Adaptive Wireless Wearable Neuro-Stimulator

Team: 22
Client: Adan Cervantes
Advisor: Swamy Ponpandi
Patrick Walsh/Communications Manager, Kevin Wang/Meeting Facilitator,
Kevin Simons/Test Engineer, Matt Stephenson/Report Manager, Brian Weber/Chief Engineer
sdmay18-22@iastate.edu | sdmay18-22.sd.ece.iastate.edu

Revised Project Design

Previous Work

Infant monitoring is not a new concept, there have been many different forms of infant monitoring throughout the years. Most people are familiar with the baby monitor, there are many different companies making these, one example would be the audio and video monitors produced by VTech [1]. These types of monitors are very passive and hands off, they won't alert you, they only send whatever is being recorded and the rest is up to the parent. These types of devices however do not offer the level of data required to diagnose sleep apnea (Ziganshin, Numerov & Vygolov, 2010).

In order to detect sleep apnea, which is pauses in breathing during sleep, a better monitoring system would be required. Some of the proposed ideas by Ziganshin, Numerov & Vygolov are a sensor pad to detect movement, a motion sensor to attach to the body, or a type of radar system to monitor the infant (2010). A version of this is the NanoPulse Baby SleepGuard, which monitors respiration, heart-beat, and general movement while the infant sleeps, and parents are alerted if any of these are abnormal. Another such device is the Sproutling Wearable Bay Monitor [2], which has a similar design to what our proposed idea is. This device monitors the infant's heart rate and movement to let the parent know if the child is asleep, waking up soon, or sleeping on their stomach.

Proposed System Block Diagram

Our prototype will consist of three separate sections. The device itself, which will do the monitoring and pulse actions. The Android app, which will be accessible for the parents, that will display active information about the child, such as sleep position. And finally, the web app, which will primarily be used by researchers to view historic data for all devices, filtered by any user specific criteria necessary, such as age, or birth month.

Each different part has its own design, as each is used for a different action. To start with the hardware portion, there are a number of different solutions to consider. The design of this part in broad terms is to be able to connect with an application, monitor biometric data, and report that data to the app. Once this portion has been completed, there can be more research done into what length and strength of vibration pulses are needed for different situations that an infant may get into. And example of this would be sending a longer pulse if the infant flips to laying face down, in order to rouse the child.

This vibration portion of the system, has to be aggressive enough to get a natural response from the infant, such as an increased heart rate, but also gentle enough to not wake the child unless absolutely necessary. In order to do this we use a viration motor that we are able to control the length and strength of the vibration. We currently have these set to some example numbers given to us as a starting point by our client, but these would need to be updated as more research is done. This research however was out-of-scope of this project.

Biometric data recorded from the child must serve two purposes, it must be accurate enough to give us enough data to make decisions off of, and it must be able to be recorded from an extremity such as the ankle. This information could include, temperature, heart rate, movement, blood pressure, or even perspiration levels. The decision to go with heart rate, temperature, and movement for recording was based of the research that we received, as well as past literature, where devices today were recording this same information [2]. The temperature sensor is also used to detect if the device is being worn currently so we can remove the need of parents to remember to turn on the device when they put their child to sleep.

The features for the two visual applications of our product were determined based on who would be using them. In the case of the Android application, we needed to create a parent-friendly interface, that would be simple to interact with, and could quickly give parents the overview information they are looking for. With this, we designed a simple alerting page, to give the basic details of the device at a glance, such as sleep position, as seen in Appendix I.B.I. The web application on the other hand, is meant to be used as a research tool. This means our interface should expose more information about the raw data, allowing the users to select exactly the data that applies to their research. For this, we created an in depth form that allows users to specify all the criteria for the data that we have, as seen in Appendix III.C.I. This data is then fetched from our database, and displayed in graph form, for ease of consumption.

An overview of our system can be seen through our block diagram in Appendix II.B.

Assessment of Proposed Methods

Our solution is a divided one, it is very good at giving the user the basic information they need, and letting researchers obtain all of the information. The device itself is also monitoring some of the most important factors in determining the sleep health of an infant. Because we need to keep the device running on low power, we do have to make sure the phone is in range at all times in order to alert and upload the data, otherwise the device will just be acting on its own with no backup of parental alerts. This is also a benefit though, in that the device can function and be useful even if the parents are not around.

A possible weakness of this device however is the fact that if the device cannot correct the problem, then essentially the danger alert to the parent is delayed, by however long the device was trying to fix it itself. This is balanced out, we believe, by the amount of times the device fixes the problem and does not have to send an alert. But, being cognizant of this weakness, the device should try a small number of times, with short intervals between attempts, in order to decrease the time before parents are alerted of the problem if they need to be.

Validation/Test Plan

For the device itself, which is a combination of hardware and software, our tests will mostly be centered around verifying the parts work as expected, and then based on specified inputs, they react accordingly. Our performance and compatibility tests will be used to verify the correctness of our parts, this will include things like testing that our vibration motor can send a strong enough pulse, and verifying the biometric sensors record data at a rate that is fast enough to be useful. Moving to the embedded code, we will have more system tests for things such as making sure the device will turn on and off when it is attached to a user. We also have some lower level integration or unit tests to verify things such as the device will be able to connect to phones when it is powered on.

The two software-only parts of our project, the Android and web app, will have a combination of some manual integration testing, to verify that as a user, the app works correctly, and responds as desired based on the input, and it also has some automated functional tests. For the web application, we have developed functional tests that we run on every deploy that verifies all of our endpoints are still behaving as expected, so we verify we can still login, and upload data. We try to test every endpoint listed in Appendix III.A.

Implementation Details

Hardware

The hardware in this project consists of a bunch of basic circuits containing sensors that are connected to an Arduino Mega. This we refer to as the wearable throughout the document, as these circuits would eventually be a part of a wearable anklet device, if it makes it to market. The wearable constantly monitors the temperature sensor to see if it is being worn. If it is connected it starts polling the other sensors (heart rate, accelerometer, and controlling the vibration motor).

It logs the data gathered from the sensors into an array of packets to be transmitted. Storing up to a max number of packets, that would be based on the size of the storage in the final design. Since our product is a proof-of-concept, this value is currently set to an arbitrary, large constant.

Once the Android device is connected via bluetooth, it begins transmitting the packets. The packets follow a specified format for the Android device. The wearable sends a FIN packet at the end of its transmission. This tells the Android app the it is done. The Android app responds with the timestamp of the last packet received.

Android App

The Android application periodically pulls from the device via Bluetooth and pushes it out to the server when possible using either WIFI or cellular data. The application is also be able to perform network transactions for various other services such as logins and account creations. To accomplish the various functionalities in the application, we made use of a variety of libraries and technologies.

Before the application is even able to communicate with the wearable device, the phone and the wearable need to be paired. To pair, the application will require the user to go through a pairing process in which the MAC address of the wearable will be stored locally on the phone for future data transactions. Once paired, the wearable and application will be able to communicate without issue. The data received from the device is comma separated, and is stored locally by way of the room persistence library. When a sufficient amount of data has been gathered and transferred to the phone by the wearable, the application will then attempt to upload the data via internet or cellular data connection.

The network functionality of the phone is managed using the Volley library. All network packets are constructed using a JSON string format with specific options put in the headers for the various requests done by the application (See Appendix III.A for details.). To create each request, the parameters map is filled out and passed to a Volley library function to generate a JSON request. The JSON request is then added to a queue of HTTP requests, and once the request is executed the return code is interpreted by the application to perform the return code-specific action. This process is used for all requests. The data post request is different in the sense that it is set up as a background process. The background process executes a thread that checks to see if there is new data in the room persistent database (See Appendix III.B for details). If there is new data to be sent, the thread packages all of the necessary information and formats it in such a way that it can be read by the web server endpoint that handles the database transactions.

Data storage on the phone is handled using the room persistence library from Android. This library is, in essence, an SQLite database with an abstraction layer over it to make data management easier. The database consists of the database itself, the entity representing the table, and Data Access Objects (DAO) with methods that allow access to the database. As data comes in via Bluetooth, the line is checked for the expected format and parsed to obtain the data consisting of timestamp, heart rate, temperature, and accelerometer magnitude. In order to store this data, a new BiometricData object is created. An ASyncTask handles the inserting of the new data into the database, using the insert method from BiometricDataDao. An instance of the BiometricDatabase should be instantiated before writing data.

Web Application

The web app for this project is used to both save device metrics long-term, in order to provide medical researchers with a tool to visualize this data, as well as track our users and registered devices. Since the biometric data from the device can be considered personal medical data, we do correlate user accounts with registered devices. Instead, we manage registered devices with less personally-identifiable, namely a latitude/longitude, as well as a zipcode. If this project were to be commercialized, and widely distributed, we found that Amazon RDS would support our storage needs, as well as allow us to meet HIPPA requirements such as restricting personal data access.

In order to communicate with our Android app, we expose REST endpoints for any needed communication. Some of these include, creating users, registering devices, uploading data, and checking the login of a user. Using REST endpoints allows us to scale our app in the future if needed by just spinning up more servers to handle the apps requests. For storing users, as well as historical device data points, we use a SQL database, and communicate with it via php.

In order to display the historic trends of data, we present any logged in users with a form that allows them to narrow their query of the data. Some options allow filtering to specific users, such as gender, or north/south hemisphere, while others allow filtering the data, such as choosing between heart rate, or accelerometer data. After specifying the bounds of data, we perform a SQL query to retrieve that data, and load that data into the javascript library Chart.js. With a line in the graph for each user that matched the filters, users are then able to see individual data points over the specified amount of time. This allows researchers to see if any naturally occurring events affected certain subsets of the user population, and what trends, if any, appear in the data.

Testing Processes and Results

Hardware

Our wearable code was tested using a testing script. The script supplies parameters to each and every method we created. It then verifies the return value. In this way, we can make sure that we didn't accidentally break something when making changes. This script tests methods created for the heart rate, temperature, vibration motor, and bluetooth.

Android Application

Testing of the Android Application was done by testing UI elements on each successive iteration of the UI, making sure that flow was as expected without any unexpected behavior. Dummy data was used to test reading over Bluetooth by sending cases covering malformed data and data in the expected format. Dummy data was also used to test interfacing with the local database, and data posting.

Web Application

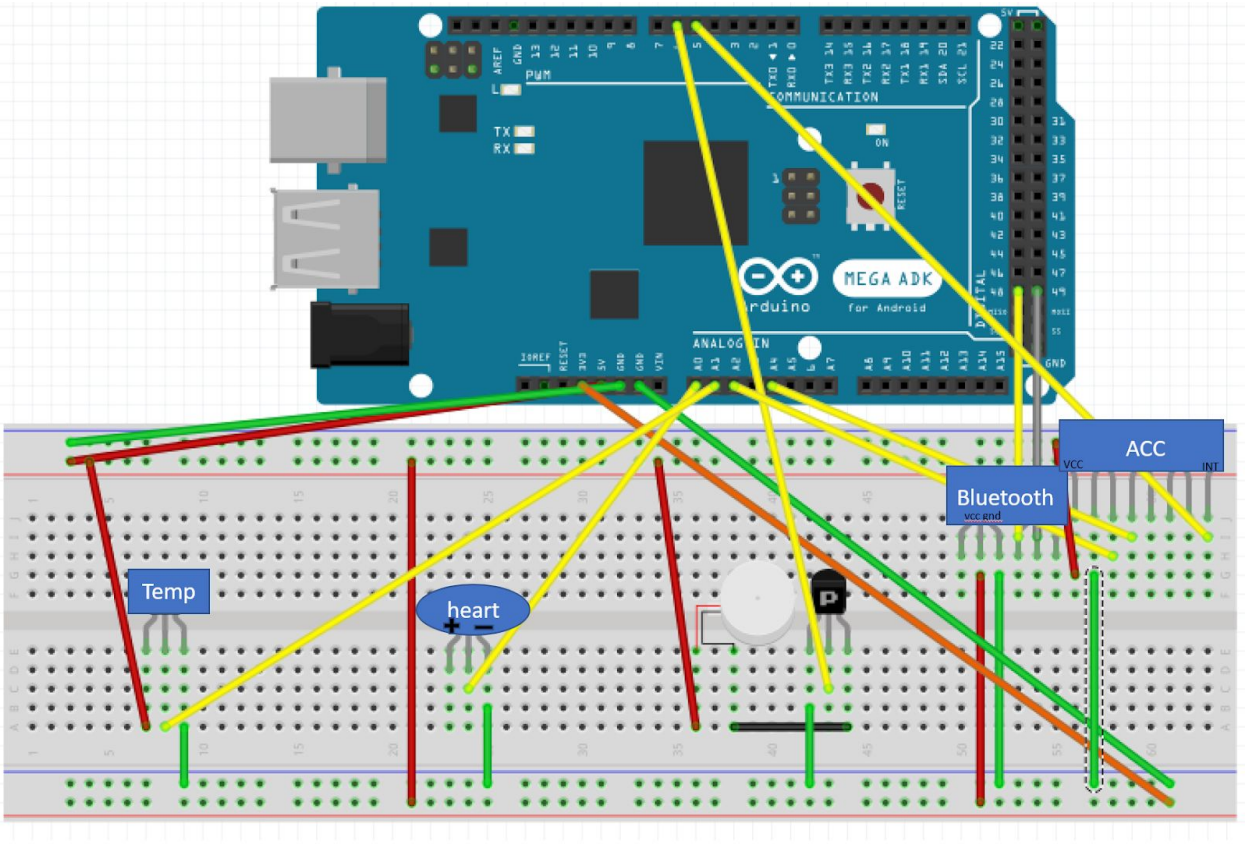
In order to verify web functionality, we have written a test script that verifies endpoints are responding as they should to different requests. In this way, we can run a script after a new deploy, that will check that each of our exposed endpoints can receive data, and respond accordingly based on that data. Some of these tests include, attempting to create a new user, verifying a login works, and verifying that we can successfully POST data. This test suite allows us to quickly realize if any problems have been introduced, and revert the last change if needed. This testing is run anytime we manually deploy, and results are reported to the user who ran the script, and they can see if any of the endpoints are not behaving correctly.

Appendix I : Operation Manual

I.A. Hardware

I.A.I Setup

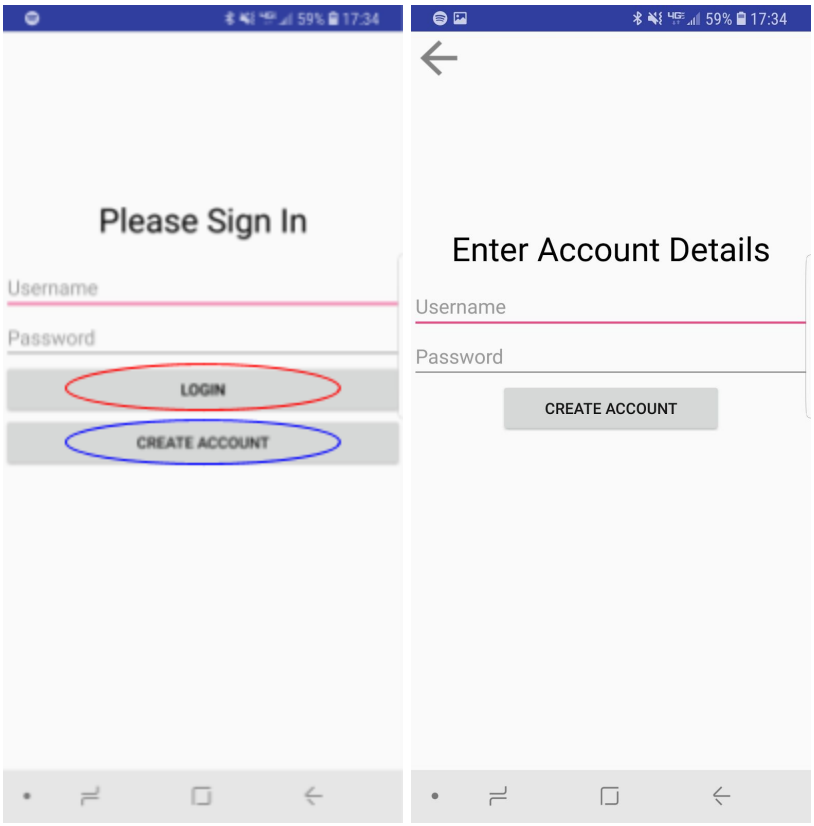
All of the hardware should be hooked up as seen in diagram below. You will need an Arduino Mega and all the necessary parts (heart rate, accelerometer, temperature, and bluetooth).



I.B. Android Application

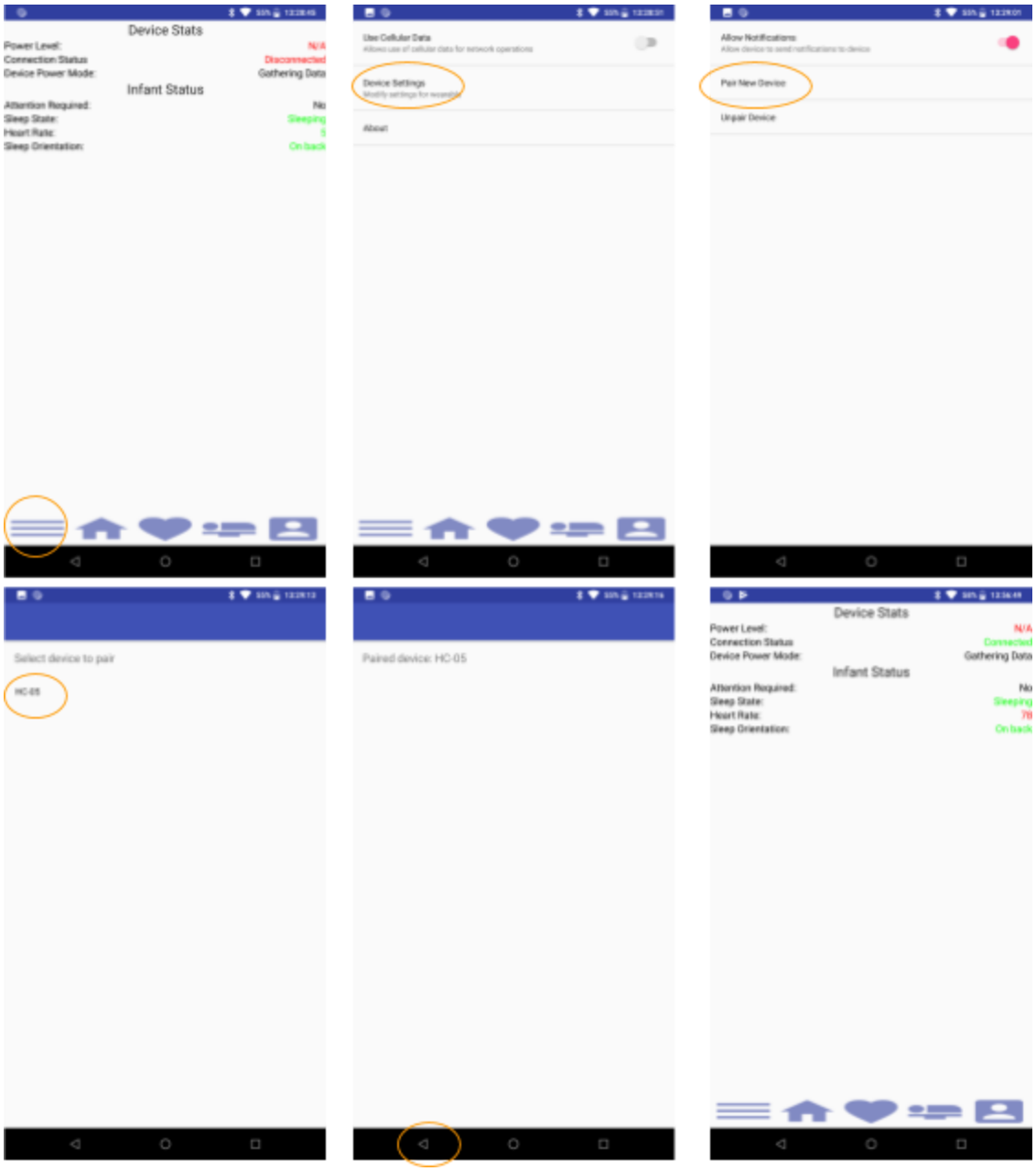
I.B.I Setup

On the opening of the application, the login screen is presented to the user. To log into the app, credentials must be used that are stored in the database. If a user does not have valid credentials, they must create an account from the login screen and use them to login after successful creation.



To login, the user simply enters their credentials and presses the login button (red). Once the login is successful, the user will be presented with the main application screen. If they do not have an account and need to create one, they must push the create account button (blue). Once the account creation button is pressed, the user is taken to the account creation screen where they must choose credentials and click the create account button. If the creation is successful, the user is taken back to the login screen where they will login using their freshly created credentials. Otherwise, if the credentials are invalid, the user will stay on the creation screen until valid credentials are used.

The application then has to be paired with the hardware (in this case, the name is HC-05) for the initial setup. The Android Application saves the mac address of the hardware's Bluetooth chip to connect automatically.



Navigate to the settings tab, select "Device Settings", select "Pair New Device", then select the device to pair. Once the device is saved, it will be indicated the device is paired. Navigate back to home screen to see the status of the connection.

I.B.II Demo

When connected, the home screen will periodically update as it reads in data over Bluetooth. The heart rate screen will also display data in real time.

I.C. Web Application

I.C.I Setup

The web application must be deployed to a server with PHP and MySQL support. Once that has been obtained, there needs to be some credential updates in the code in order to communicate with the database, which have been marked in the code base. The username and login of the database need to be updated to whatever the new database supports. The supporting tables must also be created with the correct schema. We currently have a table for each data type, accelerometer, heart rate, and temperature, which are linked to our users table, which contains the user specific data such as age. We also maintain a login table, to allow people who do not own a device, to still login, for instance to view the historic data.

I.C.II Demo

The current deploy of the code can be viewed on http://sdmay18-22.sd.ece.iastate.edu/data/index.php , where you can either create a new login, or login as an existing user. From here you are then able to specify your data parameters, with the form seen in Appendix III.C.I, and visualize and data that is stored in the database.

I.C.III Test

In order to test the website, the REST endpoints are listed in Appendix III, these can be sent data, and verify the response is successful. To test the functionality of the graphing library, you can specify the data parameters in the form at /data/projectData.php, and verify that any data POSTed to the database shows up in the graph.

Appendix II : Initial Design

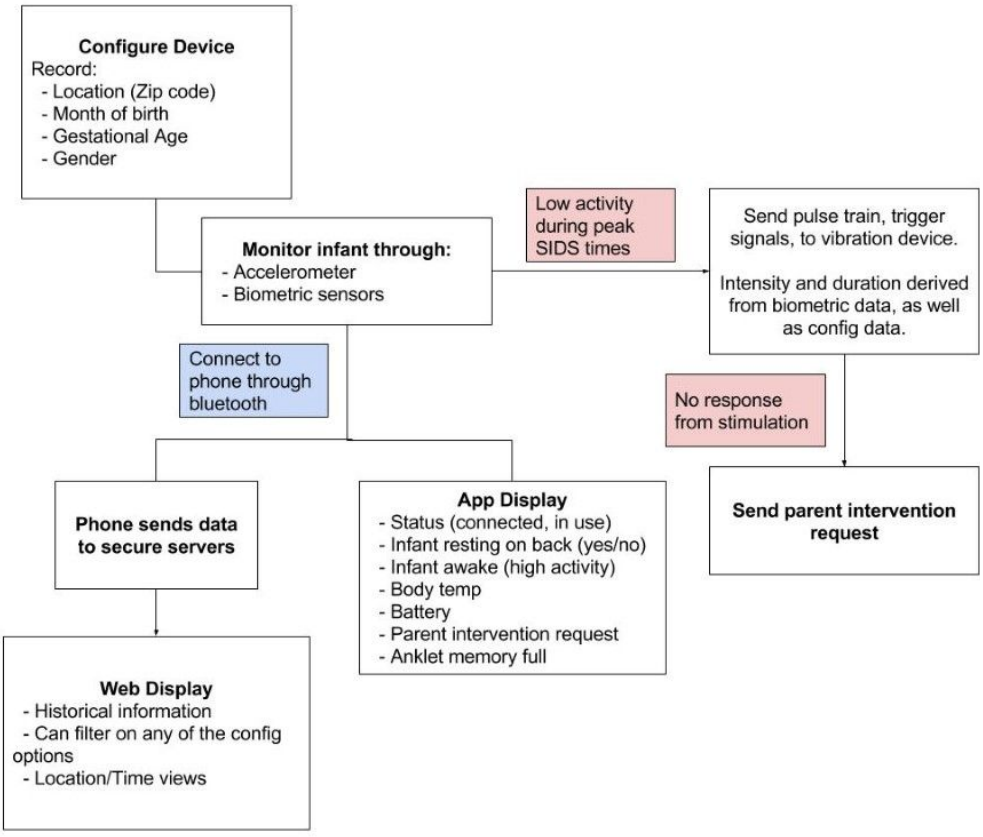
II.A Hardware

Initially our project was more of a hardware project. Our clients' initial project was to create a wearable anklet with flexible straps and a small form factor. With the android app and possible web app being side goals. We first started down that route, looking into designing PCBs and how

to best do that for our use. We started selecting parts with criteria of lowest power consumption and smallest form factor.

After a few weeks of trying to learn basics of pcb, since none of us had any experience, our adviser suggested we focus more on the software goals. We talked with our client and he agreed. We switched to proof of concept (sensors attached to an Arduino) because of that, the hardware anklet portion of the design was changed.

II.B Block Diagram



Appendix III : Other Considerations

III.A REST Documentation

III.A.I Login

```
Path : /data/login.php
Headers : {
    'Content-Type: 'application/json',
}
JSON: {
    'username': '<username>',
    'password': '<pw>'
}

Response :
    status_code => 200
    reason => "OK"
```

III.A.II Create User

```
Path : /users/create.php
Headers : {
    'Content-Type: 'application/json',
}
JSON: {
    'username': '<username>',
    'password': '<pw>'
}

Response :
    status_code => 200
    reason => "OK"
```

III.A.III Register Device

Path : /users/register_device.php
Headers : {
    'Content-Type: 'application/json',
}
JSON: {
    'gender': '<Male/Female>',
    'birth_month': '<Jan/Feb/Mar/etc.>',
    'gestational_age': <months:int>,
    'zipcode': '<12345>',
    'latitude': <-23.4572:float>,
    'longitude': <98.123:float>,
}

Response :
    status_code => 200
    reason => "OK"
    result => {
        'id': <12:int>
    }

III.A.IV Upload Data

```
Path : /data/users/upload_info.php
Headers : {
    'Content-Type: 'application/json',
}
json : {
    'user_id': <id:int>,
    'heart_rate': {
        '<Timestamp: 2018-02-08 19:46:21>': <bpm:int>,
        '<Timestamp: 2018-02-08 19:47:21>': <bpm:int>,
        ...
    },
    'accelerometer': {
        '<Timestamp: 2018-02-08 19:46:21>': [<x>, <y>, <z>, <mag>],
        '<Timestamp: 2018-02-08 19:47:21>': [<x>, <y>, <z>, <mag>],
        ...
    },
    'temperature': {
        '<Timestamp: 2018-02-08 19:46:21>': <temp:int>,
        '<Timestamp: 2018-02-08 19:47:21>': <temp:int>,
        ...
    },
}

Response :
    status_code => 200
    reason => "OK"
```

III.B Room Persistent Database (Android Application)

Local database on Android Application to store data which is read from Bluetooth.

III.B.I Biometric Data

Represents one entry in the table

Columns: timestamp (long), heart_rate (int), temperature (int), accel_mag (float), x_accel (float), y_accel (float), z_accel (float)

Methods:
setTimestamp(long timestamp)
setHeart_rate(int heart_rate)
setTemperature(int temperature)
setAccel_mag(float accel_mag)
setX_accel(float x_accel)
setY_accel(float y_accel)
setZ_accel(float z_accel)
getTimestamp()
getHeart_rate()
getTemperature()
getAccel_mag()
getX_accel()
getY_accel()
getZ_accel()

III.B.II BiometricDataDao

Interface for getting and editing entries in table

Methods:
getAll(), *Returns all entries in table*
getMostRecent(), *Returns the last entry in table*
insert(BiometricData data), *Inserts new entry in table*
delete(BiometricData data), *Deletes entry from table*

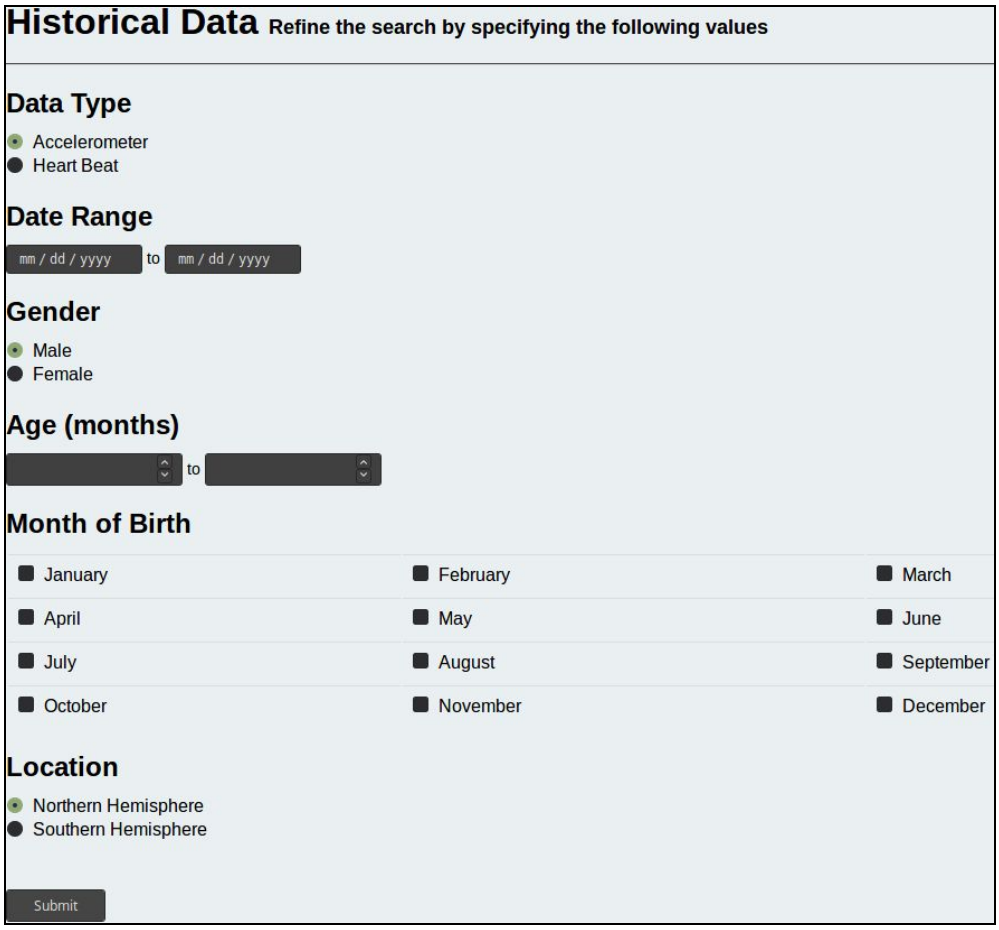
III.B.III BiometricDatabase

Represents the SQLite database, the database is instantiated following the singleton pattern.

Methods:
dataDao(), *gets the BiometricDataDao interface for this database in order to interact with entries*
getBiometricDatabase(), *gets the instance of this database, creates new instance if not instantiated*
destroyInstance(), *destroys the instance of this database*

III.C Website Images

III.C.I Data Form



Appendix IV : Sources

[1] Baby Monitor | Official VTech® Audio and Video Baby Monitors. (n.d.). Retrieved December 05, 2017, from https://www.vtechphones.com/products/baby-monitors

[2] Sproutling Wearable Baby Monitor. (n.d.). Retrieved December 05, 2017, Retrieved from http://fisher-price.mattel.com/shop/en-us/fp/sproutling-sleep-wearable-fnf59