Final Design Specification for Pulse Fitness Tracker

April 10th, 2021

ENEL 400 W2021

REJOY - Team 24

Team Members: Rafal Dzwonek, Emma Tholl, Joey Ah-kiow,
Omar Al-Ghabari, Yazan Chama

Instructor: Denis Onen

TA: King Ma

Table of Contents

Introduction	**3**
Predecessor Work	**3**
System Block Diagram	**5**
Block Description	**7**
Engineering Analysis	**19**
Enclosure	**20**
Printed Circuit Board	**20**
Regulatory Codes	**21**
Design Alternatives Considered	**22**
Future Work	**23**
References	**24**
Appendices	**26**

1. Introduction

This report contains a summarized effort of the design solution, as well as similar existing designs of our prototype was based on and possible future additions/improvements to our existing design. Demonstrated in this report is the connections between the arduino module and the power, sensors, display, data storage and rotary encoder blocks, followed by the software block diagram and the respective explanations. The report also contains an analysis/consideration of the development process, including why and how features were incorporated and the research/methodology involved in the implementation of the design along with the PCB layout. The report is concluded with the regulatory codes, alternative design solutions and future works.

2. Predecessor Work

There are many products that contain the same or similar features to the Pulse Fitness Tracker. The Apple Watch and FitBit are well known products that are popular amongst all consumers and we used an electronic watch that tracks body functions as a foundation for our design. When initially developing the vision for our design, we wanted to include some useful features that are expected to be on a fitness watch such as heart rate, number of steps and the time of day and from there we would incorporate a feature that is not as commonly used, in this case it was the oximeter sensor, where only recent or unreleased models are able to measure blood-O2 saturation levels.

Figure 1: Fitbit Device

Figure 2: Apple Watch

Unlike Apple and other larger fitness watch manufacturers, our design must consider the limited resources, so we also received inspiration from “do it yourself” projects. The heart of our design is based on an Arduino Nano 33 BLE and many similar, arduino based designs are available on the internet including different algorithms used to count steps, which he had integrated aspects of into our design.

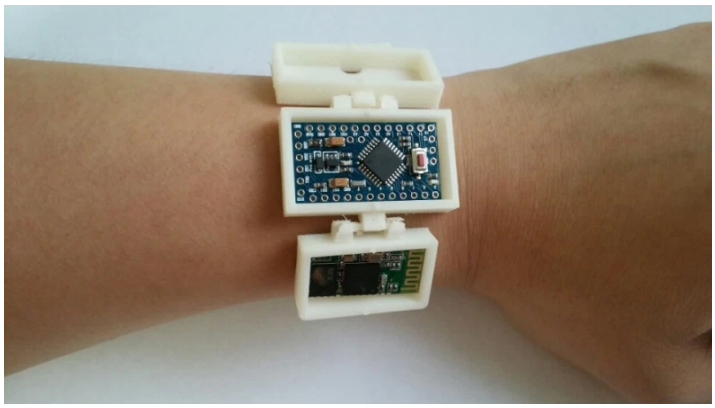


Figure 3: Predecessor Work Prototype[1]

3. System Block Diagram

3.1 Hardware

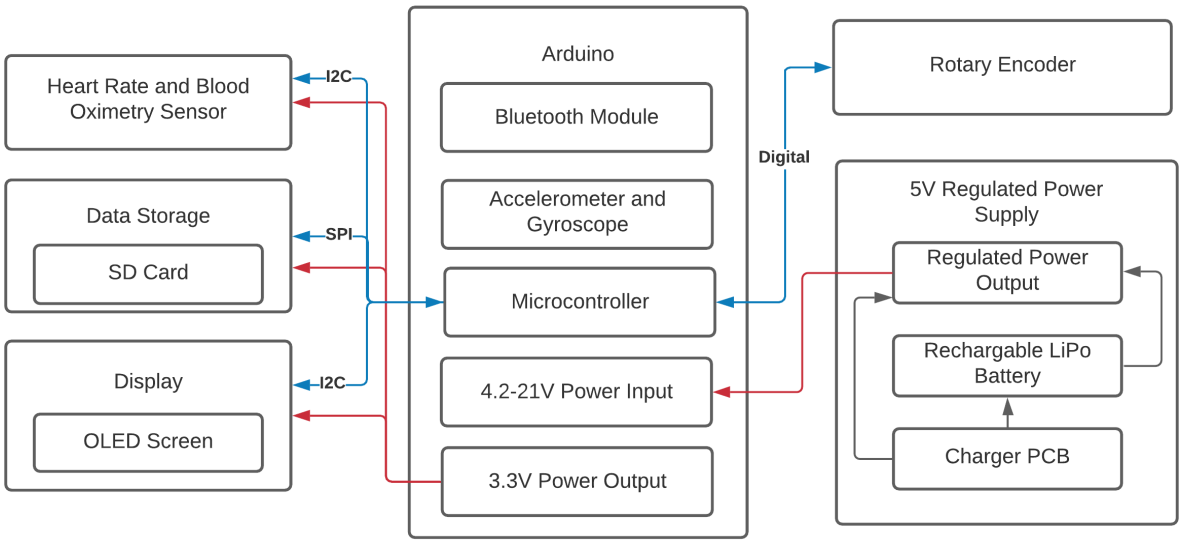


Figure 4: High Level Hardware Block Diagram

The main high level block of our system is the Arduino board. It contains the microcontroller of the device and as such, communicates and powers the rest of the device. It also contains an on-board accelerometer and gyroscope sensor and a bluetooth module, both of which are necessary for our device. Since it is a 3.3V board and requires 3.3V logic, it also has the required circuitry to step-down and regulate the 5V input down to 3.3V to power the sensors and display.

The power supply block consists of a LiPo battery and a charger PCB. The LiPo battery has a nominal voltage of 3.6-3.7V and must be stepped up to fit within the input voltage constraints of our board, which is accomplished by the charger PCB. The charger PCB is responsible for both stepping up and regulating the output voltage and being able to charge the battery by simply connecting a mini-USB cable into the device.

The remainder of the blocks consist of various IO components such as a display, a SD card module, a heart rate and blood oximetry sensor and a rotary encoder. The display and sensor, and SD card module utilize the I^2C and SPI protocols respectively while the rotary encoder utilizes digital IO, requiring 3 IO pins.

The heart rate and blood oximetry sensor is a critical part of our device, providing the heart rate and blood-oxygen data required for the core functionality of the fitness tracker. The SD card module's purpose is to act as a data logger when the watch is not connected to an external device, such that the data can later be transmitted when the watch becomes in range of the external device/bluetooth is enabled. The rotary encoder's purpose is for user input for functions

such as turning bluetooth on/off. Finally, the display block's purpose is to provide a device UI to provide basic information such as time and to interact with the rotary encoder.

3.2 Software

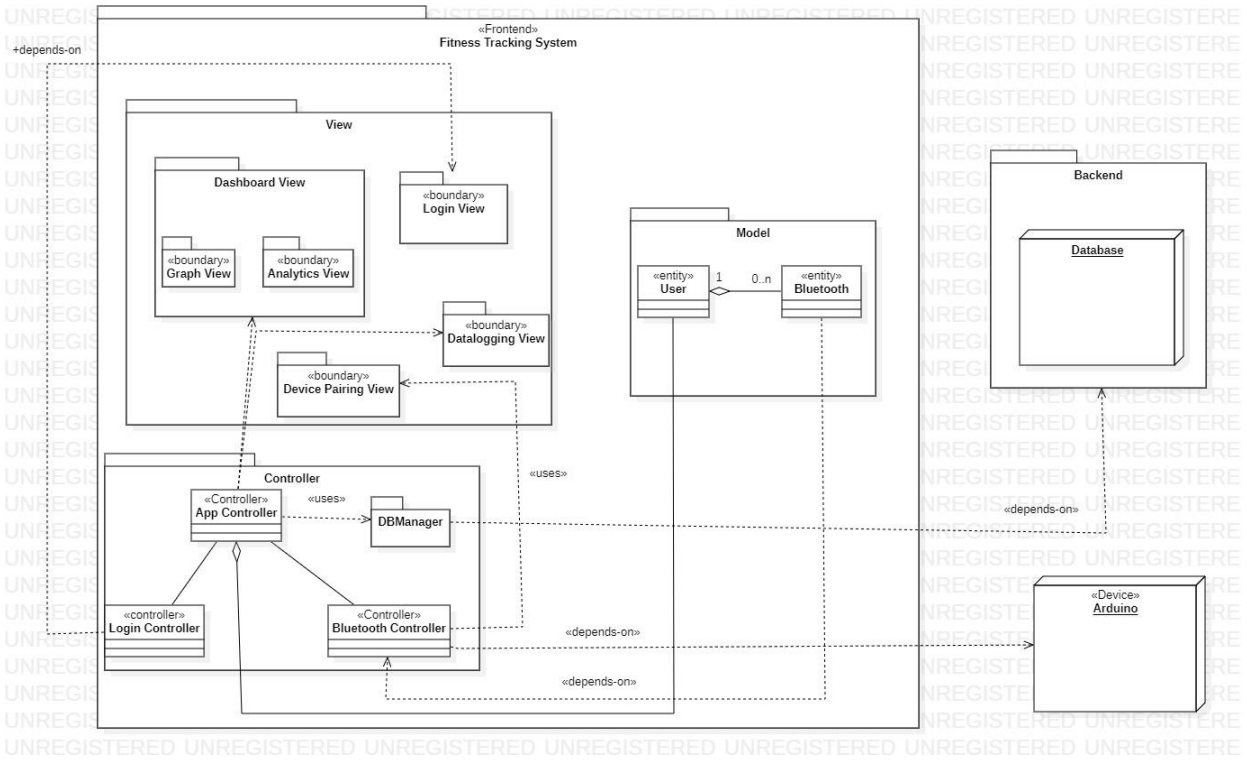


Figure 5: High Level Software Diagram

Our Software System is split up into 3 main packages. The Fitness Tracking System (Frontend), the Backend, and the Device.

The Frontend is composed of three sub-packages which employs a MVC (Model-View-Controller) design pattern. The frontend is built using React JS, which is a lightweight Javascript library that allows for painless and interactive UI (User Interface). The view package is what displays all the information to the end-user, such as the Login Page, The Dashboard, Pairing a device, and also data that was logged. To display said data, the view relies on the Controller to do so. The Controller employs all the business logic of the frontend (ie. it is the brains of all operations), from getting the data from the database, logging in, and interfacing with the Arduino via bluetooth. This module relies on the Model Package, to keep information persistent during a session of the application, such as the bluetooth device, and user information, so that a query to the database isn't required each time user information is required.

The Backend package is used to keep track of all user information and analytics. It uses a NoSQL database to store, and read information on the web-application.

4. Block Description

4.1 Hardware

4.1.1 Arduino

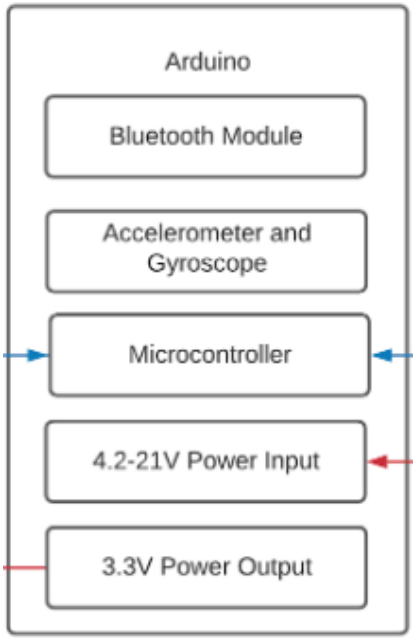


Figure 6: Arduino Block

The specific arduino board used is the Arduino Nano 33 BLE Sense. We chose this specific board due to its various integrated components that are required for our device, such as an accelerometer and gyroscope sensor and a BLE module. The board operates using 3.3V logic and has the required power electronics to step-down a 4.2V - 21V input down to 3.3V, which we used to power all the other components of our system.

The microcontroller on this board is the ARM-based nRF52840, which has all the necessary features required for this project such as I^2C and SPI communication capabilities and enough GPIO[2]. The accelerometer/gyroscope sensor was utilized to develop a step tracking feature. This was done by frequently polling data from the sensor and implementing an algorithm which utilizes this data to translate it into steps taken. The BLE module was used to transmit data from the device to an external device, to be displayed on the web application.

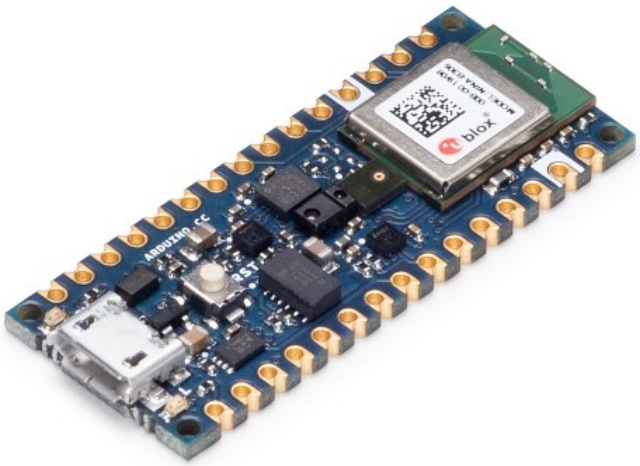


Figure 7: Arduino Nano 33 BLE Sense

4.1.2 Power Supply

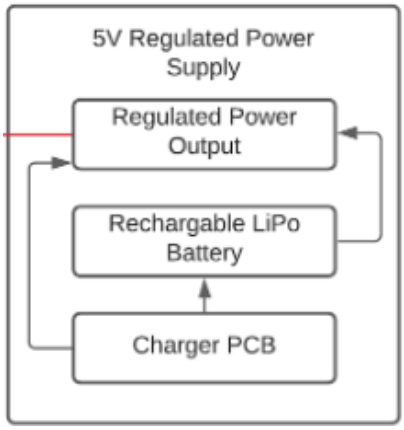


Figure 8: Power Supply Block

The power supply block consists of 2 components, a 400 mAh LiPo battery, with a nominal voltage of 3.7V(3.0V-4.2V), and the Adafruit PowerBoost 500c charger board. The charger circuit has three main connections, one to the battery through a JST connector, one to the Arduino through 5V and GND pins and one mini USB port, to charge the battery. This charger PCB was chosen for two main reasons. The first reason being that LiPo batteries require special care when charging, which this board can provide. They require a system called CC/CV charging. The charger will keep the current constant until the battery reaches its peak voltage(4.2V). Then it will maintain that voltage, while reducing the current down to zero[3]. If multiple battery cells are used, cell balancing is also a factor but only 1 cell is used here so this will not be discussed.

The second reason is the ability to charge the battery without turning off the device/removing the battery from the device. The charger board we are using has the capability to switch its power

input from and to the battery when a mini USB cable is connected, meaning that it is possible to both charge the battery and power the device while the USB cable is plugged in. Once the USB cable is removed, the power output of the board will automatically switch to the battery. The power output of the charging board, coming from either the USB cable or battery, is 5V regulated power which fits the input constraints of the arduino. As previously mentioned, the arduino then takes care of the step-down to 3.3V to power the other components.

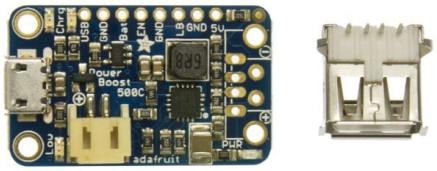


Figure 9: Adafruit PowerBoost 500c Charging Board

Figure 10: 400 mAh LiPo Battery

4.1.3 Heart Rate and Blood Oximetry Sensor

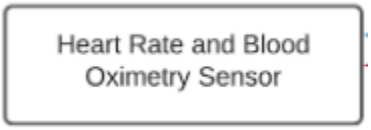


Figure 11: Heart rate and Blood Oximetry Block

This block is responsible for collecting heart rate and blood oxygen level data, one of the main functionality of our device. The board we are using is the SparkFun Electronics SEN-15219, which integrates 2 fundamental components from Maxim Integrated, the MAX30101 Pulse Oximeter and Heart-Rate Sensor and MAX32664 Ultra-Low Power Biometric Sensor Hub. The board communicates with the Arduino using the I^2C communication protocol, making it

effortless by connecting it to the SDA and SCL pins of the Arduino. It is powered by 3.3V which can be supplied by the arduino. We also used a 3rd party library to communicate with the device, so no low-level interfacing was required. The MAX30101 does all the sensing, while the MAX32664 is an incredibly small and fast Cortex M4 processor that handles all of the algorithmic calculations, digital filtering, pressure/position compensation, advanced R-wave detection, and automatic gain control[4].

The MAX30101 sensor works by utilizing its internal LEDs to bounce light off the arteries and arterioles in the skin's subcutaneous layer and sensing how much light is absorbed by them with its photodetectors(photoplethysmography). This is a common, inexpensive way of obtaining valuable cardiovascular information[5] and is widely used in the wearables industry in devices such as the Apple Watch. This data is passed onto and analyzed by the MAX32664 which applies its algorithms to determine heart rate and blood oxygen saturation (SpO2). SpO2 results are reported as the percentage of hemoglobin that is saturated with oxygen(% of red blood cells that is saturated with oxygen). It also provides useful information such as the sensor's confidence in its reporting. We utilized this confidence to determine if data points were valuable or outliers.

Figure 12: SparkFun Electronics SEN-15219 Sensor

4.1.4 Data Storage

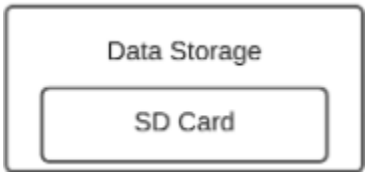


Figure 13: Data Storage Block

The data storage block consists of a Micro SD card breakout board and a SD card. The board we chose, the Adafruit 254 can be operated using either 3.3V or 5V which was an important consideration for us since our Arduino's IO must be 3.3V logic. It utilizes the SPI communication protocol which is the standard protocol used to interface SD card breakout

boards. Since our arduino does not have a dedicated CS(chip select) pin, we used a regular digital GPIO pin instead.

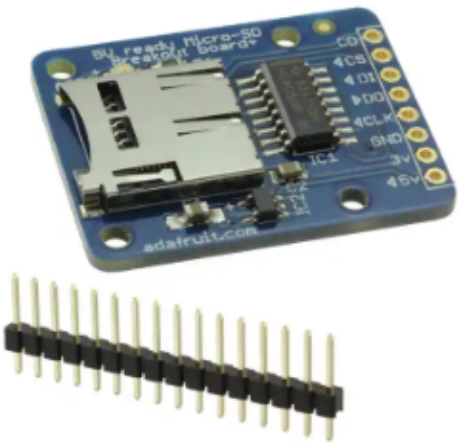


Figure 14: Adafruit 254 MicroSD breakout board

4.1.5 Display

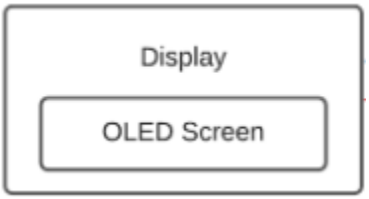


Figure 15: Display Block

The display used for this device was the Adafruit 938. It utilizes a 1.3” monochrome OLED screen, with a resolution of 128 × 64. This display was chosen due to the ease of integration with the I^2C communication protocol, the low power draw compared to LCD displays, the high contrast and its form factor which makes it perfect for devices such as this. Another reason this board was chosen over several similar boards is the fact that it can be powered from both 3.3V or 5V. Since our Arduino requires 3.3V logic, this was a very important criteria as 5V boards would utilize 5V logic and damage the MCU.

As mentioned above, one advantage of using an OLED display vs. an LCD display, is the lower power consumption. Since the OLED display consists of lighting up individual pixels instead of using a backlingh, its power consumption is directly related to the amount of pixels being used at any point in time[6]. This is perfect for our application because only text is ever displayed, using a low amount of pixels and the high contrast makes it highly readable even with the relatively small size of the display. Once again, the I^2C protocol was used, requiring connections to only the SDA and SCL pins of the Arduino.

Figure 16: Adafruit 938 OLED Display Module

4.1.6 Rotary Encoder

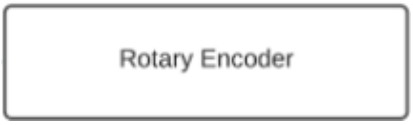


Figure 17: Rotary Encoder Block

The purpose of the rotary encoder is to act as the primary method of user input on the device. The module has 5 pins, 2 connected to GND and 3 connected to the Arduino. 1 pin for the push button and 2 pins for the rotary encoder. The pull-up resistors of the 3 GPIO pins used must be enabled, meaning the push-button logic is inverted(active low). The rotary encoder works by transmitting a pulse train through its two output pins. When the encoder is stationary, the pulses are in phase but the pulses become 90° out of phase when the encoder is rotated in either direction. For example, if the encoder is rotated clockwise, output 1 will lead output 2 by 90° and vice versa[7].

We implemented an algorithm to translate these two output signals to rotations of the encoder. In the firmware, we created 24 “positions”(0-23) that the encoder can move to and from and broke up these 24 positions into 2/3 sections(depending on which menu you are in), with each section representing an option in the menu. The user is expected to rotate the encoder to navigate between options of a menu, and press the push-button to select their option and either turn functionality on/off or navigate between different menus.

Figure 18: Generic Rotary Encoder/Push-Button Unit

4.2 Software

4.2.1 Fitness Tracking System: Frontend

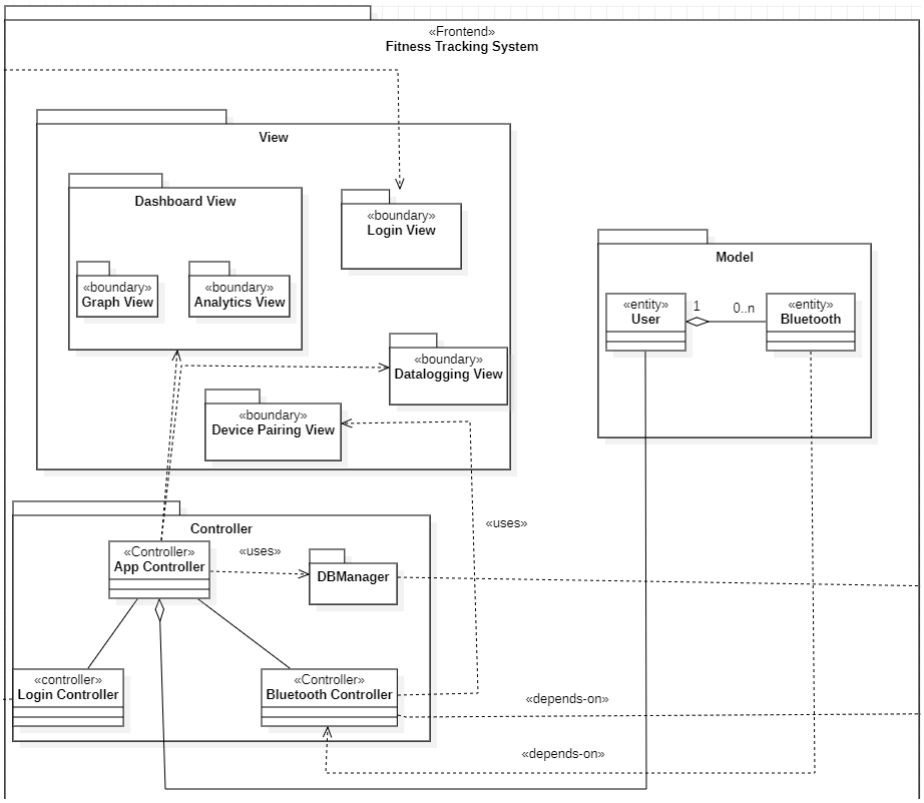


Figure 19: Fitness Tracking System Package

The entire Web-Application was built using ReactJs, which is a Javascript library that makes it painless to create interactive UIs. It is a Component-Based library that allows you to build encapsulated components that manage their own state, in order to then compose them and create complex UIs. To create an aesthetically pleasing UI, the Material-UI library was used. Material-UI is a React Component used for faster and easier Web-Development, making the styling of the website much easier.

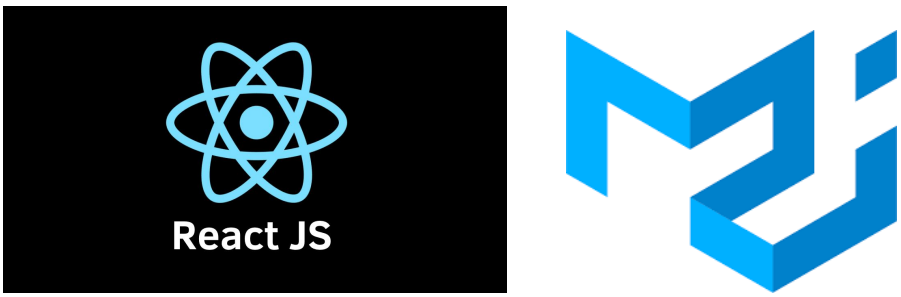


Figure 20: ReactJs + MaterialUI

4.2.2 View Package

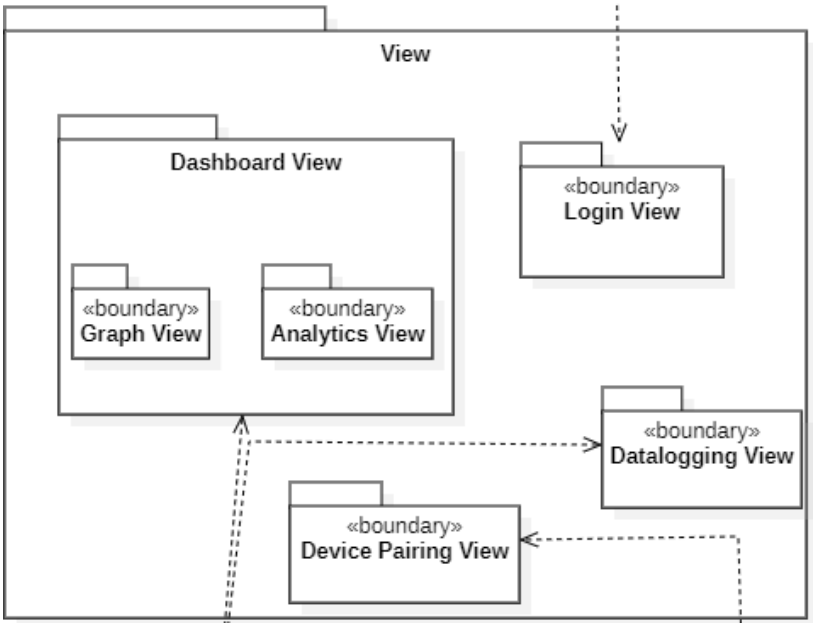


Figure 21: View Package

The View Package is responsible for displaying all the data on the screen to the end-user. This is split up into 4 main packages, to separate the logic, since the functionality of each package differs from one another. The Login view is responsible for taking in a user's username and password, into a textfield, and passing it onto the Controller Package which is responsible for authenticating a user's credentials, as shown below.

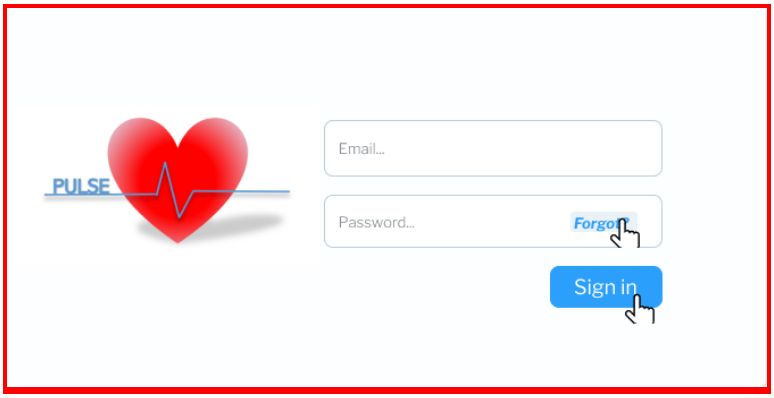


Figure 21: Login View

The Dashboard View is responsible for showing the user the graphed data, of his/her heart rate, blood oxygen level and step count, over a period of time.

Figure 22: Dashboard View

The Device Pairing View and Login View, is displayed on the dashboard page, and allows the User to pair to a Pulse Device via bluetooth, the logic and functionality is independent of the Dashboard, but accessing this feature can be done through the dashboard page, via the Uploaded data button shown on the Figure 22.

4.2.3 Controller Package

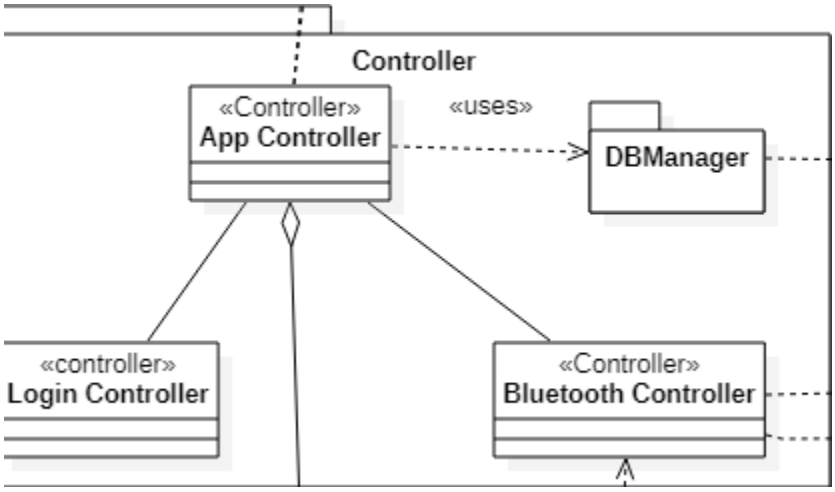


Figure 23: Controller Package

The Controller Package consists of 3 modules, and 1 sub-package. The App Controller Package is responsible for handling all the business logic of the web-application, it is essentially the brains of the frontend. The App-Controller module, controls everything from Logging into the platform, controlling the bluetooth transmission, retrieving data from the database and traversing through all the different views. The Login Controller is responsible for communicating with the login view, and retrieving the username and password entered by the user, and then communicating with the App Controller which then Communicates with the Database manager, in order to authenticate the users credentials, and fetch the users information given that he/she is authenticated. This data is then stored in the User module of the Model package, to ensure that the data is persistent across the application and multiple queries don't need to be made each time user data is required. The Bluetooth Controller communicates with the App Controller and the Device Pairing View, in order to connect to the Arduino itself, when a Bluetooth connection has been requested. To have a connection established between the Arduino and the web-application, the UUID of the service is required, and the UUID of the characteristic of the Arduino. The service that the Arduino exposes, can be thought of as a bulletin board, and the Bluetooth controller is a viewer of the board. The characteristic is like a notice posted on the board, which in our case is a set of data containing the heart rate, blood oxygen level, steps taken, and the date that this data was logged on. So once a characteristic has posted something on the board, the BLEController, can now see this data and it then passes it onto the AppController to have it stored in the database, which is required to create the graphs later on. The last package is the DBManager, and this is responsible for communicating with the Backend, and fetching data, updating data, or even creating new data in the case when new bluetooth data arrives from the Arduino, or registering for an account.

4.2.4 Model Package

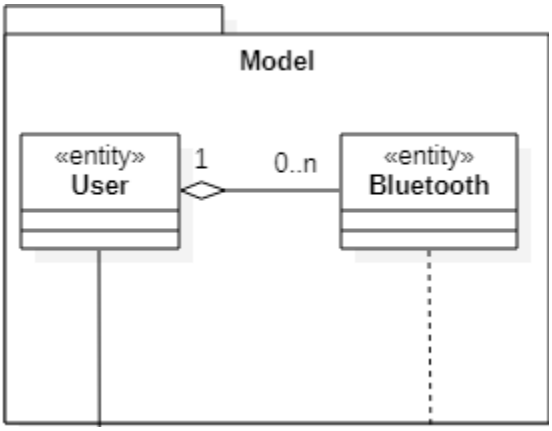


Figure 24: Model Package

The Model Package is fairly simple, it has two entities, the User entity and the Bluetooth entity. This package is required to store the User and Bluetooth Information, so that it is persistent across the application, and multiple queries don't have to be made each time a User changes pages. Information stored about a User includes, their name, their age, gender, height, and their unique ID, which is the Primary Key, necessary to make updates and access information relevant to them in the database. The following package shows that each User can have many bluetooth devices associated with them, meaning during one session they can use many Pulse devices if needed.

4.2.5 Backend Package

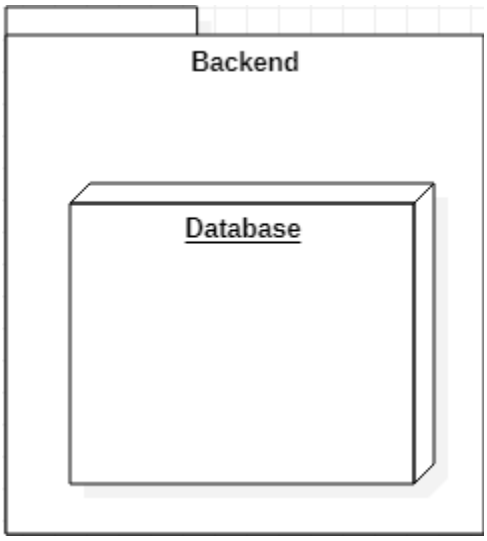


Figure 25: Backend Package

The Backend Package, contains the Database that we used and this is where all User data and User analytics is kept. The database management system is based on AWS GraphQL, which

takes care of the database, and authentication. The database that is used is a NoSQL database, and more specifically it uses dynamoDB, to store all the tables and data. With authentication being a big factor of any web-application, AWS Cognito (a subset of AWS GraphQL) was used to handle authentication. Using the username and password, it can safely authenticate any user, given the credentials are valid, and as such it minimizes the risk of SQL injection, so that only the person with valid credentials can access his/her and doesn't have access to anyone else's data in the database.

4.2.6 Arduino Package

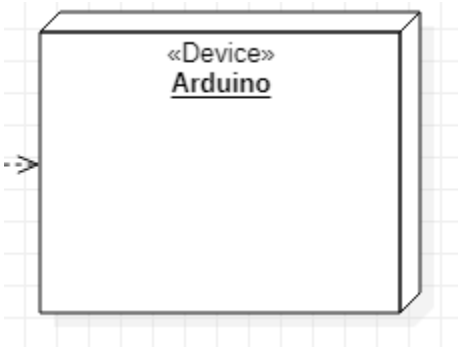


Figure 26: Arduino Package

The Arduino Package, contains all the code relevant to have the Arduino function as intended. It contains the logic for populating the screen with different menus, such as the Main Menu, the Workout Menu, WorkoutData Display, and the BluetoothTransmission Menu. Along with the different views, each view has its own functionality associated with it. The WorkoutData Display, once activated, is constantly reading the data from the SEN-15219 blood oxygen and heart rate sensor, and given that the data is not duplicated and the sensor is at least 90% confident about the data it saves the data into a JSON object, and after 10 valid reads this data is stored onto the SD card, which is necessary to transmit the data via bluetooth to the web-application. The reason for having 10 valid reads is to ensure that the data stored is less than 512 bytes of data which is the limitation that the BLE sensor has when transmitting data at a time. The data that is read regardless of its precision is then displayed onto the screen to allow the end-user to be able to view his/her heart rate and blood oxygen level during a workout, or reading. When a User once stops reading the data, they can do so using the pushbutton, and any remaining data that hasn't been saved to the SD card will be saved at that time. The MainMenu Display, is responsible for displaying the two main menus: The WorkoutMenu, and the BluetoothTransmission Menu, as well as the steps taken so far during that day. The BluetoothTransmission Menu, exposes its services so that the web-application is able to connect to it, and once a connection has been established, each file is read and open and transmitted to the device. Once completed the arduino will stop exposing its service, and return back to the main menu.

5. Engineering Analysis

Prior to the implementation and purchase of any hardware, our team considered which modules and sensors would be appropriate for our design and which Arduino model to move forward with, considering the product must be able to fit comfortably on the wrist. We chose the Arduino Nano 33 BLE Sense as it already integrates an accelerometer, Bluetooth capabilities and an attached LCD all fit in a relatively small size.

When we chose the sensors we want to include, we had 3 considerations:

1. Will the sensor provide accurate information that is useful to the user?
2. Is it an appropriate feature to have in a fitness watch?
3. How will the peripherals fit in the enclosure?

When implementing the heart rate monitor, we only needed to consider a reasonable price, as there are many available sensors that are small in size and relatively accurate. There were also many available oximeters, however we decided to trade off the accuracy of the device to have a better fitted model. Traditional oximeters will pinch the index finger to read the data, however our watch is designed to fit on the wrist, so we decided to use the SEN-15219 which integrates the heart rate monitor and oximeter into one board.

Our initial design did not contain the rotary encoder and the device would only run in an on state and having a constant transmission of data to the SD card. We ran into issues with battery life and data storage issues, we considered a push button that allows the user to turn the sensors on and off, saving battery life and limiting data storage to sessions where the user wants to record their information. The iterative design of a button feature finally led us to the inclusion of the rotary encoder, which not only added more functionality (provided the user with menu options to navigate to) but also solved the issue with data storage and battery consumption. For the step tracking algorithm we used data from the arduino's built in accelerometer and gyroscope. The algorithm consisted of summing the absolute value of the X, Y and Z components for one second which consisted of 9 data samples. From here, we looked at the change in this summed value from when the user was resting, walking and running. These values were recorded in order to create a code statement where the watch could understand when the user was in motion and track when a real step was taken, as opposed to an arm movement.

Once we had finished assembly of all the electronics, we took careful measurements and designed the smallest possible enclosure that would be both aesthetically pleasing but also functional. As we had added more functionality to improve the final product, since we were using pre built modules, the small wearable size was not possible to maintain. While we could not stay within our initial size estimations of 44x48x16mm, to continue the idea of creating a wearable device, we made adjustments to make it usable on the upper wrist or on the bicep. Since these areas would be more prone to impacts, the major focus of development shifted towards impact resistance. A key consideration was using only 35% infill as opposed to printing

one solid piece of plastic. While it also saves material, the infill and empty space inside create a “honeycomb” like structure which can deform on impact without breaking the plastic. In order to make the enclosure waterproof, we created printed plugs for the ports, used a rubber watch gasket on the encoder and then covered all exposed surfaces with 5 layers of silicon spray.

6. Enclosure

The enclosure was manufactured out of PLA on a 3D printer. Given it was the last item to be produced, and since we had very limited time to do it, this was the only choice to make something custom, quickly. PLA offers both a cheap and robust way of making the watch. While we were unable to meet the size criteria it serves the purpose of the project well providing a robust, impact proof and waterproof shell.

Figure 27: Manufactured Enclosure and 3D CAD Model

7. Printed Circuit Board

Due to the design constraints relating to the size of the watch, implementing this PCB into the final production model was completely impractical since our hardware was already oversized. Instead, it serves as a test platform for debugging, and as a base for future works. Since the next logical step of the project would be to develop our own sensor modules to reduce the overall size of the product, having a base test platform to eliminate all unnecessary wires and breadboarding for known working parts provides us with ease of ease of development and testing. Each Arduino module could be replaced by smaller test PCBs would be attached to the master PCB for testing.

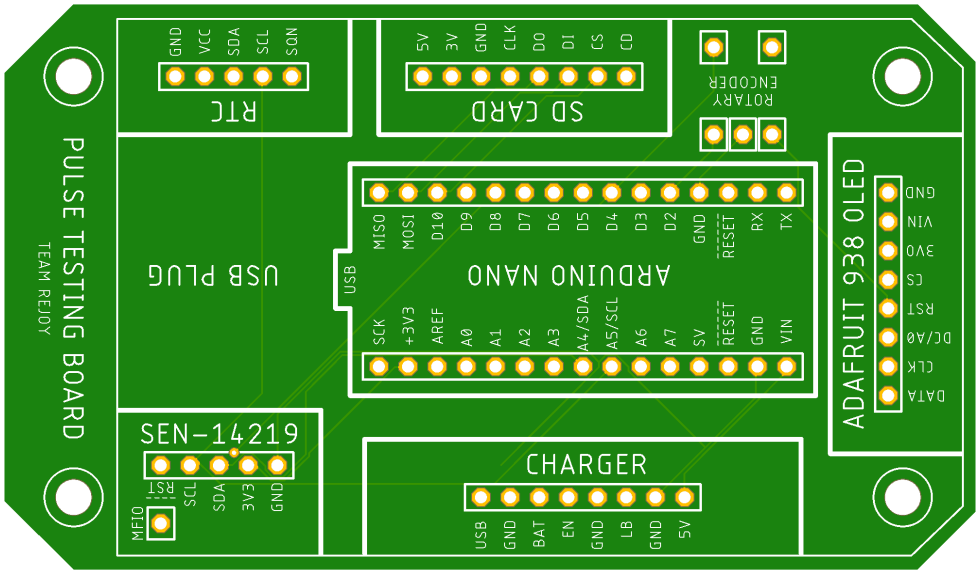


Figure 27: Top view of the 2 layer hardware testing board

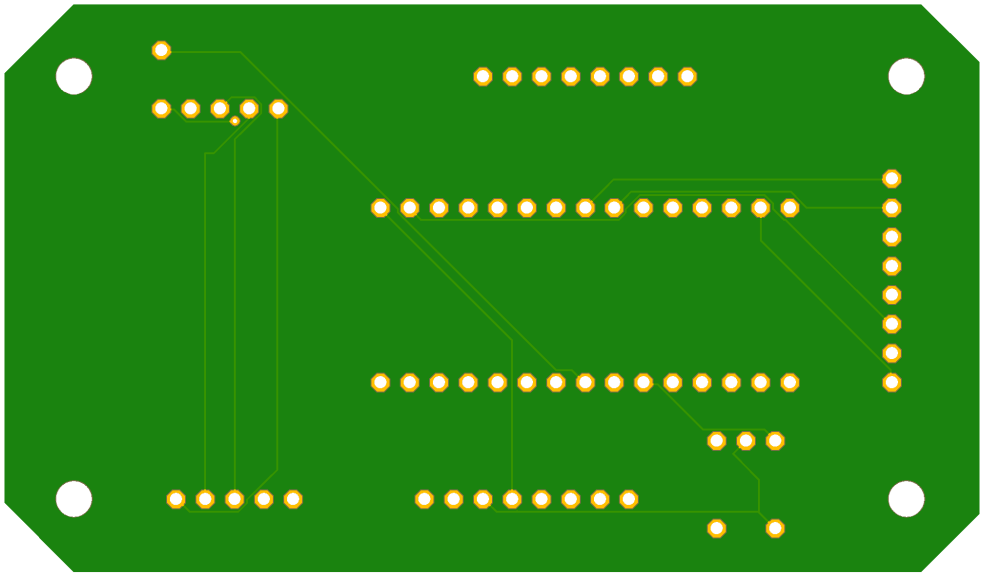


Figure 28: Bottom view of the 2 layer hardware testing board

8. Regulatory Codes

Regulatory codes set by various Canadian and global regulatory agencies were considered in the design process to ensure product reliability, consumer safety and rollout of the product into public markets.

8.1 Canada Consumer Product Safety Act (CCPSA)

8.1.1 Consumer Chemicals and Containers - Category 5 - Packaging material may not contain hazardous substances and information on potential dangerous internal components must be provided.[8]

8.1.2 Consumer Packaging and Labelling - Sections 12, 14 and 30 - ensure proper labeling and presented product information.[8]
8.1.3 Medical device regulation - Section 10 - Device shall be designed to be safe and measures taken to identify, eliminate (when possible) and inform on inherent risks and hazards. [8]
Section 14 - Characteristics and performance of the device must not deteriorate under normal use to such a degree that the health or safety of the user is affected.[8]

8.2 Office of the Privacy Commissioner of Canada

8.2.1 The Personal Information Protection and Electronic Documents Act (PIPEDA) - Division 4 Section 20 ensures compliance to in place privacy laws are maintained, data collected from the user is not shared and only used to the extent of permissions granted by the user.

8.3 CSA Group

8.3.1 CAN/CSA-IEC 62443:17 Network and System Security

8.4 Innovation, Science and Economic Development Group (ISED): Regulatory body for electronic devices in the Canadian market

8.4.1 All electronic and electrical devices must be approved by ISED before entering the Candian market.[9]

8.5 International Organization for Standardization (ISO)

8.5.1 ISO 10377 Documentation of risk assessment and management to meet requirements.[10]
8.5.2 ISO/IEC JTC 1/SC 25 Standardization of microprocessor systems interfaces, protocols, architectures and IT equipment.[10]

9. Design Alternatives Considered

9.1 Hardware

One of the first decisions we had to make was deciding which board to use. We chose to focus our search on an Arduino because it has a smaller form factor then most other boards, like a Raspberry Pi. In the end, we decided to choose an Arduino Nano 33 BLE Sense because it had many built in functions (bluetooth, accelerometer, gyroscope) that we required for our device. This saved us money by not having to purchase more components. Also by choosing an Ardunio Nano, we were able to meet size requirements that were crucial in producing a reasonably sized watch. Additionally, members of our group were familiar with the Arduino Nano which made the programming aspect less intimidating.

Initially when deciding to create a watch device we knew a screen would be necessary for displaying time and potentially other functions. We had to make a decision between either an

OLED screen or a LCD screen[11]. From research we learned that the OLED screens do not require backlighting and also had higher contrast which would allow for text to be read easier[11]. Because our device is battery powered, having a screen that requires lower power consumption is important to ensuring the size can remain as small as possible[11]. The OLED screen allowed for a smaller form factor while still producing clearly visible text. Overall, the OLED met many of the needs we were looking for by allowing us to distinctly display text while meeting dimension requirements.

From here, we decided we wanted to add functionality to the screen by adding either a pushbutton, switch or rotary encoder. All three options had pros and cons but we decided on the rotary encoder. We thought it allowed for the most aesthetically pleasing design with the benefit of providing options to both scroll and select. Both the pushbutton and switch could have been implemented easier but wouldn't have been as aesthetically pleasing or allow for as much functionality as the rotary encoder. The rotary encoder has been programmed to allow for users to scroll to various functions and select them while still providing a clean look to the device.

9.2 Software

When beginning our software implementation we had to decide on a backend database to use. We narrowed it down to either SQL or NoSQL[12]. We chose to work with NoSQL because it is simpler and we didn't need the backend to compute any logic. In our case, the backend was just responsible for storing and fetching the data therefore we didn’t need any computation to be performed in the backend. We also had members of the group who were familiar with NoSQL which made working with it less intimidating. Overall, NoSQL made more sense for our device and we didn’t see any benefit to overcomplicating the process and going with SQL.

From here, we also decided to use dynamoDB as our database management system because we were working with GraphQI through Amazon Web Services (AWS). We used AWS because members of our group were familiar with it and because it is low cost.

10. Future Work

In order to enhance our wearable fitness tracker we could implement various design and watch features. There is also additional work that could improve the functionality of this product.

Future features could include:

1. Various watchface and wristband sizes to accommodate for different wrist sizes and further prevent any bulkiness or obstruction.
2. Miniaturize the watch enclosure and components in order for it to be closer to a normal sized watch.
3. A calorie tracker that allows users to record the amount of calories burned throughout the day while wearing the watch.

4. A workout feature that displays when the watch knows a user is resting, walking or running.
5. A feature that allows users to connect with other users both through the delay and on the website to compare step counts and other parameters.

Additional work could include:

1. Further testing to improve accuracy and guarantee consistent performance.
2. Increase functionality of the rotary encoder so that users have access to more features directly on the watchface.
3. Optimizing the battery life of our device.

11. References

[1] M. Terndrup, “HOMEMADE ACTIVITY MONITOR”, 2014. [Online]. Available:

https://hackaday.com/2014/09/01/homemade-activity-monitor. [Accessed: 3-Apr-2021].

[2] Nordic Semiconductor, “nRF52840 Product Specification”, V1.1, 2019

[3] https://rogershobbycenter.com, “A Guide to Understanding LiPo Batteries”, 2012. [Online]. Available: https://rogershobbycenter.com/lipoguide. [Accessed: 3-Apr-2021].

[4] Sparkfun Electronics, “SparkFun Pulse Oximeter and Heart Rate Sensor ‑ MAX30101 & MAX32664 (Qwiic) Product Specification”

[5] D. Castaneda, A. Esparza, M. Ghamari, C. Soltanpur, H. Nazerun, “A review on wearable photoplethysmography sensors and their potential future applications in health care”, International Journal of Biosensors and Bioelectronics. Available: https://medcraveonline.com/IJBSBE/IJBSBE-04-00125. [Accessed 3-Apr-2021].

[6] Adafruit.com, “Monochrome 1.3" 128x64 OLED graphic display - STEMMA QT / Qwiic”, 2019. [Online]. Available: https://www.adafruit.com/product/938. [Accessed 4-Apr-2021].

[7] howtomechatronics.com, “How Rotary Encoder Works and How To Use It with Arduino”. [Online]. Available: https://howtomechatronics.com/tutorials/arduino/rotary-encoder-works-use-arduino. [Accessed 4-Apr-2021].

[8] Branch, L. (2021, April 07). Consolidated federal laws OF Canada, Consumer chemicals and CONTAINERS regulations, 2001., from https://laws-lois.justice.gc.ca/eng/regulations/sor-2001-269/index.html

[9] Electronic product regulations in CANADA: An overview. (2020, June 22)., from https://www.compliancegate.com/electronic-product-regulations-canada/#:~:text=The%20ISED%20regulates%20radio%20devices,Cellular%20devices

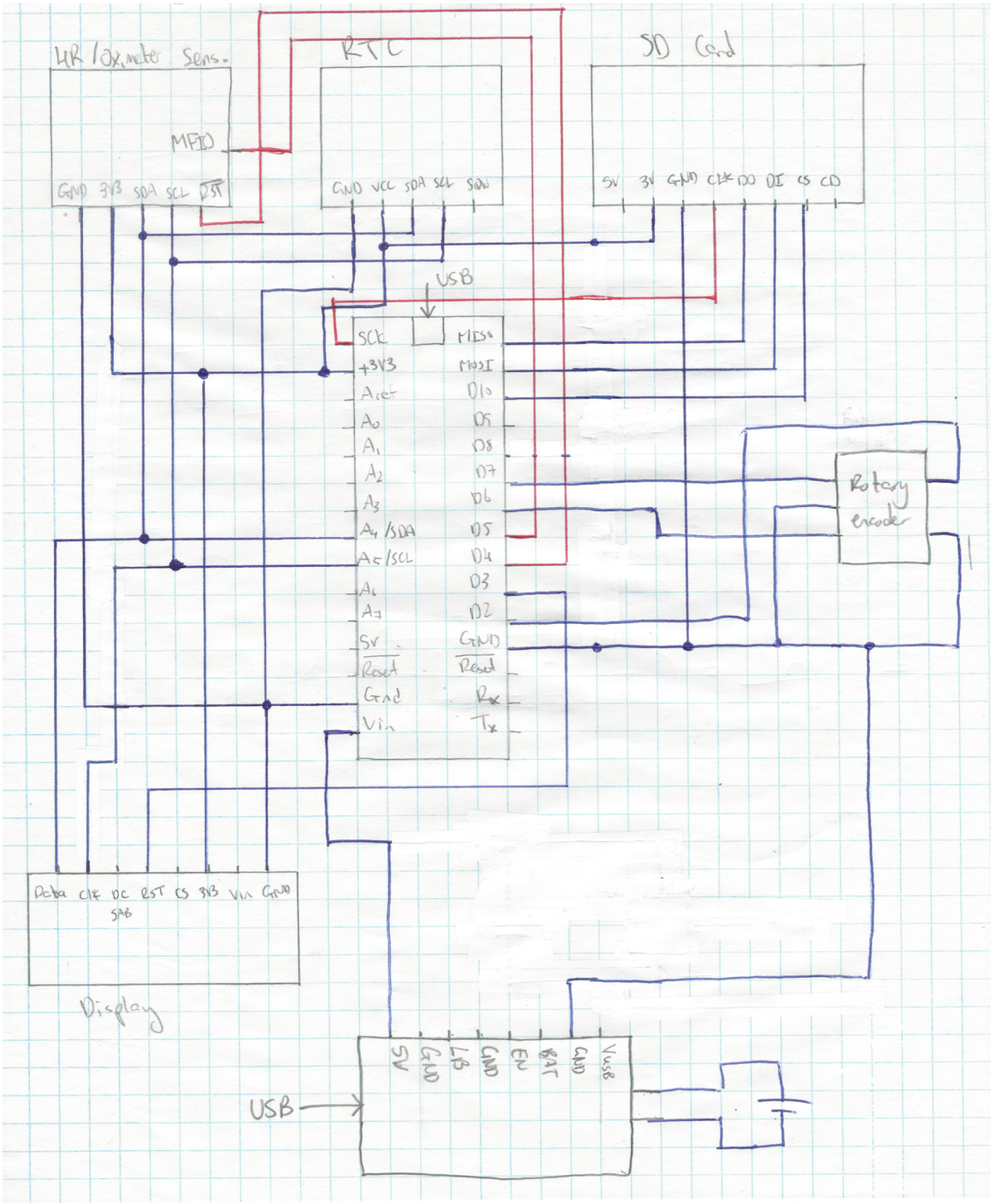
[10] Iso 10377:2013 Consumer Product Safety. (2018, October 15)., from https://www.iso.org/standard/45967.html

[11] Craig Freudenrich, P. (2005, March 24). How oleds work. Retrieved April 09, 2021, from https://electronics.howstuffworks.com/oled5.htm

[12] Chan, M., 21, M., & *, N. (2020, December 16). SQL vs. NoSQL - What's the best option for your database needs? Retrieved April 09, 2021, from https://www.thorntech.com/sql-vs-nosql/

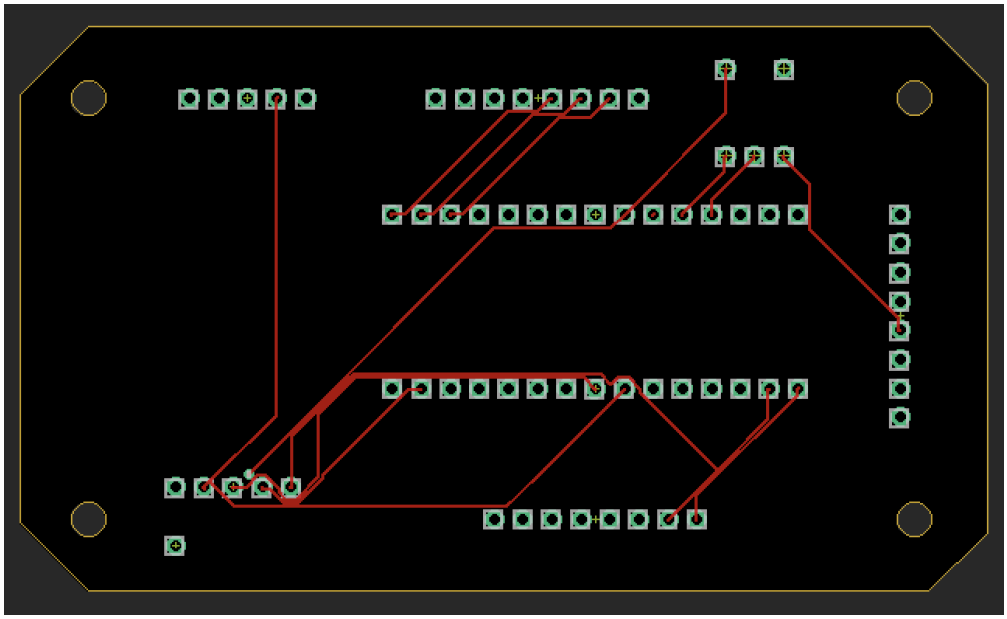
12. Appendices

12.1 Schematics

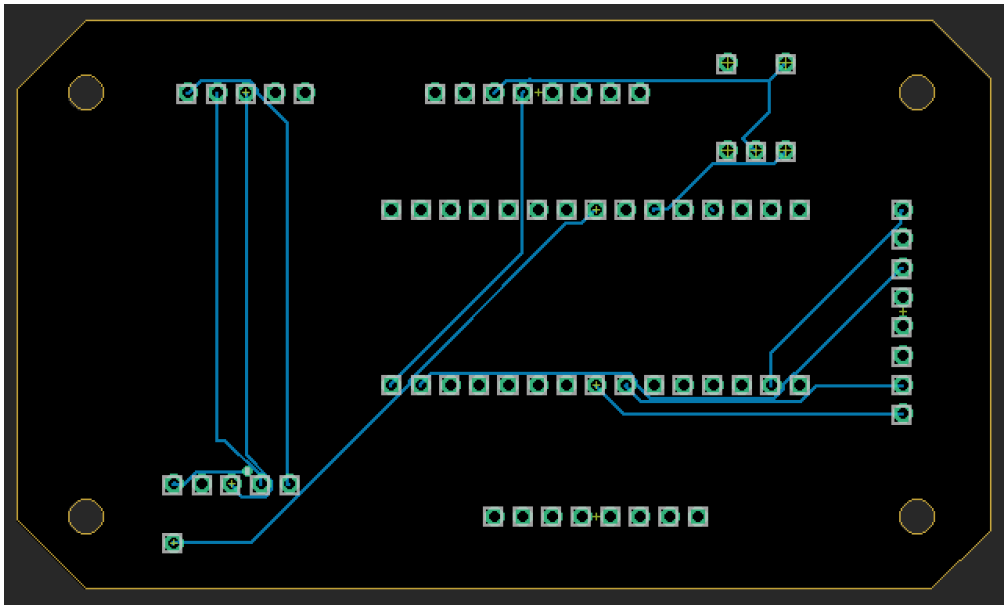


12.2 PCB Layout

Top layer connections (looking at the board from the top):



Bottom layer connections (looking at the board from the top):



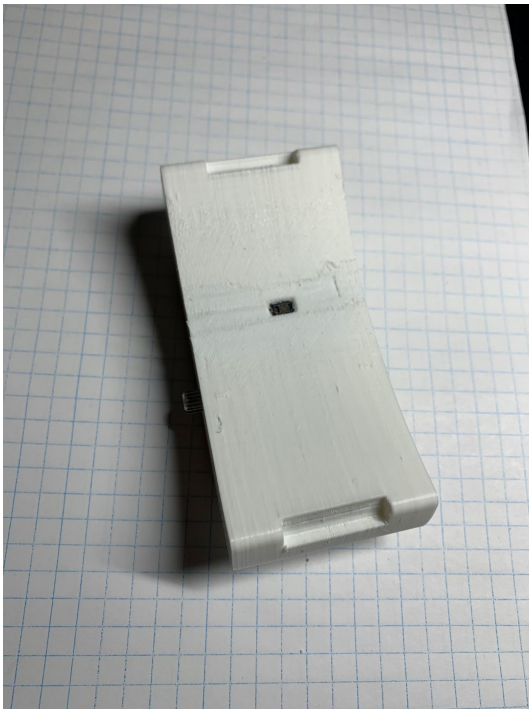
12.3 Bill of Materials

Bill of Materials

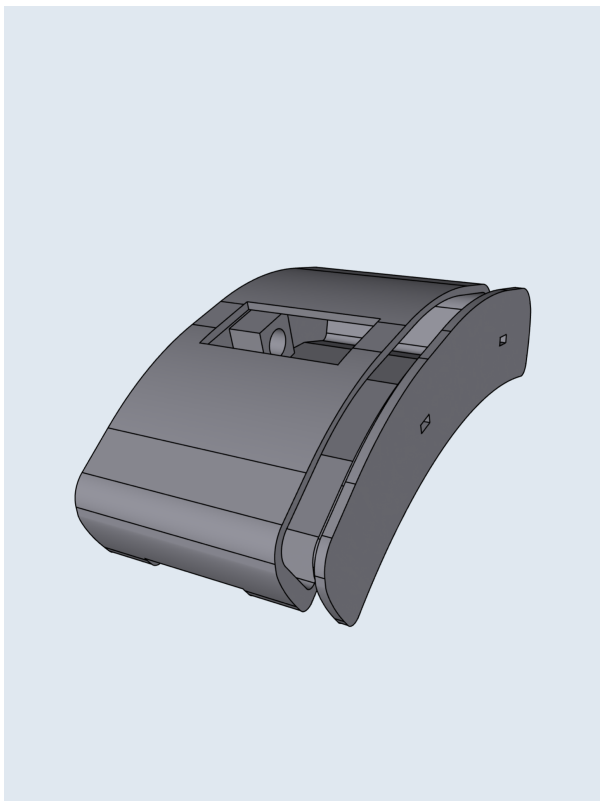
Team Name	REJOY	Approval Date	2/18/2021
		Part Count	10
Product Name	PULSE	Unit Count	10
		Total Cost	$205.04

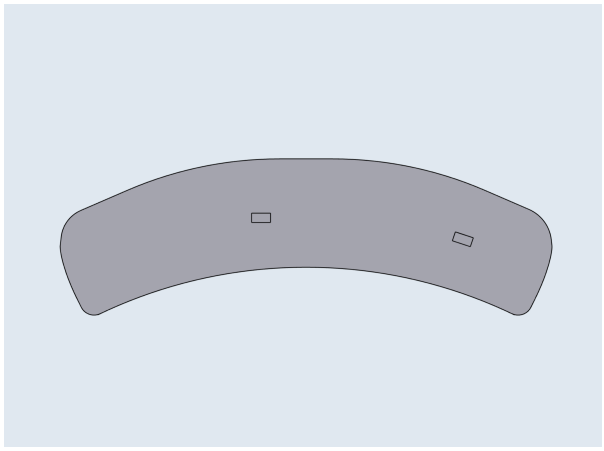
Part Name	Part ID	Description	Quantity	Unit Cost
Arduino Nano 33 BLE Sense	N/A	Main MCU of the device, includes integrated BLE module and inertial sensor https://www.digikey.ca/en/products/detail/arduino/ABX00031/10239970	1	$47.34
Pulse Oximeter and Heart-Rate Sensor	SEN-15219	PULSE OXIMETER & HEART RATE SENSOR W/ MAX30101, MAX32664 - I2C https://www.digikey.ca/product-detail/en/SEN-15219/1568-SEN-15219-ND/10483252/?itemSeq=355478928	1	$60.68
Display	938	Graphic LCD Display Module Passive White OLED - Passive Matrix I²C, SPI 1.3" (33.02mm) 128 x 64 https://www.digikey.ca/en/products/detail/adafruit-industries-llc/938/5774238	1	$30.30
RTC	1528-1787-ND	PCF8523 REAL TIME CLOCK BOARD https://www.digikey.ca/en/products/detail/3295/1528-1787-ND/6238007?itemSeq=355727742	1	$7.06
Storage	N/A	16GB SD Card	1	Free
Storage Interface	1528-1462-ND	MICROSD CARD BREAKOUT 5V OR 3V https://www.digikey.ca/en/products/detail/254/1528-1462-ND/5761230?itemSeq=355727967	1	$10.70
Batttery + Charger	N/A	Adafruit 500C charger board + 400mAh LiPo https://solarbotics.com/product/50873/ https://solarbotics.com/product/18936/	1	$37.03
Rotary Encoder	N/A	Rotary Encoder/PB Unit	1	$3.03
RTC Coin Battery	N/A	Coin Battery for RTC	1	$3.90
AWS Server	N/A	AWS for Web App	1 month	$5.00
			Total	**$205.04**
			Cost/Member	**$41.01**

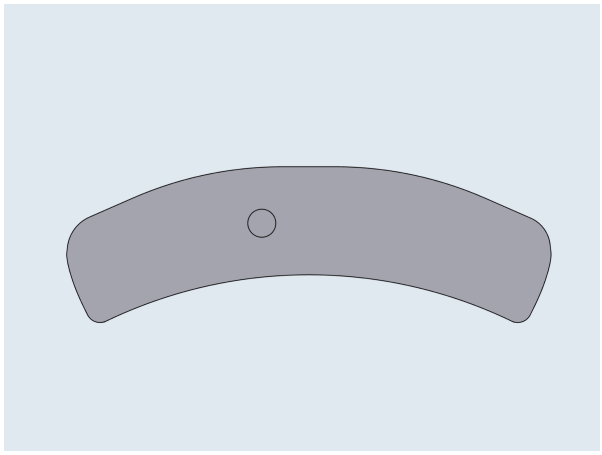
12.4 Images of Prototype



12.5 3D Model Drawings

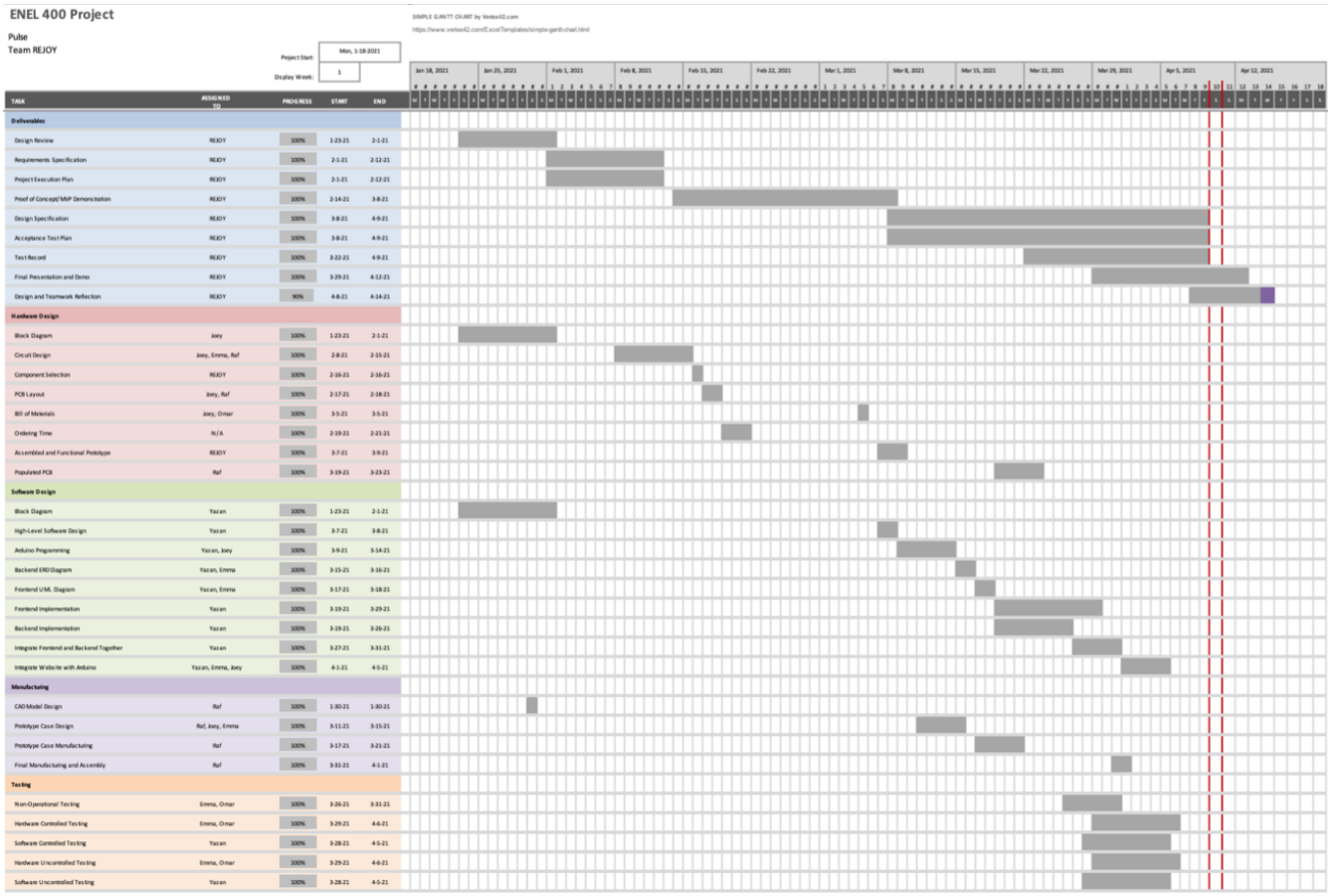






12.6 Schedule and Expenditures Variance

12.6.1 Schedule Variance



Overall the components and subcomponents of our schedule stayed the same throughout the semester. We had made the schedule thorough to begin with that way we accounted for the time every step would take. Significant changes that were made included our timeline. Throughout the semester, parts of our project got pushed back meaning not as much progress was made in the middle of the semester like we anticipated. However, our initial schedule left room for potential roadblocks and steps backward. At the end of the day, we found our schedule was a great way to keep everyone on track and visually understand our progress. We may not have had as much time for testing as we had wanted but the functionality of our project was fully completed on time.

12.6.2 Expenditures Variance

The expected expenditure was $250 CAD. The final cost of the prototype development tallied up to $205.04 CAD, as shown in the BOM above. This represents a $44.96 or 18% difference when compared to our expected value. This difference comes from the purchase of the Arduino Nano 33 BLE Sense instead of a regular Arduino Nano. The price difference between the two boards was practically negligible, but purchasing the Arduino Nano 33 BLE Sense allowed us to cut cost on the Bluetooth module and accelerometer and gyroscope since these parts are already integrated on the Nano 33 BLE Sense. The estimated savings from this is about $22($15 for the Bluetooth module and $7 for the accelerometer and gyroscope).

Another cost saving measure was the fact that we obtained most of our parts from Solarbotics, a local electronics shop instead of Digikey. The cost of similar parts from either vendors was almost identical but we were able to save on the shipping costs. In total, we made 3 “rounds” of orders, 1 for the majority of the components, 1 for the battery and changer and 1 for the rotary encoder. The first order was done on Digikey but the other 2 were done at Solarbotics, saving the team $16 of shipping costs($8 per order).