MiBEAT : The Innovative Wearable Technology

**Project By:**

Kuljeet Singh

Sahil Saini

Anshul Sanan

A technical report submitted in partial fulfillment of the requirements for

Computer Engineering Course

School of Applied Technology

Humber College of Applied Arts and Technology

24 January, 2018

# Declaration of Joint Authorship

I, Kuljeet Singh affirm that this work submitted for appraisal on the portability and software requirements of the hardware components, connection of the drivers & sensors to that of the Android application and output to be displayed is my own and is communicated in my own words. Any utilizations made inside it of crafted by some other creator, in any structure (thoughts, conditions, figures, writings, tables, programs), are legitimately recognized at the point of utilization. A list of the references used is also incorporated.

I, Anshul Sanan affirm that this work submitted for appraisal on the Android application to make it usable on Android device, display the readings retrieved from the database and used in application to produce a meaningful output is my own and is communicated in my own words. Any utilizations made inside it of crafted by some other creator, in any structure (thoughts, conditions, figures, writings, tables, programs), are legitimately recognized at the point of utilization. A list of the references used is also incorporated.

I, Sahil Saini affirm that this work submitted for appraisal such as attaching all the hardware components sensors to the Raspberry Pi Zero and configuring the hardware to store the variables and values on a remote server, Firebase and also configuring the I2C and SPI drivers of the sensors with the Raspberry Pi Zero is my own and is communicated in my own words. Any utilizations made inside it of crafted by some other creator, in any structure (thoughts, conditions, figures, writings, tables, programs), are legitimately recognized at the point of utilization. A list of the references used is also incorporated.

# Proposal

2019-01-17

***Proposal for the development of MiBEAT***

Prepared by Sahil Saini, Kuljeet Singh, Anshul Sanan  
*Computer Engineering Technology Students*

**Executive Summary**

As the students in the Computer Engineering Technology program, we will be integrating the knowledge and skills learned from our program into this MiBEAT themed capstone project. This proposal requests the approval to build the hardware portion that will connect to a database as well as to a mobile device application. The internet connected hardware will include a custom PCB with the IS31FL3731 144-LED CharliePlex driver (0x74), Pulse Sensor Amped, ADS1015 12-Bit ADC - 4 Channel with Programmable Gain Amplifier. The database will store Measured values for heart rate and number of steps travelled. In addition it willl also store user login information and user profile. The mobile device functionality will include the feature of re-examining his/her readings from last 7 captures. Readings here refer to heart rate and total distance travelled. It will also include feature of calculating total number of calories burnt and will be further detailed in the mobile application proposal. We will be collaborating with Humber’s School of Health Science. The hardware was completed in CENG 317 Hardware Production Techniques independently and the application was completed in CENG 319 Software Project. These will be integrated together in this term in CENG 355 Computer Systems Project as a 3-student group.

**Background**

This aim of this project is to create a wearable that can make people more responsible towards their health and help the medical science as well. It is our inexpensive approach so that everyone can own this device and lead a heathier lifestyle. Our project consists of a Pulse Sensor Amped which will further send the analog signal which is received by another sensor called ADS1015 12-Bit ADC. It converts the analog signal to a digital signal and further projecting it to the Charlieplexed PWM LED Matrix to aware the user of any change in heart rate.

In the Computer Engineering Technology program, we have learned about the following topics from the respective relevant courses:

* Java Docs from CENG 212 Programming Techniques in Java,
* Construction of circuits from CENG 215 Digital and Interfacing Systems,
* Rapid application development and Gantt charts from CENG 216 Intro to Software Engineering,
* Micro computing from CENG 252 Embedded Systems,
* SQL from CENG 254 Database with Java,
* Web access of databases from CENG 256 Internet Scripting; and,
* Wireless protocols such as 802.11 from TECH152 Telecom Networks.

This knowledge and skill set will enable us to build the subsystems and integrate them together as our capstone project.

**Methodology**

This proposal is assigned in the first week of class and is due at the beginning of class in the second week of the fall semester. Our coursework will focus on the first two of the 3 phases of this project:  
 Phase 1 Hardware build.  
 Phase 2 System integration.  
 Phase 3 Demonstration to future employers.

*Phase 1 Hardware build*

The hardware build was completed in the fall term. It fits within the CENG Project maximum dimensions of 12 13/16" x 6" x 2 7/8" (32.5cm x 15.25cm x 7.25cm) which represents the space below the tray in the parts kit. The highest AC voltage that is used is below 16Vrms from a wall adaptor from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption is less than 20 Watts.

*Phase 2 System integration*

The system integration will be completed in this term.

*Phase 3 Demonstration to future employers*

This project will showcase the knowledge and skills that we have learned to potential employers.

The brief description below provides rough effort and non-labour estimates respectively for each phase. A Gantt chart will be added by week 3 to provide more project schedule details and a more complete budget will be added by week 4. It is important to start tasks as soon as possible to be able to meet deadlines.

**Concluding remarks**

This proposal presents a plan for providing an IoT solution for 0. This is an opportunity to integrate the knowledge and skills developed in our program to create a collaborative IoT capstone project demonstrating my ability to learn how to support projects such as the initiative described by [3]. We request approval of this project.

# Abstract

The report discusses the MiBeat, a wearable technology as a possible substitute to modern expensive devices which are already in market. It explains the fabrication, software & hardware requirements, working and future scope of the project. Our project’s focus is to find a solution for growing obesity and diseases due to lack of motion. Here is where the smartwatch comes to play. This project can also help people with cardiovascular problems helping them to take precautions if necessary. The hardware consists of a LED Matrix which displays the status to the user which is also stored in a remote database. An android application retrieves this data and is utilized by the application for any future use. The objective of this project is to help people who are struggling to stay fit and people with heart problems without any big expenses in expensive technology.

Table of Content

Contents

[Declaration of Joint Authorship 3](#_Toc7108463)

[Proposal 4](#_Toc7108464)

[Abstract 8](#_Toc7108465)

[List of Illustrations 11](#_Toc7108466)

[Introduction 13](#_Toc7108467)

[Project Description 13](#_Toc7108468)

[Requirement Specification 15](#_Toc7108469)

[1) Software Requirements: 15](#_Toc7108470)

[a) Mobile Application: 15](#_Toc7108471)

[b) Database: 15](#_Toc7108472)

[2) Hardware Requirements: 15](#_Toc7108473)

[a) Development Platform: 15](#_Toc7108474)

[b) Interface boards and Sensors: 16](#_Toc7108475)

[c) Other Hardware and Enclosures: 16](#_Toc7108476)

[Build Instructions 17](#_Toc7108477)

[1) Equipment/Components 17](#_Toc7108478)

[2) Raspberry Pi initialization and Image creation 17](#_Toc7108479)

[3) Mechanical Assembly and Connections 20](#_Toc7108480)

[4) Install and update python libraries on Raspberry 21](#_Toc7108481)

[5) Soldering 22](#_Toc7108482)

[6) Enclosure 26](#_Toc7108483)

[7) Testing 27](#_Toc7108484)

[Android Application 31](#_Toc7108485)

[1) Programming Language and Working Studio 31](#_Toc7108486)

[2) Database 34](#_Toc7108487)

[Conclusion 37](#_Toc7108488)

[References 39](#_Toc7108489)

[Appendix-I 41](#_Toc7108490)

## List of Illustrations

[Figure 1 Raspbian being installed 19](#_Toc7108611)

[Figure 2 - Cable connections for installing operating system 20](#_Toc7108612)

[Figure 3 All the components joined together 22](#_Toc7108613)

[Figure 4 Various images of Soldering 25](#_Toc7108614)

[Figure 5 The final working project 25](#_Toc7108615)

[Figure 6 Installation of Case under-progress 26](#_Toc7108616)

[Figure 7 Enclosed Hardware 26](#_Toc7108617)

[Figure 8 testing Blinka files 42](#_Toc7108618)

# Introduction

## Project Description

This aim of this project is to create a wearable that can make people more responsible towards their health and help the medical science as well.

It is our inexpensive approach so that everyone can own this device and lead a heathier lifestyle.

We will be integrating the knowledge and skills learned from our program like coding and embedded system design into this MiBEAT themed capstone project. Also, the internet connected hardware will include a custom PCB with the IS31FL3731 144-LED CharliePlex driver (0x74), Pulse Sensor Amped, ADS1015 12-Bit ADC - 4 Channel with Programmable Gain Amplifier.

The database will store Measured values for heart rate and number of steps travelled. In addition it willl also store user login information and user profile. The mobile device functionality will include the feature of re-examining his/her readings from last 7 captures. Readings here refer to heart rate and total distance travelled. It will also include feature of calculating total number of calories burnt.

In all it will satisfying our objective of making the user responsible and aware towards his/her health.

Size is the most important and also the most problematic factor in the development the project. The wearable electronic product can only succeed in the market if its design is convenient and simple.

We overcame this obstacle with a unique approach. In spite of making it a pity looking watch we boldly changed our device to a wearable glove.

# Requirement Specification

## Software Requirements:

### Mobile Application:

**Anshul** will work on the **Android** application to make it usable on Android device and display the readings retrieved from the database. **Sahil** will configure the **I2C** and **SPI** connections of the sensors with the **Raspberry Pi Zero W. Kuljeet** will connect drivers of the sensors to that of the Android application and will control output to be displayed.

### Database:

Anshulwill be responsible retrieving values from the database and be used in application to produce a meaningful output. Sahil will configure the hardware to store the variables and values on a local or remote server, possibly **Firebase.** Kuljeet will look after the encryption, decryption and binding of the hardware with the database.

## Hardware Requirements:

### Development Platform:

Sahil will attach all the hardware components such as sensors to the Raspberry Pi Zero W**.** Anshul will troubleshoot any problems occurring with the installation. Kuljeet will work on the portability and software requirements of the hardware components.

### Interface boards and Sensors:

Our project consists of a **Pulse Sensor** Amped which will further send the analog signal which is received by another sensor called **ADS1015 12-Bit ADC**. It converts the analog signal to a digital signal and further projecting it to the **Charlieplexed PWM LED Matrix** to aware the user of any change in heart rate. These sensors will be mounted on Raspberry Pi using a PCB board.

### Other Hardware and Enclosures:

Since the project at final stage will be wearable, hence we decided to make it in the form of a glove. Some other hardware like Clips, Velcro and Battery will be used to support the basic framework.

# Build Instructions

## Equipment/Components

We need Raspberry Pi Zero, Pulse Sensor, and Analog to Digital converter (ADS1015), Seven Breadboard Jumper cables, female to female jump wires, Charliplex LED sensor

This project can be completed in 3 days.(Required all parts and necessary tools are already in hand).Sometimes it may take only 2 days where as sometimes it takes more than a month to ship the product. Generally, it should take around 7-8 hours to finish (taking into account that the PCB has been printed effectively, generally the PCB should take about a half-day to be printed). The major steps in the process are:

* [Raspberry Pi and Noobs installation](https://github.com/kuljeet-Singh/charli0x74#Raspberry-Pi-initialisation-and-Image-creation) (2.5 hours)
* [Other Installations, Connections and Verifications](https://github.com/kuljeet-Singh/charli0x74#Other-Installations-Connections-and-PowerUp) (2 Hours)
* [Soldering](https://github.com/kuljeet-Singh/charli0x74#Soldering) (1.5 Hours)

## Raspberry Pi initialization and Image creation

Using NOOBS is the easiest way to install Raspbian on your SD card. To get hold of a copy of NOOBS, visit [www.raspberrypi.org/downloads/](https://www.raspberrypi.org/downloads/)

You should see a box with a link to the NOOBS files. Click on the link.

The simplest option is to download the zip archive.

If the SD card on which you wish to install Raspbian currently has an older version of Raspbian on it, you may wish to back up the files from the card first, as they will be overwritten during this process. Now, visit the SD Association’s website and download [SD Formatter 4.0](https://www.sdcard.org/downloads/formatter_4/index.html) for Windows or Mac. Follow the instructions to install the software. Insert your SD card into the computer or laptop’s SD card reader and make a note of the drive letter allocated to it.

In SD Formatter, select the drive letter for your SD card, and format it. Now, **Extract NOOBS from the zip archive.** Next, you will need to extract the files from the NOOBS zip archive you downloaded from the Raspberry Pi website. Then, go to Downloads folder and find the zip file you downloaded. Extract the files and keep the resulting Explorer/Finder window open. Now open another Explorer/Finder window and navigate to the SD card. It’s best to position the two windows side by side. Select all the files from the NOOBS folder and drag them onto the SD card. Eject the SD card. **Booting from NOOBS,** once the files have been copied over, insert the micro SD Card into your Raspberry Pi, and plug the Pi into a power source. You will be offered a choice when the installer has loaded. You should check the box for **Raspbian**, and then click **Install**. Click **Yes** at the warning dialog, and then sit back and relax. It will take a while, but Raspbian will install.

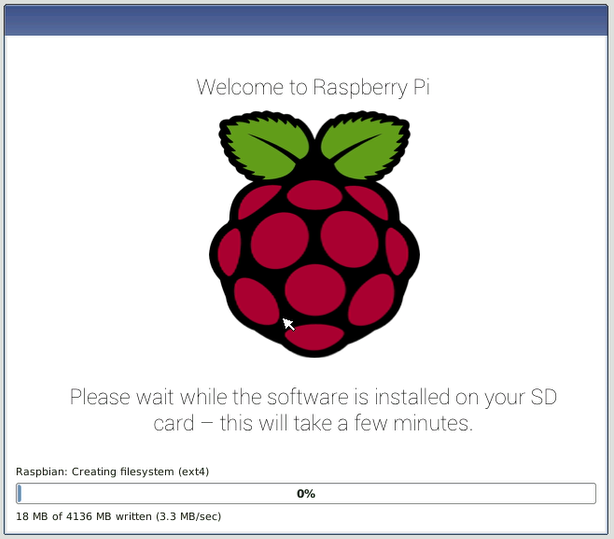


Figure 1 Raspbian being installed

When Raspbian has been installed, click **OK** and your Raspberry Pi will restart and Raspbian will then boot up. Download the latest version of [**NOOBS** OS](https://www.raspberrypi.org/downloads/noobs/). Downloading the software takes a while. Click on the link for step by step [video](https://www.raspberrypi.org/help/videos/#noobs-setup) to flash the image of OS to your micro-SD card. When downloading is done connect your Raspberry Pi to a screen and plug a mouse and keyboard to it. Once setup is done, just enable I2C, VNC and SSH interfaces.

This can be done by selecting Preference from Start Menu and then clicking Raspberry Pi configuration and then select Interfaces and now set I2C, SSH and VNC to enable mode.



Figure 2 - Cable connections for installing operating system

## Mechanical Assembly and Connections

Here's the how the Raspberry Pi wired to with I2C, Connect

* 3V3 to ADS1015, Pulse Sensor and LED’s VIN
* GND to ADS1015, Pulse Sensor and LED’s GND

The ADS1015 and Charliplex LED uses I2C communication protocol so,

* SCL to ADS1015, LED’s SCL
* SDA to ADS1015, LED’s SDA
* Then connect the pulse sensor to the ADC on channel A0 (make sure to check the channel in the code below).

Once I2C is detected we are ready for next step.

## Install and update python libraries on Raspberry

Now create a new file using nano editor by name it as a ".py" file as you you did earlier and write following code into it

* import board
* import busio
* import adafruit\_is31fl3731
* display = adafruit\_is31fl3731.Matrix(i2c)
* When the display initializes it will go through and clear each frame (there are 8 frames total) of the display. You might see the display momentarily flash and then turn off to a clear no pixel lit image.  
  You can control all of the board's pixels using the fill function. Send to this function a value from 0 to 255 where 0 is every LED pixel turned off and 255 is every LED pixel turned on to maximum brightness. For example to set all the pixels to half their brightness run:
* display.fill(127)

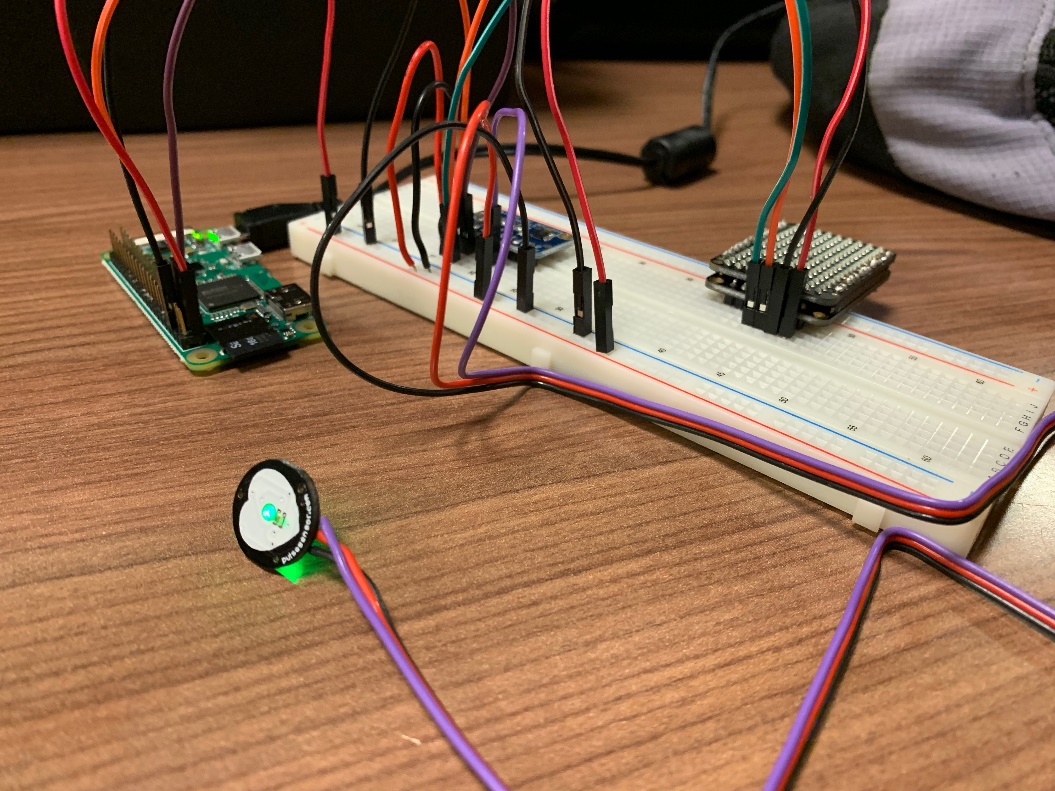


Figure 3 All the components joined together

## Soldering

Soldering the LED's matrix and driver together. This is required before any further step. Also solder headers to ADC and PCB.

###### Soldering Iron Safety

* Never touch the element or tip of the soldering iron. They are very hot (about 400°C) and will burn.
* Hold wires to be heated with tweezers or clamps.
* Keep the cleaning sponge wet during use.
* Always return the soldering iron to its stand when not in use. Never put it down on your workbench.
* Turn unit off or unplug it when not in use.

###### Work Safely with Solder, Flux and Cleaners

* Wear eye protection. Solder can "spit".
* Use lead free solder.
* Keep cleaning solvents in dispensing bottle to reduce inhalation hazards.
* Always wash your hands with soap and water after soldering.
* Read and understand the Safety Data Sheets (SDS) for all materials before beginning work.

###### Dangers of Lead Exposure

* Lead on your skin can be ingested and lead fumes can be given off during soldering. Other metal fumes can also be hazardous. Lead can have serious chronic health effects, such as reproductive problems, digestive problems, nerve disorders, memory and concentration problems, muscle and joint pain.

###### Avoid Toxic Fumes

* Work in a well-ventilated area. The smoke formed is mostly from the flux which can be irritating, a sensitizer and aggravates asthma. Avoid breathing it by keeping your head to the side of, not above, your work.
* A benchtop fume extractor may be necessary to remove harmful fumes caused by solder and flux from the soldering workstation by filtering the air.

###### Reduce Risk from Electricity

* Always use a grounded outlet and grounding prong to reduce the risk of electrical damage if a short circuit occurs in the equipment.
* Prevent damage to electrical cords during soldering. Keep them away from heated tips.

###### Fire Prevention

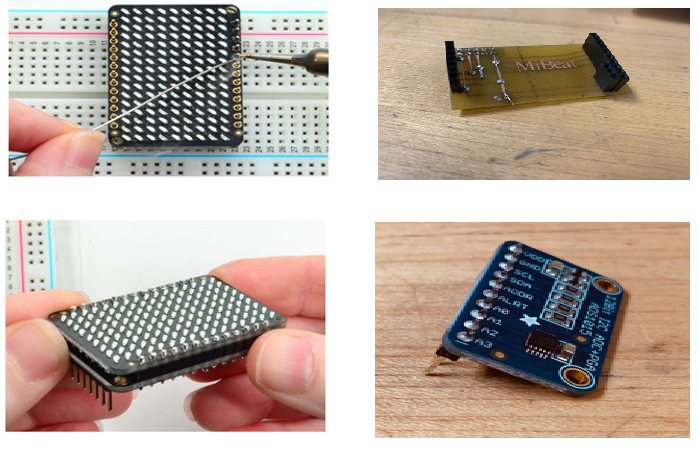
* Work on a fire-proof or nonflammable surface that is not easily ignited.
* Wear nonflammable or 100% cotton clothing that covers your arms and legs to help prevent burns.
* Know where your fire extinguisher is and how to use it.

###### First Aid

* Immediately cool the affected area under cold water for 15 minutes.
* Do not apply any creams or ointments. Cover with a band-aid.
* Seek medical attention if the burn covers an area bigger than 3 inches across.

###### Waste

* Discard lead and silver solder and dross in a container with a lid.
* Label the container: "Lead (Silver) Solder Waste for Recycling".
* Used solder sponges and contaminated rags must be disposed of as hazardous waste.

Figure 4: Various images of Soldering

Final look of soldered PCB attached to Raspberry Pi all sensors connected to it looks like this:

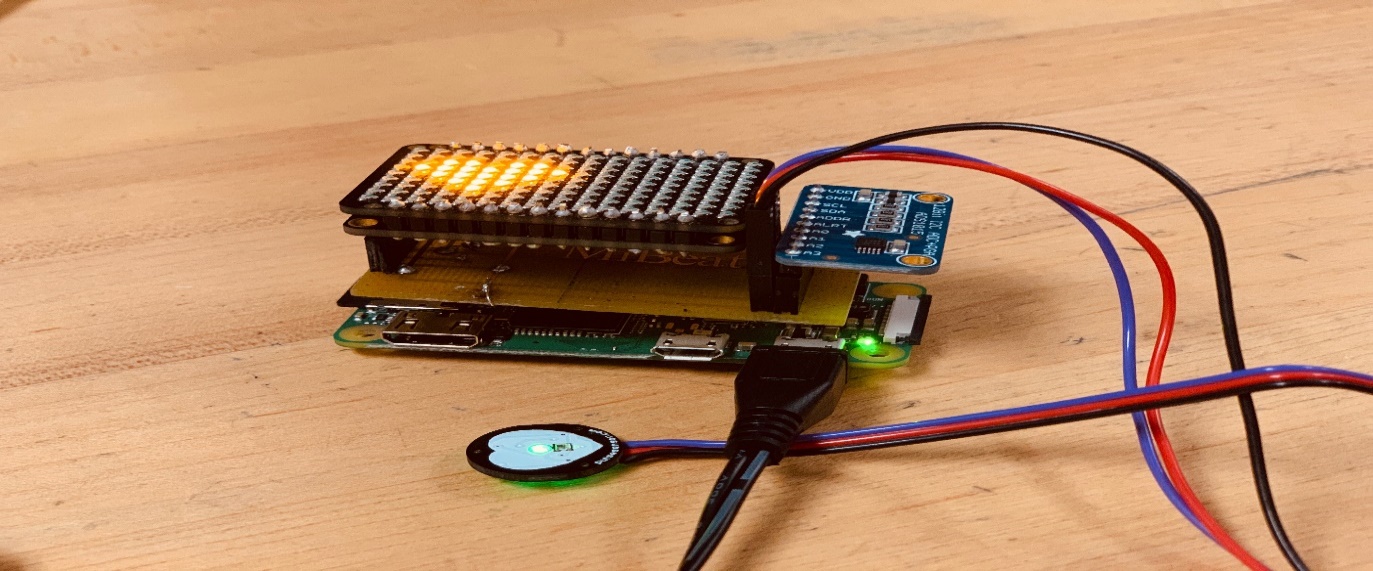


Figure 5 The final working project

## Enclosure

The software used to make framework of soft-file is known as Coral draw. It is a highly professional tool hence difficult to use. After watching some videos, we were able to make a case but it did not work due to improper positioning of holes. We contacted prototype lab, they helped us to design a perfect case.

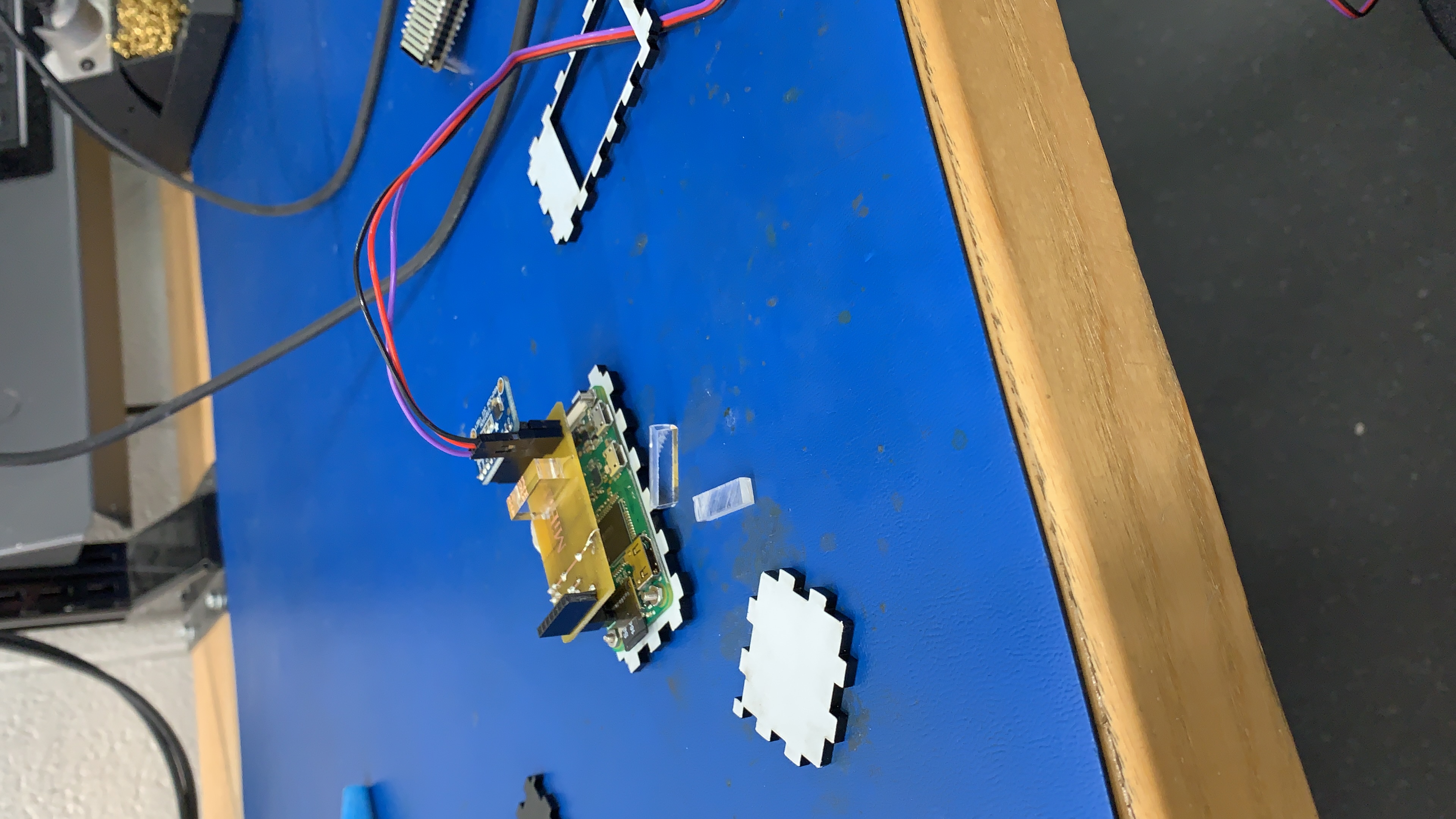


Figure 6 Installation of Case under-progress

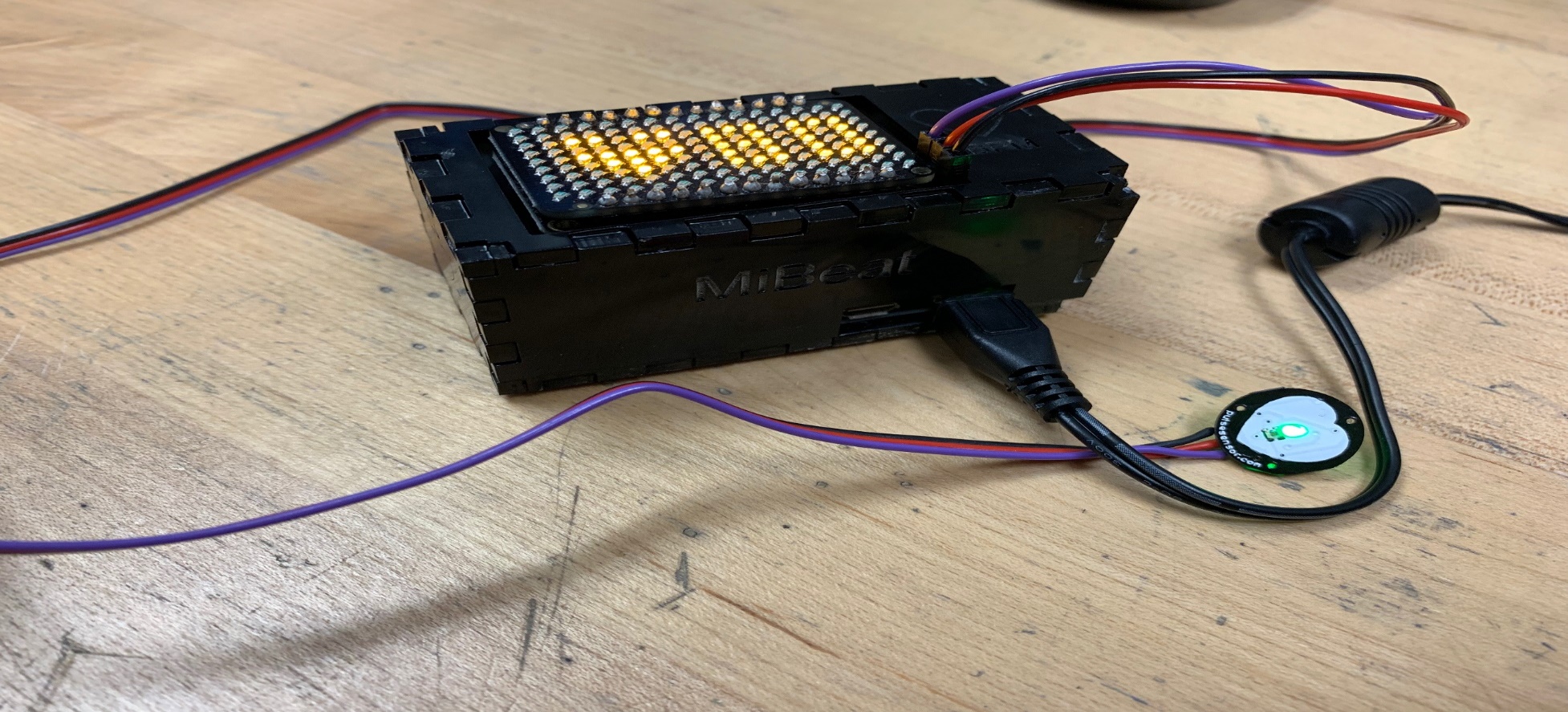


Figure 7 Enclosed Hardware

## Testing

If we want this product to produce on large scale and then the testing of this hardware can be done. Soldering part can be easily done by the machines on very fast pace.

As from top to bottom of this page we have elaborated each and every step clearly and in very easy way. So to reproduce the same hardware should not be much difficult now. You just have to focus on the instructions we gave and the instructions in the links that I have provided. You should not miss any step while installing anything.

Personal Testing   
PowerUp the Raspberry Pi. Go to terminal check for I2C detect again as we did earlier.  
If detected run the "board.py" file. And matrix should light up.  
If not, check that all connctions are tightly made and "VIAs" are properly soldered.

Production Testing   
Since soldering and PCB desiging are major causes for errors, when they will be done on industrial skale with high professional skills adavannced machines I think it will eliminates a lage probability of error.  
One out every 100 devices should be checked at personal level.

Test Case:

|  |  |
| --- | --- |
| **Test Case ID** | [The ID of the test case.]  TC-01 |
| **Test Case Summary** | [The summary / objective of the test case.]  To check data retrieval |
| **Prerequisites** | [Any prerequisites or preconditions that must be fulfilled prior to executing the test.]  User must be a valid[registered] user |
| **Test Procedure** | [Step-by-step procedure to execute the test.]   1. Login 2. Measure readings using sensors 3. Press any of the four buttons to retrieve data. |
| **Test Data** | [The test data, or links to the test data, that are to be used while conducting the test.]  HeartRate (75 Bpm), Calories Burnt (120 Cal), Steps walked (1500), distance travelled (1 km) |
| **Expected Result** | [The expected result of test.]  When the following buttons are pressed it should display following things:  Heart Rate = < timestamp > 75 Bpm  Calories Burnt = < timestamp > 120 Cal  Steps Walked = < timestamp > 1500  Distance travelled = < timestamp > 1 Km |
| **Actual Result** | [The actual result of the test; to be filled after executing the test.]  If result matches above value database is connected  If not then report a problem. |
| **Status** | Pass or Fail. Other statuses can be ‘Not Executed’ if testing is not performed and ‘Blocked’ if testing is blocked. |
| **Remarks** | [Any comments on the test case or test execution.]  This is a sample test. |
| **Created By** | [The name of the author of the test case.]  Team Smart Health Tracker |
| **Date of Creation** | [The date of creation of the test case.]  12/13/2018 |
| **Executed By** | [The name of the person who executed the test.]  Team Smart Health Tracker |
| **Date of Execution** | [The date of execution of the test.]  12/13/2018 |

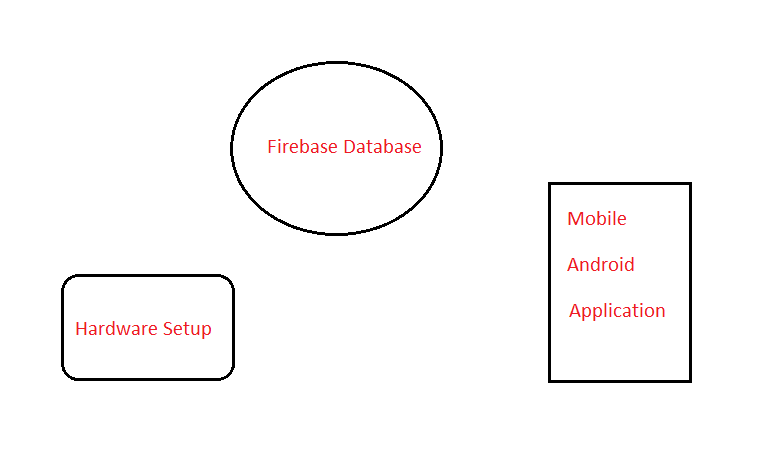
# Android Application

## Programming Language and Working Studio

The Android application consists of several language including Java, Kotlin, JSON, and Python. The android application consists of several XML files, Java classes encapsulated in a single program named Android Studio. Android Studio supports all the same programming languages of [IntelliJ](https://en.wikipedia.org/wiki/IntelliJ) and has several important functions, [Gradle](https://en.wikipedia.org/wiki/Gradle) - based build support, [Kotlin](http://kotlinlang.org/) programming language support, [Android profiler](https://developer.android.com/studio/preview/features/android-profiler.html): Memory, CPU, network; [Java 8](http://www.oracle.com/technetwork/java/javase/overview/java8-2100321.html) language features built-in, quick build instructions, device file explorer; [Android Instant Apps](https://developer.android.com/topic/instant-apps/index.html) support; [adaptive icon wizard](https://developer.android.com/guide/practices/ui.../icon_design_adaptive.html), XML and downloadable fonts, [Android Things](https://developer.android.com/things/hardware/index.html) support, layout editor improvements, APK profiling and debugging, Layout inspector improvements, Improved Gradle sync speed, AAPT2 is now enabled by default, [Firebase app](https://firebase.google.com/) indexing assistant, App links assistant and many more making it a reliable option for developing an Android application.

In the meantime, it screens your pulse and fills you in regarding whether it identifies something of concern. This item is a part of portable health gadgets which will demonstrate heart beat and level of oxygen in blood (using oximeter), the application will likewise contain a pedometer to check the separation strolled day by day. The market of compact/potable health devices is our significant target. Our point is to make an application which will accumulate information from the sensors/gadget, process it and transfer to our database and will incorporate reports on day by day movement and heart.

The sensor records reading which represents to the electrical heartbeats that make your heart beat. The application checks these heartbeats to get your pulse and check whether they are healthy. Strangely high or low pulses could be indications of a serious condition. Be that as it may, numerous individuals don't perceive the manifestations, so the basic causes regularly go undiscovered. SHT checks your heart and cautions you to these abnormalities.

Our sensor will send data to the database and app will retrieve data from there and display them to the user. Oximetry Sensor placed inside the Smart Health Tracker will send the readings or data to the database and app will fetch data from there. In general, there is no direct contact or communication between the hardware and software part. For now, every reading is fake and is being retrieved from database. But, in next phase (next semester) we will only be fetching distance travelled and heart rate measured from database and will be calculating number of steps walked and calories burnt locally. 

This software will work on Android Jelly Bean and above and not IOS. Using this system is fairly simple and intuitive. A user familiar with basic mobile using skills should be able to understand all functionality provided by the system. A first-time user of the mobile application should see the Home page which will be displaying “Get started button”. Then a log-in page will display when they proceed through the application. If the user has not registered, they will get a link to do that on the log-in page. If the user is not a first time user, First main screen is opened which ask various options like heart-beat, distance walked and distance to be walked.

System features:

User Registration: The user should be able to register through the mobile application. The user must provide user-name, password and e-mail address.

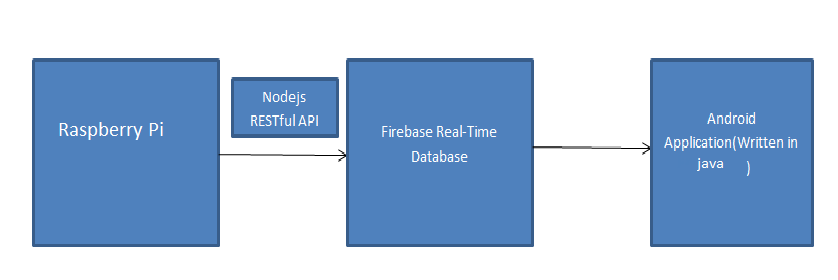
User login: Given that a user has registered, then the user should be able to log in to the mobile application.

Profile-Page: On the mobile application, the user will have a profile page. On the profile page a client can alter his/her data.

Performance Requirement: The information will be store locally and whenever the user will sign out and sign in again all the information will be pulled out from the DB and all the reading recorded on the device will be saved on the DB.

Security Requirements: According to the COOPA(Children's Online Privacy Protection Act) Children under the age of 13 can use the product only on the supervision of the parent, the password used to sign in will require to be at least 8 character long and include at least one capital letter, one number and one special character. To use the product only a sign up is required.

1. Data structure
2. Database design, including the authentication and database structure
3. Plan to implement your hardware design/code to your database structure



According to our architecture, if we take an example of a simple sensor whose output needs to be processed real-time in a Mobile App. First through an hosted Node-js RESTful API, Arduino(or any other hardware with a Wi-Fi module) sends the required data to Firebase real-time database which is then stored.

**Login Screen**

I’d had a struggle with designing a login/sign up screen for the Android app before I stumbled upon this excellent design concept. So, I decided to implement the idea. Apart from making this splendid interaction, I want to make the code reusable, and to separate the main concerns. That is, my login screen should know nothing about the sign up screen and vice versa.

The use of Background image, Sign up screen, Login screen, Gap at the edges of each screen, Input fields, Logo, bottom buttons and login and sign up labels.

In my opinion it is not a significant effort to interpret these components in terms of Android Views. I also used a ViewPager to switch between them. The background, logo, and buttons are simple ImageView objects.

**Fingerprint Sensor**

Finger print authentication helper is an android library to simplify the authentication with finger print process. This will provide a common platform for finger print authentication for all the android versions.

To use the Android Fingerprint API you must:

* Create a new or load an existing SecretKey
* Create a Cipher with a created or loaded SecretKey
* Create a CryptoObject with a created Cipher
* Start Fingerprint authentication with a created CryptoObject
* Handle possible exceptions at every step due to complexity of the Android Fingerprint API

The CryptoObject is locked when created and it is unlocked when the user successfully authenticates. Once it is unlocked, you can use it to cipher data.

Fingerprint authentication is used to either:

1. Authenticate the user, e.g. for payment
2. Perform encryption or decryption operations over user’s case-sensitive information, e.g. passwords

FingerPrint Sensor wraps everything mentioned and provides an intuitive and easy-to-use interface.

Feedback and code contributions are very much welcome. Just make a pull request with a short description of the changes. By making contributions to this project you give permission for the code to be used under the same [license](https://github.com/infinum/Android-Goldfinger/blob/master/LICENSE).

## Database

The Android application uses a remote database structure provided by Google Inc. for the users, individuals and big corporations for easy accessibility to the data with low or no extra costs making it reliable for everyone. Firebase has several features making it an optimum choice for everyone including Real-time Database, Hosting, Authentication, Storage, Cloud Messaging, Remote Configuration, Test Lab, Crash Reporting, Notifications, App Indexing, Dynamic Links, and Invites. It also included AdWords and AdMob making it optimum for developers for making income without the need of in-app purchases. The real-time database helped in updating and making changes to the database in the real time without the need to wait for the specialized ticketing system in the discreet database. Also, it supports many microcomputers and ARM – based microprocessors to connected at the same time.

Firebase Remote Config is a cloud service that lets you change the behavior and appearance of the app without requiring users to download an app update. When using Remote Config, you create in-app default values that control the behavior and appearance of the app. Then, you can later use the Firebase console or the Remote Config REST API to override in-app default values for all app users or for segments of the user base. The app controls when updates are applied, and it can frequently check for updates and apply them with a negligible impact on performance.

Remote Config includes a client library that handles important tasks like fetching parameter values and caching them, while still giving you control over when new values are activated so that they affect the app's user experience. This lets you safeguard the app experience by controlling the timing of any changes.

The Remote Config client library get methods provide a single access point for parameter values. The app gets service-side values using the same logic it uses to get in-app default values, so you can add the capabilities of Remote Config to the app without writing a lot of code.

To override in-app default values, you use the Firebase console or the Remote Config REST API to create parameters with the same names as the parameters used in the app. For each parameter, you can set a service-side default value to override the in-app default value, and you can also create conditional values to override the in-app default value for app instances that meet certain conditions. This graphic shows how parameter values are prioritized in the service and in the app.

**Problems in Firebase Connection:**

**In the starting:-**

Previously, I had tried to connect my project on Android studio to Firebase database. While trying to do so I have added some code lines to the app gradle file. Then I got an error that I can't solve for a week. Although this error I can run the app on the emulator so I continued the process of getting connected to Firebase. After I finished that, I have tried to add something to the database- and no success. The error was fixed using the line “implementation 'com.android.support:support-v4:28.0.0' in the application.

**During the Process:-**

I had a problem in reading , pushing and retrieving data to and from the Firebase database which was further solved using the specific troubleshooting process and implementation of tests services in the Android Studio .

**Debugging Process:**

Android studio is much clever. It understands context in which you are working (because it's built over IDEA). And for now it is stable enough to work with. I switched IDE from eclipse to IDEA Android Studio, in the early process and then continued with it. It allows better debugging of the application and object troubleshooting aiding the user in better performance.

# Code:

// You should use the CancellationSignal method whenever your app can no longer process user input, for example when your app goes  
 // into the background. If you donâ€™t use this method, then other apps will be unable to access the touch sensor, including the lockscreen!//  
  
 private CancellationSignal cancellationSignal;  
 private Context context;  
  
 public FingerprintHandler(Context mContext) {  
 context = mContext;  
 }  
  
 //Implement the startAuth method, which is responsible for starting the fingerprint authentication process//  
  
 public void startAuth(FingerprintManager manager, FingerprintManager.CryptoObject cryptoObject) {  
  
 cancellationSignal = new CancellationSignal();  
 if (ActivityCompat.checkSelfPermission(context, Manifest.permission.USE\_FINGERPRINT) != PackageManager.PERMISSION\_GRANTED) {  
 return;  
 }  
 manager.authenticate(cryptoObject, cancellationSignal, 0, this, null);  
 }  
  
 @Override  
 //onAuthenticationError is called when a fatal error has occurred. It provides the error code and error message as its parameters//  
  
 public void onAuthenticationError(int errMsgId, CharSequence errString) {  
  
   
 Toast.makeText(context, "Authentication error\n" + errString, Toast.LENGTH\_LONG).show();  
 }  
  
 @Override  
  
 //onAuthenticationFailed is called when the fingerprint doesnâ€™t match with any of the fingerprints registered on the device//  
  
 public void onAuthenticationFailed() {  
 Toast.makeText(context, "Authentication failed", Toast.LENGTH\_LONG).show();  
 }  
  
 @Override  
  
 //onAuthenticationHelp is called when a non-fatal error has occurred. This method provides additional information about the error,  
 //so to provide the user with as much feedback as possible Iâ€™m incorporating this information into my toast//  
 public void onAuthenticationHelp(int helpMsgId, CharSequence helpString) {  
 Toast.makeText(context, "Authentication help\n" + helpString, Toast.LENGTH\_LONG).show();  
 }@Override  
  
 //onAuthenticationSucceeded is called when a fingerprint has been successfully matched to one of the fingerprints stored on the userâ€™s device//  
 public void onAuthenticationSucceeded(  
 FingerprintManager.AuthenticationResult result) {  
 Toast.makeText(context, "Success!", Toast.LENGTH\_LONG).show();  
 FingerprintHandler.this.context.startActivity(new Intent(FingerprintHandler.this.context,MainScreen.class));  
 }  
  
}

# Conclusion

The report discussed MiBeat, a wearable technology as a possible substitute to modern expensive devices which are already in market. It explained the fabrication, software & hardware requirements, working and future scope of the project. Our project’s focus was to find a solution for growing obesity and diseases due to lack of motion. Here is where the smartwear comes to play. This project also helped people with cardiovascular problems by helping them to take precautions if necessary. The hardware consists of a LED Matrix which displayed the status to the user, which also stored in a remote database. An android application retrieved this data and then utilized for any future use. The objective of this project is to help people who are struggling to stay fit and people with heart problems without any big expenses in expensive technology.

In future we are looking towards making it more compact and easily portable. In addition, we also want to add Bluetooth connectivity to it. Power supply will be replaced by battery.

# References

1. Raspberry Pi Heartbeat / Pulse measuring. (n.d.). Retrieved from <https://tutorials-raspberrypi.com/raspberry-pi-heartbeat-pulse-measuring/>
2. Adafruit Industries. (n.d.). Pulse Sensor Amped. Retrieved from <https://www.adafruit.com/product/1093>
3. Udayan Kumar. (2016, May 17). Heart Rate monitor using Raspberry Pi and Pulse sensor. Retrieved from <http://udayankumar.com/2016/05/17/heart-beat-raspberry/>
4. Python Code Examples. (n.d.). Retrieved from <https://www.pythonforbeginners.com/code-snippets-source-code/python-code-examples>
5. Learn Java - Free Interactive Java Tutorial. (n.d.). Retrieved from <https://www.learnjavaonline.org/>
6. Raspberry Pi Heartbeat / Pulse measuring. (n.d.). Retrieved from <https://tutorials-raspberrypi.com/raspberry-pi-heartbeat-pulse-measuring/>
7. Adafruit Industries. (n.d.). Adafruit 15x7 CharliePlex LED Matrix Display FeatherWings. Retrieved from <https://www.adafruit.com/product/2965>
8. Friesen, J., & Friesen, J. (2019, February 14). Tutorial series: Android Studio for beginners. Retrieved from <https://www.javaworld.com/article/3340234/tutorial-series-android-studio-for-beginners.html>
9. (n.d.). Retrieved from <https://www.w3schools.com/Xml/xml_examples.asp>
10. Googlesamples. (2018, February 16). Googlesamples/android-FingerprintDialog. Retrieved from <https://github.com/googlesamples/android-FingerprintDialog>
11. Azzola, F. (2018, January 18). Fingerprint app: How to use fingerprint authentication in Android. Retrieved from <https://www.survivingwithandroid.com/2016/12/android-fingerprint-app-authentication-tutorial.html>

# Appendix-I

These are the few terms or codes used during the development of the project.

Python Files.

Update your Pi and python

1. sudo apt-get update
2. sudo apt-get upgrade
3. sudo pip3 install --upgrade setuptools

Once you're done with both and have rebooted, verify you have the I2C and SPI devices with the command

1. ls /dev/i2c\*

Install Python libraries

1. pip3 install RPI.GPIO
2. pip3 install adafruit-blinka

Then create a new named bilnka.py using nano text editor:

import board

import digitalio

import busio

print("Hello blinka!")

# Try to great a Digital input

pin = digitalio.DigitalInOut(board.D4)

print("Digital IO ok!")

# Try to create an I2C device

i2c = busio.I2C(board.SCL, board.SDA)

print("I2C ok!")

# Try to create an SPI device

spi = busio.SPI(board.SCLK, board.MOSI, board.MISO)

print("SPI ok!")

print("done!")

Then run this file

python3 blinkatest.py

Your output should look similar to this. If SPI is not working; it’s not a problem because we are not using it.

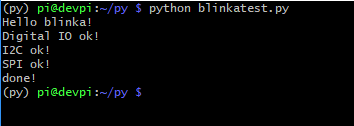


Figure 8 testing Blinka files