

**An educational Android application – The SeeLife project**

Zhonghao Zhan, Jiakai Chen, Gyurae Kim

Instructors: Dr. Michael Fu <EECS dept.>, Dr. Gregory S. Lee <EECS 398>, Erika Woodrum <Art Design & Project Management>

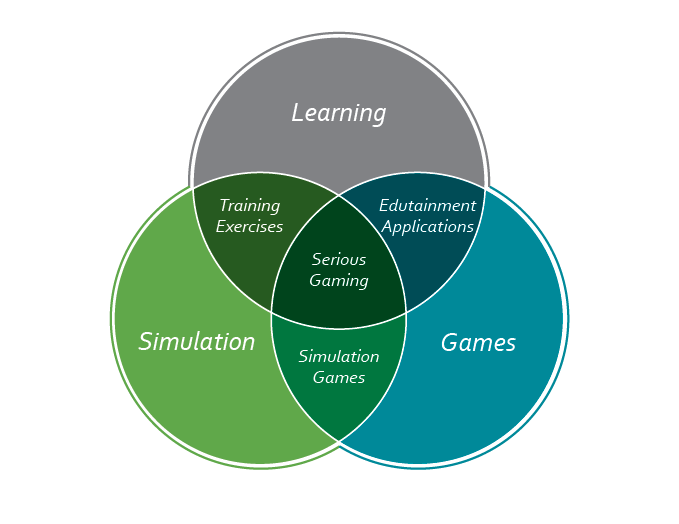
Email: zxz397@case.edu, jxc890@case.edu, gxk163@case.edu

Present date: 7/25/2017

**Executive Summary**: The goal of the SeeLife project is to produce an educational biofeedback game on mobile devices to help children at ages 6-8 overcome their anxiety and to teach them how to calm themselves down. To achieve this objective, we utilize the heart rate sensor and connect it to the device via Bluetooth Low Energy (BLE) connection. On the other hand, the user interface is designed to be user-friendly and cartoonish. All of these features, including the wireless connection and the biofeedback mechanism, effectively train the children at ages 6-8. The heart rate data can be easily synchronized to computer database for further analysis and researches by parents and doctors.

**1. Introduction & background**

Seelife Production is a project that introduces a new treatment for the anxiety of children age 6-8. For those who experience difficulties when facing stressful circumstances, the effect of anxiety will cause poor self-esteem and social interaction failure. As a solution, Seelife provides a biofeedback therapy via introducing an engaging game. We provide a two-step design to achieve the goal. Firstly, we transplant the Seelife game demo from Windows to Android platform. We also change the control system of the game from the keyboard to accelerometer. In this way, the player can control the moving direction of the game character by tilting the device. Then, we retrieve the heart rate data wirelessly by using BLE technology without a long delay. BLE technology is a relatively new technology that has the simplicity, low power, and compatibility advantages. The goal of this project is to successfully build a BLE connection between the heart rate sensor and the Android system.



**1-1. Concept of Serious Game**

A serious game is a game designed for a primary purpose other than pure entertainment. Serious games use game environments and techniques to train or educate users or to promote a product or service in an engaging and entertaining way. The “serious” aspect of SeeLife comes from the fact that it enhances the ability to control anxiety and emotions of young children.

**1-2. Mechanism of the game**

The basic idea of this game’s mechanism is the player controlling a blue fish swimming in the sea, gathering coins as many as possible to score. During the game, there will be two obstacles: sea monster and fish hooks prevent the player to finish the level. Touching either of the two obstacles will cause the level to be lost, and the player has to replay from the beginning (Woodrum, Erika. 31). The obstacles appear randomly. The appearing frequency and moving speed of the obstacles are increasing as game levels going further. The final goal is to reach the Golden Starfish at the end of each level to win the game.

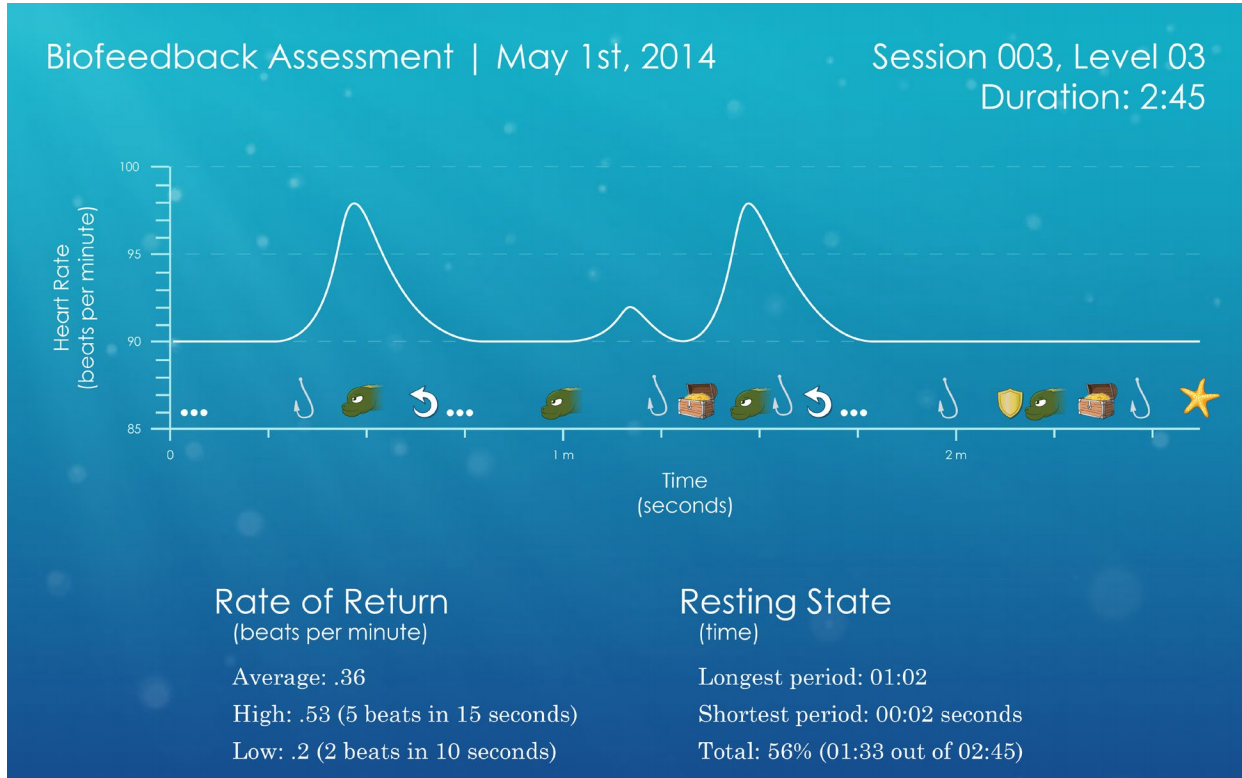


Figure 1. The design of heart rate analysis plot

During the game, the player's heart rate will be monitored. When the heart rate is steady, a bonus meter will fill and will reward the player with coin collection. When the player's heart rate affected by the obstacles, the sea water will get murkier, letting the obstacles being hard to see in advance. The face of the game character - the fish, will also become unhappy. At the end of the game, an analysis plot of heart rate will be presented. The design of this mechanism is to teach the players autonomic self-relaxation.

**Hardware setup**

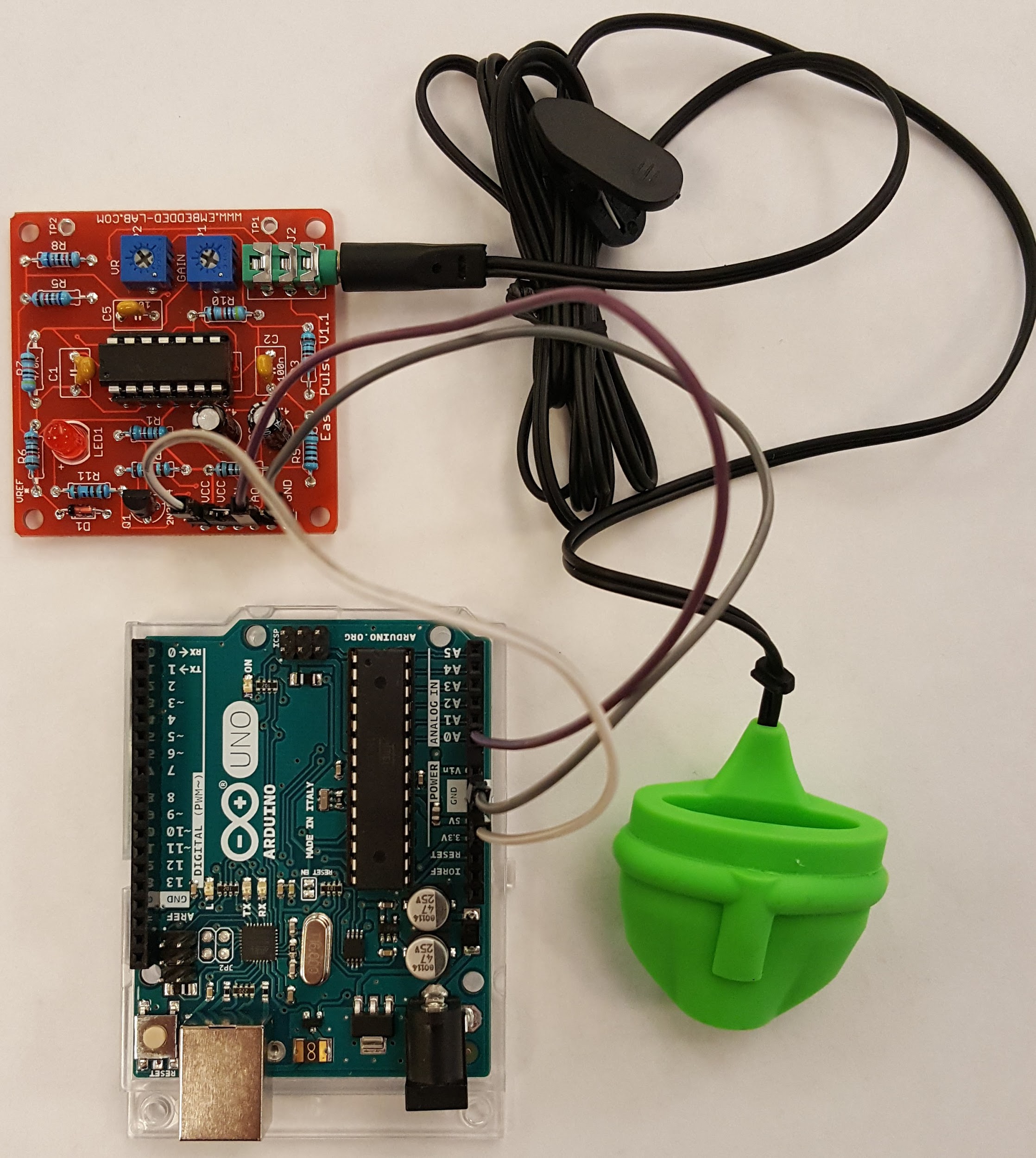


Figure 2. the serial setup

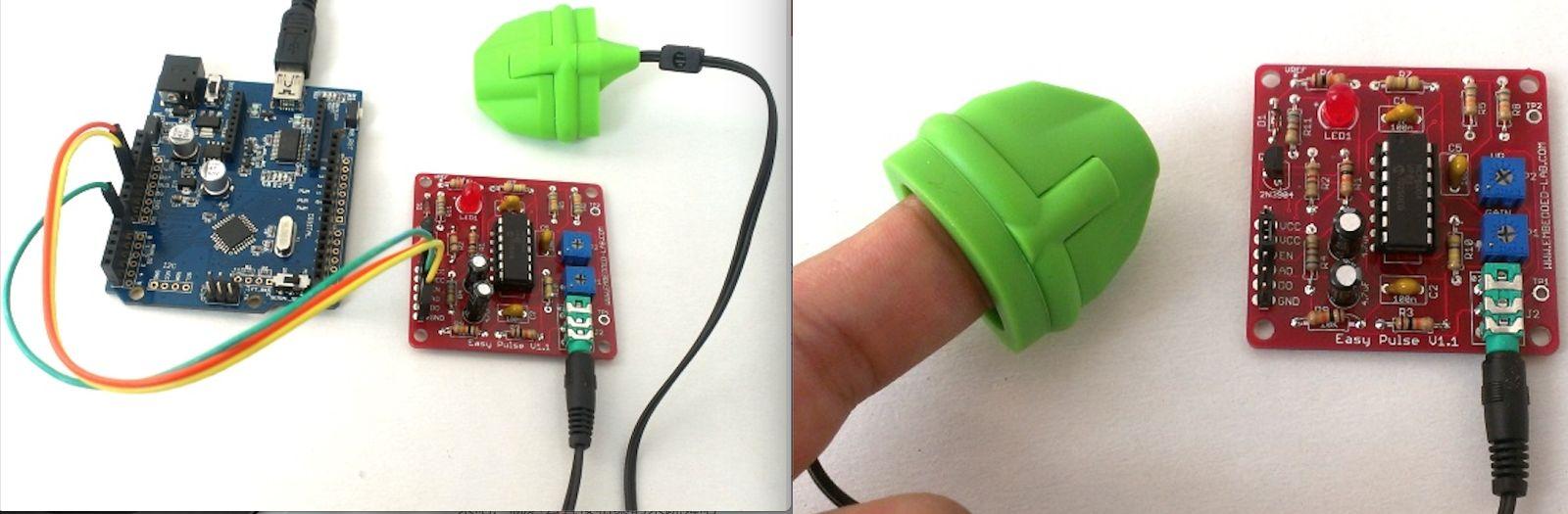


Figure 3. A wired heart rate sensor on the fingertip

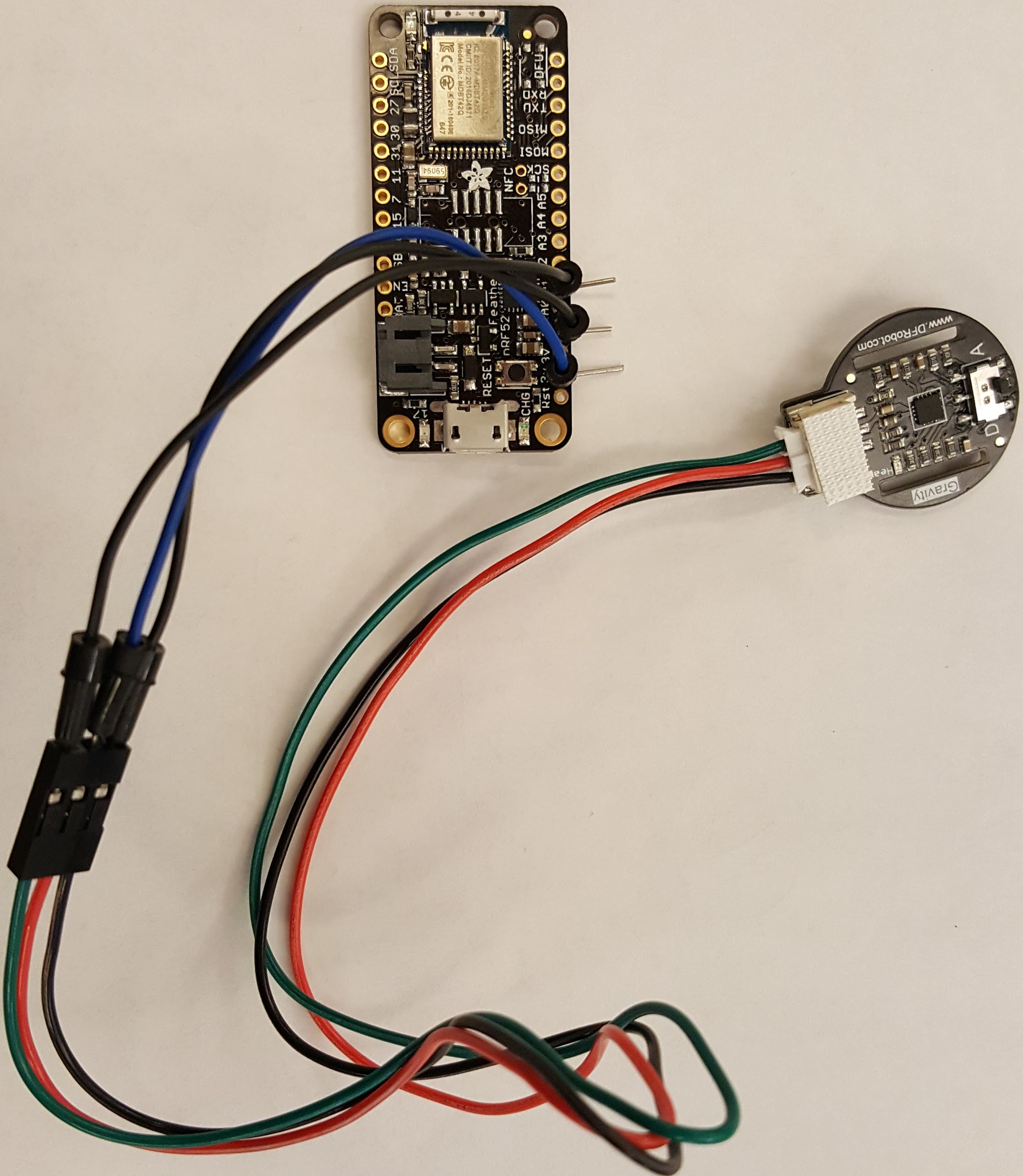


figure 4. the BLE setup

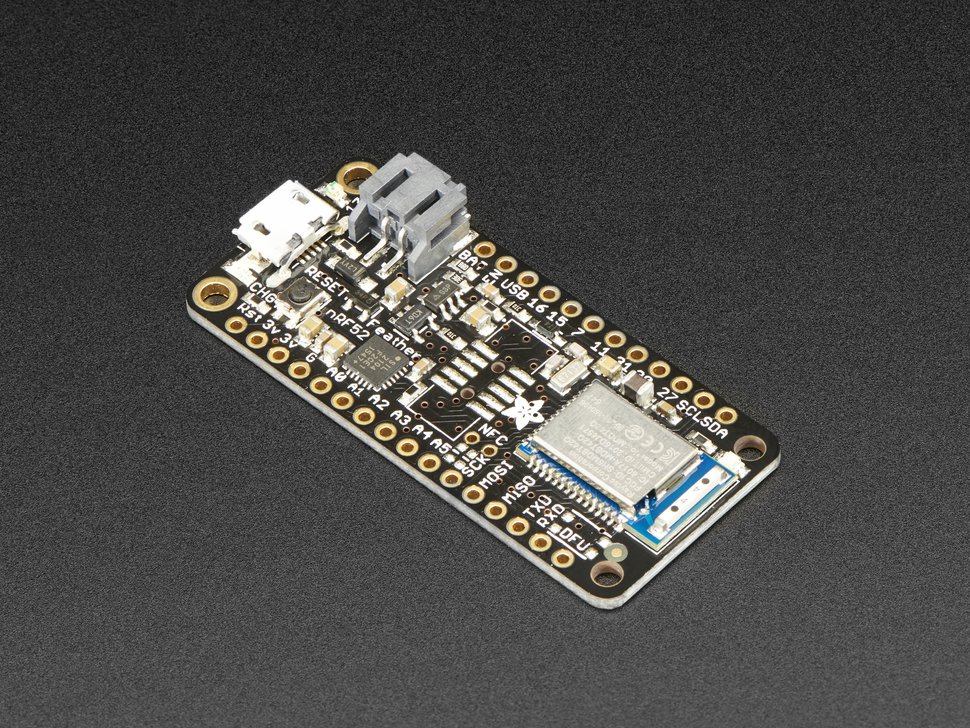


figure 5. the BLE board: Adafruit nRF52 Feather

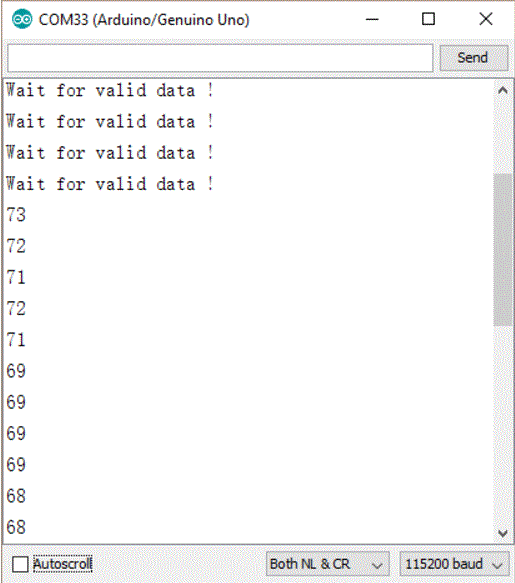


Figure 6. the heart rate reads with sending rate of 115200 baud



Figure 7. The movement control

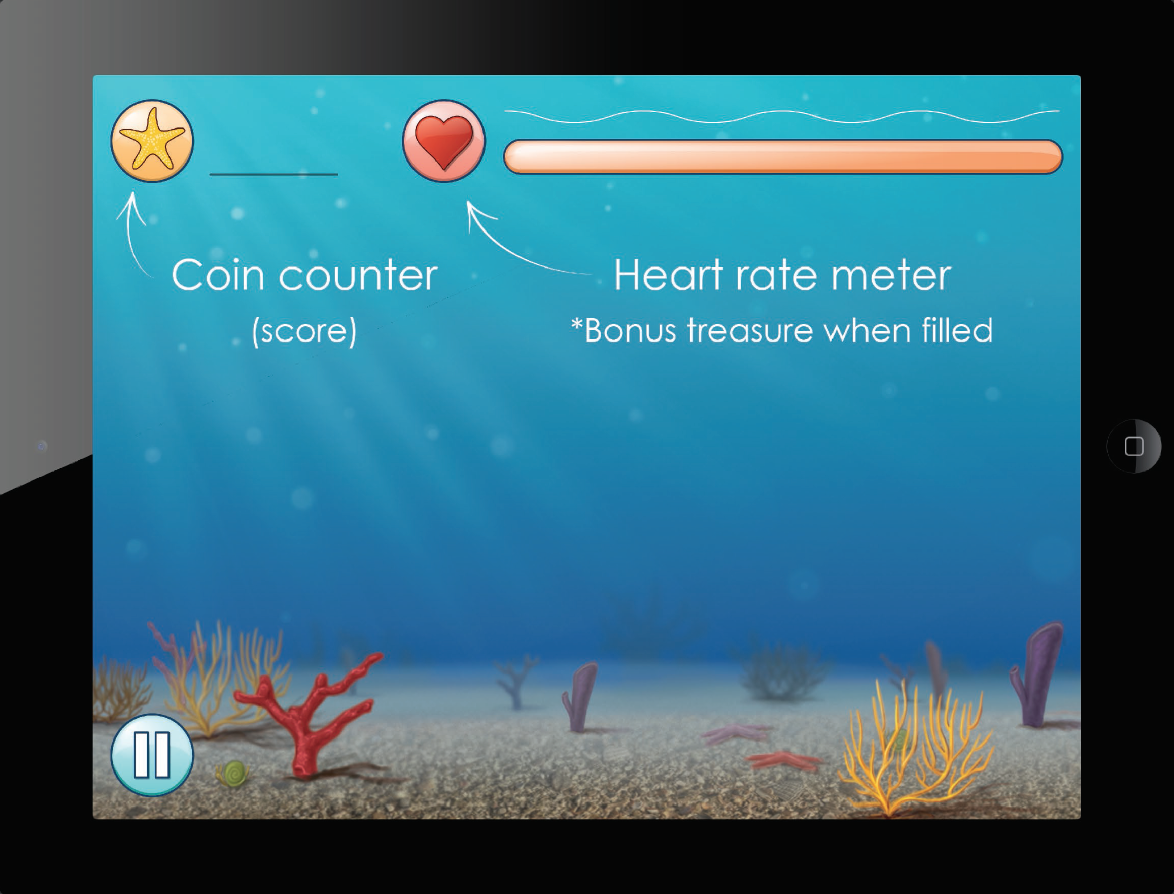


Figure 8. The HUD (head-up display) layout

**1-3. Potential Setbacks**

The potential problems of our design including the strength of the BLE connection between devices and sensors, the power consumption and the compatibility of the application on various Android devices. These problems will affect the quality of the game experience and the accessibility to the users.

Since SeeLife is a biofeedback game, the precision of the data procession is the key of the game mechanism. If the BLE connection is not strong enough, the heart rate data will not properly sync with the game application and the user will not get a correct feedback. On the accessibility side, the power consumption of the sensor and the board must be low enough so that the user does not need to change or recharge the battery frequently. These problems will definitely reduce the accessibility. Furthermore, the game must be compatible with most of the major mobile devices in the market.

In order to resolve these possible setbacks, we designed and conducted several testings and analysis.

**1-4. Industrial Standards**

The industry standard for this project, such as create a more “complete” hardware set is also a point of consideration. Although the stability of the hardware circuit has been proved, the layout of the circuit is still a barebone. To industrialize this product, the next step is to make a proper “shell” for the prototype circuit, or directly implement the heart rate sensor algorithm in the industrial product such as Fitbit Flex or Apple watch.

**1-5. Relate Patents**

There is no patent that has a direct relation to this project. Programming the hardware, we referred to the open source codes specified under Reference section.

**1-6. Relevant Papers**

See Reference section.

**1-7. Other references**

1. Dr. Michael Fu <mjf24@case.edu>: Assistant Professor of Robotics and Haptics of EECS department, Case Western Reserve University. The technical advisor of the SeeLife project. He also provides research orientation instructions on both software and hardware level.
2. Erika Woodrum <ewoodrum@fescenter.org>: Medical Illustrator of Cleveland FES Center. The art designer of the SeeLife game and the project manager.
3. David Miron <david.miron@duke.edu >: EE Student of Duke University. The chief programmer of the prototype of SeeLife game on PC platform and the designer of serial connection between wired sensor and computer. He is the designer of basic game mechanics.
4. Brian Li <bvl8@case.edu >: CS Student of Case Western Reserve University. Experienced Android developer. Helps BLE connection handshake problem between Unity and Android Studio during our BLE develop process.

**2. Objectives and Technical Specifications**

There are three main objectives of the SeeLife project:

1. Transplant the PC platform prototype game into Android system, and change the game control from the keyboard to the accelerometer.

2. Establish Bluetooth Low Energy connection and a serial connection between the device and the heart rate sensor.

3. Calculate the user’s heart rate and transmit the data from the sensor to the device via BLE connection. Address the heart rate data inside the SeeLife application, and process the data to control the game factors.

To achieve the first objective, we use Unity Engine (ver 5.6.2) to process and develop the project files. Since the game will be transplanted into the mobile platform, we have to change the game control from the keyboard to the accelerometer. In this part, we enable the accelerometer input function in Unity engine, and utilize filter algorithm to adjust the sensitivity of the game control. Since the player character can only change the angle of where he is moving to. We did not use the gyroscope. A Gyroscope is 3D sensor that might cause some bugs, if we just need to monitor 2D instead. We only take accelerometer values for x and z axis, and it allows at least 5 degrees of rotation as the input. Then, by utilizing the “building” function of Unity Engine, we generate the .apk file of SeeLife application and install it on the Android system.

After successfully transplant the game, we have to design both the serial and BLE connection circuit for establishing the connection between the device and the heart rate sensor.

First, we reproduce the original serial circuit design by David Miron. This circuit includes a Easy Pulse v1.1 sensor set by Embedded Lab, and an Arduino Uno Rev3 by Adafruit. By connect A1 (analog input), 3.3V (power supply) and GND (ground) pins, the Easy Pulse v1.1 will transmit raw heart rate data to Arduino Uno board.

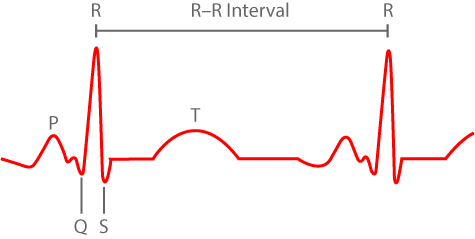


Figure 8. An example of heart rate waveform

Since there are four spikes in the waveform of a complete heart beat, but only the R wave will be counted, we set a threshold on the Arduino Uno board to filter the raw data. Firstly, we set the amplitude of R wave as 100%, and any wave with less than 50% amplitude will be ignored. Then the board will count the number of the R wave and the time elapsed to calculate the BPM (beat per minute). The calculated data will be sent in bytes to the computer for further uses.

On the computer side, we use Unity engine to process the raw data in bytes. We appoint a specific USB port on computer “usbmodem 1411” to receive the data, with sending rate of 115200 baud. With these setups, the data can be sent to the computer. On Unity side, we get the heart rate data update from port “usbmodem 1411”, and use the data to control the brightness of the game’s background and the game character. At this point, the technical setup of the PC prototype has been successfully reproduced.

Then, to transplant this heart rate sending-processing procedure to mobile platforms, we have to redesign the serial connection software code. On the other hand, since our final goal is to build a steady Bluetooth Low Energy connection between the device and the sensor, we also have to design a BLE connection at the same time. Therefore, from week 4, we use a parallel work schedule.

Firstly, for the serial connection design, we use a USB OTG (on the go)cable to connect the circuit to the Android devices. The algorithm of calculating heart rate on the Arduino board stays unchanged. Most modifications are on the Unity engine side. Because building the connection with Android device is very different with building connection with PC, we look into the original code of “getting heart rate data” in Unity. This time, we disable the port “usbmodem 1411”. Instead, we query for all connected devices on the Android platform, and set a specific “vender ID” to match and pair our Arduino board.

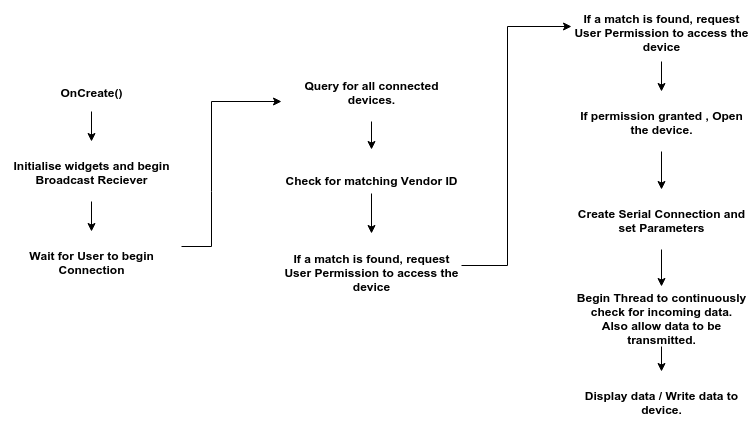


Figure 9. The program flow[1]

For each USB slave device, there is a vendor ID for identification. If the vendor ID of the Arduino board matches that of a connected device, the program will identify it, and get data from that device. Although most of Arduino devices’ vender IDs are 0x2341, the Arduino Uno Rev 3’s ID is 0x2a34 after inspection. After setting the matching ID as 0x2a34, we can get the data from our serial circuit.

On the other side, the BLE connection design is different with the serial connection. Instead of sending data via cable to a specific device, BLE connection is actually broadcasting the data in order to communicate to the outside world. Therefore, we have to figure out how to make a proper “handshaking” procedure between the devices and the BLE sensor circuit. In the profile called Heart Rate, there are two characteristics, Heart Rate Measurement and Body Sensor Location. We did not use Body Sensor Location characteristic for the project. The Heart Rate profile has a UUID of 0000180d-0000-1000-8000-00805f9b34fb. Heart Rate Measurement characteristic has a UUID of 00002a37-0000-1000-8000-00805f9b34fb. Then we have to set these two UUIDs in our software code in Unity as the matching code for both “Notify” and “Read” function of BLE connection. This step is similar with setting the vendor ID in the serial connection part, but more complicated.

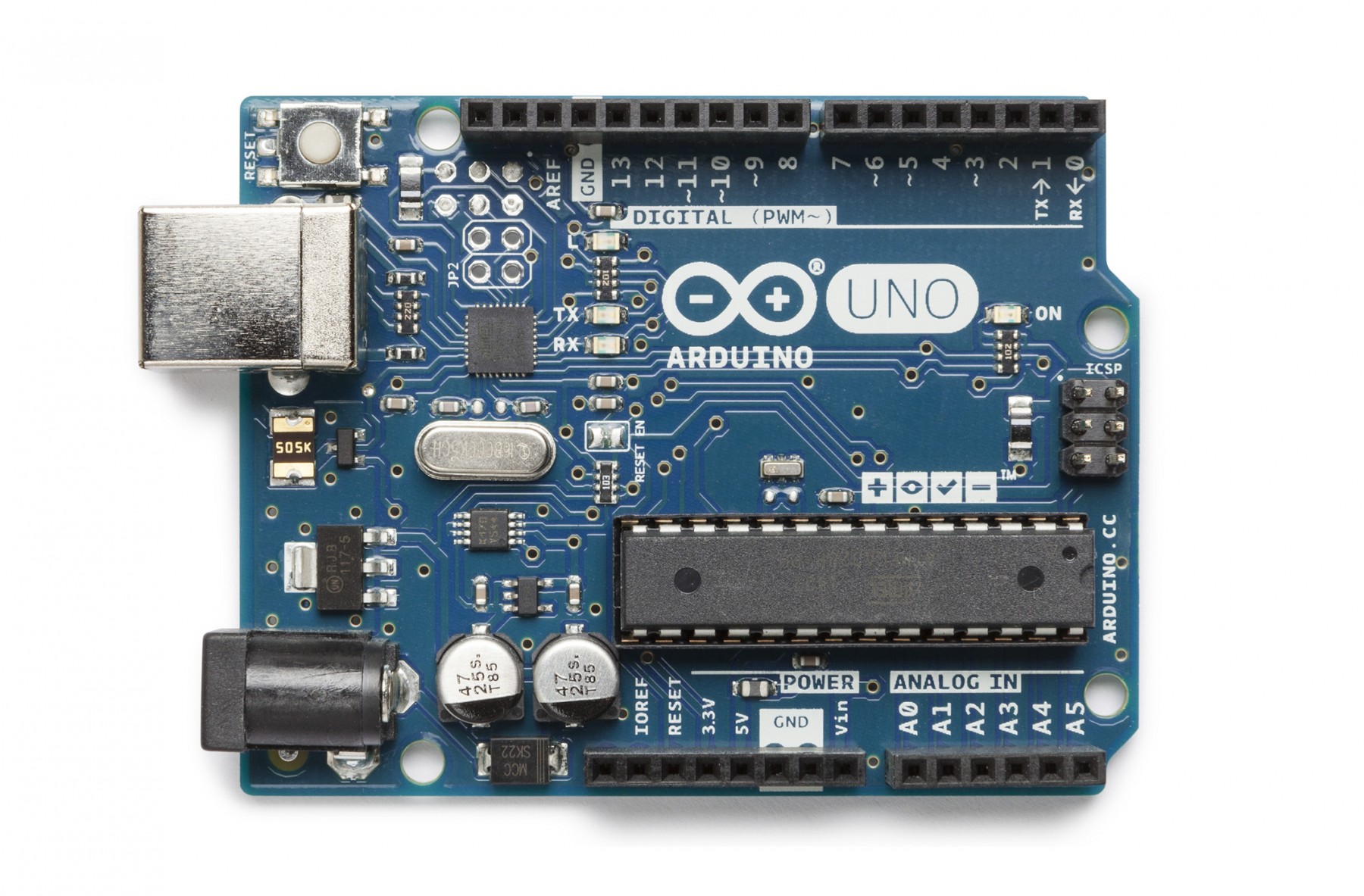


Figure 10. The Arduino Uno Rev3 Board

After finishing the software level design of both serial and BLE connection, we look into the circuits. For serial connection, we still choose to use Arduino Uno Rev 3 board. It’s working voltage is 3.3V, with a clock speed of 16 MHz. Because this board is programmed by us, we are not sure the operating current at this point. But we measure and specify this data later in the testing report. One component different with the PC prototype game is the sensor. We substitute the Easy Pulse V1.1 with the Gravity Heart Rate Sensor. The working voltage of this heart rate sensor is also 3.3V, with an operating current of less than 10mA.

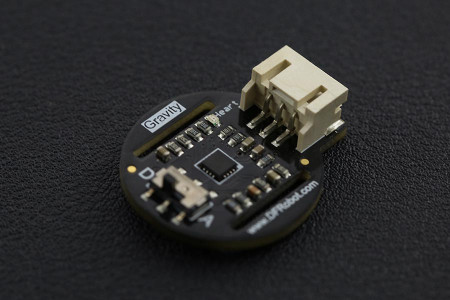


Figure 11. The Gravity Heart Rate Sensor

About the BLE circuit design, we use the same heart rate sensor as the serial connection circuit. However, we change the Arduino Uno Rev 3 board to Adafruit nRF52 Feather. It’s working voltage is 3.3V, with an operating current of 10mA.

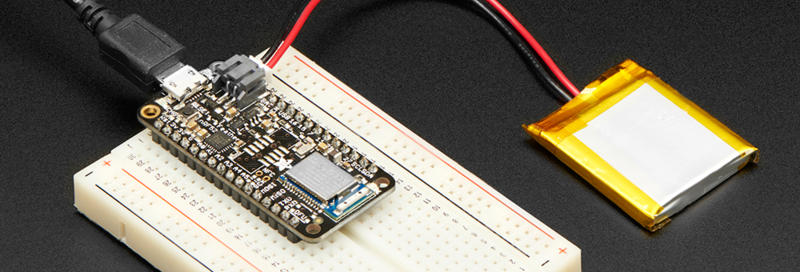


Figure 12. The Adafruit Feather with a battery

To supply this board, we use a 350mAh Lithium Ion Polymer battery as the power supply.With these two sets of circuits, we start our demonstration of the SeeLife project. Although the serial connection circuit is only for test purpose and will not be used in the final version of our product, it could be the backup plan since the serial connection is more robust. It is also easier to sync data to the computer via the serial connection. On the other hand, the BLE setup, although still “bareboned”, will be used as the final hardware setup for the SeeLife project.

With these specified specs and setups, the BLE circuit works smoothly in the final demonstration of the SeeLife project. Although several technical points have changed compare with the PC prototype of the SeeLife application, there is no major change in the orientation of the project. And the game performance and simulation data will be provided on Part 4 Testing to prove that our designs meet our technical objectives.

**3. Methodology**

**3-1. Introduction**

The Seelife game is originally programmed on Unity, which can only be executable on Windows operating system. It also requires the player to sit down and wear a serial heart rate sensor. Our basic software design of the game is to let the player control the fish character, avoiding the obstacles randomly showing in a steady moving canvas. The heart rate changes will be monitored and recorded. By the end of the each level, there will be an assessment chart that shows the overall heart rate changes with a timeline of when the obstacles showed. The parent or doctor can view the assessment chart and better understand the children’s conditions. (Figure 1)

Since the game is primarily for 6-8 years old children and children are always active when they are playing the game, we want to transplant the game to Android devices. It will be much more convenient for the children to play. Second, we need to decide a way to get the heart rate wirelessly. The serial connection is not suitable for children, because the sensor in serial connection requires the player to be steady and patient. We decided to use Bluetooth Low Energy to transmit the heart rate data because it is simple, low powered, and flexible. Since none of the team members have studied BLE connection before, we cannot ensure that we can complete the BLE feature when the course ends. We consider the serial connection with the Android device by using OTG cable as a backup plan. If we fail on BLE connection, at least, we can work on serial connection with the Android device by using the OTG cable.

**3-2. Design procedure**

In the first week of our project, we were working on writing the proposal, choosing the compatible hardware and improving the game mechanics. Zhonghao Zhan and Gyurae Kim were working on choosing the hardware, and Jiakai Chen was working on improving the game, which is adding a gyro-control to the game.The hardware selection was made in accordance with specific criteria.

1) A programmable board with Bluetooth Low Energy capability.

2) The board must be light and small enough to be worn on a wrist.

3) The heart rate sensor must be sensitive enough to read the pulse on a wrist.

4) The sensor and the board must be compatible.

5) The entire hardware prototype must run at least 24 hours with 350mAh Lithium Ion Polymer battery.

6) The devices must be affordable.

Following these requirements, we chose to purchase Arduino 101, designed and provided by Arduino and Adafruit Feather nRF52, designed and provided by Adafruit. We used Arduino 101 for testing purposes. However, we ended up not using 101 because it did not meet some of the design requirements. We specifically chose Adafruit Feather nRF52 because it fits our needs and requirements. This chip is light (5.7 grams) and has a small dimension (51mm x 23mm x 8mm) so that the prototype could be worn on a wrist. Also, its CPU (ARM Cortex M4F) has enough calculation power to accurately find out the BPM from the raw EKG data with little delays. This board has a built-in 2-pin JST-PH connector for the battery and a micro USB connector to recharge the battery connected to the board. Most importantly, this device is Bluetooth Low Energy compatible (BLE technology requires significantly lower amount of energy compared to the previous Bluetooth models). We chose Pulse Sensor Amped for the heart rate sensor. Later on, however, we had to discard this device because it was not sensitive enough to collect reliable EKG data on a wrist. Our final selection for the sensor was Gravity Heart Rate Monitor Sensor designed and provided by DFRobot. This sensor met the purpose of this project much better than the previous sensor in that it is very sensitive and able to deliver both analog and digital pulse signal. We chose 3.7V 350mAh Lithium Ion Polymer battery with 2-9in JST-PH connector to power the prototype design.

The final prototype contains three different hardware parts, which are Adafruit Feather nRF52, Gravity Heart Rate Monitor Sensor, and a Lithium Ion Polymer battery. The sensor collects the raw EKG data from the user’s wrist. This sensor has a built-in ADC so that the developer can use both analog and the digital EKG signals. Adafruit nRF 52 receives the raw data from the sensor to calculate and send the BPM data to the mobile game. The 350mAh Lithium Ion Polymer battery provides 3.7V~4.2V and has a total capacity of 1.3Wh. This battery is light and small enough for the prototype and it is rechargeable at a rate of 350mAh.

The experimental prototype was built by connecting the sensor to 3.3V Vcc, Ground, and A1 pin of Adafruit nRF52. We conducted three different simulations with the hardware.

1) Receiving and monitoring the RAW analog/digital EKG data from the sensor

2) Testing the validity of the calculated BPM.

3) Sending data via BLE using specific UUIDs.

The handshaking problem between the Unity and Android Studio is considered the most difficult part in the whole design. The Unity does not have the feature that can activate the BLE connection in Android, but Android Studio does. After discussion, we concluded two options. One is Unity-main development. We developed a Unity plugin based on Android Studio to receive data via BLE. The other option was Android-main development. We would export the whole project to Android Studio, add a line of code to Unity that would direct the calculation process to the Android Studio, and do all the BLE features in Android Studio.

At the same time, Jiakai also achieved the gyro feature of the game. Since the game only needs a 2-D gyro-control. Instead of activating the gyroscope of the Android device, we decide to activate the accelerometer on the phone to accomplish the gyro feature, because accelerometer takes less load than the gyroscope. The game will run more smoothly and generate less heat during the playing process.

Then, we were working on BLE connection, which was the most challenging part of our project. Dr. Michael Fu suggested us to meet with Brian Li, who has the decent experience on BLE connection and handshaking problem between the Unity and Android Studio. Brain planned to deal with the handshaking problem just as our original second plan, which is by exporting the whole project to the Android Studio, activating the BLE feature directly in Android Studio code. Brian helped us a lot. He gave us many suggestions on programming the code and helped us debug the Android Studio code. Since all the team members were not familiar with the Android Studio, we had a hard time on studying it at the beginning.

The Bluetooth LE board has many characteristics. If we want to access one characteristic, we need to know its unique UUID. The UUID is an abbreviation of the universally unique identifier. Every board has a UUID for the device. As a result, if we want to access the heart rate characteristic in the specific BLE board, we need to activate the Bluetooth on the phone first. Then, search the paired Bluetooth devices by using the device UUID. If device UUID matches, then go on search the characteristic by using the characteristic UUID. If all UUIDs match, we can access the program that was coded in the specific characteristic. For the Feather nRF52 board, the device UUID is 0000180d-0000-1000-8000-00805f9b34fb. Heart Rate Characteristic UUID is 00002a37-0000-1000-8000-00805f9b34fb. In Android Studio, we first use “bluetooth.getBondedDevices” code to get boned Bluetooth device. Then, we use the build in method called “onServicesDiscovered” to check and connect to the BLE board. Within the method, we use “gatt.getService(UUID.fromString())” to specify the board we tried to connect to. Then, we use “gatt.getCharacteristic(UUID.fromString())” to specify the characteristic in the board that we tried to access to. Next, if the characteristic has “READ” property, we can use “gatt.readCharacteristic()” to read the data and store the data, the heart rate in digital, to an integer variable. Finally, on the Unity part, we created a javaObject called “heartrate” and use “heartrate = javaObject.Call<int>(“getHeartRate”)” code to transfer the heart rate value from Android Studio to Unity. It requires the Android Studio has a method called “getHeartRate”, which returns the integer value, so we create a “get method” and made it return the heart rate as an integer. After getting the heart rate value in the Unity, we can then attach the variable to different scripts and affect the game mechanics.

However, we met some severe problems on handshaking. In the Brian suggested workflow, we need to first export the Unity project into Android Studio project. Then we will work all the code in Android Studio. Even though we re-checked our code and tried to fix possible bugs, the Android Studio keeps sending us fatal errors, which we cannot run the game on the Android device. The error message is “Fatal signal 11 (SIGSEGV), code 1, fault addr 0xe59fc000 in tid 14007”. We searched on the internet, but we found that the error is unique. We tried to build the APK file in Unity, but the game then worked well on the device. We used Android monitor to record and see the working process of Android Studio. After we worked a whole week on debugging, we found that the real problem is not the code problem. Android Studio has some code running environment that needs to be pre-setted in order to run the code correctly. We need to change the API level through the project structure panel. Under the project structure panel, we need to change the Flavors Min Sdk Version from None to the API of your Device. For the Samsung Galaxy 8 Android Phone, the API is 24: Android 7.0 (Nougat). After we changed the running environment, we could successfully build the APK and made it run on the Android device. During the process, we studied Android Studio at a deep level and the teamwork on fixing the most difficult problem stimulated and inspired us on completing the whole project in the next several weeks.

At this point, we decided to make a parallel work schedule to make sure that all of us would constantly make meaningful progress when facing obstacles. Zhonghao Zhan would take the responsibility of building a serial connection between the device and the sensor as a backup plan. For this part, we use the USB OTG(on the go) cable to send data from sensor to the device. Different with the complicated handshaking procedure of BLE connection, the serial connection in Android system only requires us to specify the Vendor ID of the USB slave devices and query for the external connections. To test the feasibility of the serial connection, Zhonghao implements a serial data receiver application on Android system. This attempt makes a good progress at the beginning. By specify the Vendor ID 0x2A34, the data receiver successfully identifies the Arduino Uno Board in the serial connection circuit, and updates the sensor readings to the Android device. Although the data receiver can address the data, there are still problems for transmit this data to the SeeLife application. To address this issue, Zhonghao tries several methods. The first one is to build the data receiver as an Android plugin, and attach it to the SeeLife application.The second is to use the TCP(Transmission Control Protocol) protocol. When testing the first solution, we noticed that the Android plugin inside an Unity project can only print out texts or pictures, but it is hard to address data outside of the application which it plugs. After facing this setback, Zhonghao moved to the TCP protocol. But this attempt is also time-consuming.

In the next few weeks, we used the “Debug.log” built-in code, which is the most common debug method, to show the running process and decide where should be improved. We changed the hardware from keep sending the dummy heart rate data to the real heart rate data, and it worked with the software correctly. We improved some game bugs that required the player to rerun the game in order to play it from the beginning.Since this is the project that all the team members do not have experience, we have plenty alternative designs both on hardware and software. If we cannot succeed on our original plan, we can at least achieve the alternative design.

**3-3. Alternative Design plan**

For the software, we have two choices on designing the BLE connection in our initial plan. One is programming on Unity by making an asset from Android Studio, and another is programming on Android Studio. As the description from above, we met lots of problems on BLE connection by programming on Android Studio. Therefore, we tried to use Unity asset to achieve the BLE connection. We bought a Unity BLE asset from the asset store in the Unity. However, the Unity asset is not working on Bluefruit Feather nRF52 board. Although we asked the author on how to use the BLE Unity asset on another board, he said we need to reprogram the whole asset, which is much harder than we thought. Since we only had two weeks left at that time, and we had already done so many works on Android Studio, we decided to abandon this alternative choice.

For the handshaking problem, the original plan, completing the BLE connection, met many different problems on interacting two software. We cannot ensure that we can achieve the BLE connection on time. As a result, we plan to start working on the serial connection alternative design simultaneously. We used the OTG cable to make a serial connection between the Android device and the heart rate sensor. We have succeeded in making an independent Android application that can receive the data and show on the screen. Then, Dr. Michael Fu suggested us using the TCP socket to deliver the data The TCP method is a to skip the connection between the game and the BLE board. Instead, we use an application to get the BLE data first, and then use the TCP socket to send the data to the game. When one of the group members started combining the application with the game, the BLE connection had a significant progress. Then, we decided to abandon the serial connection alternative design and work harder on improving the BLE connection.

**4. Verification and Results**

**4-1. The software testing details**

According to the result of the final demonstration on July 25th, 2017, the final project of SeeLife works properly. With the BLE circuit, the heart rate data transmits steadily, and the game will give feedback for the change in data and parameters.



Figure 13. The SeeLife project during the test at Case Western Reserve University, 7/25/2017

This fact shows that our design of the BLE circuit meets the technical objectives which we set at the beginning of the semester. On the other hand, the performance of data transmission is also strong. The setting of data sending rate is 2 data points per second, which means any subtle changes of heart rate will be monitored and updated on time. In the later test, it is also proved that this data transmission connection is reliable up to 40 ft. Within this range, no data point will be missed during the transmission procedure.

The completion of the game play, including the HUD design, menu design, level design, and the accelerometer application have been proved during the demonstration of the game play. The game character moves forward in a proper speed, and the obstacles (sea monster and hook) appear as expected in the level. The filtering of the accelerometer, which was a concern during the first half of the developing period also has been solved. Since this game’s mechanism is based on the biofeedback (heart rate), we have to make the game control a little bit harder. The game character will not have a “dead zone”, which keep it go in straight line. The player has to keep tilting and handling the character to make sure it will move in the desired trace. The “Flappy Bird” like design make sure the players will provide evident biofeedback to trigger the game mechanism, and to be trained effectively. The At this point, the front-end design (client/user side) has been proved that it meets the technical objectives.



Figure 14. The SeeLife project during the test in VA center Cleveland 7/26/2017

**4-2. The hardware testing details**

The first test we ever made is the waveform and data log test. When we attach the device to the waveform monitor/plotter in the Arduino development kit, the data which transmitted from the sensor to computer will be reflected. Firstly, we verified that whether the analog of heart rate data input is good.

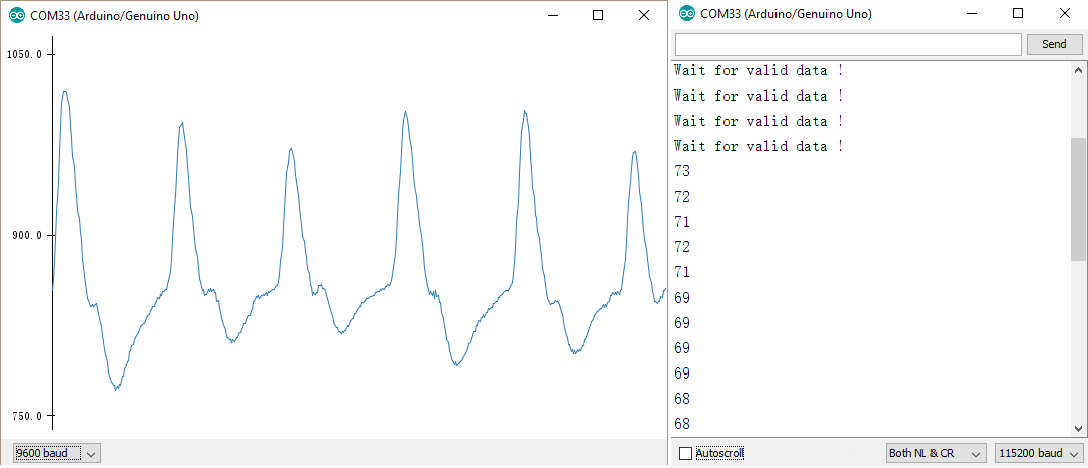


Figure 15. The heart rate waveform and the testing log in Arduino

As the figure shown above, the waveform here matches the model of heart rate waves. This fact shows the data collected by the Gravity heart rate sensor is in the correct form. On the other hand, by comparing the waveform and the digital number printed in Arduino log, we are sure that the number of BPM in the log is correct.

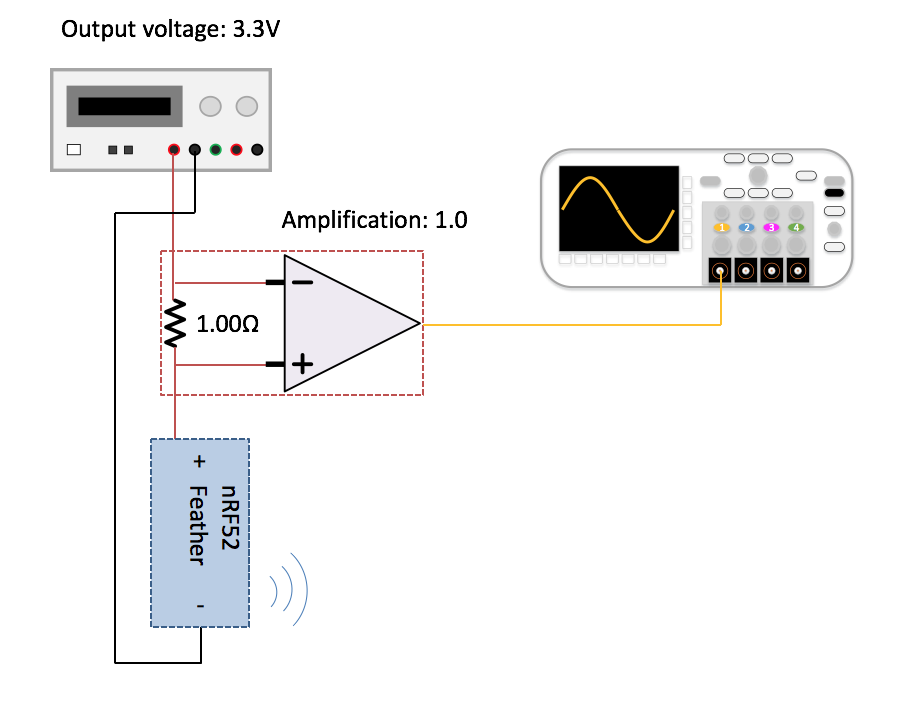


Figure 16. The setup of power consumption test

After verifying the data is correct, Zhonghao builds a power consumption test of the BLE circuit. In this test, we utilize a power supply, a function generator, an amplifier/ resistor circuit (actually does not use to make amplification) and an oscilloscope. After making this setup, we can monitor the waveform produced by the BLE board and collect the waveform of data transmission. Firstly, we capture the waveform data, and import the data into a computer. Then, we use OriginLab 2017 to process the data. Both the original data waveform and the dataset which processed by the OriginLab 2017 have been saved for further research.

In the analog plot below, you can see there are two major spikes in a single data transmission action. We use the “area integration” function in the OriginLab to calculate the area here. Since this plot is time elapsed vs. current, according to the formula :

E = P \* t = V \* I \* t

We can calculate the power it consumed here, since the working voltage of the BLE circuit is constantly 3.3V. Refer to OriginLab integration, the area here is 4.929 E-4. Therefore, the power consumption of a single data transmission will be 1.873mJ.

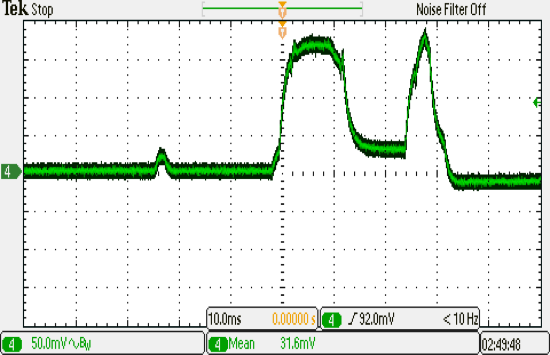


Figure 17. The analog waveform of data transmission

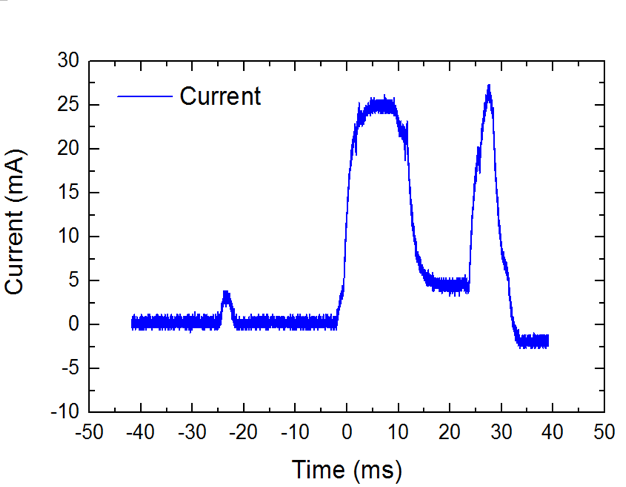


Figure 17. The integrated waveform in OriginLab 2017

After finishing the power consumption test, we start the test of signal transmission range. In this test, we set up 6 checkpoints by increasing 10 ft for each gateway, up to 60 ft. The result shows that the data transmission will keep steady up to 40 ft (12.2m). This result demonstrates that the transmission range will not be a issue for the BLE circuit.

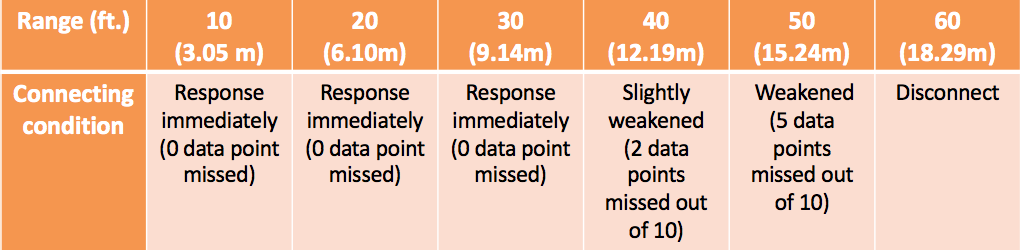


Figure 18. The range test of the BLE circuit

The last part of hardware testing is the CPU usage. We test the game on the Samsung Galaxy 8 Android Phone. The CPU type of Samsung Galaxy 8 is Octa-core (2.3GHz Quad + 1.7GHz Quad), 64 bit, 10nm processor. According to the CPU usage test function in Android studio, the maximum CPU usage of SeeLife application is 19.5%, and the avg. usage is 9.7% .This result shows that the CPU usage will not be a problem for the SeeLife project.

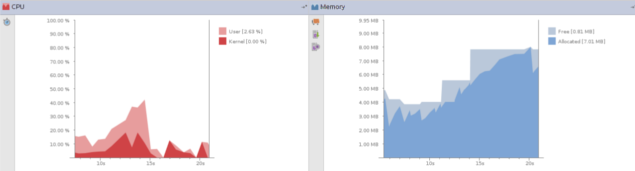


Figure 19. An example of CPU usage testing in Android Studio.[2]

**5. Project Management**

Erika Woodrum is the representative of the FES center, Cleveland in VA center. She is also the project manager and consultant. Technical advisor Dr. Michael Fu also gives instruction on research orientation. We have weekly meetings with them to show our progress in a week.

However, since we have a relatively small team, and everyone has his tasks in different fields, our team does not have a project leader in fact. We took the agile project management technique to regulate the project. We split the project into different parts and each group member takes the part that he is willing to do.

Zhonghao Zhan has a great experience on hardware protocol development and testing process. He took the part on the alternative plan of BLE connection, the OTG serial connection with Android device, and the TCP socket. On the other hand, Zhonghao also took the responsibility of designing the testing circuit (analog & digital) & analyzing the datasets. After finishing the design and demonstration of BLE connection of the SeeLife project, Zhonghao makes the scientific poster for the future Showcase.

Jiakai Chen has taken the game development course based on the Unity engine. He took the part on improving the game mechanics, debugging in the Unity, finding the way to solve the handshaking problems. He also worked with Zhonghao on making the application to receive the data with OTG serial connection.

Gyurae Kim has done many Bluetooth connection projects before. He took the part on choosing and programming the suitable Bluetooth hardware that meet the specification. He also was working with Zhonghao Zhan on the alternative design option that uses Android plugin on Unity to receive heart rate data from the sensor and send it to Unity while Jiakai Chen was working on solving the handshaking problem on BLE connection.

**6. Design Implications**

**6-1.** **Manufacturing cost (including both serial and BLE circuits)**

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Manufacturer | Purpose | Price |
| Adafruit Bluefruit nRF52 Feather | adafruit | Data Transmission | $24.95 |
| Gravity: Heart Rate Monitor Sensor for Arduino | DFRobot | Heart rate monitor | $16.00 |
| Easy pulse sensor V1.1 | Embedded Lab | Heart rate monitor | $20.05 |
| Arduino Uno Rev3 | ARDUINO | Data processing unit | $24.95 |
| Intel Arduino 101 | adafruit | Data processing unit | $39.95 |
| Lithium Ion Polymer Battery - 3.7v 350mAh | adafruit | Battery | $6.95 |
| Total | - | - | $132.85 |

**6-2. Sustainability**

The sustainability of the prototype mainly depends on the longevity of the heart rate sensor and the microcontroller chip. Since this prototype is the first working version that still requires major improvements and testings, we expect its longevity to be shorter than the final product. It would be easier and wiser to replace any broken parts, rather than trying to fix them. The broken parts can be recycled just like other electronic parts.

**6-3. Safety concerns**

Children need to wear a small heart rate sensor to their wrists. The heart rate sensor and the BLE chip are not radioactive and totally safe. The only concern is the choking hazard. Because the heart rate sensor circuit contains small parts, it should be kept out of reach of children under 3, or use under the guidance of doctors and parents.

**6-4. Commercial and societal impact**

Children age 6-8 have a difficult time calming themselves down in stressful situations. Anxiety often leads to poor self-esteem, difficulty in making friends, and trouble in school.By implementing the engaging game design into biofeedback therapy in the form of a mobile app based serious game, a six-year-old child’s attention may be captivated for the required periods of time to successfully teach autonomic self-relaxation.(Woodrum, Erika. 9). The prototype design will be open-sourced because the prototype we built still need improvements. It still requires a game level map and a better control throughout the game. We plan to make it open-sourced so that other people can keep improving the game.

**7. Conclusions — Discussion and Suggestions**

Overall, the most important project objective is completed. We transfer the game from keyboard-control to gyro-control and transplant it to the Android device. We can receive the heart rate data from BLE connection with Android device in a very short delay. However, there are some missed objectives. We did not have enough time on making the assessment chart at the end of the each level. The game still needs many improvements, but since we had met the required objective, the project work will not be continued. There will have some other engineering students work on revising and improving the game. The next step of game development should:

1. add the setting feature in the game. The setting can show the BLE chips available around and show their signal strength.
2. add the assessment chart. The chart is important for doctors and parents to monitor and summarize the condition of their kids.
3. embellish the art design by adding animes. The anime can stimulate the children to play.
4. advanced game level design and game experience evaluation.

**Reference**

1. “SeeLife Production Book” Woodrum, Erika (2014).
2. linfeng (08162016) DFRobot\_Heartrate\_Digital\_Mode (Version 1.1) [Arduino Source Code]. Available at https://github.com/DFRobot/DFRobot\_Heartrate/blob/master/examples/DFRobot\_Heartrate\_Digital\_Mode/DFRobot\_Heartrate\_Digital\_Mode.ino
3. adafruit customhrm [Adafruit nRF52 Source Code] Available at https://learn.adafruit.com/bluefruit-nrf52-feather-learning-guide/custom-hrm
4. "GATT Specifications" Bluetooth Sig. Inc
5. K, Melissa. “EKG Interpretation.” WWW.RN.ORG, Aug. 2017. Web.
6. "Product Specification." Nordic Semiconductor Infocenter, 03 Feb. 2017. Web.
7. “feather nRF52 pinout diagram.” adafruit industry, 20 Apr. 2017. Web.
8. [1]"PC-based heart rate monitor using Arduino and Easy Pulse sensor." Embedded Lab. N.p., 15 Nov. 2016. Web. 28 July 2017.
9. [2]"How to show current CPU load with Android Device Monitor?" Stack Overflow. N.p., n.d. Web. 28 July 2017.

**Appendices**

**Documents**

1. SeeLife Senior Project Documentation

2. nRF52832 Product Specification v1.3

3. feather nRF52 pinout diagram

4. feather nRF52 circuit schematic

**Workflow**

1. Get the Gyroscope working and export the project from Windows(Unity Engine) to Android devices(Tablet or phone).

Since the player character can only change the angle of where he is moving to(which is only the x and z direction). We did not use the gyroscope. Gyroscope is 3D sensor that might cause some bugs, if we just need to monitor 2D instead. We finally choose to use accelerometer. It is less complicated and easy to set which direction we want to monitor.

1. Then we move to the most important part of the project: BLE connection with Unity.

First, we ask for someone had done before for help. Dr. Michael Fu suggested us to ask Brian, who has the experience on BLE connection in Unity and Android Studio. We contacted with Brian Li. We made several online meetings with him. He taught us how to program in Android Studio and some tricks on monitor the data that we want to know. We really appreciate his helps on this project.

1. However, after we did what Brian suggested us to program, there were unexpected bugs that kept crushing the game. The Android Studio monitor kept sending the message:“Fatal signal 11 (SIGSEGV), code 1, fault addr 0xe59fc000 in tid 14007”. It looked like the game is trying to connect with a IP address online but it cannot, so it returned the error. We searched on the Internet, but the error message is kind of unique, so we cannot get many helps on someone had experienced this error. Brian and we did not know why the error came out, so we stuck for a long time that trying to fixed the bug.
2. In the first week, we were trying to solve the problem on Android Studio, but the error showed in every version, even the original version of the game. We tried to use another way to get BLE connection done, because the Android Studio problem seemed not be easy. Since the game crushed when it was trying to open, we cannot get debug information on Android Studio debug monitor. We tried to use Unity Engine asset to achieve the BLE connection. However, the asset is not working, because it was written for a specific BLE device, not our Adafruit Bluefruit Feather nRF52. We investigate the code a little bit, then we abandoned this method, because we judged it will be harder than the previous one.
3. Prof. Fu asked us to do the back-up plan instead. We then worked on serial connection first. We first rebuilt the serial connection by using serial chip instead of BLE chip. Then, in that week, we successfully get the data from the BLE chip to our made Android App by serial connection, but we still cannot implement this feature into the game. We also investigate more on the BLE connection on Unity Asset. For the BLE part, we can search and detect the BLE chip address and show it on a Unity application, but we cannot see the data.
4. Then, Dr. Michael Fu suggested us using the TCP socket to get the data. None of us have experience on TCP connection. At the same time, Jiakai Chen decided he will work alone on finding the bug in Android Studio or making the Unity asset work. Zhonghao Zhan and Gyurae Kim would work together on studying the TCP socket in a quick and tried to solve the data receiving problem by using TCP. (Later, Zhonghao Zhan and Gyurae found a way to show the data sent by another Android Studio on the game’s screen. However, at the same time, Jiakai Chen debugged the Android Studio and the BLE connection accomplished, so they abandoned the TCP socket alternative design to devote themselves on testing the BLE connection.)
5. We then tried to redo the process on the Android Studio methods by methods to try to find the error. This time, we can run the App in Android Studio. We find that it is not our device or code has some problems, but the Android Studio itself. After we changed the API level to 24, the whole application is working successfully. However, we still cannot get the data from BLE.
6. Then, in another meeting with Brian, he noticed that our data property, which sending from the chip, is NOTIFY. NOTIFY means we can only see the data, but the data is not readable. We then fixed the hardware code, changing the property from NOTIFY to NOTIFY and READ. And we now can get the data.
7. We then use the code from Unity to set a variable to store the data from Android Studio. This part worked great, without any unexpected errors. We now can get the dummy data from the BLE chip. Next step is to try the real data.
8. We tried the real heart rate data, and the App works fluently without any bugs. Now we can say the BLE connection and handshake between Unity and Android Studio problem were solved. The BLE connection is achieved. We still need to improve the gaming experience, fixing some bugs, but after the game can get the data, we can solve the software problem easily.