting	2/1/2021	3/1/2021		28
5.1 Progress Report	2/1/2021	2/10/2021		9
5.2 In-progress report	2/10/2021	2/20/2021		10
5.3 Final Report	2/20/2021	3/1/2021		9
6. Milestones/Demos	3/1/2021	5/1/2021		61
6.1 Demo#1	3/1/2021	3/20/2021		19
6.2 Demo#2	3/20/2021	4/10/2021		21
6.3 Demo#3	4/10/2021	5/1/2021		21

Fig. 16. Schedule of Tasks for ECE 493

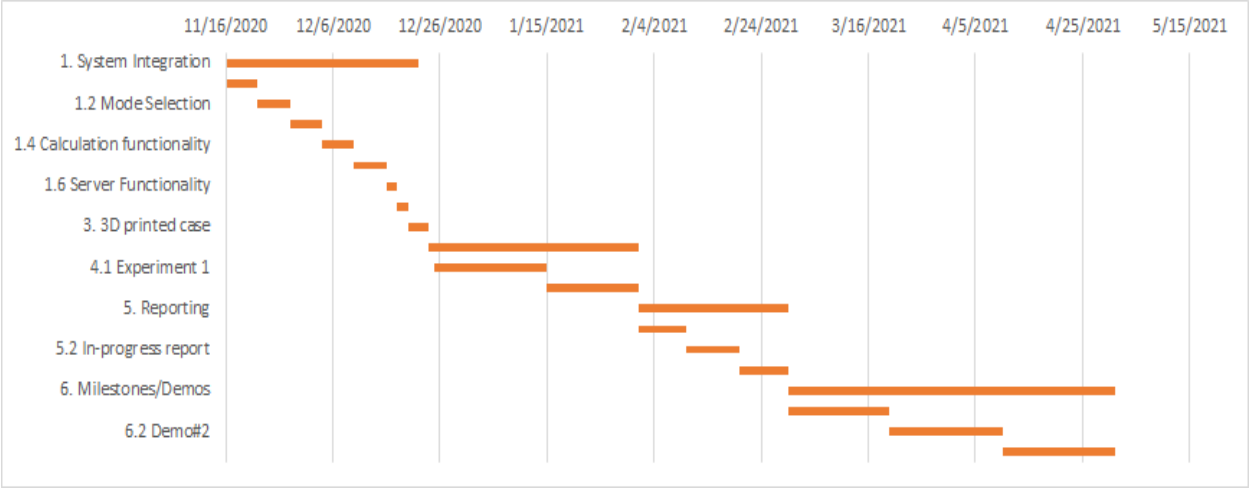


Fig. 17. Gantt Chart for ECE 493.

X. Summary of the project.

For our project we wanted to create a health device that was designed specifically for gamers. The device would be able to measure a user's body temperature, wrist angle, heart rate and stress level. Body temperature would be monitored to ensure that a user is staying well hydrated. Wrist angle check for and to prevent early signs of CTS. Heart rate and stress level would be used to advise the user to take a break from. The device would be able to alert the user of this important vital information through an LCD screen as well through a text message alert. A user would also be able to access their information through an off-site server, which they can review at a later time.

XI. Links to important files:

Design Review Presentation:

https://www.youtube.com/watch?v=ZtVC0dlqUE4&ab_channel=MoneebAhmad

Early Prototyping Video:

https://www.youtube.com/watch?v=kHXUFNpBC6k&ab_channel=MoneebAhmad

Heart Rate Prototyping

https://www.youtube.com/watch?v=vG8EGvoUSFI&ab_channel=MoneebAhmad

SMS Prototyping

https://www.youtube.com/watch?v=qaZ_8nI56wY&ab_channel=MoneebAhmad

3D model:

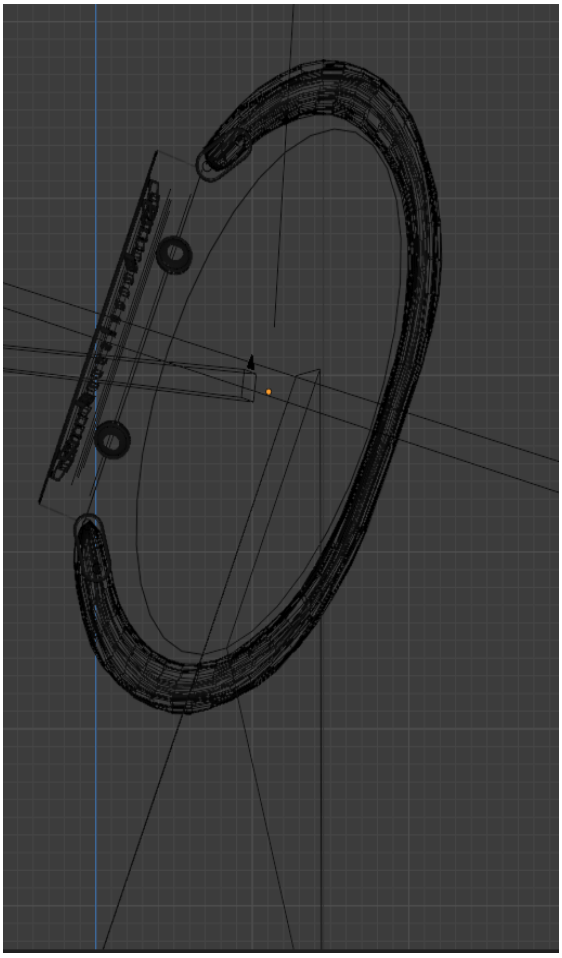


Figure 18: 3D Model of the Device

Link to .stl file:
https://drive.google.com/file/d/1yPVlIbf0Pa-xF3l3GcpZ_00di7R4Yaoi/view?usp=sharing

XII. References:

[1] J. M. von der Heiden, B. Braun, K. W. Müller, and B. Egloff, “The association between video gaming and psychological functioning,” *Frontiers in psychology*. Jul. 2019, DOI: 10.3389/fpsyg.2019.01731.

[2] “Carpal Tunnel Syndrome Fact Sheet,” *National Institute of Neurological Disorders and Stroke*. Apr. 2020, [Online]. Available: https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Fact-Sheets/Carpal-Tunnel-Syndrome-Fact-Sheet

[3] “How to Send Data to ThingSpeak Cloud using Raspberry Pi.” Jan. 2019. https://iotdesignpro.com/projects/how-to-send-data-to-thingspeak-cloud-using-raspberry-pi .

[4] Dibya Chakravorty, “STL File Format (3D Printing) – Simply Explained,” All3DP, Jun. 13, 2017. https://all3dp.com/what-is-stl-file-format-extension-3d-printing/.

[5] L. Haghshenas and M. Pooyan, "Investigating the Effect of Computer Games on the Heart Signal," 2019 27th Iranian Conference on Electrical Engineering (ICEE), Yazd, Iran, 2019, pp. 1775-1778, doi: 10.1109/IranianCEE.2019.8786686.

[6] H.-G. Kim, E.-J. Cheon, D.-S. Bai, Y. H. Lee, and B.-H. Koo, “Stress and Heart Rate Variability: A Meta-Analysis and Review of the Literature,” Psychiatry Investigation, vol. 15, no. 3, pp. 235–245, Mar. 2018, doi: 10.30773/pi.2017.08.17.

[7] Abyarjoo, Fatemeh & Barreto, Armando & Abyarjoo, Somayeh & Ortega, Francisco & Cofino, Jonathan. (2013). Monitoring Human Wrist Rotation in Three Degrees of Freedom. Conference Proceedings - IEEE SOUTHEASTCON. 10.1109/SECON.2013.6567517.

[8] M. Bahrudin, “Carpal Tunnel Syndrome (CTS),” Saintika Medika, vol. 7, no. 1, Oct. 2012, doi: 10.22219/sm.v7i1.1090.

XIII. Appendix C:

Main.py

```
#obtained from
https://github.com/adeept/Adeept_Ultimate_Starter_Kit_Python_Code_for_RPi/b
lob/master/ADC0832.py
import ADC0832
#std library
import time
#obtained from https://pypi.org/project/msa301/
import msa301
#std library
import math
#std library
import board
#std library
import subprocess
#obtained from https://pypi.org/project/adafruit-circuitpython-mcp3xxx/
import digitalio
#obtained from
https://learn.adafruit.com/1-8-tft-display/python-wiring-and-setup
from PIL import Image, ImageDraw, ImageFont
#obtained from
https://learn.adafruit.com/1-8-tft-display/python-wiring-and-setup
import adafruit_rgb_display.st7735 as st7735
#obtained from
https://github.com/WorldFamousElectronics/Raspberry_Pi/blob/master/PulseSen
sor_Processing_Pi/PulseSensor_Processing_Pi.md
from pulsesensor import Pulsesensor
#std library
import sys
#obtained from https://pypi.org/project/thingspeak/
import thingspeak
#std library
import os


bpm_channel_id = 1280632 # BPM CHANNEL ID
bpm_write_key  = 'ME9U0JXEMH8AECDG' # BPM WRITE KEY
bpm_read_key   = 'Y4C1ID3DPQMMCFBG' # BPM YOUR READ KEY

acc_channel_id = 1313730 # Accelerometer CHANNEL ID
acc_write_key  = 'ML8HKXWFO7W09PX4' # Accelerometer YOUR WRITE KEY
```

```
acc_read_key   = 'HDZXREAGRUOCNOUE' # Accelerometer YOUR READ KEY

hrv_channel_id = 1313732 # HRV CHANNEL ID HERE
hrv_write_key  = 'HEGT9HXJLJS92FWH' # HRV YOUR WRITE KEY HERE
hrv_read_key   = 'GVBMMMHV4FK3JY99' # HRV YOUR READ KEY HERE

temp_channel_id = 1313731 # TEMP CHANNEL ID HERE
temp_write_key  = 'K6I61OT7SH3KUCYG' # HRV YOUR WRITE KEY HERE
temp_read_key   = '83EJJWQTK873JHRA' # HRV YOUR READ KEY HERE

acc_sms = 'cd /home/pi/Desktop ; node main_acc.js'
hrv_sms = 'cd /home/pi/Desktop ; node main_hrv.js'
bpm_sms = 'cd /home/pi/Desktop ; node main_bpm.js'
temp_sms = 'cd /home/pi/Desktop ; node main_temp.js'

p = Pulsesensor()
p.startAsyncBPM()
old = 0
old_ibi = 600
ibi = 600
count = 0
upload = 0
hrv_array = [0] * 16
hrv = 0.0
bpm_avg = 0.0
setupold = 1

acc_array = [0] * 16
temp_array = [0] * 16
bpm_array = [0] * 16

acc_avg = 0.0
temp_avg = 0.0

accel = msa301.MSA301()
time.sleep(2)
accel.set_power_mode('normal')
time.sleep(2)
cs_pin = digitalio.DigitalInOut(board.CE0)
dc_pin = digitalio.DigitalInOut(board.D25)
reset_pin = digitalio.DigitalInOut(board.D24)
backlight = digitalio.DigitalInOut(board.D26)
backlight.switch_to_output()
```

```
backlight.value = True

# Config for display baudrate (default max is 24mhz):
BAUDRATE = 100000

# Setup SPI bus using hardware SPI:
spi = board.SPI()

disp = st7735.ST7735R(
    spi,
    rotation=90,  # 2.2", 2.4", 2.8", 3.2" ILI9341
    cs=cs_pin,
    dc=dc_pin,
    rst=reset_pin,
    baudrate=BAUDRATE,
)
# pylint: enable=line-too-long

# Create blank image for drawing.
# Make sure to create image with mode 'RGB' for full color.
if disp.rotation % 180 == 90:
    height = 128  # we swap height/width to rotate it to landscape!
    width = 160
else:
    width = 128  # we swap height/width to rotate it to landscape!
    height = 160

image = Image.new("RGB", (width, height))

# Get drawing object to draw on image.
draw = ImageDraw.Draw(image)

# Draw a black filled box to clear the image.
draw.rectangle((0, 0, width, height), outline=0, fill=(0, 0, 0))
disp.image(image)

# First define some constants to allow easy positioning of text.
padding = -2
x = 0
font =
ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSansMono.ttf",
24)
```

```
def hrvcalc(lst1):
    sumarray = [0] * 16
    sum = 0.0
    HRV = 0.0
    count = 0
    for i in range(0, len(lst1), 1):
        IBI = lst1[i]
        IBI **=2
        sumarray[i] = IBI


    for i in range(len(sumarray)):
        sum += sumarray[i]

    HRV = (1/(16-1))* sum
    return (math.sqrt(HRV))


def avg_list(lst1):
    sum = 0.0
    for i in range(0, len(lst1), 1):
        sum = sum + lst1[i]

    if sum <= 0:
        return 0
    return sum / 16.0

ADC0832.setup()

while True:
    try:
        #--------Thingspeak setup-------
        bpm_channel = thingspeak.Channel(id=bpm_channel_id,
api_key=bpm_write_key)
        acc_channel = thingspeak.Channel(id=acc_channel_id,
api_key=acc_write_key)
        hrv_channel = thingspeak.Channel(id=hrv_channel_id,
api_key=hrv_write_key)
        temp_channel = thingspeak.Channel(id=temp_channel_id,
api_key=temp_write_key)

        #-----Sensor Info Gathering----------
```

```
        #BPM
        bpm = p.BPM
        ibi = p.IB
        time.sleep(1)

            #Accl
        pos = accel.get_measurements()
        if pos[0] != 0:
            theta = abs((((math.atan(pos[1]/pos[0]))*62)))
            thetaInt = int(theta)
            if thetaInt <= 90:
                thetaInt = thetaInt
            elif thetaInt >= 90 and thetaInt <= 180:
                thetaInt = 180 - thetaInt
            elif thetaInt >= 180 and thetaInt <= 270:
                thetaInt = thetaInt - 180
            elif thetaInt >= 270 and thetaInt <= 361:
                thetaInt = 360 - thetaInt
            acc_array[count] = thetaInt
        else:
            thetaInt = 0
            acc_array[count] = thetaInt

        #Temp
        Temp = ADC0832.getResult()-41
        temp_array[count] = Temp
        if Temp < 0:
            Temp = 0
            temp_array[count] = Temp
        print("theta =", thetaInt)
        print("temp =", Temp)

        #--------LCD Output------------
        ThetaText = "Angle = " + str(thetaInt)
        TempText = "Temp = " + str(int(Temp))
        BPMText = "BPM = " + str(int(old))
        HRVText = "HRV = " + str(int(hrv))

        #clear screen
        draw.rectangle((0, 0, width, height), outline=0, fill=0)

        #write all text data
        y = padding
```

```
        draw.text((x,y), ThetaText, font=font, fill = "#FFFFFF")
        y += font.getsize(ThetaText)[1]
        draw.text((x,y), TempText, font=font, fill = "#FFFFF0")
        y += font.getsize(TempText)[1]
        draw.text((x,y), BPMText, font=font, fill = "#00FF00")
        y += font.getsize(BPMText)[1]
        draw.text((x,y), HRVText, font=font, fill = "#00FF00")

        #Draw Image
        disp.image(image)
        time.sleep(.75) #change this if you want to go faster.


        #----Thingspeak output-------
        if(upload == 1):
            bpm_avg = avg_list(bpm_array)
            acc_avg = avg_list(acc_array)
            temp_avg = avg_list(temp_array)
            if bpm_avg >= 100: #send BPM alert
                os.system(bpm_sms)
                print("-----------bpm message sent---------\n")
            if acc_avg >= 35: #send accelerometer alert
                os.system(acc_sms)
                print("-----------acc message sent---------\n")
            if hrv >= 130: #send hrv alert
                os.system(hrv_sms)
                print("-----------hrv message sent---------\n")
            if temp_avg >= 100: #send temp alert
                os.system(temp_sms)
                print("-----------temp message sent---------\n")
            try:
                response_bpm = bpm_channel.update({'field1': bpm_avg,
'field2': bpm_avg})
                read_bpm = bpm_channel.get({})
                print("Read:", read_bpm)
                print("\n")

                response_acc = acc_channel.update({'field1': acc_avg,
'field2': acc_avg})
                read_acc = acc_channel.get({})
                print("Read:", read_acc)
                print("\n")
```

```
                response_hrv = hrv_channel.update({'field1': hrv, 'field2':
hrv})
                read_hrv = hrv_channel.get({})
                print("Read:", read_hrv)
                print("\n")

                response_temp = temp_channel.update({'field1': temp_avg,
'field2': temp_avg})
                read_temp = temp_channel.get({})
                print("Read:", read_temp)

                upload = 0

            except:
                print("connection failed")
                upload = 0

        count = count + 1
        if(count >= 16):
            count = 0
            upload = 1
            hrv = hrvcalc(bpm_array)
            print("HRV: %d" %hrv)
    except Exception as e:
        p.stopAsyncBPM()
        ADC0832.destroy()
        print(e)
```