liftVectr Design Prototype

Praveen Anbu, Micah Bowonthamachakr, Antoine Ferguson, Cole Fisher, Timothy Krause

EFFORT

A comprehensive log of team contributions (contrib_log.txt), research ("Research" Folder), build and run instructions (README.md), and the full codebase is included in the submission through a Github Repo link: https://github.com/micahbow/liftVectr. A percentage breakdown for individual team members' work as well as their overall contribution to the project is shown in *Table A.*

TEAM MEMBER INDIVIDUAL PERCENT CONTRIBUTIONS					
Contribution	Praveen Anbu	Micah Bowonthamachakr	Antoine Ferguson	Cole Fisher	Timothy Krause
Research	61%	48%	60%	63%	65%
Interfaces	11%	20%	15%	21%	24%
Systems	28%	32%	25%	16%	11%
Margin Total	100%	100%	100%	100%	100%

TEAM MEMBER CONTRIB_LOG CONTRIBUTIONS						
Contribution	Praveen Anbu	Micah Bowonthamachakr	Antoine Ferguson	Cole Fisher	Timothy Krause	Margin Total
Research	19%	20%	21%	21%	19%	100%
Interfaces	11%	31%	16%	21%	21%	100%
Systems	21%	37%	21%	13%	8%	100%

Table A: Individual and overall team member contributions.

THEORETICAL BACKGROUND AND PRIOR ART

Weight lifting is often thought of as simply a measure of strength and the way to improve one's lifts is to lift heavier weights for higher repetitions. This however, is an incomplete view of the sport, in that it's becoming increasingly appealing to use velocity-based training (VBT) to measure and improve one's workout intensity [1]. Using IMUs, it's possible to use measurements of velocity and displacement to measure things like concentric and propulsive velocity during a lift which can be used to track and reap the full benefits of VBT [2]. Fully, these measurements can be used with Newton's Second Law and force profiles in order to increase strength and endurance, stamina, and exercise safety amongst other benefits of VBT [2]. BarSense is one existing app that attempts to track bar path and velocity for weight lifters using a video recording from the user's smartphone to track these measurements, but provides less accurate results than what can potentially be obtained from IMU readings coming from a wrist strap like the liftVectr Watch [3].

COMPONENT ARCHITECTURE

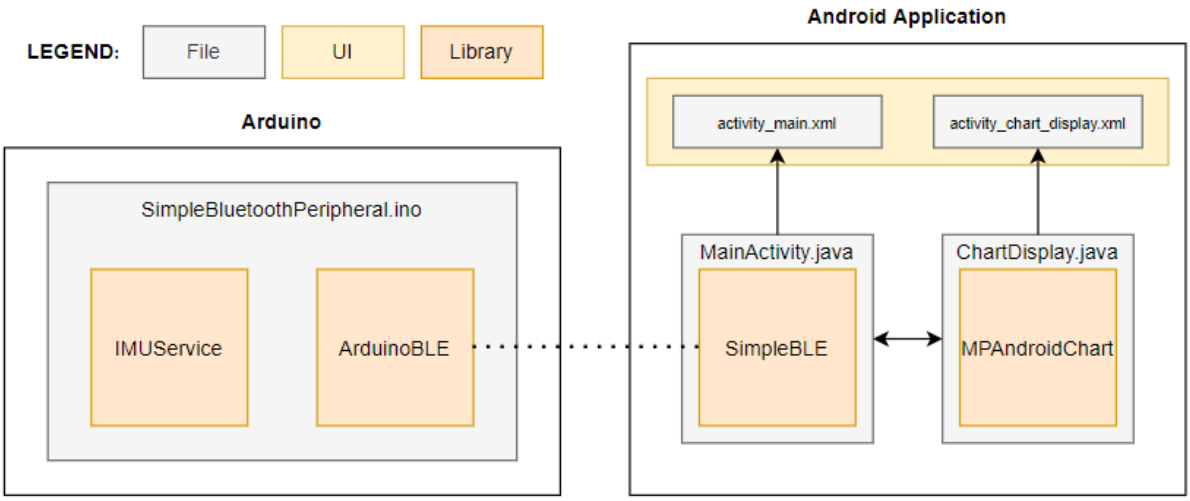

Figure 1.1: First Semester Component Architecture.

Our current prototype has a working Arduino driver that utilizes the Arduino IMUService and ArduinoBLE libraries to send live 9-axis IMU data to the Android Application. The application connects to the device and receives the data using Android SimpleBLE. Anytime data needs to be plotted, the ChartDisplay activity can plot the data utilizing the MPAndroidChart library.

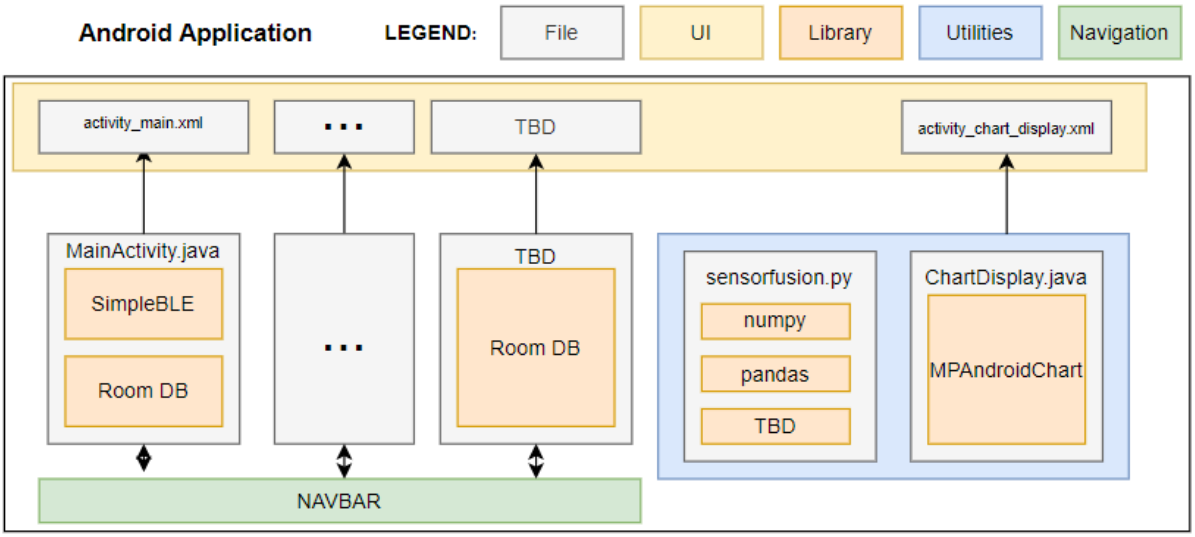

Figure 1.2: Second Semester Component Architecture Plan.

The Arduino architecture for the second semester will likely stay structurally similar to as it is now, experiencing minor optimizations. The Android Application, however, will get a navigation bar and several new activities/app pages. Example pages could include a StartExercise page, UserProfile page, and Settings page. These pages will utilize Room Database, which hasn't been fully integrated into the first semester architecture. The app will utilize a python program to perform sensor fusion calculations on the IMUData, and this may be implemented internally or externally depending on further research.

COMMUNICATION MECHANISMS AND RESILIENCE

In order to transmit data from the Arduino Nano 33 BLE to our Android Studio application, we are using bluetooth for wireless transmission. We are using the ArduinoBLE module within the Arduino programmer in order to set up the broadcasting of the data. This is done by setting up the service and characteristic for the data, gathering data from IMU, and outputting data to the service's characteristic as soon as new data is gathered. The service and characteristics are identifiable, through the broadcast, through UUIDs. Within Android Studio, in order to read the data from the specified UUID, we are using the SimpleBLE Android package. With this package, we are able to identify the Arduino Nano device through its device name and call a read function using the same GATT service and characteristic UUIDs.

With the current prototype, some errors in bluetooth transmission are identified as malformed data fields, such as floats with multiple decimal points or no decimal points. In order to account for this, we have implemented field checking to only account for accurate data readings. In the next batch of planned features, we plan to implement a filter that will remove distinct outlier data that is gathered from the IMU for the final product.

PRESENTATION AND PERCEPTION

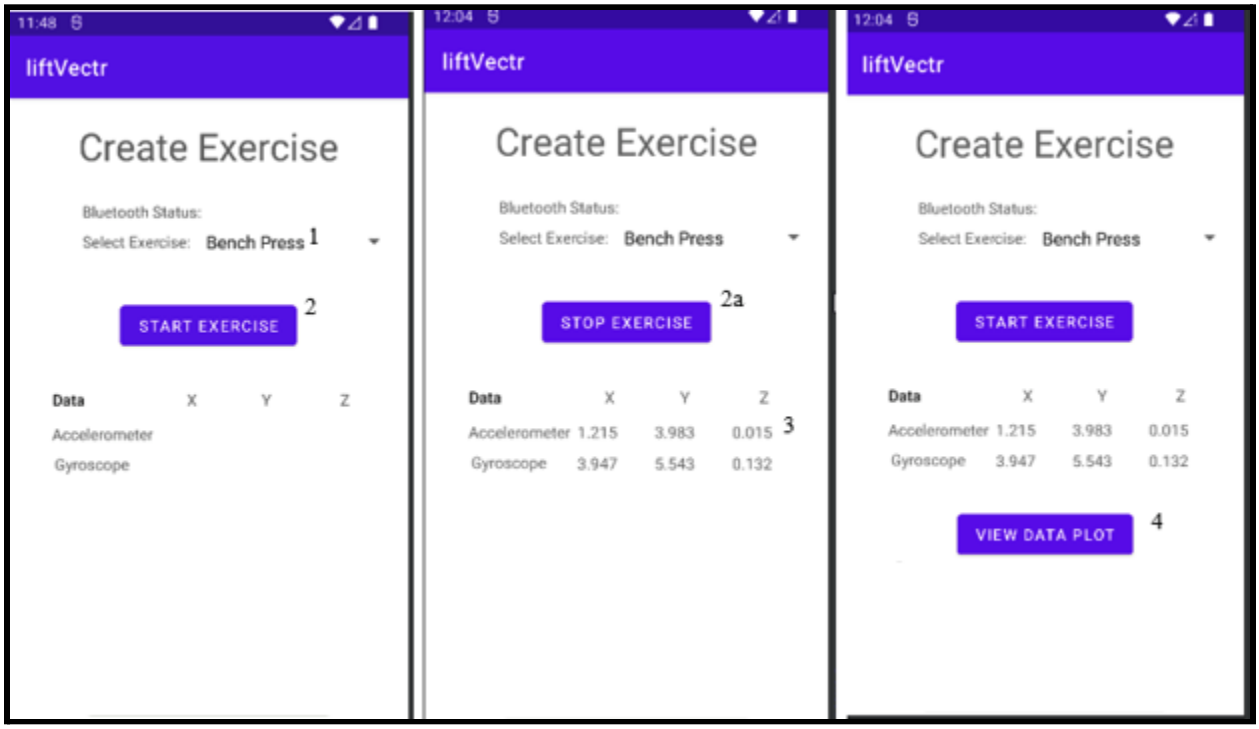

Figure 2.1: App UI and Key.

1. A drop-down menu choosing the exercise type will determine where the data points are stored within our database.

2. A toggle button will enable/disable data collection from the fitness watch. If "start exercise" text is present, data collection will begin when the button is clicked. A short countdown will begin and a sound clue will indicate when data collection has begun.

2a. If "stop exercise" text is present, data collection will stop when the button is clicked, and a data plot button will become visible. A short sound clue or screen flash will signify data collection has concluded.

3. Data currently being collected will be visible from the chart and values will be updated at a set frequency. The goal is for users to see some data points, instead of rapidly displaying every data point.

4. When clicked the data points will be graphed. View Figure 1.0 for an example graph.

Future interactions will include bluetooth buttons that handle bluetooth pairing with previously connected and newly paired devices. When pairing new devices, a dialog alert will indicate Bluetooth scanning has begun, then a new screen will display a list of available BLE devices. When connecting to known devices, a screen will display a list of previously connected BLE devices. Once a device is connected, the app will return to the initial screen.

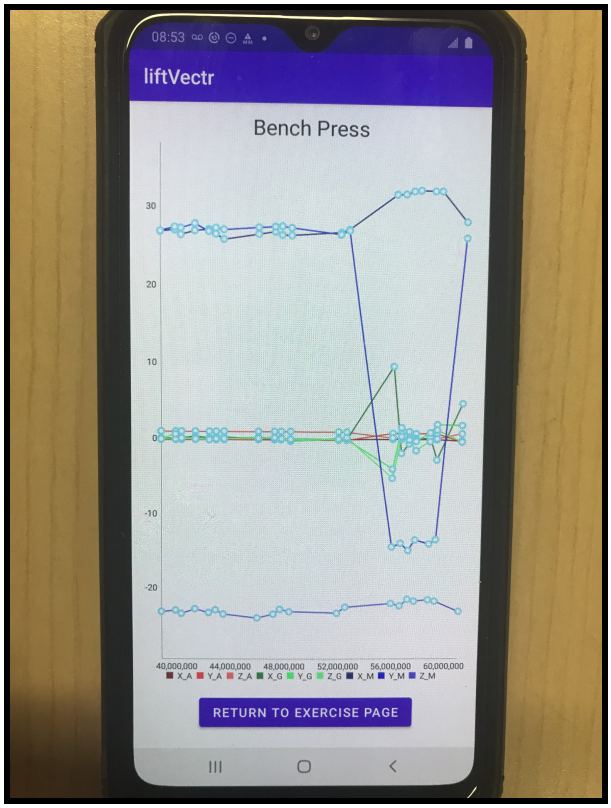

Figure 2.2: The graph displays the values (x, y, z coordinates) from the accelerometer, gyroscope, and magnetometer data using valid data points read from the fitness watch.

EMPIRICAL EVIDENCE

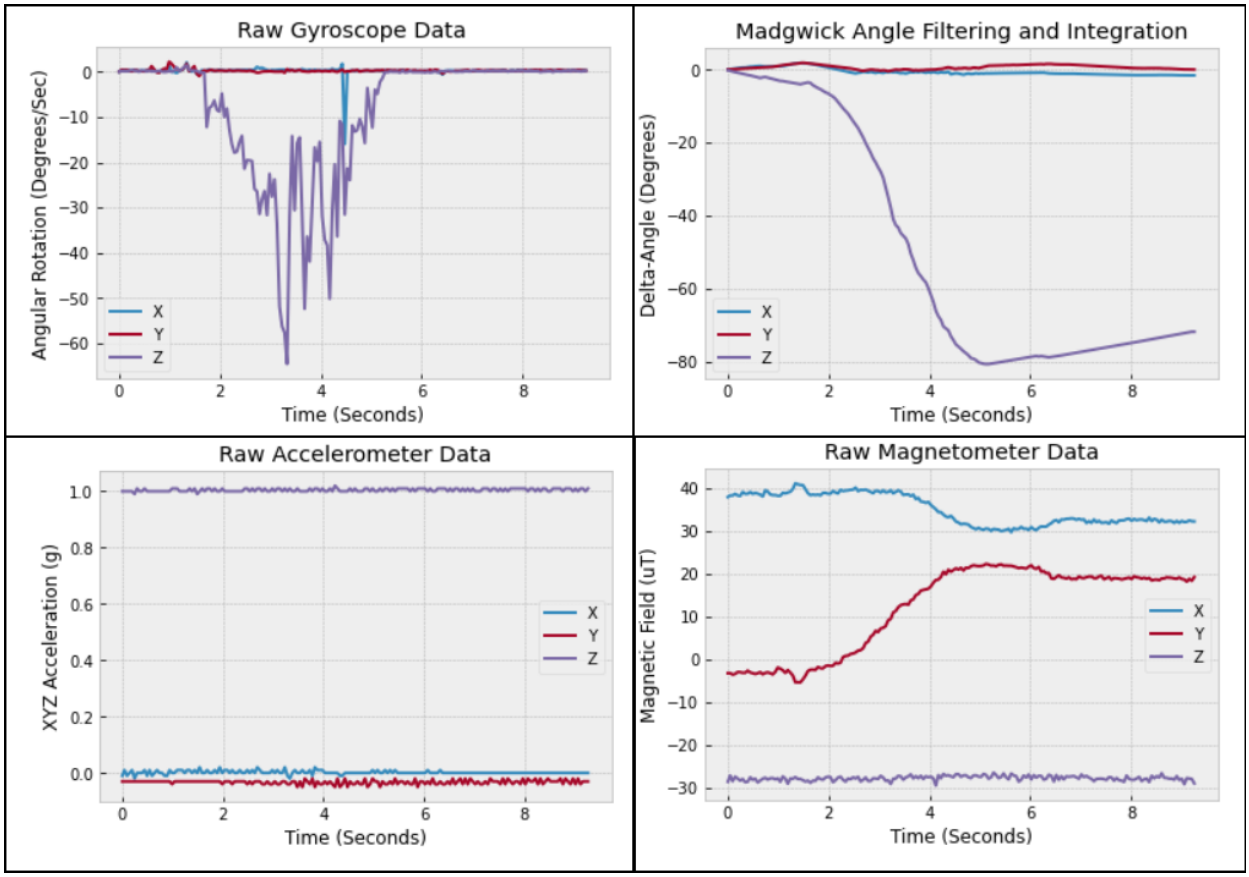

Figure 3.1: IMU-generated test data demonstrating a roughly 90 degree yaw with raw and processed results through Python.

Aside from visually validating the UI functionality of the Android App, much of the experimentation/testing involved was debugging the IMU data and Bluetooth. In order to verify the accuracy and functionality of the IMU, a basic CSV IMU-data logger program was written and the data then plotted in a JuPyter Notebook. One test case, shown in *Fig.3.1*, shows the data recorded when the Arduino board was rotated 90 degrees in the yaw axis. This data was also processed using the first part of the data processing algorithm [4] to be further implemented in the future.

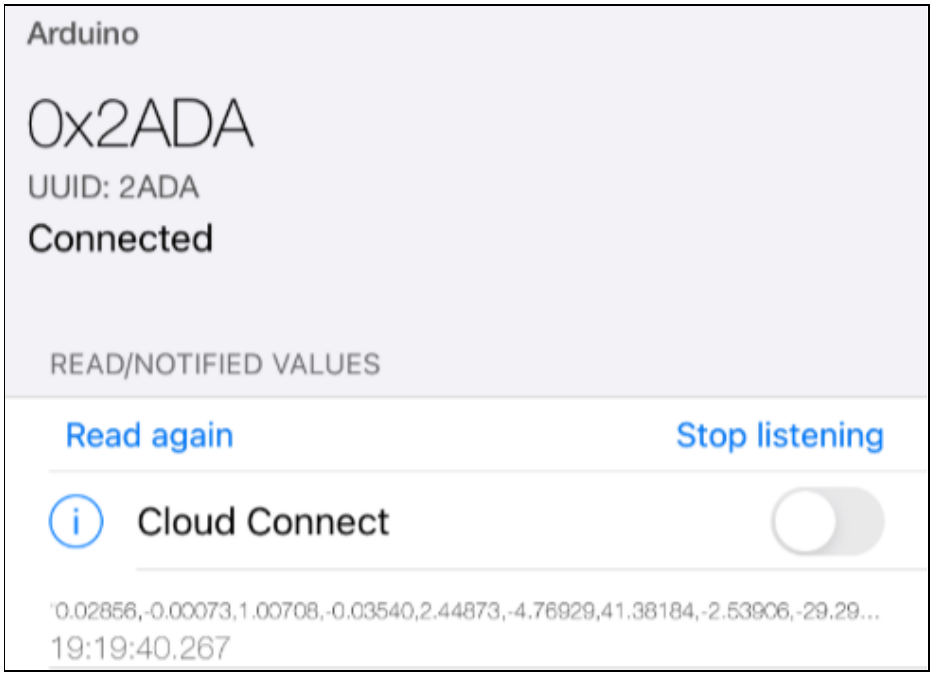

Figure 3.2: LightBlue Android App demonstrating successful read of Arduino IMU data characteristics as a CSV string.

To simplify debugging, testing the bluetooth transmission functionality of the hardware was first done using a third-party application, LightBlue, before testing it with the app, shown in *Fig. 3.2*. This ensured that the hardware side was bug-free. Finally, testing the Android App was the most intensive, requiring the debugging of code in the app to pair and read from the BLE device.

USABILITY

The interface and use-process of the hardware and Android app are predictable and consistent. The README documents the power-on and pairing process required to start the Arduino board for each use, and the app automatically initiates pairing and will not allow further functionality until the hardware has been connected. Because the IMU data is accurate and shown to be consistent with intended movement patterns, readings and error checking done by the App should also reflect this reliability. The watch's Universally-Unique-Identifier (UUID) is also coded in the firmware, assuring the app pairs with the correct device each time.

References

[1] T. L. Pro, “Velocity-based training for powerlifting,” *Testo Lab Pro®*, 18-Jan-2021. [Online]. Available: https://www.testolabpro.com/blogs/nutrition/velocity-based-training-for-powerlifting. [Accessed: 03-Apr-2022].

[2] L. Radenković, M. Milošević, and L. Nešić, “Physics of powerlifting: Bar force ... - iopscience.iop.org,” *IOPscience*, 16-Nov-2020. [Online]. Available: https://iopscience.iop.org/article/10.1088/1361-6404/abb4f0/meta. [Accessed: 03-Apr-2022].

[3] “Analyze and improve your lifts,” *BarSense*. [Online]. Available: http://www.barsense.com/. [Accessed: 03-Apr-2022].

[4] W. Buczko, “IMU Velocity and Displacement Measurements,” *GitHub,* Oct. 2nd, 2019.

[Online]. Available: https://github.com/Wojtek120/IMU-velocity

-and-displacement-measurements. [Accessed: April 4th, 2022].