1. ACKNOWLEDGEMENT

We would like to express our deepest appreciation to the mentors, Dr. Ram Kumar Maharjan and Dr. Diwakar Raj Pant for giving us an opportunity with this minor project to apply our theoretical knowledge on circuits and microcontrollers into practical implementation. We extend our gratitude to Mr. Durga Prasad Khatiwada, who is willing to provide us help in regards to the workspace and testing of our prototype.

We would like to acknowledge the Department of Electronics and Communication, Pulchowk Campus, for providing us an opportunity to learn and showcase our skills by creating innovative projects for the benefit of the society.

Finally, we would like to thank our fellow classmates, Mr. Rashik Shrestha for providing his help with the optimization of our project and 3D printing and Mr. Raji Bajracharya for helping us while finding the appropriate project name and providing help with data analysis. And we would also like to express our gratitude for all the people who gave us suggestions regarding our project.

Regards,

Amrita Thakur	(073BEX405)
Pujan Budhathoki	(073BEX428)
Sarmila Upreti	(073BEX439)
Shirish Shrestha	(073BEX440)

2. ABSTRACT:

This project report deals with the essence of the Polysomnographic Analysis Band and Fitness Companion application in sleep and health analysis. It also discusses about their working and limitations. This report highlights the analysis of the heart rate dataset by comparison with standard heart rate curve using correlation analysis for finding the relation between standard dataset and obtained dataset. It also highlights the deep sleep time duration for a dataset that follows standard heartrate curve. This report also provides the movement rate or roll rate of a person while asleep. The expenses excluding NRE cost are also mentioned in the report. It talks about the issues we faced while creating the product along with the limitations of the product.

TABLE OF CONTENTS

1. **ACKNOWLEDGEMENT** 1
2. **ABSTRACT:** 2
3. **OBJECTIVES:** 5
4. **PURPOSE:** 6
5. **THEORETICAL BACKGROUND:** 7
   - 5.1. **Sleep and its analysis:** 7
     - 5.1.1. Sleep Deprivation and the Importance of Healthy Sleep: 7
     - 5.1.2. Standard heart rate curve: 8
   - 5.2. **Hardware Parts:** 9
     - 5.2.1. Pulse Sensor: 9
     - 5.2.2. Accelerometer Gyroscope Sensor: 10
     - 5.2.3. Arduino Pro Mini: 11
     - 5.2.4. USB to TTL adapter: 11
   - 5.3. **3D printing:** 12
   - 5.4. **Software Parts:** 13
     - 5.4.1. MIT App Inventor: 13
   - 5.5. **Analysis Part:** 16
     - 5.5.1. MATLAB: 16
     - 5.5.2. Correlation for statistical analysis: 16
6. **METHODOLOGY** 17
   - 6.1. **Hardware part:** 17
     - 6.1.1. Interfacing Pulse sensor: 17
     - 6.1.2. Interfacing GR-521: 17
     - 6.1.3. Interfacing HC-06: 18
     - 6.1.4. Interfacing with USB to TTL Serial Converter: 18
     - 6.1.5. Finalization: 19
     - 6.1.6. Hardware Product Comparison: 20
   - 6.2. **Software part:** 21
     - 6.2.1. Main Screen: 21
     - 6.2.2. Bluetooth Screen: 22
     - 6.2.3. Breathing Screen: 23
     - 6.2.4. Software Product Comparison: 23
   - 6.3. **Analysis part:** 24
     - 6.3.1. Generation of Standard Heart Rate Curve: 24

6.3.2. Correlation coefficient analysis: 25

6.3.3. Patient's Movement while asleep: 28

7. PROJECT EXPENSES 30

8. PROBLEMS FACED AND LIMITATIONS 31

9. CONCLUSIONS 32

References: 32

10. APPENDIX: Code for Arduino Pro Mini 33

TABLE OF FIGURES

Figure 1: Standard Hammock Curve 8
Figure 2: Pulse Sensor 9
Figure 3: Circuit of Pulse Sensor 9
Figure 4: MPU5060 with GY 521 10
Figure 5: Arduino Pro Mini 11
Figure 6: USB to TTL adapter 11
Figure 7: 3D printing 12
Figure 8: MIT app inventor interface 13
Figure 9: Component Designer 14
Figure 10: Blocks Editor 14
Figure 11: Arduino pro mini to USB-to-TTL connection 19
Figure 12: 3D layout (top view) 19
Figure 13: 3D layout (bottom view) 19
Figure 14: Orthographic View of the 3D model 19
Figure 15:Hardware Components 19
Figure 16: Layered Arrangement of components 20
Figure 17: Battery fitted on the side 20
Figure 18: Final output (Polysomnographic Analysis Band) 20
Figure 19: Hardware product till midterm defense 20
Figure 20: Hardware Product till Final Defense 21
Figure 21: Component Design for Main Screen 21
Figure 22: Blocks code for Main Screen 21
Figure 23: Component Design for Bluetooth Screen 22
Figure 24: Blocks code for Bluetooth Screen 22
Figure 25: Blocks Code for Breathing Screen 23
Figure 26: Video played on Breathing Screen 23
Figure 27: Software Product till midterm defense 23
Figure 28: Added features in Software Product till Final defense 24
Figure 29: Time conversion table 24
Figure 30: Standard Heart Rate Curve in MATLAB 25
Figure 31: Comparison plot between standard data and GoodBPM.txt data 28
Figure 32: Comparison plot between standard data and BadBPM.txt data 28
Figure 33: Change in sign of roll variable visualized using a stem graph 29

3. OBJECTIVES:

- To make a sleep quality analyzing gadget using Arduino Pro Mini.
- To measure heartbeat per minute using heart beat sensor.
- To measure the movement of the patient while asleep using accelerometer and gyroscope sensor.
- To develop an android application using MIT app inventor reflects the measurements done by the gadget.
- To make the gadget portable and wearable.
- To compare the obtained readings for heartrate with the standard heartrate dataset.
- To find the number of times a person moved from left to right and vice versa while asleep.

4. PURPOSE:

This section deals with the purpose, importance and reason for the necessity of the product. We know that, sleep deprivation is a condition that occurs if you don't get enough sleep. It is caused when you don't get enough sleep, you sleep at the wrong time of day or you have a sleep disorder that prevents you from getting enough sleep or causes poor quality sleep. Sleep deficiency can interfere with work, school, driving, and social functioning. You might have trouble learning, focusing, and reacting. Also, you might find it hard to judge other people's emotions and reactions. Sleep deficiency also can make you feel frustrated, cranky, or worried in social situations. So, polysomnography has been one of the rising platforms in the modern world.

The main aim of our project is to perform sleep analysis on a patient with the help of various parameters such as pulse rate in beats per minute (bpm) and movement of patient while asleep. The project will combine hardware components composed of the following components:

- Arduino Pro Mini
- Pulse Sensor
- Accelerometer with Gyroscope (MPU6050-gy521)
- Bluetooth Module (HC-06)

and software components which involves an app which will display the measured data and MATLAB software which will compare the data obtained (like pulse rate) with the ideal curve's data set with the help of correlation coefficient between the ideal data set and the obtained data set. Similarly, the movement of the person while asleep measured using Accelerometer with Gyroscope will be used to depict the person's movement rate or roll during sleep.

5. THEORETICAL BACKGROUND:

5.1. Sleep and its analysis:

5.1.1. Sleep Deprivation and the Importance of Healthy Sleep:

Sleep deprivation is a condition that occurs if you don't get enough sleep. Sleep deficiency is a broader concept. It occurs if you have one or more of the following:

- ❖ You don't get enough sleep (sleep deprivation)
- ❖ You sleep at the wrong time of day (that is, you're out of sync with your body's natural clock)
- ❖ You don't sleep well or get all of the different types of sleep that your body needs
- ❖ You have a sleep disorder that prevents you from getting enough sleep or causes poor quality sleep

To understand sleep deficiency, it helps to understand how sleep works and why it's important. The two basic types of sleep are rapid eye movement (REM) and non-REM.

Non-REM sleep includes what is commonly known as deep sleep or slow wave sleep. Dreaming typically occurs during REM sleep. Generally, non-REM and REM sleep occur in a regular pattern of 3–5 cycles each night. Your ability to function and feel well while you're awake depends on whether you're getting enough total sleep and enough of each type of sleep. It also depends on whether you're sleeping at a time when your body is prepared and ready to sleep. You have an internal "body clock" that controls when you're awake and when your body is ready for sleep. This clock typically follows a 24-hour repeating rhythm (called the circadian rhythm). The rhythm affects every cell, tissue, and organ in your body and how they work.

If you aren't getting enough sleep, are sleeping at the wrong times, or have poor quality sleep, you'll likely feel very tired during the day. You may not feel refreshed and alert when you wake up. Sleep deficiency can interfere with work, school, driving, and social functioning. You might have trouble learning, focusing, and reacting. Also, you might find it hard to judge other people's emotions and reactions. Sleep deficiency also can make you feel frustrated, cranky, or worried in social situations. The signs and symptoms of sleep deficiency may differ between children and adults. Children who are sleep deficient might be overly active and have problems paying attention. Sleep deficiency, which includes sleep deprivation, affects people of all ages, races, and ethnicities. Certain groups of people may be more likely to be sleep deficient. Examples include people who:

- ❖ Have limited time available for sleep, such as caregivers or people working long hours or more than one job
- ❖ Have schedules that conflict with their internal body clocks, such as shift workers, first responders, teens who have early school schedules, or people who must travel for work
- ❖ Make lifestyle choices that prevent them from getting enough sleep, such as taking medicine to stay awake, abusing alcohol or drugs, or not leaving enough time for sleep
- ❖ Have undiagnosed or untreated medical problems, such as stress, anxiety, or sleep disorders
- ❖ Have medical conditions or take medicines that interfere with sleep

5.1.2. Standard heart rate curve:

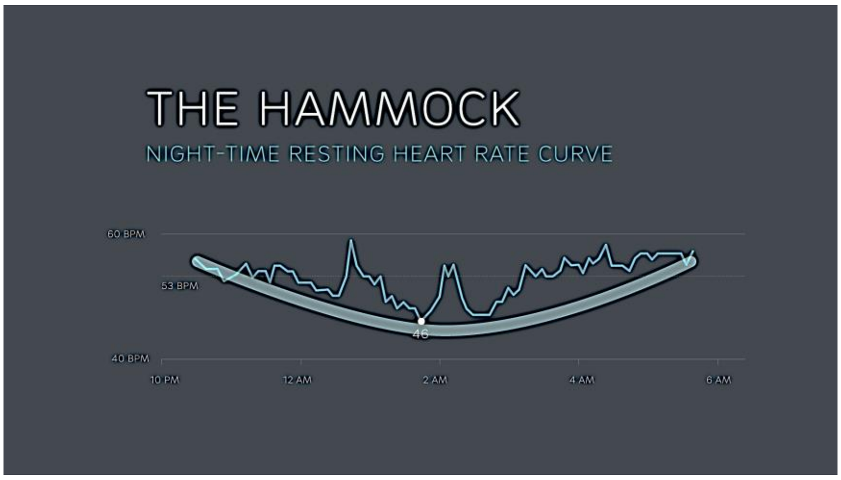


Figure 1: Standard Hammock Curve

The hammock curve is one of the standard heart rate curves used for sleep analysis. This curve depicts a sound quality of sleep. If a heartbeat pattern follows this curve during night time, then that patient had had a quality sleep that night. The pattern of heart beat in this curve is overall explained by a decrement in heart beat rate till it reaches the lowest point (least value of bpm) where deep sleep is achieved, and after that again the heart beat starts to increment.

Equation of Hammock Curve:

To make a hammock curve, we have used the equation of parabola given by:

$$y = ax^2 + bx + c$$

where, a, b and c are appropriate constants that would make the parabola shaped as a hammock curve.

5.2. Hardware Parts:

5.2.1. Pulse Sensor:

Pulse Sensor is used to measure the pulse rate per minute. It is based on the principle of photo plethysmography. It measures the change in volume of blood through any organ of the body which causes a change in the light intensity through that organ (a vascular region). In our application where heart pulse rate is to be monitored, the timing of the pulses is more important. The flow of blood volume is decided by the rate of heart pulses and since light is absorbed by blood, the signal pulses are equivalent to the heart beat pulses.

Figure 2: Pulse Sensor

Working:

The basic heartbeat sensor consists of a light emitting diode and a detector like a light detecting resistor or a photodiode. The heart beat pulses causes a variation in the flow of blood to different regions of the body. When a tissue is illuminated with the light source, i.e. light emitted by the led, it either reflects (a finger tissue) or transmits the light (earlobe). Some of the light is absorbed by the blood and the transmitted or the reflected light is received by the light detector. The amount of light absorbed depends on the blood volume in that tissue. The detector output is in form of electrical signal and is proportional to the heart beat rate. This signal is actually a DC signal relating to the tissues and the blood volume and the AC component synchronous with the heart beat and caused by pulsatile changes in arterial blood volume is superimposed on the DC signal. Thus, the major requirement is to isolate that AC component as it is of prime importance.

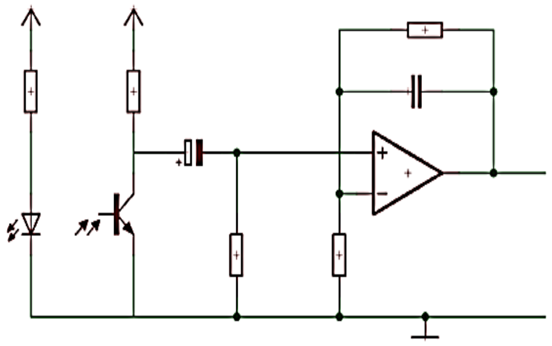


Figure 3: Circuit of Pulse Sensor

Features:

- ❖ Biometric Pulse Rate or Heart Rate detecting sensor
- ❖ Plug and Play type sensor
- ❖ Operating Voltage: +5V or +3.3V
- ❖ Current Consumption: 4mA
- ❖ Inbuilt Amplification and Noise cancellation circuit.
- ❖ Diameter: 0.625”
- ❖ Thickness: 0.125” Thick

5.2.2. Accelerometer Gyroscope Sensor:

Accelerometers in general are used to detect the orientation of the object. The gyroscope, or gyro for short, adds an additional dimension to the information supplied by the accelerometer by tracking rotation or twist. An accelerometer measures linear acceleration of movement, while a gyro on the other hand measures the angular rotational velocity. Both sensors measure rate of change; they just measure the rate of change for different things. We have used GY 521 module of the sensor. The MPU-6050 sensor module contains an accelerometer and a gyro in a single chip. It is very accurate, as it contains 16-bits analog to digital conversion hardware for each channel. Therefore, it captures the x, y, and z channel at the same time. The sensor uses the I²C-bus to interface with the Arduino.

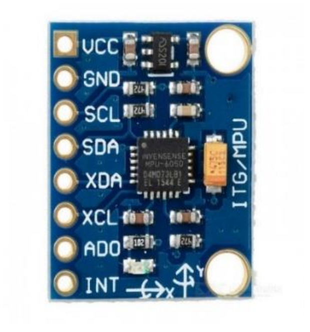


Figure 4: MPU5060 with GY 521

Working of Accelerometer:

Most accelerometers are Micro-Electro-Mechanical Sensors (MEMS). The basic principle of operation behind the MEMS accelerometer is the displacement of a small proof mass etched into the silicon surface of the integrated circuit and suspended by small beams. Consistent with Newton's second law of motion (**F = ma**), as an acceleration is applied to the device, a force develops which displaces the mass. The support beams act as a spring, and the fluid (usually air) trapped inside the IC acts as a damper, resulting in a second order lumped physical system. This is the source of the limited operational bandwidth and non-uniform frequency response of accelerometers.

Working of Gyroscope:

When the gyroscope is applied with external torques or rotations about the given axis, the orientation can be measured by a precession phenomenon. When an object rotating about an axis is applied with external torque along a direction perpendicular to the rotational axis, the precession occurs. This rotation about the spin axis is detected and information on this rotation is delivered to a motor or other device that applies torque in an opposite direction thereby canceling the precession and maintaining its orientation. The precession can also be prevented using two gyroscopes that are arranged perpendicular to each other. The rotation rate can be measured by the pulsation of counteracting torque at constant time intervals.

Features:

- ❖ Uses the chip: MPU-6050 and Power supply: 3-5v (internal low dropout regulator).
- ❖ Communication modes: standard I²C communications protocol.
- ❖ Chip built-in 16-bit ADC, 16-bit data output.
- ❖ Immersion Gold PCB machine welding process to ensure quality.
- ❖ Tri-Axis angular rate sensor (gyro) with a sensitivity up to 131 LSBs/dps and a full-scale range of ±250, ±500, ±1000, and ±2000dps
- ❖ Tri-Axis accelerometer with a programmable full-scale range of ±2g, ±4g, ±8g and ±16g

5.2.3. Arduino Pro Mini:

The Arduino Pro Mini is an ATmega168 based microcontroller board. The board comes with built-in Arduino bootloader. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 8 analog inputs, an on-board resonator, a reset button, and holes for mounting pin headers. The board can be connected to the PC using USB port and the board can runs on USB power.

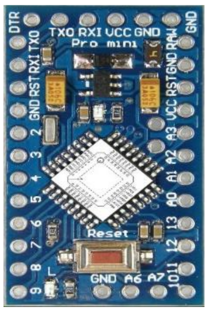


Figure 5: Arduino Pro Mini

Feature:

- ❖ There is a voltage regulator on board so it can accept voltage up to 12V DC. If you're supplying unregulated power to the board, be sure to connect to the RAW pin and not VCC.
- ❖ VCC accepts only 5v.
- ❖ The latest version of this board breaks out the ADC6 and ADC7 pins, also adds footprints for optional I2C pull-up resistors.
- ❖ To upload code on to pro mini, a USB-TTL adapter is required.

Properties:

Power:

- ❖ Raw:5V-16V
  - o Recommended: 6V-12V
- ❖ VCC:5V
- ❖ Maximum current: 150mA @5V

LEDs:

- ❖ Power: Red
- ❖ User (D13): Green

ATMega328P:

- ❖ Absolute maximum VCC: 6V
- ❖ Maximum current for chip: 200mA
- ❖ Maximum current per pin: 40mA
  - o Recommended: 20mA
- ❖ 8-bit Atmel AVR
- ❖ Flash Program Memory: 32kB
- ❖ EEPROM: 1kB
- ❖ Internal SRAM 2kB
- ❖ ADC:10-bit
- ❖ PWM:8bit

5.2.4. USB to TTL adapter:

In simple words, adaptor is our own personal translator between the device and the computer. Pro mini requires USB to TTL convertor to get program uploaded in it from computer. The connector is external since the goal is to make Arduino as small as possible for greater embeddability.

Figure 6: USB to TTL adapter

Features:

- ❖ Use this device to connect your PC to a serial (TTL level) device.
- ❖ Uses CP2102 chipset and has a standard 0.1" pitch terminal strip to connect directly to UART or I/O pins
- ❖ USB specification 2.0 compliant with full-speed 12Mbps
- ❖ 6 pins for 3.3V, RST, TXD, RXD, GND & 5V
- ❖ All handshaking and modem interface signals
- ❖ Baud rates: 300 bps to 1.5 Mbps

- Byte receive buffer; 640 bytes transmit buffer
- Hardware or X-On/X-Off handshaking supported
- Event character support Line break transmission
- USB suspend states supported via SUSPEND pins
- Supports Windows 98SE, 2000, XP, Vista, Window7, Mac OS 9, Mac OS X & Linux 2.40
- Size: 42mm X 15mm (approx.)
- Weight: 4g (approx.)

5.3. 3D printing:

3D printing or additive manufacturing is a process of making three dimensional solid objects from a digital file. The creation of a 3D printed object is achieved using additive processes. In an additive process an object is created by laying down successive layers of material until the object is created. Each of these layers can be seen as a thinly sliced horizontal cross-section of the eventual object. 3D printing is the opposite of subtractive manufacturing which is cutting out / hollowing out a piece of metal or plastic with for instance a milling machine. 3D printing enables you to produce complex shapes using less material than traditional manufacturing methods.

It all starts with a 3D model. You create one yourself or download it from a 3D repository. When creating it yourself you can choose to use a 3D scanner, app, haptic device, code or 3D modeling software. There are many different 3D modeling software tools available. Slicing is dividing a 3D model into hundreds or thousands of horizontal layers and is done with slicing software. When file is sliced, it’s ready to be fed to 3D printer. This can be done via USB, SD or internet. Your sliced 3D model is now ready to be 3D printed layer by layer.

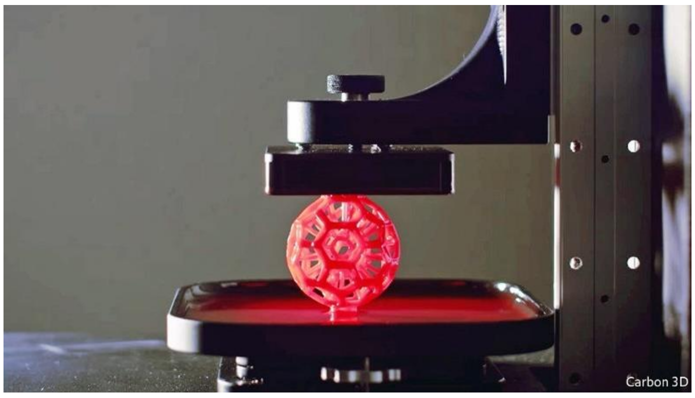


Figure 7: 3D printing

5.4. Software Parts:

5.4.1. MIT App Inventor:

MIT App Inventor is an intuitive, visual programming environment that allows everyone even children to build fully functional apps for smartphones and tablets. Those new to MIT App Inventor can have a simple first app up and running in less than 30 minutes. And what's more, our blocks-based tool facilitates the creation of complex, high-impact apps in significantly less time than traditional programming environments. The MIT App Inventor project seeks to democratize software development by empowering all people, especially young people, to move from technology consumption to technology creation. The MIT app inventor was written in Java, Kawa, Scheme languages.

It uses a graphical user interface (GUI) which allows users to drag and drop visual objects to create an application that can run on mobile devices. Google drew upon significant prior research in educational computing, and work done within Google on online development environments in creating this. The blocks-based tool facilitates the creation of complex, high-impact apps in significantly less time than traditional programming environments. The MIT App Inventor project seeks to democratize software development by empowering all people, especially young people, to move from technology consumption to technology creation.

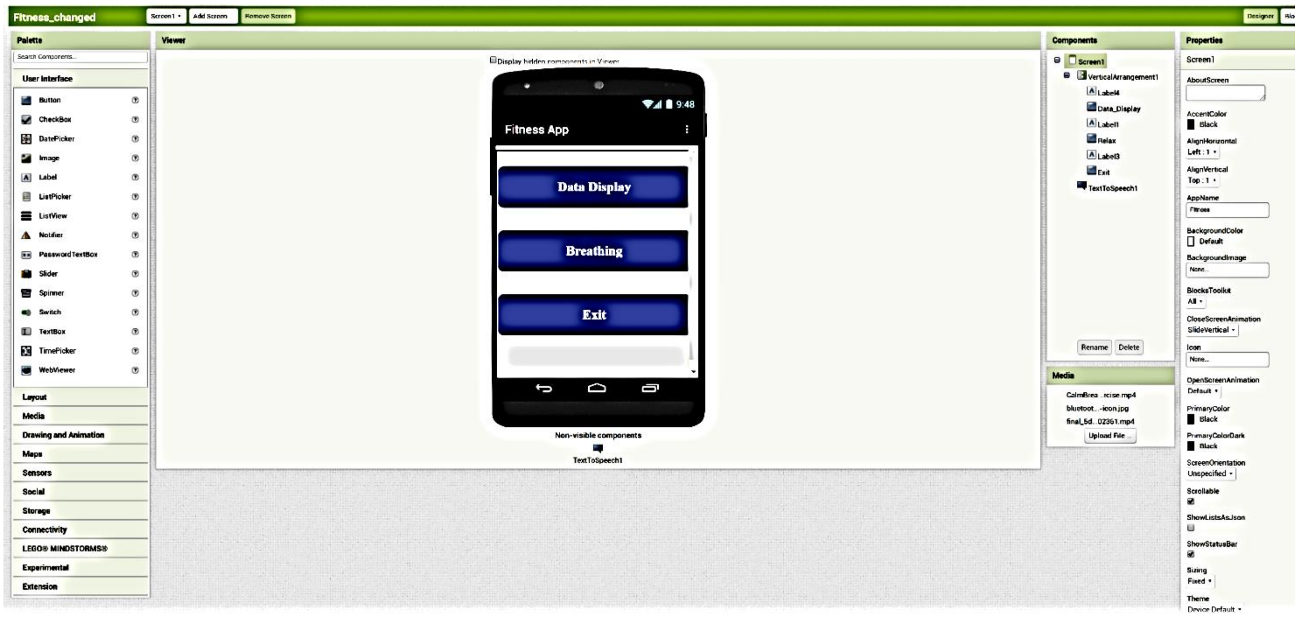


Figure 8: MIT app inventor interface

The App Inventor programming environment has three key parts:

- ❖ The Component Designer (Figure 6). We use it to select components for our app and specify their properties.
- ❖ The Blocks Editor (Figure 7). We use it to specify how the components will behave (e.g., what happens when a user clicks a button).
- ❖ An Android device with which we can actually run and test your app as you are developing it. If you don’t have an Android device handy, you can test the apps you build by using the Android emulator that comes with the system.

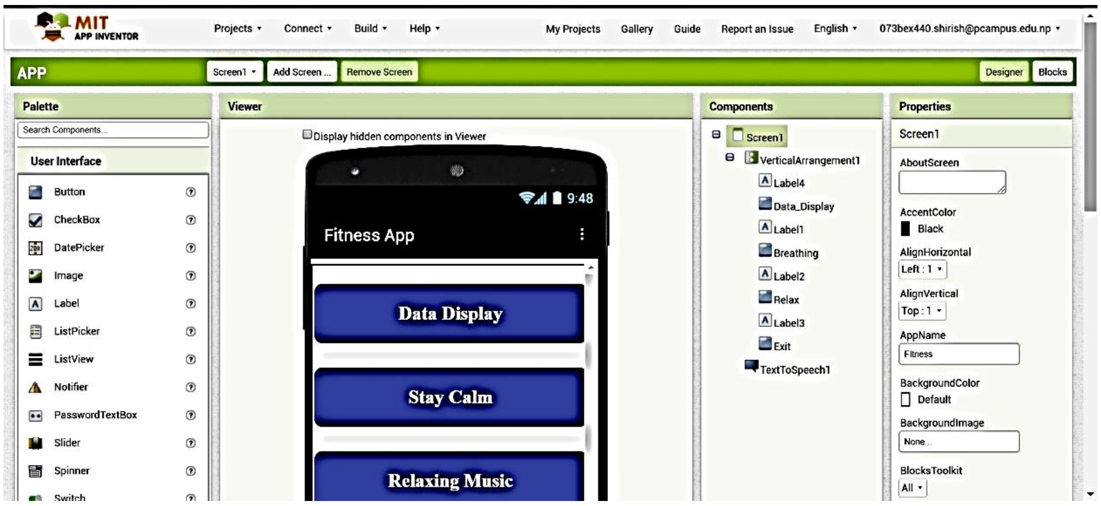


Figure 9: Component Designer

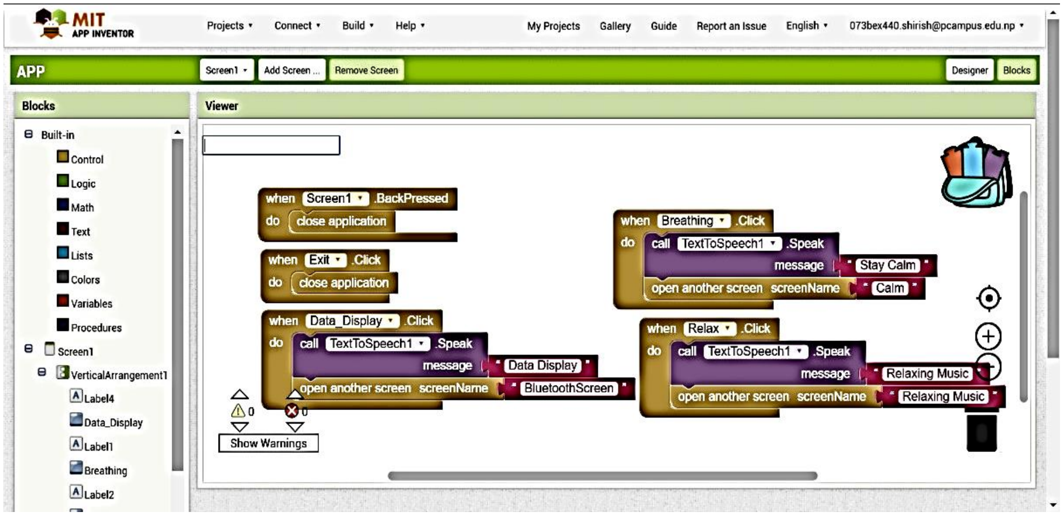
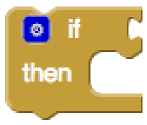

MIT APP INVENTOR
APP
Blocks
Viewer
Built-in
Control
Logic
Math
Text
Lists
Colors
Variables
Procedures
Screen1
when Screen1 .BackPressed do close application
when Exit .Click do close application
when Data_Display .Click do call TextToSpeech1 .Speak message Data Display open another screen screenName BluetoothScreen
when Breathing .Click do call TextToSpeech1 .Speak message Stay Calm open another screen screenName Calm
when Relax .Click do call TextToSpeech1 .Speak message Relaxing Music open another screen screenName Relaxing Music
Show Warnings

Figure 10: Blocks Editor

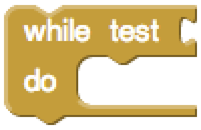
The first time we browse to ai2.appinventor.mit.edu, you'll see the Projects page, which will be mostly blank because you haven't created any projects yet. To create a project, at the upper left of the page, click "New Project," and then enter the project name. The first window that opens is the Component Designer. The Blocks Editor is available by clicking on the "Blocks" button in the upper-right corner of the window. App Inventor is a cloud computing tool, meaning that your app is stored on an online server as you work. So, if you close App Inventor, your app will be there when you return; you don't have to save anything on your computer as you would with, for example, a Microsoft Word file.

<u>Basic functions in MIT App Inventor:</u>

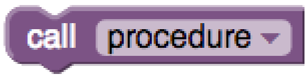
Some of the basics blocks that are most often used in development of apps are:

i. 

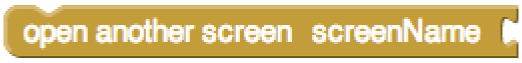
Tests a given condition. If the condition is true, performs the actions in a given sequence of blocks; otherwise, the blocks are ignored.



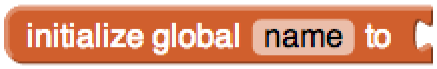
ii. Tests the -test condition. If true, performs the action given in -do, then tests again. When test is false, the block ends and the action given in -do is no longer performed.



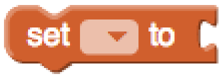
iii. When you create a procedure, App Inventor automatically generates a call block and places it in the My Definitions drawer. You use the call block to invoke the procedure.



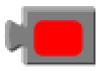
iv. Opens another screen and passes a value to it.



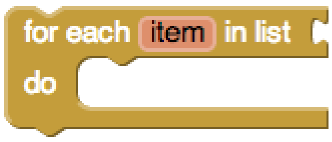
v. This block is used to create global variables. It takes in any type of value as an argument. Clicking on name will change the name of this global variable. Global variables are used in all procedures or events so this block will stand alone.



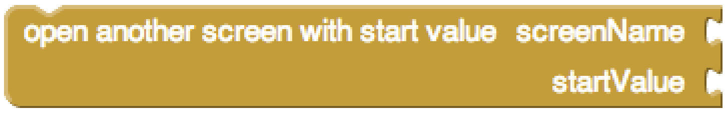
vi. This block follows the same rules as get. Only variables in scope will be available in the dropdown. Once a variable v is selected, the user can attach a new block and give v a new value.



vii. This is a basic color block. It has a small square shape and has a color in the middle that represents the color stored internally in this block. If you click on the color in the middle, a pop-up appears on the screen with a table of 70 colors that you can choose from. Clicking on a new color will change the current color of your basic color block.



viii. Runs the blocks in the do section for each item in the list. Use the given variable name, item, to refer to the current list item. You can change the name item to something else if you wish.



ix. Opens another screen and passes a value to it.

5.5. Analysis Part:

5.5.1. MATLAB:

MATLAB (Matrix Laboratory) is a matrix programming language popular for numerical computing environment. It is a high-level language and interactive environment that lets you focus on your course work and applications, rather than on programming details. It is quite popularly used for data analytics, wireless communications, deep learning, computer vision, signal processing, robotics, control systems and so on. Engineers and scientists worldwide rely on MATLAB. You can also use MATLAB to analyze and visualize data using automation capabilities, thereby avoiding the manual repetition. A vast library of prebuilt toolboxes lets you get started right away with algorithms essential to your domain.

MATLAB also includes Simulink which adds graphical multi-domain simulation and model-based design for dynamic and embedded system. It allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including, C, C++, C#, Java, Fortran and Python. It enables you to solve many numerical problems in a fraction of the time it takes to write a program in a lower-level language such as Java, C, C++, or Fortran.

5.5.2. Correlation for statistical analysis:

Correlation is a statistical measure, which is used to study the degree of relationship between two or more than two variables. Therefore, correlation exists between two variables when one of them is related to the other in some way. When there exists some relationship between two variables, we have to measure the degree of relationship and this measure is called correlation coefficient, denoted by r. It helps us to know the nature, direction and degree of relationship between two or more variables. It helps in making predictions and decision making. Here, we are only dealing with simple correlation where the linear relationship between only two variables is considered.

Among many properties of correlation coefficient, we are going to discuss about the one which is the basis of our analysis. The value of r lies always in between -1 and 1 inclusive. Symbolically, $-1 \leq r \leq 1$. It is not designed to measure the strength of a relationship that is not linear. The magnitude of r describes the strength of a linear relationship and its sign indicates the direction. If r is closer to +1, then there is high degree of positive correlation and if r is closer to -1, there is high degree of positive correlation. So, for example; Value of r that is greater than +0.6, shows the two data sets are significantly correlated in positive direction.

<u>Formula for Correlation Coefficient:</u>

Suppose we have two data sets X and Y with same number of elements denoted by n, then the correlation coefficient can be calculated as:

$$r = \frac{n\sum XY - \sum X \sum Y}{\sqrt{n\sum X^2 - (\sum X)^2}\,\sqrt{n\sum Y^2 - (\sum Y)^2}}$$

6. METHODOLOGY

6.1. Hardware part:

In our hardware part, the required data from the sensors are sent to the software application via Bluetooth. The microcontroller used is Arduino Pro Mini and other hardware components comprises of Pulse sensor, Accelerometer-Gyroscope, Bluetooth module and USB-to-TTL Serial Converter. Like in every hardware part, initially the individual hardware elements were tested. The initial testing and coding were done in Arduino Uno. After successful testing of individual components, the combined circuity is done in Arduino pro mini for product finalization.

6.1.1. Interfacing Pulse sensor:

The pulse sensor is interfaced with Arduino Pro Mini with sensor reading taken via analog pin A3 of the Arduino. It is connected to: +V (red), Ground (black), and Analog Pin 3(purple) on our Arduino Pro mini.

The front of the sensor is the pretty side with the Heart logo. This is the side that makes contact with the skin. On the front you see a small round hole, which is where the LED shines through from the back, and there is also a little square just under the LED. The square is an ambient light sensor, exactly like the one used in cellphones, tablets, and laptops, to adjust the screen brightness in different light conditions. The LED shines light into the fingertip or earlobe, or other capillary tissue, and sensor reads the light that bounces back. The back of the sensor is where the rest of the parts are mounted. We put them there so they would not get in the way of the of the sensor on the front. Even the LED we are using is a reverse mount LED.

The red pin powers the LED and it is also the enable pin and pulling it high turns the LED on and the sensor starts working. After the code is uploaded it outputs the BPM reading to the serial monitor. The communication is UART TTL serial communication, which is available on digital pins 0 (RX) and 1 (TX). The transmitting UART converts parallel data into serial form, transmits it in serial to the receiving UART, which then converts the serial data back into parallel data for the receiving device. The baud rate fixed is 9600.

6.1.2. Interfacing GR-521:

The sensor values are retrieved by using the I2C serial data bus, which requires only two wires SCL and SDA, SCL being connected to A5 and SDA being connected to A4. I2C is a serial communication protocol, so data is transferred bit by bit along a single wire (the SDA line). Like SPI, I2C is synchronous, so the output of bits is synchronized to the sampling of bits by a clock signal shared between the master and the slave. The clock signal is always controlled by the master.

- ❖ **SDA (Serial Data)** – The line for the master and slave to send and receive data.
- ❖ **SCL (Serial Clock)** – The line carries the clock signal.

Because I2C uses addressing, multiple slaves can be controlled from a single master.
With a 7-bit address, 128 (2^7) unique address are available. Initially the I2C address of the MPU is accessed through ***Wire.beginTransmission(0x68),*** then ***Wire.write(0x1B)*** and ***Wire.write(0x1C)*** is used for accessing Gyroscope and Accelerometer Configuration. So, the

raw values can be directly read of the accelerometer and gyroscope registers by the Arduino. The MPU-6050 chip also includes a 1024-byte FIFO buffer where data can be placed and read off. The MPU-6050 always acts as a slave when connected to the Arduino with SDA and SCL pins connected to the I2C bus.

Roll is the rotation around x-axis and pitch is the rotation along y-axis. Both are calculated using the equation below:

$$rollangle = arctan\frac{-G_x}{Gz} \qquad (1)$$

$$pitchangle = arctan\frac{G_y}{\sqrt{G_x{}^2 + G_z{}^2}}$$

Where G_x = acceleration in x direction

G_x = acceleration in y direction

G_x = acceleration in z direction

6.1.3. Interfacing HC-06:

The HC-06 is a class two slave Bluetooth module designed for transparent wireless serial communication. The Bluetooth 2.0 protocol operates at 2.4Ghz ISM frequency band with an EDR. Here, it is paired to a master Bluetooth device (smartphone) such that all data received through the serial input is immediately transmitted over the air. Master device search and make pair with the slave device automatically. Arduino default pins D0 and D1 are used for Rx and Tx respectively.

- ❖ TXD: serial data is transmitted by module through this pin
- ❖ RXD: serial data is received by module through this pin with 3.3V logic

For pro mini uses 5V, a voltage divider circuit comprising of 1K and 2.2K resistors is used to supply 3.3V to the Rx pin. First the HC-06 module communicates with microcontroller through UART interface. Here the 5V TXD of MCU connects to HC-06's UART_RXD and the 3.3V RXD of MCU connects to HC-06's UART_TXD. Default baud rate is 9600. For the slave HC-06 receives the data, it needs to transmit the data to the master device. Master device search and make pair with the slave device automatically. Slave devices cannot initiate a connection which means you cannot link 2 HC-06s together. For pairing it needs the authenticated password which is 1234. Then the module sends the serial data to master through wireless Bluetooth.

The HC-06 has 2 modes of operation; AT mode and transmission mode. When the modules are first powered on, they go in to AT mode. Here AT commands can be entered via the wired serial connection. After a connection has been made the modules go in to transmission mode. Here everything the module receives via the wired serial connection is sent to the connected device. At commands cannot be entered again until the connection is broken.

6.1.4. Interfacing with USB to TTL Serial Converter:

The USB TTL Serial cables are a range of USB to serial converter cables which provide connectivity between USB and serial UART interfaces. Unlike Arduino Uno, pro mini is not provided with in built USB port so we need to interface the controller with USB to TTL serial convert to upload programs in our IDE.

Connections USB-to-TTL 5pins to Arduino Mini Pro:

- USB-to-TTL: TXD (green) -> Arduino Mini Pro RX
- USB-to-TTL: RXD (yellow) -> Arduino Mini Pro TX
- USB-to-TTL: GND (brown) -> Arduino Mini Pro GND
- USB-to-TTL: +5V (red) -> Arduino Mini Pro VCC

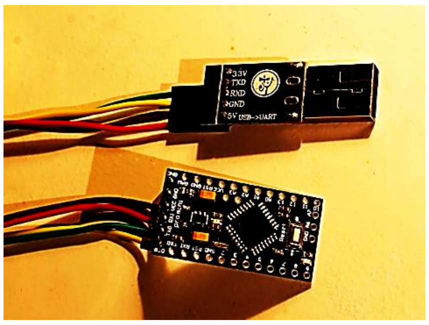


Figure 11: Arduino pro mini to USB-to-TTL connection

The Arduino IDE board is configured as Arduino Pro or Pro Mini with Processor ATmega328 (5V, 16 MHz). The USB-to-TTL hasn't 6th pin for auto reset Arduino board so you have to click on reset button on Arduino Mini Pro manually.

6.1.5. Finalization:

For the finalization of our project we first did the 3D printing of the layout of our band. Then all the hardware components are comprised within the band structure.

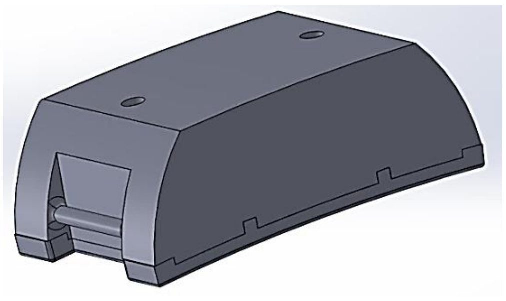


Figure 12: 3D layout (top view)

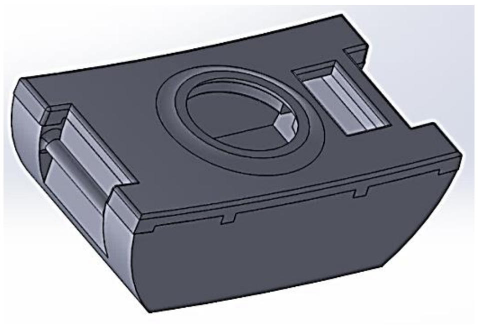


Figure 13: 3D layout (bottom view)

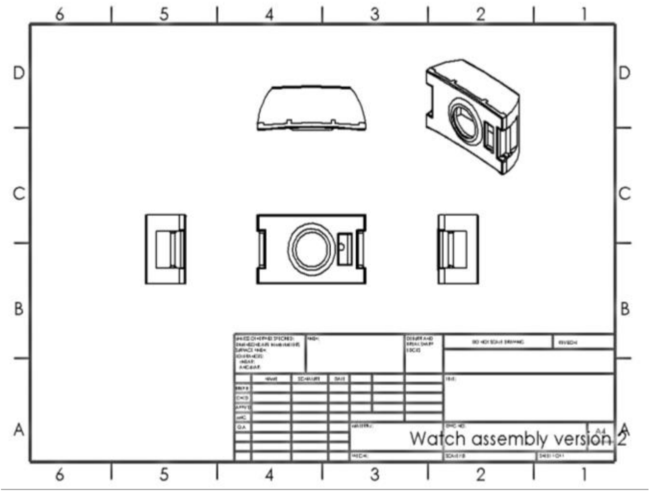


Figure 14: Orthographic View of the 3D model

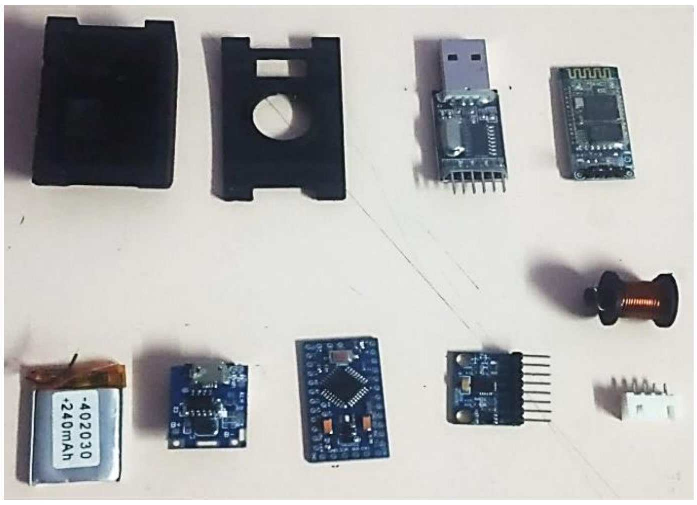


Figure 15:Hardware Components

Then, the components are placed in layered configuration such that three-layer soldering is done. Layering is done such that pulse sensor is place at the bottom which remains in contact with our skin. Then Bluetooth module is placed over it with pro mini on its top. Then GY- 521 module is placed over it such that it doesn't overlap with the reset pin of the Arduino pro mini. For the reset pin should be used manually, the pin can be accessed from outside. Inter layer insulation is done with thin double tape. Overall connection and layering can be seen in picture below. There is also a voltage divider circuit for RX pin of HC-06. A led is also attached in pin D7 of Arduino Pro mini. The battery is fitted on the side.

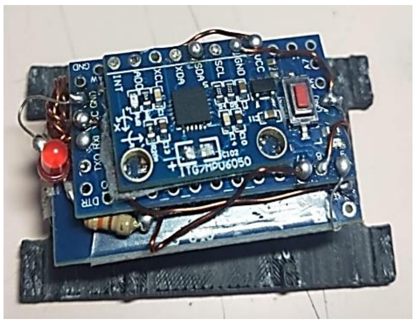
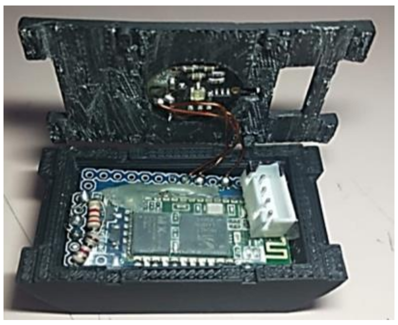



Figure 16: Layered Arrangement of components

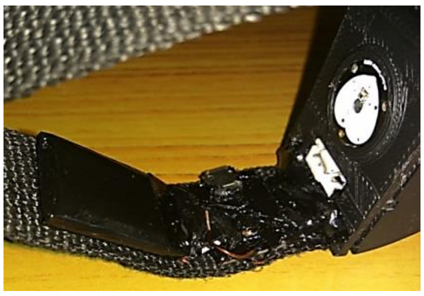


Figure 17: Battery fitted on the side

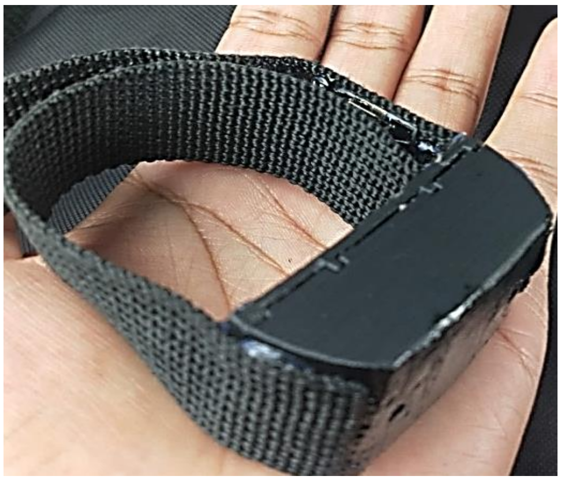
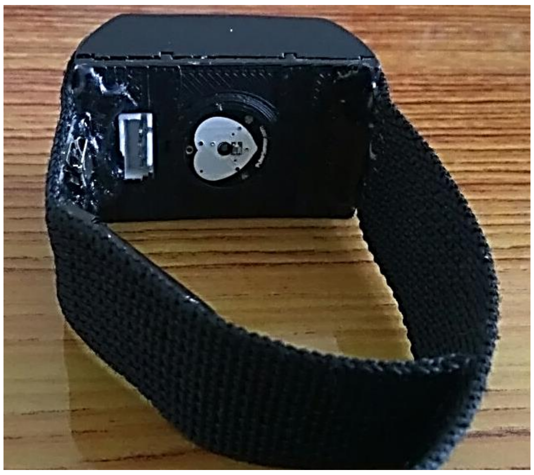



Figure 18: Final output (Polysomnographic Analysis Band)

6.1.6. Hardware Product Comparison:

6.1.6.1. Till Midterm Defense:

- ❖ We were able to interface pulse sensor, accelerometer gyroscope sensor and temperature sensor with Arduino Uno.
- ❖ We were able to display the obtained data in our fitness app using Bluetooth module.

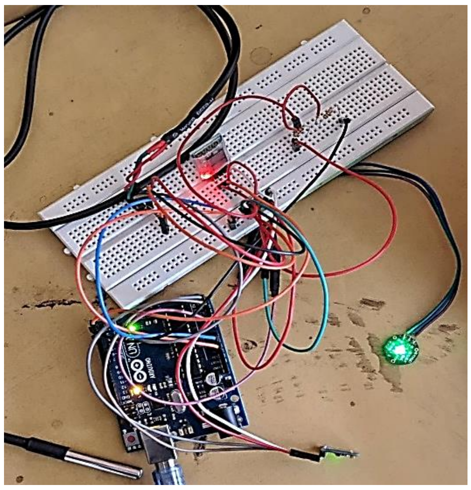


Figure 19: Hardware product till midterm defense

6.1.6.2. Till Final Defense:

- ❖ We have replaced Arduino Uno with pro mini.
- ❖ We have been able to compile all the hardware in the form of a wrist band using 3D-printing.
- ❖ We have been able to compare the curve obtained from observed set of data with standard curve using correlation coefficient of the two datasets.
- ❖ Finally, our project can perform sleep analysis.

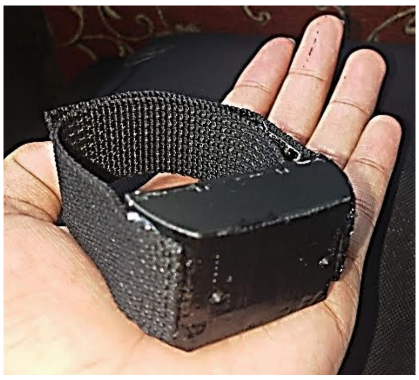


Figure 20: Hardware Product till Final Defense

6.2. Software part:

In the software part, we created an application which displays the data sent by the band via Bluetooth. Along with this feature, the application also has “calm breathing” feature which lets the patient when unable to sleep, breathe in and out accordingly to the video played in the app for relaxing the patient. And using a simple text-to-speech feature, the application reads out the buttons pressed and also tells the user to do certain things, so that he/she wouldn’t feel uncomfortable using this application. The application has three major screens which have a Component Design and Blocks Editor each. These screens are described below:

6.2.1. Main Screen:

This is the block for the start screen. This is the first screen that app greets us with “Data Display”, “Breathing” and “Exit” buttons.

- ❖ **<u>When back key is pressed:</u>** The first block closes the screen and closes the application when the back key is pressed.
- ❖ **<u>When “Data Display” button is clicked:</u>** It calls the TextToSpeech function with the message given in the text box “Data Display”. It also opens another screen with screen name, “BluetoothScreen”.
- ❖ **<u>When “Breathing” button is clicked:</u>** It calls the TextToSpeech function with the message given in the text box “Relax”. It also opens another screen with screen name,” Relax”.
- ❖ **<u>When the “Exit” button is clicked:</u>** It closes the application.

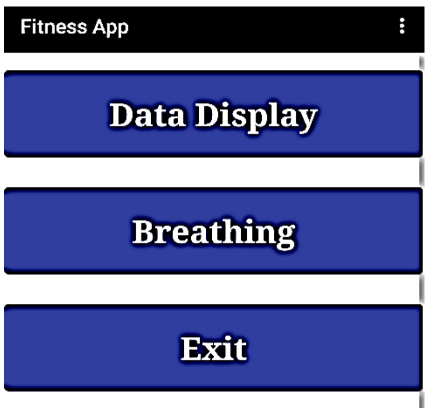
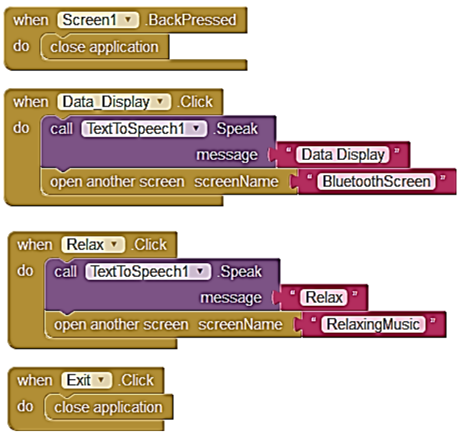


Figure 21: Component Design for Main Screen



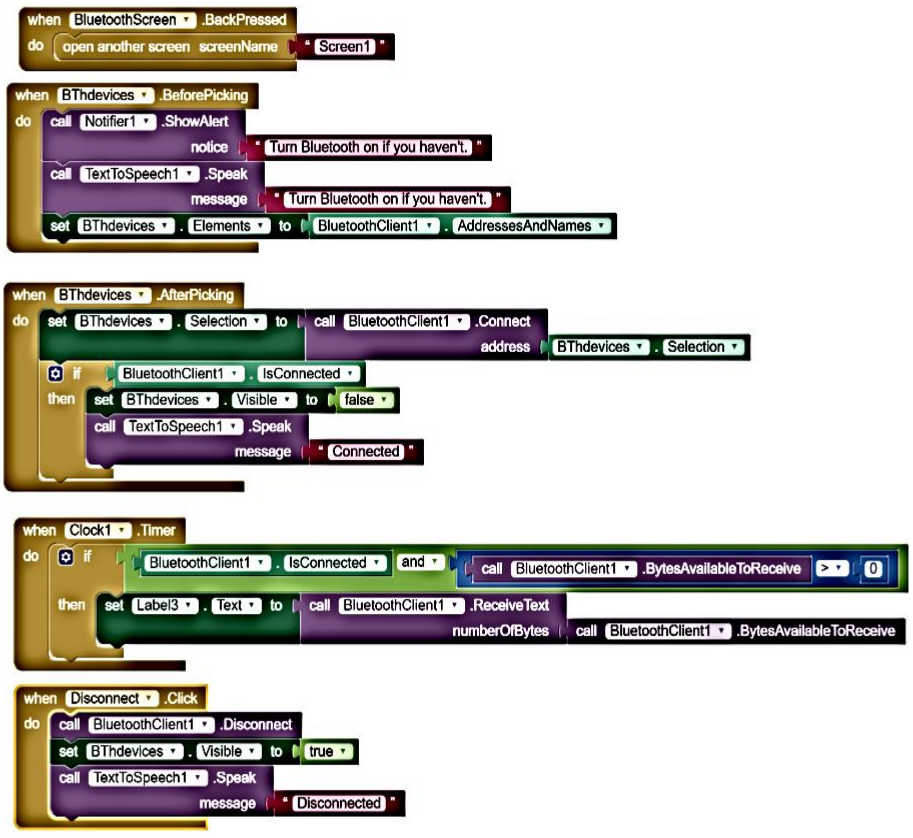
```
when Screen1 .BackPressed
do   close application

when Data_Display .Click
do   call TextToSpeech1 .Speak
                         message  " Data Display "
     open another screen  screenName  " BluetoothScreen "

when Relax .Click
do   call TextToSpeech1 .Speak
                         message  " Relax "
     open another screen  screenName  " RelaxingMusic "

when Exit .Click
do   close application
```

Figure 22: Blocks code for Main Screen

6.2.2. Bluetooth Screen:

This screen deals with Bluetooth devices, connecting them and receiving data.

- ❖ **<u>When Back key Pressed:</u>** When the back key is pressed, the closes the current screen and opens the main screen.
- ❖ **<u>When BThdevices before picking:</u>** Before the Bluetooth device is picked, a warning to turn on Bluetooth devices is given and it displays a list of Bluetooth devices to pick.
- ❖ **<u>When Bthdevies after picking:</u>** After picking among the devices, it sets the visibility of the Bluetooth device to OFF. After that it speaks a message to confirm that the device has been connected.
- ❖ **<u>Clock timer:</u>** The Clock component has a key event, Clock.Timer, which acts as an alarm clock, and one that can go off repeatedly (unless you turn it off). The Clock.TimerInterval property determines how often the Clock.Timer event will be triggered. By default, the TimerInterval is set to 1000 milliseconds (1 second). This sets the data to be received every 1000 milliseconds.
- ❖ The received text is displayed on the Text Box named label 3.
- ❖ **<u>When disconnect clicked:</u>** When the disconnect button is clicked, the following series of actions are performed:
  - ▪ The Bluetooth device is disconnected.
  - ▪ The Visibility of Bluetooth device is set to true.
  - ▪ To confirm the disconnection a message, disconnected is read out.



```
when BluetoothScreen .BackPressed
do  open another screen  screenName  " Screen1 "

when BThdevices .BeforePicking
do  call Notifier1 .ShowAlert
                      notice  " Turn Bluetooth on if you haven't. "
    call TextToSpeech1 .Speak
                      message  " Turn Bluetooth on if you haven't. "
    set BThdevices . Elements to  BluetoothClient1 . AddressesAndNames

when BThdevices .AfterPicking
do  set BThdevices . Selection to  call BluetoothClient1 .Connect
                                                    address  BThdevices . Selection
    if    BluetoothClient1 . IsConnected
    then  set BThdevices . Visible to  false
          call TextToSpeech1 .Speak
                            message  " Connected "

when Clock1 .Timer
do  if    BluetoothClient1 . IsConnected  and   call BluetoothClient1 .BytesAvailableToReceive  >  0
    then  set Label3 . Text to  call BluetoothClient1 .ReceiveText
                                              numberOfBytes  call BluetoothClient1 .BytesAvailableToReceive

when Disconnect .Click
do  call BluetoothClient1 .Disconnect
    set BThdevices . Visible to  true
    call TextToSpeech1 .Speak
                      message  " Disconnected "
```

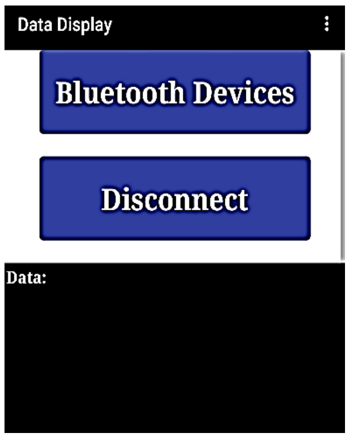


Figure 23: Component Design for Bluetooth Screen

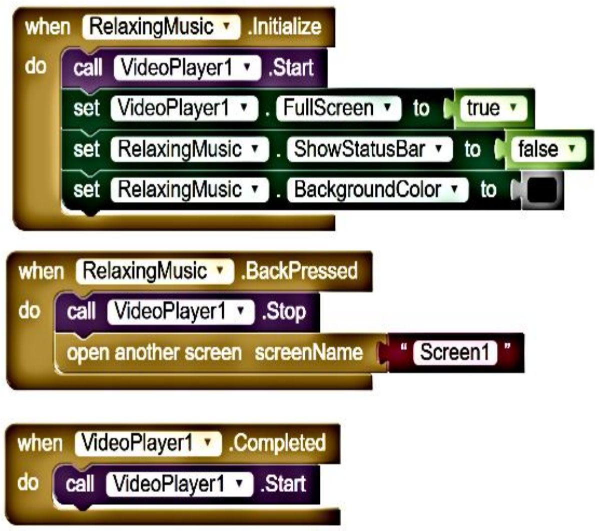
Figure 24: Blocks code for Bluetooth Screen

6.2.3. Breathing Screen:

This screen displays a soothing breathing way to calm ourselves during stressful situations.

- **<u>When RelaxingMusic Initialized:</u>** When this screen is called, initially it calls the video player and sets the screen to fullscreen and hides the status bar and sets the background color to black.
- **<u>When back key is pressed:</u>** When the back key is pressed, it returns to the main screen.
- When the playback is completed, it starts over again and loops over until the back key is pressed.

Figure 26: Video played on Breathing Screen



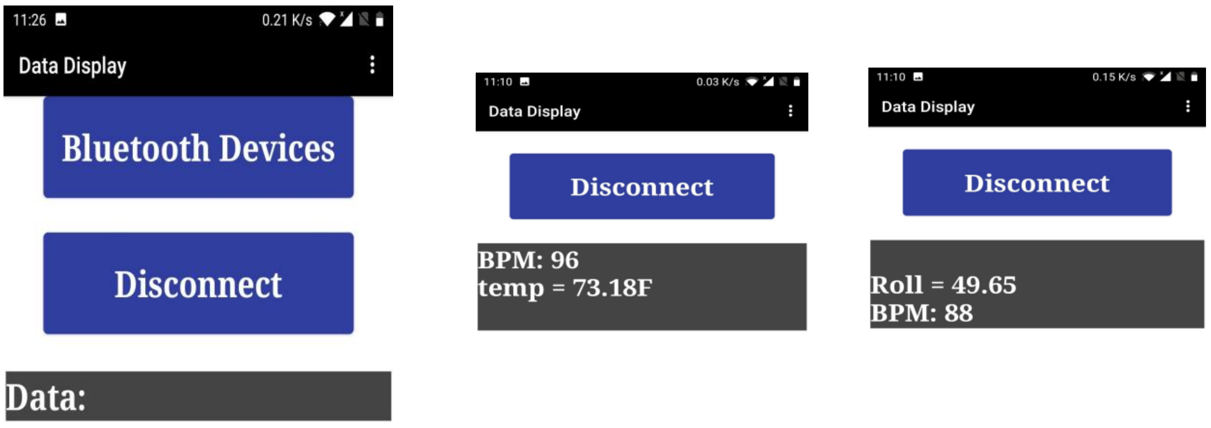
```
when RelaxingMusic .Initialize
do  call VideoPlayer1 .Start
    set VideoPlayer1 . FullScreen to true
    set RelaxingMusic . ShowStatusBar to false
    set RelaxingMusic . BackgroundColor to

when RelaxingMusic .BackPressed
do  call VideoPlayer1 .Stop
    open another screen screenName " Screen1 "

when VideoPlayer1 .Completed
do  call VideoPlayer1 .Start
```

Figure 25: Blocks Code for Breathing Screen

6.2.4. Software Product Comparison:

6.2.4.1. Till Midterm Defense:

- We managed to create two screens: Main Screen, where we could select which feature we wanted to perform and Bluetooth screen, which was for connecting with the hardware and receiving the sent data.
- We were able to display the data sent by the hardware part. The data consisted of heart rate in BPM, roll to measure the movement of the patient while asleep and temperature.



BPM: 96
temp = 73.18F

Roll = 49.65
BPM: 88

Figure 27: Software Product till midterm defense

6.2.4.2. Till Final Defense:

- ❖ We had to remove the temperature reading display as in the final product, we couldn’t fit the temperature sensor.
- ❖ Along with the above-mentioned features, we managed to add one more screen called Breathing. This screen provides a way to relax for a moment and practice calm breathing technique. It plays a video for the user to follow along with the breathing.

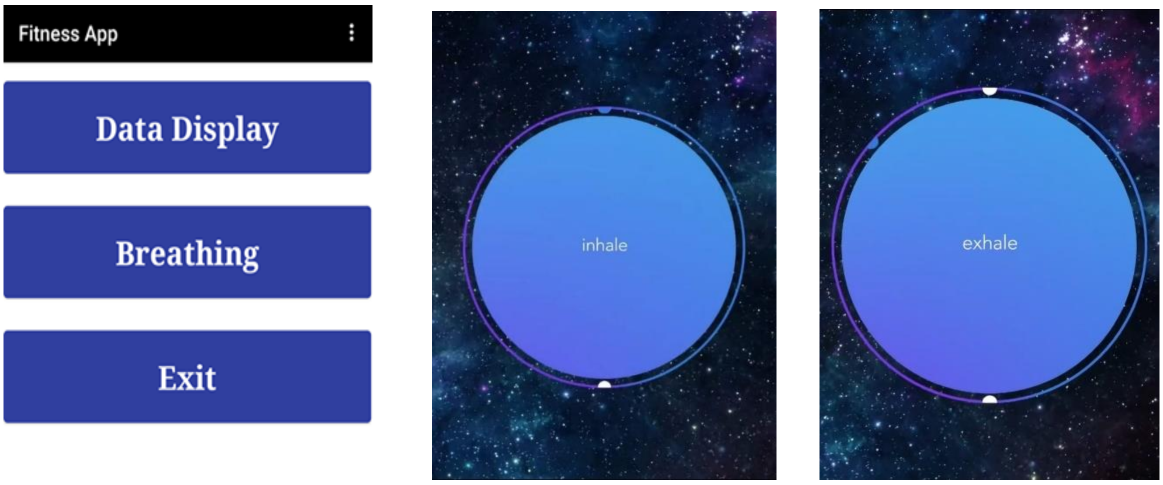


Figure 28: Added features in Software Product till Final defense

6.3. Analysis part:

In analysis part, we at first generated the data set for the standard heartrate curve, then we calculated the correlation between the generated standard data and obtained data from the band which has been stored in a text file, which will be loaded in MATLAB and there, correlation coefficient is calculated. After calculating the correlation coefficient, we get the clear picture of whether the obtained data matches our standard data. Along with this, we also calculated the number of times a person rolls from left to right and vice versa while asleep. This can also be a parameter to check if the patient had trouble sleeping or not.

6.3.1. Generation of Standard Heart Rate Curve:

The sleep time duration for a quality sleep is from 10 pm to 6 am. Thus, for generating a hammock curve, we divided the time axis which is of in total eight hours. We calculated a total of seven reading in an hour, thus giving us, fifty-six readings in eight hours. So, x-axis that is time axis has values from 0 to 55. And ‘0’ indicating 10 pm that is sleep start time and ‘55’ indicating 6 am that is wake up time. Thus, we have some known points as observed from the curve in Figure 1. Points (0,54), (27, 46) and (55, 63) can be obtained from the above figure. The points describe the heart rate at different times. At 10 pm, the heart rate is 53 bpm, at 6 am, the heart rate is 63 bpm and at 2 am, the heart rate is 46 bpm.

Figure 29: Time conversion table

Time in am-pm	Time in x-axis
10 pm	0
11pm	6
12 pm	13
1 am	20
2 am	27
3 am	34
4 am	41
5 am	48
6 am	55

So, by putting the points (0,54), (27, 46) and (55, 63) in the equation $y = ax^2 + bx + c$, we get the values for a, b and c as: $a = \frac{683}{41580}$, $b = -0.7398$ and $c = 54$

So, the equation of the standard curve becomes: $\mathbf{y = \frac{683}{41580}x^2 - 0.7398x + 54}$

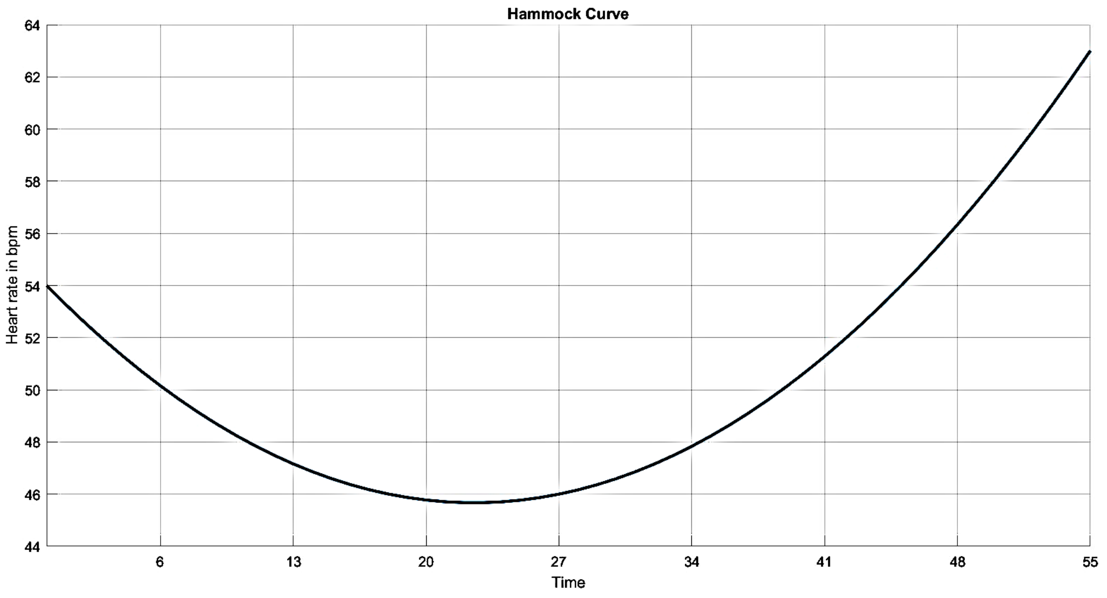


Figure 30: Standard Heart Rate Curve in MATLAB

6.3.2. Correlation coefficient analysis:

6.3.2.1. Loading the bpmData.txt file to MATLAB:

At first, loaded the data file that contained the obtained heartrates in bpm. For this, we browse for the folder where our text file is saved. Then, we select that folder. And, in command window, we write *load bpmData.txt*, now we can use the data in the file as the array "*bpmData*" is created by MATLAB with the data in the file as its elements from 1 to n.

6.3.2.2. Creating a function called "correlation":

Now, we create a function called correlation which takes two data arrays as its input and gives the correlation coefficient as its output. We simply applied the following formula.

$$r = \frac{n\sum XY - \sum X \sum Y}{\sqrt{n\sum X^2 - (\sum X)^2}\ \sqrt{n\sum Y^2 - (\sum Y)^2}}$$

Where, $Sxy = n\sum XY - \sum X \sum Y$, $Sxx = n\sum X^2 - (\sum X)^2$ and $Syy = n\sum Y^2 - (\sum Y)^2$

Therefore,

$$r = \frac{Sxy}{\sqrt{Sxx}\ \sqrt{Syy}}$$

MATLAB code:

```
function [ r ] = correlation( Xs,Xo )
    n = length(Xs);
    sumStd = sum(Xs);
    sumObtd = sum(Xo);
    sumBoth = sum(Xs .* Xo);
    sumStd2 = sum(Xs.^2);
    sumObtd2 = sum(Xo.^2);
    Sxy = n*sumBoth - sumStd*sumObtd;
    Sxx = n* sumStd2 - sumStd.^2;
    Syy = n* sumObtd2 - sumObtd.^2;
    %Correlation Coefficient
    r = Sxy /(sqrt(Sxx) * sqrt(Syy));
end
```

6.3.2.3. Finding correlation coefficient and giving remarks:

Thus, we use the above function to calculate the correlation coefficient between the standard data and obtained data. The MATLAB code is given below and for two data sets, named as GoodBPM and BadBPM, we obtained the following results:

MATLAB code:

```
%Heart Rate Curve
load bpmData.txt;

x=0:1:55;
%Standard data set
a1 = 683/41580;
b1 = -0.7398;
c1 = 54;
std = a1*x.^2+b1*x+c1;

%Obtained data set
obtd= bpmData;

%Correlation Calculation
r = correlation(std,obtd);
display(r);

%Decision Making
if r >= 0.7
    display('Good sleep');
else
    display('Bad sleep');
end
```

Result for GoodBPM:

```
 >> SleepAnalysis

r =

    0.9483

Good sleep
```

Result for BadBPM:

```
>> SleepAnalysis

r =

   -0.8796

Bad sleep
```

6.3.2.3. Deep Sleep time duration calculation:

Deep sleep time duration can be calculated for a dataset that follows the hammock curve, the interval where the heartrate is minimum is considered as the deep sleep time duration. We calculated this in following way:

At first, we check for the dataset which has higher correlation with the standard data, then only we can find the deep sleep duration.

```
if r >= 0.7
    display('Good sleep');
    %deep sleep time duration
    Dtime = mean(find(obtd==min(obtd)));
    display(Dtime);
    DeepSleep(Dtime);
```

Here, the ‘Dtime’ variable finds out the time at which the heart rate is minimum with the help of *find(obtd==min(obtd))* which will give the value of index at which the minimum valued element is located, and if more than on it gives an array of indices. And then it will call DeepSleep function, which is a bunch of if-elseif statements that give us the time duration on the basis of table in figure 29. In DeepSleep( ) function:

```
function [] = DeepSleep(Dtime)
    if ((0<= Dtime) &&(Dtime< 6))
        display('Deep Sleep Duration = 10pm to 11pm');
    elseif ((6<= Dtime) &&(Dtime<13))
        display('Deep Sleep Duration = 11pm to 12pm');
    elseif ((13<= Dtime) &&(Dtime<20))
        display('Deep Sleep Duration = 12pm to 1am');
    elseif ((20<= Dtime) &&(Dtime<27))
        display('Deep Sleep Duration = 1am to 2am');
    elseif ((27<= Dtime) &&(Dtime<34))
        display('Deep Sleep Duration = 2am to 3am');
    elseif ((34<= Dtime) &&(Dtime<41))
        display('Deep Sleep Duration = 3am to 4am');
    elseif ((41<= Dtime) &&(Dtime<48))
        display('Deep Sleep Duration = 4am to 5am');
    elseif ((48<= Dtime) &&(Dtime<55))
        display('Deep Sleep Duration = 5am to 6am');
    end
end
```

Results for GoodBPM.txt:

```
Dtime =

   25.6667

Deep Sleep Duration = 1am to 2am
```

6.3.2.4. Plotting graphs for standard data and obtained data:

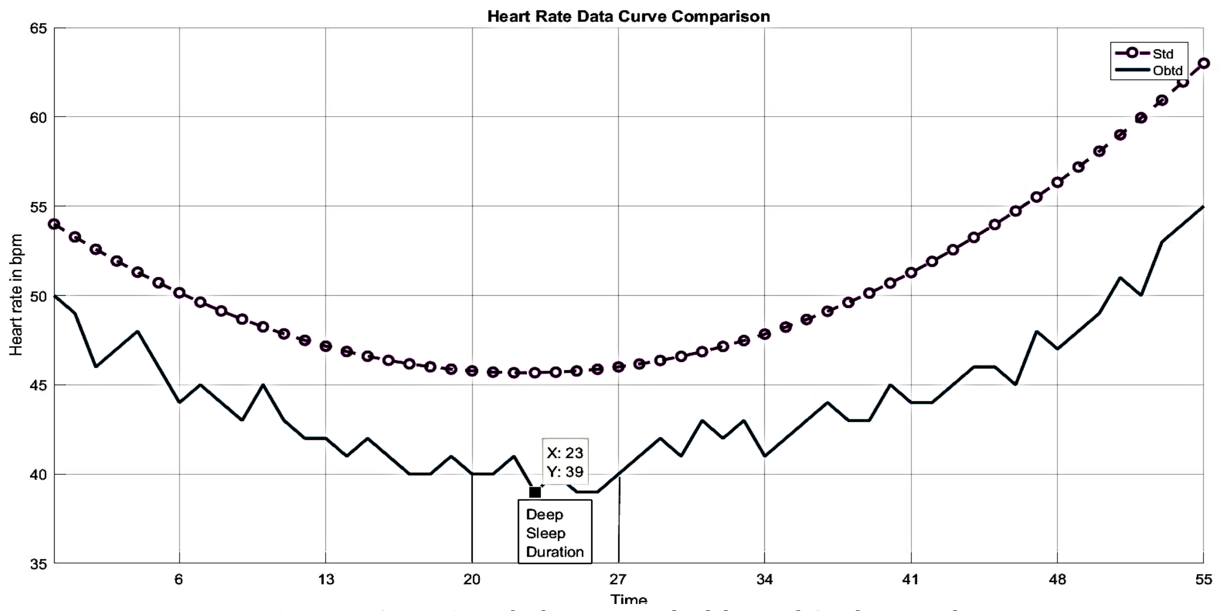


Figure 31: Comparison plot between standard data and GoodBPM.txt data

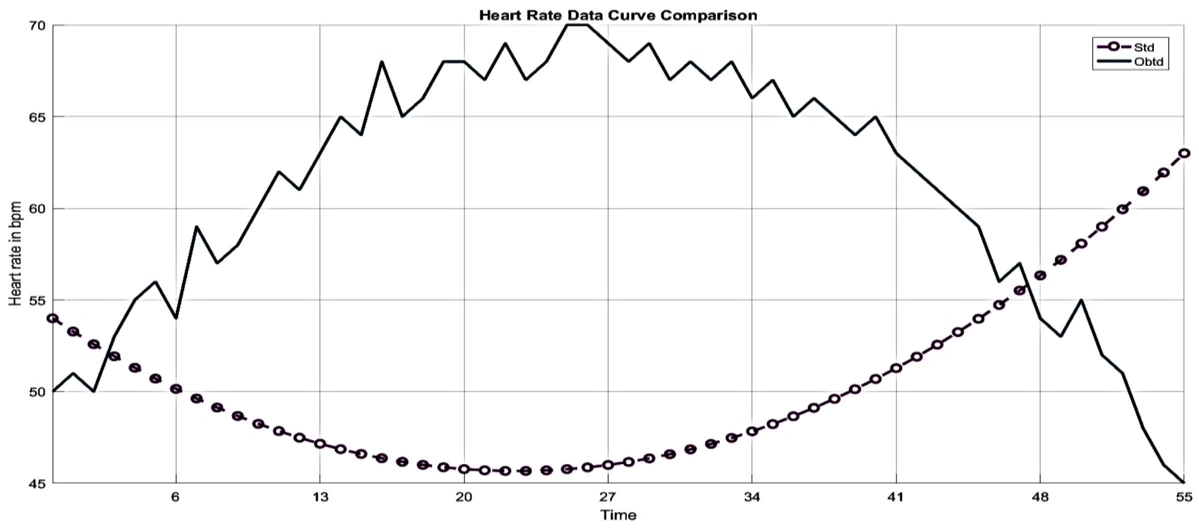


Figure 32: Comparison plot between standard data and BadBPM.txt data

6.3.3. Patient’s Movement while asleep:

For analyzing the movement of a patient while asleep, we have created a variable called “roll” while coding Arduino Mini. This variable depicts the movement of the gyroscope in right or left direction with the help of sign changes. Positive value of the variable “roll” signifies the Gy-521(eventually the person wearing it) is rolling or moving towards right and the negative value signifies the roll towards the left. The magnitude gives the extent of rolling towards left or right. So, to determine how much the patient moves while he/she is asleep, we can check the number of sign changes in a “roll” dataset recorded while the person is asleep. Thus, the patient’s movement will be equivalent to the number of changes in sign in the “roll” data array.

<u>MATLAB code:</u>

```
pos = roll>=0;
changes = xor(pos(1:end-1),pos(2:end));
movementIndex = sum(changes);
display(movementIndex);
```

In MATLAB, at first, we loaded the roll.txt file with 116 data points measured by the band while the person is asleep. The following code is used to calculate the number of sign changes. We set “pos” as 1 if roll>=0 and 0 otherwise. Then “changes” variable is equated to xor operator for each two adjacent elements, and xor acts as follows: xor(1,1)=0, xor(0,0)=0, xor(1,0)=1, and xor(0,1)=1. And the sum of the elements of the “changes” array gives the movementIndex.

For this data of roll.txt, movementIndex = 10 so the movement of the patient while asleep was of 10 times from left to right and vice versa,

To understand this better, we take an array say a = [1 -3 1 4 5] here we know that the number of sign changes is 2. So, we have pos(1:end-1) or pos(1: 5-1) i.e. pos(1:4) = [1 0 1 1 1] as pos is 1 if a >=0 and 0 otherwise. And similarly, we have pos(2:end) or pos(2:5) = [0 1 1 1 1] so now we do xor of these two variables as: changes = [1 1 0 0 0] and therefore, movementIndex = 1+1+0+0+0 = 2.

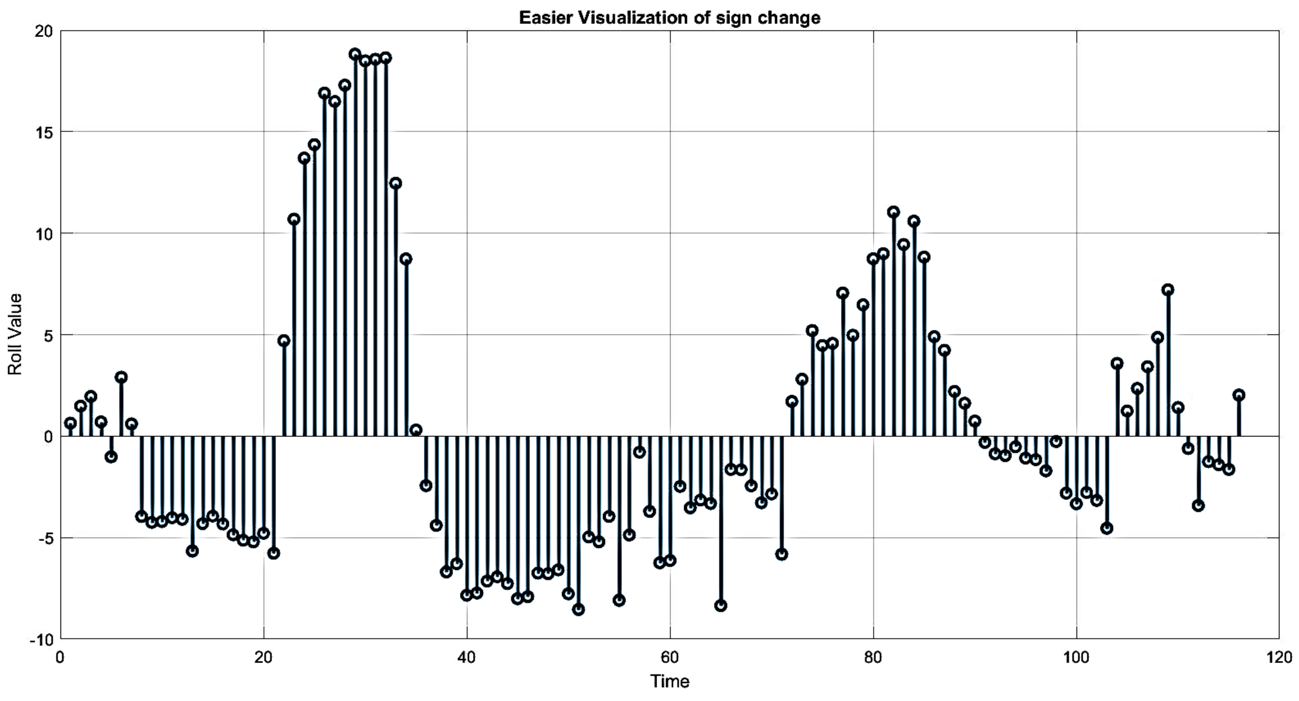


Figure 33: Change in sign of roll variable visualized using a stem graph

7. PROJECT EXPENSES

The expenses incurred till the completion of this project are tabulated below:

S/N	**Components**	**Quantity**	**Cost (Rs)**
1.	Arduino Pro Mini	1	550
2.	MPU-6050	1	400
3.	Pulse Sensor	1	200
4.	HC-06 Bluetooth Module	1	700
5.	USB-to-TTL Serial Connector	1	200
6.	240mAh Battery	1	250
7.	Battery Charger	1	125
8.	Others (Resistors, soldering wire, 3D printing and other materials used in the band)	As per need	500*
9.	Additional cost (Temperature sensor and Microphone)	-	500*
Total Cost excluding NRE cost:			**Rs. 3425***

*May be more or less than estimated.

8. PROBLEMS FACED AND LIMITATIONS

i. Problems Faced:

Some problems faced by us while the course of this minor project are listed below:

- ❖ While choosing the appropriate sensors, we had to make a compromise between cost and accuracy of the sensors.

- ❖ We encountered a problem regarding heart beat sensor as it was difficult to find the heart rate sensor in the market as well as while measuring the heartbeat, the placement of the sensor played a vital role.

- ❖ We wanted to fit a temperature sensor in our gadget but while optimizing its size, we were compelled to remove this sensor.

- ❖ We tried using Flutter for making an application but it turned out to be difficult for use to learn and complete our project so we choose an easier alternative in MIT app inventor.

- ❖ We also wanted to use the microphone with amplifier but it wasn't able to receive the audio unless we placed it near the mouth. But we wanted it to be fitted in a band, so we had to drop the idea of snoring time calculation.

ii. Limitations of the project:

- ❖ Till midterm defense, we had also interfaced temperature sensor with Arduino Uno, but while optimizing the size of the band, we couldn't include it, thus the product is unable to measure temperature and do analysis on the basis of temperature.

- ❖ Analysis of data is done in MATLAB using correlation coefficient comparison and the data of current instant is display in Fitness Companion App. We weren't about to develop a bridge between analysis part and software part; thus, the product is unable to show any kind of data analysis on our android application, but only displays the graphs and sleep quality results in MATLAB.

9. CONCLUSIONS

In hardware part, the final product is presented in the form of wrist band which comprises all the hardware components interfaced within it. The ATmega328 microcontroller Arduino pro mini made the miniature form possible with its small size. Along with that the layered placement of hardware components made the system more compact and portable. The wrist band is powered with a rechargeable battery and the data about the pulse reading in BPM and movement data can be obtained with a simple Bluetooth connection.

In software part, the app was designed using MIT app inventor with the primary goal of displaying the data of the band. The app works with Android phones and connects through Bluetooth to the HC-06 module of the Arduino Pro Mini. The data is sent serially from the Arduino and the app displays the data using a timer updating every 1000 milliseconds. There is also another menu item called “Breathing” that helps the patient who is unable to sleep or under stress by guiding effective breathing technique.

In analysis part, we used correlation coefficient to check whether the standard and obtained heart rate data have good positive correlation or not. And thus by that determined the quality of sleep. And similarly, plotted the graph for standard and obtained data to visualize and compare the standard and obtained data. Also, we calculated the deep sleep time duration, and can easily identify the region in the plot using Time conversion table. Along with this, we also calculated the movement index which gave the number of times a patient roll left from right or vice versa, indicating the movement of the patient while asleep. We also plotted the graph of the roll data to visualize the sign changes and thus movement index.

Therefore, we were able to make a sleep quality analyzing gadget using Arduino Pro Mini in which we measured heartbeat per minute using heart beat sensor and the movement of the patient while asleep using accelerometer and gyroscope sensor. Then, we were able to develop an android application using MIT app inventor which reflected the measurements done by the gadget. We were able to make the gadget portable and wearable. And finally, we also analyzed the measured data for heartrate and roll variable indicating the movement of a person while asleep.

References:

- Interfacing gyroscope to Arduino. GY521 module based on MP6050. (2019). Retrieved from http://www.circuitstoday.com/interfacing-gyroscope-arduino
- author: M. (2019). Pulse Sensor With Arduino Tutorial. Retrieved from https://www.instructables.com/id/Pulse-Sensor-With-Arduino-Tutorial/
- Arduino lesson – HC-06 Bluetooth Module « osoyoo.com. (2019). Retrieved from http://osoyoo.com/2017/10/25/arduino-lesson-hc-06-bluetooth-module/
- Heart Rate While Sleeping: Look for These 3 Patterns | Oura Ring. (2019). Retrieved from https://ouraring.com/heart-rate-while-sleeping/
- Apple Watch. (2019). Retrieved from https://en.wikipedia.org/wiki/Apple_Watch
- Heart Rate While Sleeping: Look for These 3 Patterns | Oura Ring. (2019). Retrieved from https://ouraring.com/heart-rate-while-sleeping/
- The Science Behind Oura | Oura Ring. (2019). Retrieved from https://ouraring.com/the-science-behind-oura/
- Smart sleep analysis. (2019). Retrieved from https://phys.org/news/2013-08-smart-analysis.html
- Xiaomi Mi Band (2019). Retrieved from https://en.wikipedia.org/wiki/Xiaomi_Mi_Band

10. APPENDIX: Code for Arduino Pro Mini

```
//Pulse Sensor
// Set-up low-level interrupts for most accurate BPM math.
#define USE_ARDUINO_INTERRUPTS true
#include <PulseSensorPlayground.h>     // Includes the PulseSensorPlayground Library.
PulseSensorPlayground pulseSensor;  // Creates an instance of the PulseSensorPlayground object called
"pulseSensor"
//Gyro + Accelerometer
#include <Wire.h>
//Bluetooth HC06
#include <SoftwareSerial.h>
SoftwareSerial HC06(8,9); //(Rx,Tx)
// Pulse Sensor Variables
const int PulseWire = 3;   // PulseSensor PURPLE WIRE connected to ANALOG PIN 0
int Threshold = 500;    // Determine which Signal to "count as a beat" and which to ignore.
//MPU6050 variables:
long accelX, accelY, accelZ;
float Ax, Ay, Az;
float p=0,r=0;  //movement
//for average
unsigned long previousMillis = 0;
const long interval = 6000;
int totalSamples = 0;
int count = 0;
int value;
int calculatepulseValue()
{
     int myBPM = pulseSensor.getBeatsPerMinute();
     totalSamples+= myBPM;
     count += 1;
     return totalSamples/count;
}
void heartbeat()
{
unsigned long currentMillis = millis();
int totalSamples = 0;
int count = 0;
if (pulseSensor.sawStartOfBeat())
  {
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    value = calculatepulseValue();
     Serial.println("BPM = "+String(value));
  }
  }
}
void setup(void)
{
  Serial.begin(9600);
 // Configure the PulseSensor object, by assigning our variables to it.
  pulseSensor.analogInput(PulseWire);
  pulseSensor.setThreshold(Threshold);
  pulseSensor.begin();
  HC06.begin(9600);//Bluetooth Module
  Wire.begin();// Start up the library for MPU6050
  setupMPU();
}
void loop()
{
   heartbeat();
   sleepMovement();
   HC06.print("BPM : ");
   HC06.println(value);
   HC06.print("ROLL : ");
   HC06.println(r);
}
void setupMPU()
{
  Wire.beginTransmission(0x68);   //This is the I2C address of the MPU (b1101000/b1101001 for AC0
low/high)
  Wire.write(0x6B);                        //Accessing the register 6B - Power Management
  Wire.write(0b00000000);           //Setting SLEEP register to 0.
  Wire.endTransmission();
  Wire.beginTransmission(0x68);   //I2C address of the MPU
  Wire.write(0x1B);                        //Accessing the register 1B - Gyroscope Configuration
  Wire.write(0x00000000);          //Setting the gyro to full scale +/- 250deg./s
  Wire.endTransmission();
  Wire.beginTransmission(0x68);   //I2C address of the MPU
  Wire.write(0x1C);                     //Accessing the register 1C - Accelerometer Configuration
  Wire.write(0b00000000);          //Setting the accel to +/- 2g
  Wire.endTransmission();
}
void sleepMovement()
{
   //Update Acc data:
  Wire.beginTransmission(0x68);            //I2C address of the MPU
  Wire.write(0x3B);                                //Starting register for Accel Readings
  Wire.endTransmission();

  Wire.requestFrom(0x68,6);                  //Request Accel Registers (3B - 40)
  while(Wire.available() < 6);
  accelX = Wire.read()<<8|Wire.read();    //Store first two bytes into accelX
  accelY = Wire.read()<<8|Wire.read();    //Store middle two bytes into accelY
  accelZ = Wire.read()<<8|Wire.read();    //Store last two bytes into accelZ
  //Process acclerometer data:
  Ax = accelX / 16384.0;
  Ay = accelY / 16384.0;
  Az = accelZ / 16384.0;
  Ax *= 9.8;
  Ay *= 9.8;
  Az *= 9.8;
  p = atan2 (Ay ,( sqrt ((Ax * Ax) + (Az * Az))));
  r = atan2(-Ax ,( sqrt((Ay * Ay) + (Az * Az))));
  p = p*57.3;
  r = r*57.3;
  Serial.println("\nRoll = "+ String(r) );
  delay(500);
}
```