Motion Tracking and Analysis of Basketball Free Throws using Inertial Motion Measurement

ECE4011 Senior Design Project

Free Throw Form Analytics
Dr. Erick Maxwell

Edison Carrick, ecarrick3@gatech.edu
Adam Jackson, ajackson76@gatech.edu
Kyle Kizirian, kkizirian3@gatech.edu
Patrick Miller, pmiller38@gatech.edu
Mickeal Taylor, mtaylor81@gatech.edu

Submitted

4/27/17

Table of Contents

Executive Summary

The Smart Sleeve is a device worn on the arm that tracks the user's motion and provides feedback on the consistency of their basketball free throw form. The system consists of attached inertial measurement units (IMUs) and a cloud based service. The system tracks the user's motion and the cloud service collects data and provides the user feedback on the variations in their form.

The device will allow for players to more precisely practice their shot, allowing for faster muscle memory development and reducing the amount a player shoots with bad form. By allowing the player to receive shot feedback without the need for a coach, he or she will be able to practice more conveniently. The Smart Sleeve technology is applicable to many sports beyond basketball, and could potentially redefine how athletes practice in the future.

Construction of the sleeve was successful. The sleeve was constructed using 4 IMUs placed around the wrist and the elbow. A Wi-Fi enabled microcontroller, battery, and I2C multiplexer were placed mid arm to balance and weight. The backend required writing of libraries for hardware integration which took up more time than anticipated, so hardware issues weren't realized until it was too late to order new components.

The frontend software created successfully displays the data from the database and gives feedback to the user through shot metrics. The modeling of positional data was impossible due to error induced by the IMUs and the low sampling rate of the microcontroller which limits the resolution of the data. With

improved components, the software in place will allow for proper positional data and improved possibilities for the application of the device.

Motion Tracking and Analysis of Basketball Free Throws using Inertial Motion Measurement

1. Introduction

The Smart Sleeve for Free Throw Shooting (SSFTS) is an affordable sports motion tracking system that provides feedback on the user's free throw shooting form. The system consists of a wearable motion tracking sleeve and a laptop.

1.1 Objective

The objective of the SSFTS was to create a motion tracking sleeve which will track data for a free throw shooter, and help them improve their form. The technology for this device can also be extrapolated to other sports with repeated motions.

1.2 Motivation

As a team consisting of basketball fans, we wanted to create a device which will improve the ability of the everyday basketball player's free throw by tracking the motion and providing feedback on their shot. This affordable product will be the only one in the market and will be used by amateur and professional players alike.

1.3 Background

There are currently no devices on the market that are wearable free throw tracking devices. The use of inertial measurement devices for this is therefore also not widespread. There has been some interest into the use of these devices for tracking other sports, but this has not hit the commercial market yet.

2. Project Description and Goals

The goal of the Smart Sleeve for Free Throw Shooting was to create a system that gives the user feedback on their basketball free throw shooting form. The system consists of a wearable sleeve with attached IMUs to capture data as the user shoots free throws. The microcontroller then transmits the data to the web for the user to view feedback on their shooting form at any time. The system is intended for basketball players wishing to improve their free throw shooting ability and will cost an estimated $400 each. The features of the system include

- Motion tracking of the user's arm movements
- Data transmission over Wi-Fi and cloud storage
- Analysis of the user's motion
- Graphical user interface for the user to view feedback on their free throw shot
- Minimal impedance on user's free throw shooting motion
- At least two hours of battery life
- Rechargeable battery via USB

3. Technical Specifications and Verification

Table 1 contains device specifications for analyzing free throw data collected during a practice session up to two hours in duration, which can be viewed immediately after completing any number of attempts. Table 2 shows a comparison between our proposed project ‘shalls’ and the resulting attribute of the final prototype.

Table 1. Device Specifications

Item	**Specification**
Battery Life	2 hour min.
Sampling Rate	30-60 Hz
Linear Acceleration Rate	±15 g
Linear Acceleration Accuracy	±1 mg
Roatational Velocity Range	±1500 dps
Rotational Velocity Accuracy	±100 mdps
Spatial Data Points	4 IMUs (along arm
Sensor Communication	I2C
Data Communication	Wi-Fi
Processing Time	10 sec max.
Interface	Web Application

Table 2 Comparison of Proposed Shalls and Resulting Prototype

Proposed Shall:	Prototype Results
be a wearable sleeve,	Sleeve is fully functional and wireless.
fit average young adult males,	Average flexed male bicep circumference is 13 inches [8], sleeve expands to 25 inch

	circumference maximum.
transmit data from the sleeve wirelessly,	Sleeve transmits data via Wi-Fi (802.11 b/g/n).
not restrict normal arm movement in any considerable way,	Range of motion tests passed from physical therapy checklist [9].
start and stop shot recording with the press of a button,	Prototype has a button for control.
be built with a wireless enabled microcontroller package,	Microcontroller has built-in Wi-Fi.
display an RGB LED that provides device status indication,	RGB LED found to be unnecessary, on-board LED used instead.
be powered using a rechargeable lithium ion battery,	Prototype uses an 800 mA hour lithium ion battery
operate for a minimum of 2 hours,	Prototype was tested to operate for least 3.5 hours.
operate from -20°C to 60°C,	All components rated -20°C to 60°C minimum operating rage (Battery charging minimum 0°C).
save form data locally for future comparison,	Software application saves shots into cloud database
allow users to select a primary comparison form.	Software application allows users to compare any two shots.

4. Design Approach and Details

4.1 Design Approach

There are two main components of this prototype system: the data collection system on a wearable sleeve, and the software package that processes and analyzes data for display to the user.

4.1.1 Hardware

The hardware system consists of the Adafruit Feather HUZZAH microcontroller, UnionFortune Polymer Lithium Ion 850 mAh battery PRT-00341, and four STMicroelectronics LSM9DM1 inertial measurement units that were proposed originally, as well as an I2C multiplexor, Adafruit TCA9548A. Early attempts to integrate the sensors revealed an I2C addressing limitation, which necessitated a multiplexor to use all four sensors. A pushbutton controls the function of the data collection system. A wiring diagram for the hardware system is shown in Figure 1. The diagram excludes the battery.

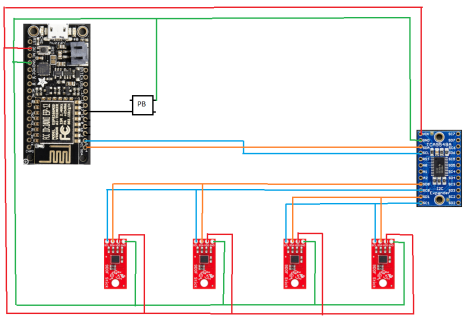


Figure 1 Wiring Diagram

4.1.2 Sleeve Construction

The sleeve was constructed using pockets made from a disassembled sleeve to stretch and accommodate the components without causing any discomfort to the user. The components were placed in locations which would not impede the user while providing maximum data for the user. The IMUs were placed on both sides of the wrist and the elbow to track the full motion of the shot. The microcontroller, I2C multiplexer, and battery were placed on the middle of the top of the forearm so the user does not feel the components and the motion remains unimpeded. In order to insure the sensors are always in the proper area, markings on the sleeve indicate the proper location of the wrist and elbow.

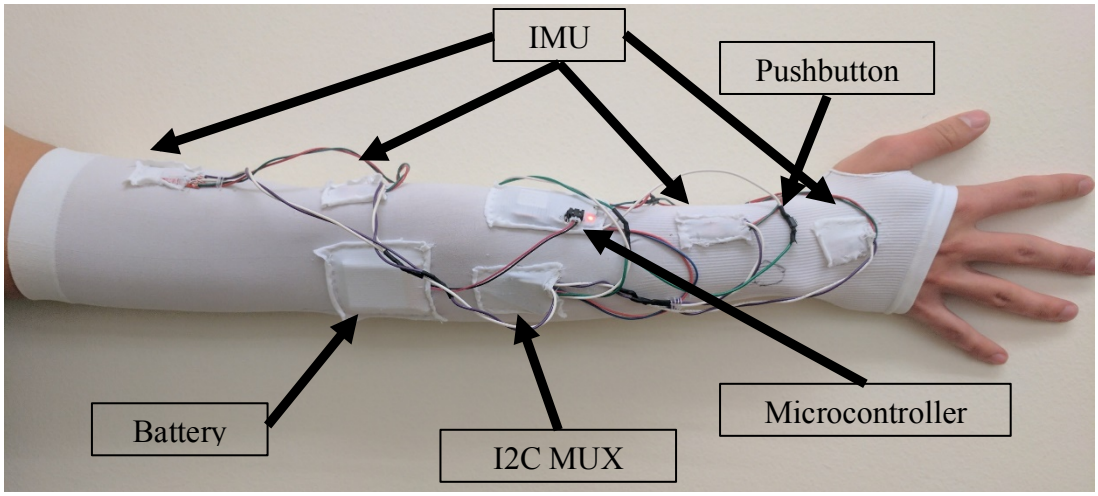


Figure 2 Sleeve Construction Layout

The wires were soldered and spliced together to minimize the amount of wires on the sleeve. Due to safety concerns the location of the battery was changed from the bottom of the arm to the top of the arm to reach

the port on the microcontroller without alteration to the battery wire. The wires were then sewn onto the sleeve to keep them from moving the IMUs and for aesthetic purposes. During the construction stage, the RGB LED was replaced with the onboard LED for ease of wiring, and user comfort.

4.1.3 Microcontroller Firmware

The Huzzah Feather ESP8266 is an open source firmware device with active community development on GitHub. Recent developments in the latest firmware introduced code breaking bugs to the HTTP module that could not be solved. In an effort to salvage firmware development our team had to build from source an older, stable version of the firmware.

The firmware update was necessary due to the memory management enhancements and increased stack space which was required to overcome our memory limitations. With control over our firmware build, we were able to optimize other parameters to fit the project's needs such as reducing timeouts in the TCP/IP stack, versioning our builds, and customizing display messages.

4.1.4 Sensor Library

Software for interacting with the sensors was written from scratch using the sensor datasheet, as open source code can only be found for Arduino implementations. These functions, found in Appendix D, initialize the sensors, select measurement range, and read data over I2C. Early iterations requested data from the sensors individually and required I2C operations for each of the six measurements (12 registers)

per sensor. Additionally, the microcontroller only supports a slower I2C protocol of 100 Kbps, making each I2C operation a time-consuming request. This lead to a major obstacle in data collection: sampling rate. Using the simplistic I2C functions resulted in a sampling rate of 4 Hz when all four sensors were read. Further use of the sensor documentation revealed a "burst read" functionality of the sensors in which a single I2C operation requests all measurements of a single sensor. The number of required I2C operations was reduced from 48 to 4, and the sampling rate was increased to 33 Hz.

4.1.5 Data Collection and Transmission

The measurements received from the sensors are stored locally on the microcontroller during recording and then transmitted after recording is finished. The Adafruit Feather HUZZAH's RAM is a major limitation at only 44 Kbit. Attempts to use RAM for temporary data storage proved infeasible. Additionally, early iterations attempted to convert and filter the values on-chip before transmission, which exacerbated the memory limitation. During the recording, data values are written to a text file in nonvolatile memory to be accessed for transmission. No conversion or filtering is done on the microcontroller before sending the data in order to maximize our sampling rate. The text file is formatted as a list of JSON compatible arrays that, when sent to Firebase, are automatically formatted and stored thanks to Firebase's built in API.

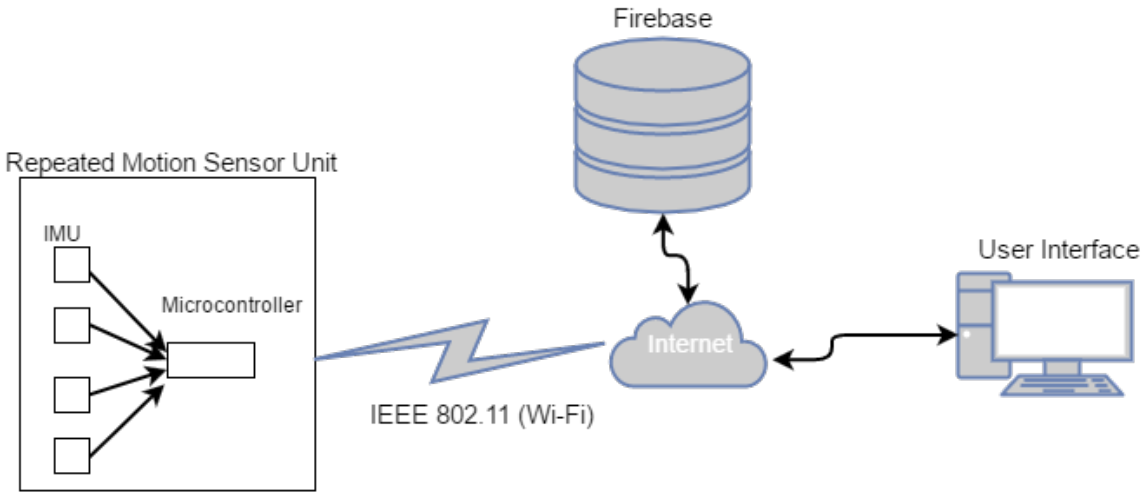


Figure 3. The System Level Overview Outlining the Flow of Data and Major Software Components

After the device has finished sampling the IMUs, the microcontroller reads the created text file line by line uploading the compatible arrays. By minimizing the amount of data per HTTP session, we were able to minimize the transfer and connection times per sample which maximized our throughput. Due to the functional design of the Lua language, the HTTP calls were asynchronous. We had to force the system to send the data in order to keep the samples in order through time, and to minimize our RAM limitations.

4.1.6 Data Analysis

The software package receives the data as strings of hex values and a timestamp for each sample. The data is first parsed and converted to the 24 measurements that make up each sample. The 24 sets of data are filtered using a low pass filter to remove sensor noise. It is assumed that the user takes a

moment at the beginning of recording to place the sleeve in a calibration position, with arm outstretched and parallel to the ground. The measured acceleration will be nearly zero for x and y axes, and 1g for z axes, as seen in figure 4. Non-zero gyroscope readings during this motionless moment, as seen in figure 5, are assumed to be the inherent bias of the measurement device, and are removed from subsequent readings. Nonzero x and y acceleration readings are used to calculate the initial orientation of each sensor, and a provided length from the shoulder is used to provide initial position.

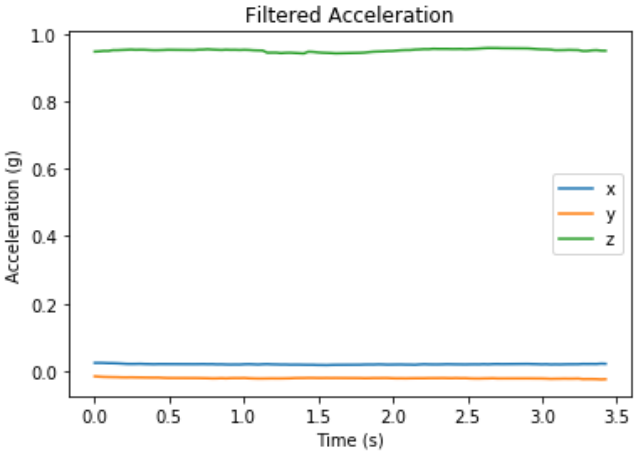


Figure 4. Acceleration data of motionless sensor

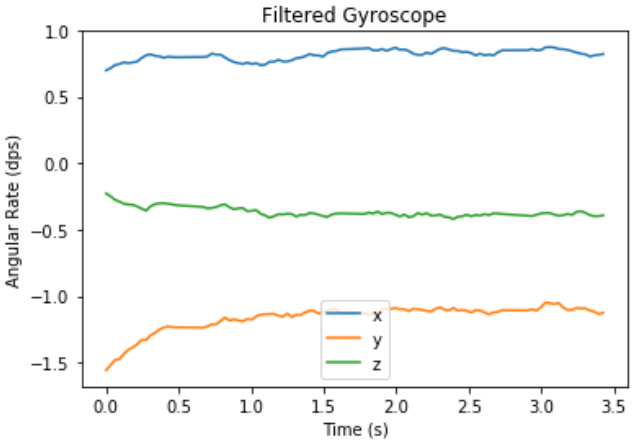


Figure 5. Gyroscope data of motionless sensor

Special Euclidean (SE) groups combine position and orientation in a matrix representation that allows for calculation of sensor configuration over time. Each 4x4 matrix is created from three rotational angles and three translational offsets. Multiplying one group by another applies the rotation and translation of the second group to the reference frame of the first group. Functions to create and use SE groups are included in Appendix D. The initial position and orientation calculated in the calibration

position are included in the first SE group. Euler discrete integration is used at each subsequent sample to calculate change in position and angle for each axis. These are included in a SE group "update" multiplied to the previous group to calculate the next. Based on the calculated orientation, gravity is subtracted from the acceleration measurements to obtain absolute linear acceleration.

The attempts to calculate position proved unsuccessful. Though the device met sampling rate specifications, the specified sampling rate was not high enough to produce data with resolution to accurately track sensor orientation. Errors in orientation compound as time goes on, and the actual orientation becomes unknown. The gravity correction step then produces more error than it corrects, and the position calculation is inaccurate, as seen in figure 6. Filtered acceleration and gyroscope data, as well as calculated linear velocity, are used in the prototype for user interface.

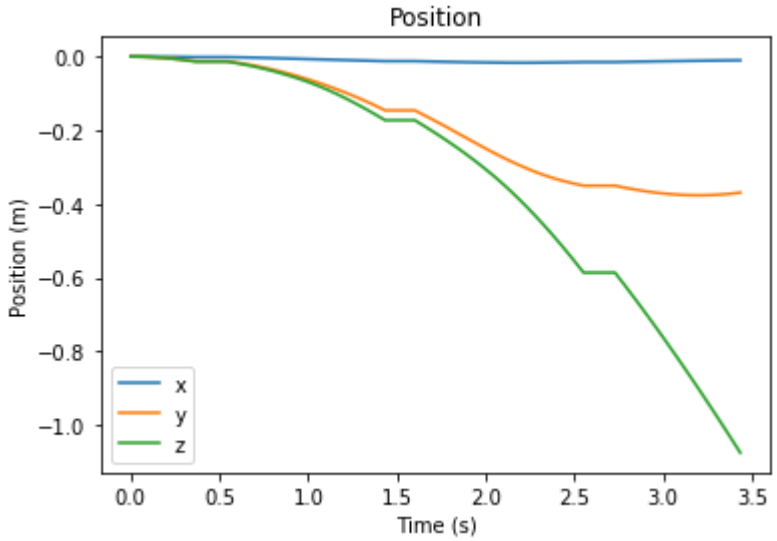


Figure 6. Positional output of motionless sensor data

4.1.7 User Interface

The software interface allows the end user to examine, compare, and save their shots for reference. The full code can be found in Appendix D. All data is stored in the cloud database making the software portable yet robust enough to perform the calculations required to produce the graphs. As shown in Figure 4 the software GUI allows the user to retrieve database data, select shots, save shots, and simulate the shot graphically.

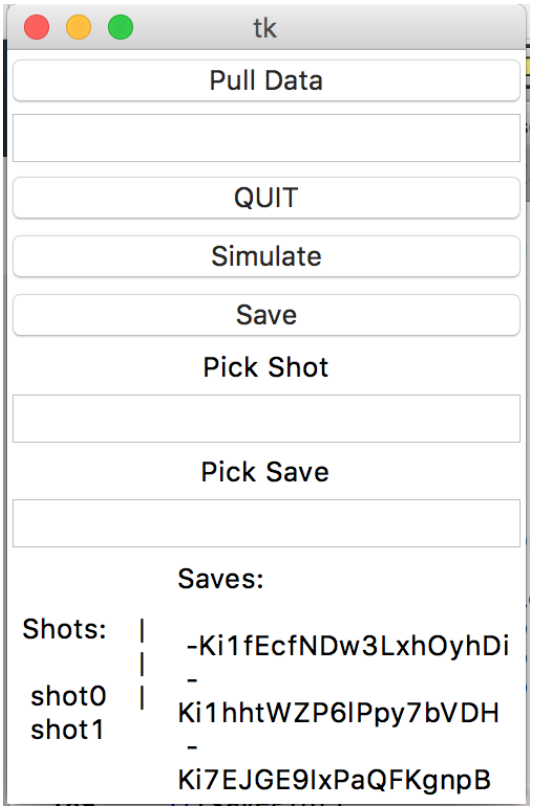


Figure 4. A screen capture of the Python software application GUI.

An example of the simulation is given in Figure 5. The right side of the display gives minimum and maximum parameters for various aspects of their shot compared to the reference shot if chosen. The left side of the figure replays the shot over time.

Although we were able to recreate a graphical representation of the shot, the previously mentioned challenges associated with position calculation mean the GUI cannot graph sensor position as we had originally hoped for.

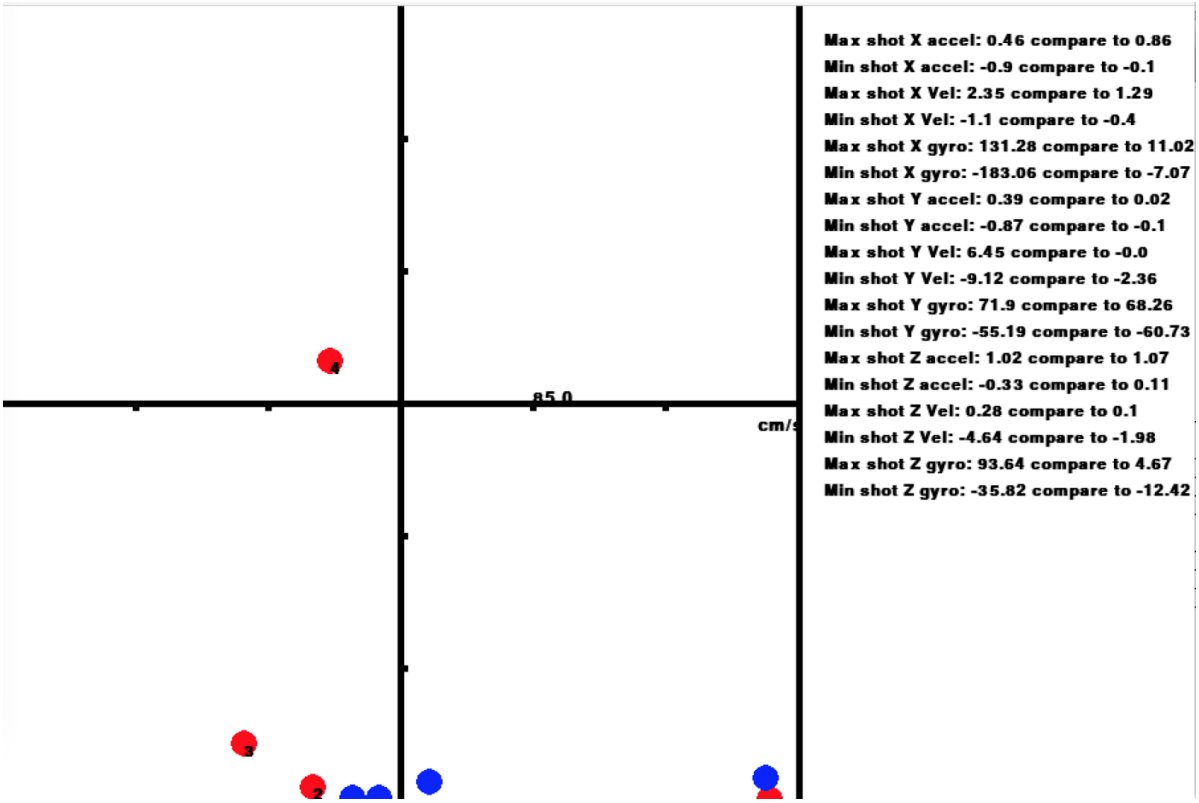


Figure 5. The simulation window showing the graphed points on the left and comparative shot metrics on the right.

4.2 Codes and Standards

Inter-Integrated Circuit (I2C) is the selected communication standard for the interface between sensors and the microcontroller. The hard-coded addresses of sensors posed a problem when on the same device. We had to purchase additional hardware to deal with the address conflict limitations.

Wi-Fi communication will use the IEEE 802.11n standard for wireless communication between the microcontroller and the web server. The impact of this standard is limited. It is well established and works well. The only downside to using 802.11 standards is the power consumption, but that was planned for.

UL 2054 provides codes for safe usage of wearable battery packs.

4.3. Constraints, Alternatives, and Tradeoffs

An alternative to Wi-Fi communication is Bluetooth communication. Bluetooth communication is cheaper and requires less power, but requires a local signal and additional hardware to receive and log the data. Wi-Fi can be used anywhere with a wireless internet connection and can store data using cloud storage. We chose to utilize Wi-Fi communication over Bluetooth for development simplicity. Bluetooth would have required our team to write a separate application to process the data. Future implementations may choose to incorporate this protocol instead due to the lack of Wi-Fi in many places and the prevalence of mobile phones with Bluetooth capability.

An alternative to external web server data processing is processing using a mobile app. This would

localize the processing to the user's own phone, but a phone has limited computing resources. For users with less powerful phones, this would lead to a diminished user experience. Similarly to the choices described above we aimed to complete our project as efficiently as possible; opting to use Python's powerful plotting and computation libraries.

One of the largest constraints, as described earlier in the paper, was memory and speed limitations of the microcontroller. We optimized for cost instead of functionality which resulted in a lack of useable data. Future projects may look to use more powerful hardware and to minimize costs at scale rather than up front in development.

The number of sensors being used along the arm comes with the tradeoff of added weight, power consumption, and cost. The number of sensors was kept low while still collecting useful data of overall arm motion. Selection of four sensors provided data points along the arm (including one on the hand) without excessive wiring or obstruction of motion.

5. Schedule, Tasks, and Milestones

Appendix A lists all the tasks over the duration of the project as well as risk level and owner, Appendix B contains the Gantt chart with legend, and Appendix C contains the list of all group members and their effective contributions to the project. The distribution of labor between all group members was fairly even and can be seen in both appendix A and C.

6. Final Project Demonstration

All desired specifications were met and demonstrated during final testing to the group's advisor. The following is a list of major specifications and the manner in which they were verified

- To demonstrate that the sleeve was fully wireless, the system was run with no wired connectivity and verified that data was sent correctly to the cloud and available through the application's user interface.

- To demonstrate that the sleeve did not restrict arm movement in any considerable way, users wearing the sleeve compared their range of motion to a Washington State DSHS guideline on acceptable range of motion and verified that the sleeve did not prevent the user from performing any normal arm movement.

- To demonstrate that the battery life lasted a minimum of two hours, the sleeve was fully charged and then run continuously without being shut off or plugged in for upwards of three hours to verify its longevity.

- To demonstrate that recording data was controlled by the press of a button, the database was monitored while the system was running and, once the button was pressed, it was verified that data began to be sent to the cloud database for storage.

- To demonstrate that the system achieved a sampling rate of at least 30 Hz, timestamps were

included with each sample from the microcontroller's system clock and measurements were taken to verify that all recording exceeded a rate of 30 Hz.

- To demonstrate that specific shot data could be stored for future comparison, the user interface was used to bring up a desired shot, save it, and then the entire system was turned off and turned back on to prove that the desired shot was stored using a designated area in the cloud storage database that was maintained even when the system was off.

- To demonstrate that the user interface allowed the user to select a primary shot form for comparison, the user interface was used to bring up two shot recordings at the same time for their data to be displayed comparing one another.

7. Marketing and Cost Analysis

7.1 Marketing Analysis

The target market consists of individuals wishing to improve their basketball free throw shot. With an estimated number of basketball participants over 25 million [4] there is a great demand for training equipment. With only 1% of high school varsity players able to play Division I college basketball [5], even at the high school level there is an immense pressure on players to gain an edge over the competition. There are a few systems which have attempted to use technology to assist free throws. The Pistons, a National Basketball Association (NBA) team, worked with STRIVR Labs to build a system utilizing

virtual reality to assist one of its players [6]. The virtual reality system was developed to ease the psychological pressure of free throws, not to assist with free throw mechanics. There are currently no systems that utilize inertial measurements to provide feedback during practice.

7.2 Cost Analysis

7.2.1 Parts and Materials

The hardware used in the development cost $106.20 [1] [2] [3] [7] as described in Table 2 The miscellaneous category is an estimation of the materials such as wiring, solder, etc. used through the development of one prototype. To mitigate catastrophic failure of any one part, we purchased enough prototype materials to build two sleeves. Having two sleeves allowed development teams to work in parallel on different aspects of the integrated system. The total hardware cost was $212.40.

Table 2. Prototype Hardware Costs

Component	**Cost**
IMU x4	$ 56.80
Microcontroller	$ 15.95
Battery Pack	$ 9.95
Shooting Sleeve	$ 8.50
Misc.	$ 15.00
Total	**$106.20**

7.2.2 Development Cost and Profits

There were five total engineers working on the project, and the total labor in hours was recorded in Table

3. With an assumed labor cost of $50 per hour and using the total hours from Table 3, the total labor cost was $4,550 per engineer. Assuming 30% fringe benefits of labor and 120% overhead on materials/labor/fringe benefits, the total development cost of the system is shown in Table 4.

Table 3. Estimated Development Hours per Engineer

Task	**Hours**
Hardware/Software Development	39
Team Meetings	13
Reports	10
Research	8
Presentation	5
Fabrication	6
Testing	10
Total	**91**

Table 4. Total Development Cost

Component	**Cost**
Parts	$ 207.48
Labor	$ 22,750.00
Fringe Benefits, % of Labor	$ 6,825.00
Subtotal	$ 29,782.48
Overhead, % of Material, Labor, Benefits	$ 35,490.00
Total	**$ 65,272.48**

The production run will consist 5000 units sold over a 5-year period at a price of $400 per sleeve. Assembly and test technicians will be employed at $15 per hour to build and finalize the products. Advertising will be 7% of the total input costs which is $21. Total revenue for first production run will be $2,000,000. Selling the sleeves for $400 per unit, a profit of $78 would be made, a 24% profit per unit as

outlined in Table 5. This is $390,250 of profit over the five-year period. The high cost per sleeve is due to the estimation of parts cost from the prototype. By using custom hardware breakouts as opposed to development breakout boards, the price per sleeve could be significantly reduced. While development costs could go up, the current amortized development cost per unit is small compared to the parts cost.

Table 5. Profit Per Unit Breakdown

Component	**Cost**
Parts	$ 98.74
Assembly	$ 15.00
Test	$ 10.00
Total Labor	$ 25.00
Fringe Benefits, % of Labor	$ 7.50
Sub Total	$ 131.24
Overhead, % of Material, Labor, Fringe	$ 157.49
Input Costs	$ 288.73
Sales Expense	$ 20.21
Amortized Development Costs	$ 13.01
Total, All Costs	$ 321.95
Selling Price	$ 400.00
Profit	**$ 78.05**

8. Conclusion

The Smart Sleeve capture acceleration and gyroscope data for the X, Y, and Z axis on all 4 IMUs at a rate of roughly 33 Hz. The system captures this data, sends it up to a cloud storage database, extracts and filters the data, and displays metrics for the user on how two free throw shots compare to one

another. For future iterations of the sleeve, it would be desirable to utilize acceleration and gyroscopic data to reconstruct position so that the user has a visual representation of their shot in comparison to another shot.

With the current system, low sampling rates and readings of acceleration due to gravity made reconstructing position from acceleration data infeasible. While the sleeve is moving, each of the IMUs is not able to distinguish whether the acceleration values being captured are due to motion or due to the effects of gravity. To remedy this, the sleeve goes through a quick calibration step in which it is held still momentarily, this allows the sleeve to capture its current orientation and gravity vector. After this, the sleeve utilizes gyroscopic data to determine how the orientation of each IMU has changed to determine in which direction the gravity vector is currently pointing and subtract gravity off from subsequent calculations. However, the sampling rate of the sleeve proved to be insufficient such that it was not able to keep track of the orientation of each IMU with enough accuracy to correctly remove gravity from the acceleration readings. Future iterations of the Smart Sleeve would benefit from IMUs that filter out gravity and only return linear acceleration, one such sensor is the Bosch Sensortec BNO055.

The Smart Sleeve would also benefit from sampling data at a higher rate which would allow for more granular reconstruction of position. The sampling rate of the current system is limited by both the speed of the Adafruit Feather's processor, the I2C bus speeds that the Adafruit Feather supports, and the fileIO. The Adafruit Feather only supports I2C speeds of 100 Kbit/s but newer I2C standards

support speeds of 400 Kbit/s, 1 Mbit/s, and 3.4 Mbit/s. Additionally, due to limited RAM on the Adafruit Feather the recordings cannot be stored in RAM and must be written to flash memory while recording is ongoing. This slows down the recording process and a microcontroller with more RAM would be desirable to speed up sampling. One possibility for a future version of the Smart Sleeve would be to use the Intel Edison microcontroller. The Intel Edison [10] provides I2C bus speeds up to 3.4 Mbit/s as well as a 500 MHz processor, 1 GB of RAM, and on-chip Wi-Fi and Bluetooth capabilities. This would provide IMU sampling rates of up to several hundred hertz and powerful on-board computation such as filtering data in real-time before storing in the cloud.

The Smart Sleeve currently provides feedback and analysis for the user through a Python application on the desktop. The Python application reads data from the cloud database, filters and processes it, and displays information for the user to view. Future generations of the Smart Sleeve would benefit from an interface either through a web application or mobile application so that users could receive instant feedback from the system on their phone or other mobile device.

The Smart Sleeve currently contains wiring that runs on the outside of the sleeve to power each device as well as provide the I2C bus. While the wiring is lightweight and nonintrusive, future generations of the Smart Sleeve could utilize conductive wiring that is sewn into the sleeve so that no wiring is visible, providing a cleaner look.

While the Smart Sleeve is targeted at basketball free throw shooting, the technology to track motion using inertial measurement units has great potential outside of basketball. The Smart Sleeve would be

easily expandable to capture and analyze motion for other sports, such as bowling, tennis, golf, or baseball. Additionally, many possibilities exist in capturing motion data for medical, veterinarian, and physical rehabilitation applications.

10. Leadership

Appendix C contains a table summarizing each individual member's contribution and leadership roles.

11. References

[1] SparkFun, "SparkFun 9DoF Sensor Stick," SparkFun, [Online]. Available: https://www.sparkfun.com/products/13944. [Accessed 26- Apr- 2017].

[2] Adafruit, "Adafruit Feather HUZZAH with ESP8266 WiFi," Adafruit, [Online]. Available: https://www.adafruit.com/product/2821. [Accessed 26- Apr- 2017].

[3] SparkFun, "Lithium Ion Battery - 850mAh," SparkFun, [Online]. Available: https://www.sparkfun.com/products/341. [Accessed 26- Apr- 2017].

[4] B. R. H. Brad R. Humphreys, "The Size and Scope of the Sports Inudstry in the United States," *IAS*, 2008.

[5] "College Basketball & Scholarship Opportunities," Scholaship Stats, [Online]. Available: http://www.scholarshipstats.com/basketball.htm. [Accessed 26- Apr- 2017].

[6] M. J. Burns, "Andre Drummond turns to virtual reality to improve free throw shooting," *Sports Illustrated,* 28 9 2016.

[7] H2H SPORT, "H2H SPORT Unisex Compression Fit Hand Cover Cooling Arm Sleeves UV Protection," Amazon. [Online]. [Accessed 26- Apr- 2017].

[8] "Are You an Average Man?", Elitefeet.com, 2017. [Online]. Available: http://www.elitefeet.com/ar you-an-average-man.

[9] Washington State DSHS, "Range of Joint Motion Evaluation Chart", 2014.

[10] Intel Corp, "Intel® Edison Development Platform", 2015. [Online]. Available: http://download.intel.com/support/edison/sb/edison_pb_331179002.pdf. [Accessed: 27- Apr- 2017]

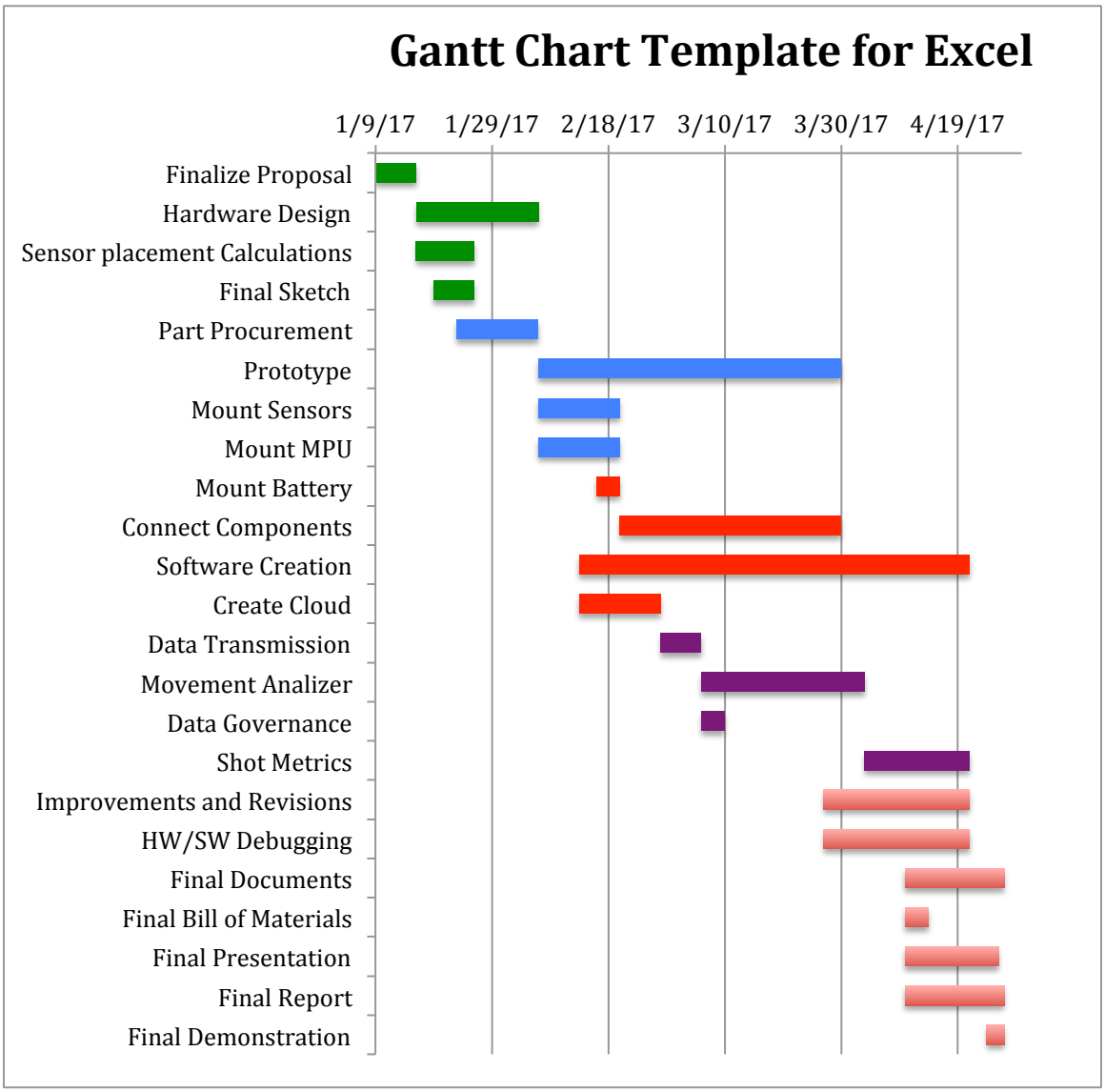
Appendix A

Task Lead List with Risk Levels

Task	Task Lead	Risk Level
Finalize Proposal	All	Low
Hardware Design	Edison, Patrick, Kyle, Mickeal	High
Sensor placement calculations	Edison	Medium
Final sketch of design	Patrick	Low
Complete layout of data store	Kyle	Low
Complete design of data push system	Mickeal	Medium
Part Procurement	Adam	Medium
Prototype	Adam, Patrick, Edison, Kyle	Medium
Mount sensors on shooting sleeve	Adam	Low
Mount microcontroller on shooting sleeve	Adam	Low
Mount battery pack to shooting sleeve	Adam	Low
Connect components to microcontroller	Edison, Patrick, Kyle	High
Hardware-Hardware integration	Edison, Patrick, Kyle	High
Software	Mickeal, Kyle, Edison, Patrick	Medium
Create cloud storage system using Firebase	Mickeal, Edison	Low
Transmit data from microcontroller to database	Edison, Patrick, Kyle	Medium
Create program to analyze movement of sensors	Mickeal	Medium
Ensure data transmitted is accurate	Mickeal, Patrick	Medium
Create metrics for comparison between shots	Mickeal, Patrick	High
Improvements and Revisions	All	High
Debugging	All	Medium
Documentation	All	Low
Final Presentation	All	Low
Final Report	All	Low
Final Project Summary	All	Low
Final Demonstration	All	Low

Appendix B

Project Gantt Chart



Task Name	Start	End	Duration (days)
Finalize Proposal	1/9/17	1/16/17	7
Hardware Design	1/16/17	2/6/17	21
Sensor placement Calculations	1/16/17	1/26/17	10
Final Sketch	1/19/17	1/26/17	7
Part Procurement	1/23/17	2/6/17	14
Prototype	2/6/17	3/30/17	52
Mount Sensors	2/6/17	2/20/17	14
Mount MPU	2/6/17	2/20/17	14
Mount Battery	2/16/17	2/20/17	4
Connect Components	2/20/17	3/30/17	38
Software Creation	2/13/17	4/21/17	67
Create Cloud	2/13/17	2/27/17	14
Data Transmission	2/27/17	3/6/17	7
Movement Analizer	3/6/17	4/3/17	28
Data Governance	3/6/17	3/10/17	4
Shot Metrics	4/3/17	4/21/17	18
Improvements and Revisions	3/27/17	4/21/17	25
HW/SW Debugging	3/27/17	4/21/17	25
Final Documents	4/10/17	4/27/17	17
Final Bill of Materials	4/10/17	4/14/17	4
Final Presentation	4/10/17	4/26/17	16
Final Report	4/10/17	4/27/17	17
Final Demonstration	4/24/17	4/27/17	3

Appendix C

Team members and their responsibilities.

Group Member	Roles and Tasks
Edison Carrick	Team Leader, helped write IMU firmware, helped create soldiering diagram and wiring diagram, worked on Lua code for microcontroller, helped implement microcontroller http post code
Kyle Kizirian	Expo Coordinator, Lead firmware development, helped write IMU firmware, wrote python code to convert data from Unicode string to usab values, helped write I2C code library, handled microcontroller firmware integration and development
Patrick Miller	Document Coordinator, Lead analytics and algorithmic development, Researched algorithmic solutions to gravity negation, implemented low pass filter algorithms on data sets, wrote algorithms to cancel gravity and calculate true x,y,z acceleration from raw acceleration and gyroscope dat
Adam Jackson	Procurement manager, Lead prototype engineer, constructed prototype sleeve with pockets for electronic components, tested flexibility and durability of sleeve design, created test-rig for optimal testing and debugging, handled part orders and requests, calculated bill of materials
Mickeal Taylor	Web Master, Lead front end engineer, created animated interface to play back shot data in real time, organized database and data storage structure created GUI with the ability to pull data from firebase, save data to firebase, simulate shot form, and display shot next to saved baseline shot, created website

Appendix D

NodeMCU Firmware:

https://github.com/ecarrick/nodemcu-firmware

Adafruit Feather Huzzah Code:

https://github.com/ecarrick/seniordesign_firmware/blob/master/simple.lua

Python User Interface Application:

https://github.com/ecarrick/seniordesign_firmware/blob/master/AnalyticsApplication.py