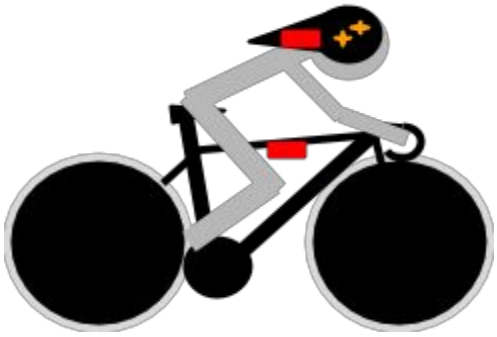
Head For Success



Client: Joel Bach

Team Members:
Brandon Parrish
Isaac Valdez
Kyle Dymowski
Alex Patel

CSCI 370
Summer 2015

The Client

Our client is Dr. Joel Bach, an Associate Professor at the Colorado School of Mines in the Mechanical engineering department. Dr. Bach teaches courses related to Biomechanics, Prosthetics and Implants engineering, biomedical instruments, Dynamics, and more. In addition to teaching, Dr. Bach is also the Director of the Center for Biomechanics and Rehabilitation Research at CSM, collaborating with researchers at Assistive Technology Partners and the Orthopedic Biomechanics Lab at CU Denver. Some of Dr. Bach's projects in bioengineering include: instrumentation and sensor development, biomedical modeling, and injury prevention and repair.

In cooperation with Dr. Bach, our team consulted with 2 individuals who provided significant contribution to this project. Firstly, Dr Mounir Zok, Senior Sports Technologist at the United States Olympic Committee; Dr. Zok leads the development of innovative sports technology solutions for the US Olympic program and has experience in wearable technology and a background in human-centered design. Secondly, Coach Andy Sparks, Director of Track Programs at USA Cycling; Andy Sparks is an athlete turned coach with over ten years of experience whose innovative methods have helped his athletes earn 11 World Championship titles, 3 Olympic medals, 2 World records, and 20 National records.

Product Vision

Head for Success is an innovative solution designed to assist cyclists overcome aerodynamic drag during training and racing. Using a gyro and accelerometer to measure a cyclist's head position and movement, Head for Success is designed to notify the athlete in real-time when his or her head is out of the ideal position. The solution includes an Android application that communicates with Arduino microcomputers via Bluetooth using a Lightblue Bean microcontroller. The mobile application stores data for an athlete's head position, graphing the coordinates of the position over time, showing where an athlete's head moved out of position. The application also displays on an interactive chart the number of times an athlete's head moved of position during a workout. Using these tools, Head for Success aims to help athletes minimize the number of times their head move out of position and ultimately save crucial time that could be the difference between Gold and Silver.

Functional Requirements

1. Mobile application
    a. Display multiple users
    b. Set desired head position
    c. Store the number of times an athlete's head moved out of position
2. Arduino hardware
    a. Measure relative angle of the head
        i. Gyroscope in helmet
        ii. Accelerometer on bike
    b. Connect over wireless network

        i. RF communication between gyroscope and accelerometer
        ii. Bluetooth from central hub to Android device
    c. Connect to mobile application through Bluetooth
    d. Send haptic feedback to user

Non-functional Requirements

1. Arduino memory limits
    a. Flash: 32k bytes
    b. RAM: 2k bytes
2. Graph display for an athlete's activity history
    a. Using GraphView, a plotting library for Android
    b. Display the number of times an athlete's head moved out of position during each activity
3. Threshold of angle and acceleration measurements
    a. How far out of position an athlete needs to be to trigger the haptic driver
    b. How long an athlete needs to be out of position to trigger the haptic driver

System Architecture

The construction of Head For Success includes many working parts. Below is a diagram of the Head For Success architecture. Following the diagram is a description of each part and how it interacts with other parts in the system.

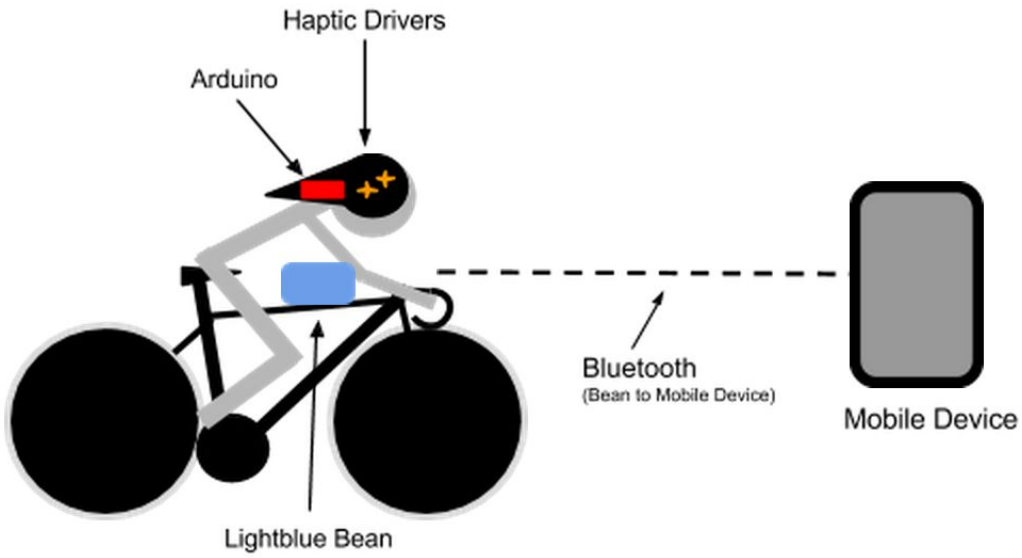


Figure 1. Architecture Diagram

Arduino and Haptic Drivers

The task of getting hardware components consisting of the haptic drivers working with the Arduino was taken on by Dr. Bach himself. The purpose of the haptic drivers is to provide feedback to the rider when their head moves out of the set position. They provide feedback by vibrating in distinction of where the athlete has moved out of position. So moving too far upwards creates a different vibration pattern than moving too far down.

Lightblue Bean

The Lightblue Bean is a microcontroller with several key pieces such as the accelerometer and Bluetooth communication. Bluetooth communication between the Bean and the mobile application is mainly the values taken from the accelerometer readings. The readings are taken in the form of x, y, and z coordinate axis points. Using the readings, the mobile application can set an athlete's desired position as well as record an activity's position date with respect to time.

Bluetooth communication

Our project uses Bluetooth Low Energy (BLE) to send and receive data packets to and from the Lightblue Bean and the mobile application. BLE is very useful in mobile applications for its low energy use, saving the mobile device's battery power.

SQLite database

A SQLite database is used to store athlete data. Although there are many database management systems, SQLite is required to store locally to an android device.

Mobile Device

The mobile device has an architecture of its own. At a very high-level, the mobile application communicates with the Bean to send and receive data. Below is a description of each view and how each view transitions to another. The first view is a splash screen with the team logo and the name of our project. The splash screen lasts for a few seconds and then transitions into a listview of all the athletes that are locally stored on the device.

Figure 2. Splash screen

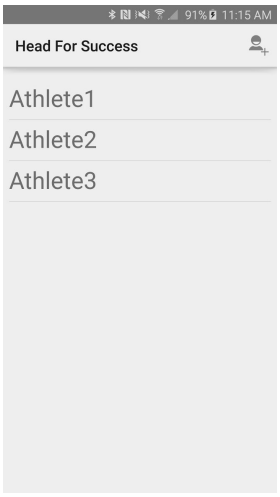


Figure 3. List view

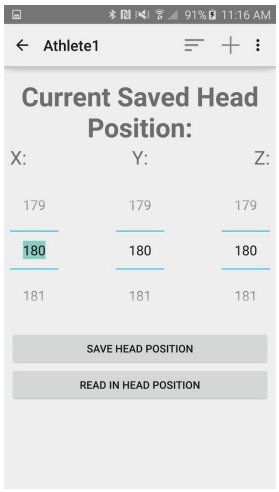


Figure 4. Set head position view

This page also allows you to add a new athlete into the database. When you select an athlete, you are taken to a view containing the athlete's currently set position, buttons allowing you to set a new position, and a menu bar that has several options.

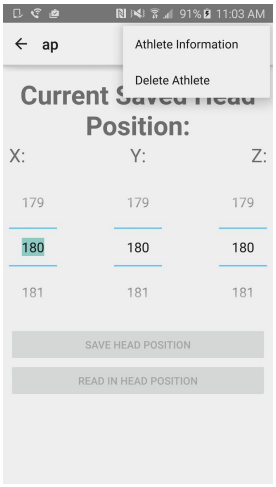


Figure 5. Edit/Delete settings

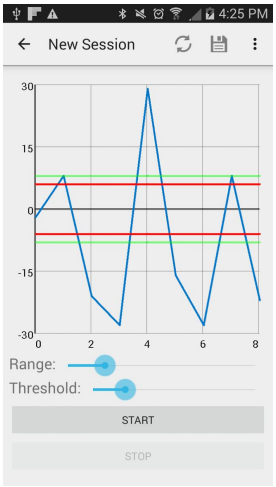


Figure 6. New activity view

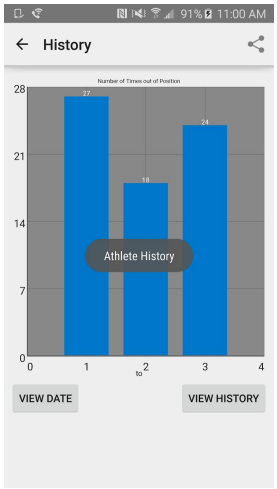


Figure 7. Activity history view

The first option is to select the far right item, which allows you to edit or delete the current athlete's name to/from the database. The middle item allows you to start a new activity, where you set the range of feedback and threshold for how far out of position you can go before losing feedback using slider bars, as well as set the amount of time out of position you can go before triggering feedback. The third and final menu item takes you to a view of your activity history. Every time you complete an activity, a new

entry is added to the history graph. You can choose a specific date or view your entire history. Selecting the icon on the top right of the menu bar exports the graph, converts it to a bitmap and saves locally on your device.

Technical Design

Figure 8. Sarah Hammer, Dotsie Bausch, and Jennie Reed. Team USA, Women's Track Cycling Team Pursuit, USA vs GBR. London 2012 Olympics.

GraphView

The activity history and session activity views in the mobile application utilize an open source graph plotting library called GraphView. GraphView works well with Android to programmatically create flexible, user-friendly graphs. It is as simple as creating a series of data points and then adding the series to the graph. Formatting is easily done by setting parameters such as axis labels, text, text size, colors, and more. GraphView

also works well with built-in Android components. The session activity view uses slider bars to adjust the range and threshold limits. In Android the slider bars are called seekbars, a good example of a seekbar is in Figure 5, on the new activity view. Below is a fragment of code taken from the range seekbar that checks to see if the current value of the range seek bar is greater than the value of the threshold seekbar.

```
public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {

  if(progress > thresholdSeekBar.getProgress())
     thresholdSeekBar.setProgress(progress);
```

The reason for this is because the threshold seekbar value must be equal to or larger than the range seekbar. If the range seekbar value is larger than the threshold value, then the threshold value is set to be the same as the range value.

Bluetooth Communication

Bluetooth connection uses GATT protocol to group attributes into a service. The service provided by the Lightblue Bean and the open source APK from Little Robots gives access to all of the functions that are inherited by the hardware of the LightBlue Bean, i.g., temperature readings, accelerometer readings, and access to the LED's on the device.

The structure of the stream of data is defined by these attributes. None of which fit our needs to send the values to the Android device.

Sending the data from the LightBlue Bean uses Arduino function Serial.print(); this sends an ASCII encoded string containing a max of twenty-three bytes with a sixteen bit handle. The handle is what defines the service and how the packets are processed into useable data. Our team created custom handlers that are read in when the message is received on the the Android device, the handle defines that our custom service is requested and the String is parsed by a delimiter. The data is then used accordingly throughout the program.

Design and Implementation Decisions

- Android Studio, after multiple attempts of trying to develop a cross platform mobile application, we decided instead to focus development on Android for its ease of use and support using Android Studio. This allowed us to write the background code in java, an object-oriented language with useful containers and libraries. Android Studio also allows the ability to test applications on nearly every Android API level. Other options we pursued include Kivy, a cross-platform IDE using python, and Xamarin, cross-platform IDE using C#
- Used SQLite as the database as this is the database that Android provides documentation on and is easier to implement than other databases. Also has cross platform code. SQLite is local database which is all that is currently needed for this application.
- Used a stock Android theme and app icons to follow Google's recommendations on app development and made the app look considerably nicer without extra hassle of creating a custom theme and app icons.

Results

Goals

Create an interface that allows a coach to monitor the head position of an athlete during training or a race, as well as provide feedback to the rider to let them know when they are out of the ideal aerodynamic position. The said interface should include a mobile application that allows an athlete to add his/her name to a database, set the athlete's head position, set the distance needed to travel to trigger the haptic device, and display the number of times an athlete's head moves out of position during a given activity.

Accomplished

The interface we created has a splash page displaying the team's logo, adds athletes to a SQLite database, set a head position with an X, Y, and Z axis using the Bean or manually using number pickers, display a live graph of an athlete's head position, and display an athlete's activity history with a chart of each activity and the number of times the athlete went out of position for each activity.

Lessons Learned

One of the biggest lessons our team learned was not to take too much on at once because things are always harder than they seem. In the beginning stages of our project, not much was known about the software-hardware interface. Feeling overly ambitious, the team decided to take on all aspects of the project and develop the mobile app as well as program the Arduino boards. It was assumed that because a few members had some prior mobile and hardware experience, that the learning curve would not be too steep, we were wrong. The hardware proved to be much more challenging than we had originally thought and required us to make some major implementation changes. After weeks of research with very little results, we came to the realization that maybe we had bitten off more than we could chew and it became overwhelming.

Another lesson the team learned was to ask ourselves, not only if something was possible but also how feasible it is. Our team originally wanted to develop cross platform for iOS and Android because it would make code management much easier. After some research, the team decided to use Kivy, a python based IDE, and began reading how to use it. Kivy, at the time, was still in alpha stages and did not work as expected. Xamarin was the next option but this also had it's shortcomings and after days/weeks of development the team discovered it would not be able to fit our needs. As a team we decided to start doing more research into how feasible something would be instead of just assuming "It worked for them, it should work for us". Doing so allowed us to

discover that although crossplatform is possible, it's not recommended and should only be used to develop native apps. Asking how feasible a bluetooth interface would be lead the team to discover that just because a component supports bluetooth does not mean it will work with other bluetooth devices and some bluetooth devices are more user friendly and easier to program than others.

Additional Development Information

HeadPositionActivity.java (Saving the Head Position)

One of the last component that needs to be added is the light blue bean communication with the gyroscope. When reading in the three integers for the head position in HeadPositionActivity.java method retrieveXYZ(), the light blue bean needs to communicate with the gyroscope to produce numbers that represent the actual head position. As of now the function reads in the three values corresponding the accelerometer in the light blue bean.

SessionActivity.java (Reading in Live Data)

As of now, when you press the start button to begin the session and read in values from the light blue bean, the bean sends the "X" value of the accelerometer inside the bean. This is then loaded onto the graph seen in the activity. The value read to the graph can be changed by changing the code inside bluetoothRunnable method called run(). Comments are provided in code to explain further.

The SeekBars that determine the range and threshold have constant max values (SEEKBAR_MAX) that can be changed at the top to better suit incoming data.

Time between each data point also has a constant at the top called TIME_BETWEEN_POINTS.

Lightblue Bean

The software utilizes an open source APK to connect the Lightblue Bean. The APK read in the packages of information using a handler to define the what package was received.

Custom handlers were created to define what information was being sent between the LightBlue Bean and the Android device. An exclamation mark (“!”) sent from the Android device tells the Arduino software on the Bean to send the current values for X, Y, Z axis back to the device (right now it is returning the accelerometer readings). An at sign followed by a defined set on X, Y, Z coordinates (“@,” + “Integer.toString(x)” + …) will be received by the Bean and stored int to an integer array called arrXYZ. This may be used to set new head position values.

Bluetooth

The app currently only handles one Lightblue Bean at a time. A system needs to be put into place to connect different Lightblue Beans to each rider.

Our team assumed that the data sent between the Arduino and the Android device is an X,Y,Z axes reading. The functions that handle the communication are in the Bean.java class, and are easily modifiable.

The head position set in HeadPositionActivity.java is passed to the Arduino when the start button is pushed in SessionActivity.java it currently only passes in the X,Y,Z axes and needs to send the threshold and seconds before the haptic device is activated. Another option is to send a signal to the Arduino telling the haptic feedback to activate. This would require a function to handle the amount of time out of position and test to see if the position is in between the threshold and activation button. Currently more information is needed to make this decision.

The library for the unofficial APK for Android contains The MIT Licence Agreement please read it.

Arduino

The code for the Arduino runs a loop that listens for data. When a message is received command is the first piece of information found. This is done by checking if the string contains the command character either returning the position of the character or negative one. The string is then passed to the parsing function that removes the information and stores it into an array. The array must be extended to for in information that is mentioned in the Bluetooth section above.

Resources

Information on GATT protocol -

https://developer.bluetooth.org/TechnologyOverview/Pages/GATT.aspx

Example of Arduino code using Lightblue Bean Library -

https://punchthrough.com/bean/examples/

The unofficial APK for using on Android -

https://bitbucket.org/littlerobots/beanlib

Android Development Tutorial (Would recommend understanding most of Getting Started)

http://developer.android.com/training/index.html