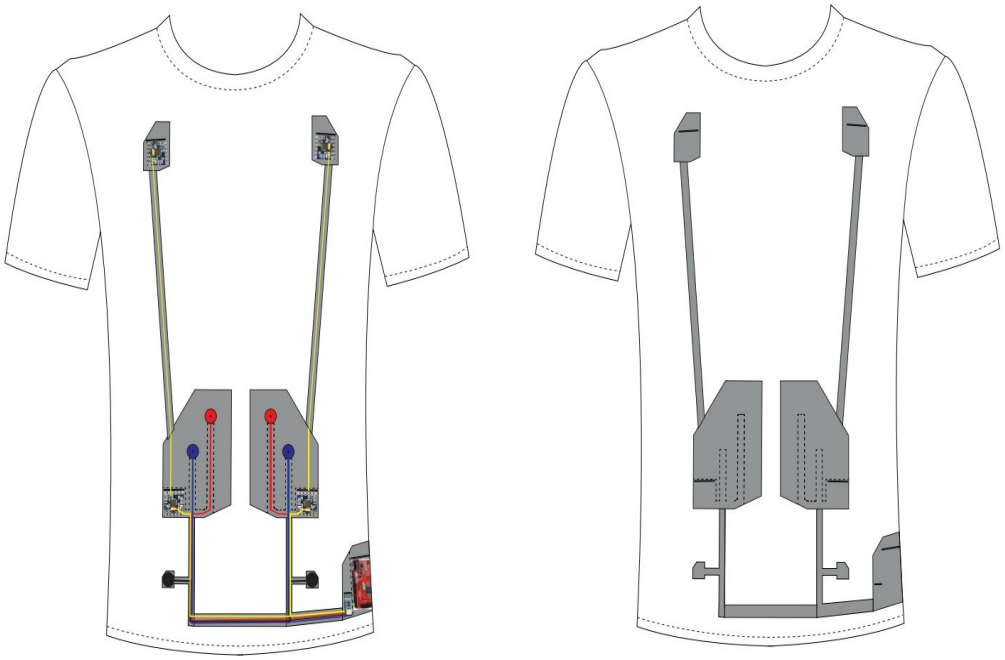
Future Wearables

Design Documentation

Advisors:	Suraj Kothari	ECpE
	Carey Novak	ISU Industry Relations
Clients:	Ted Kepros	Kepros PT
	Jeremías Sauceda	Ensoft, Inc.
Members:	Aaron Reyes	Team Leader
	Nick Plutt	Webmaster
	Will Park	Communication Leader
	Josh Cline	Key Concept Holder #1
	Nick Gonner	Key Concept Holder #2



Introduction

Over the past few decades, electronic devices have become an increasingly popular aspect of society, and the medical industry is no exception. The advent of digital medical records, and electronic devices have revolutionized the way medical professionals evaluate and treat patients. Ted Kepros, a physical therapist from Cedar Rapids, IA commissioned a group of Iowa State Students to design, build, and test a new wearable electronic device to monitor back movement and posture. While this device is not the first of its kind, it is unique because it is built by students, and it monitors muscle movement in the lower back, which is a feature that is absent from other devices of its kind.

This device utilizes the data from several motion sensors and a pair of muscle activity sensors to measure and record the activity of a patient under test. That data is then sent to an Android device, where the it is displayed graphically with respect to time. This data can be then used to evaluate the patient's posture and movement, providing detailed medical records for the physical therapist to review and analyze.

The group of Iowa State students is comprised of Electrical, Computer, and Software Engineers who were advised by Jeremías Sauceda, co-founder of Ensoft, Inc. and ISU alum. This project was completed to meet the ECpE Senior Design requirement.

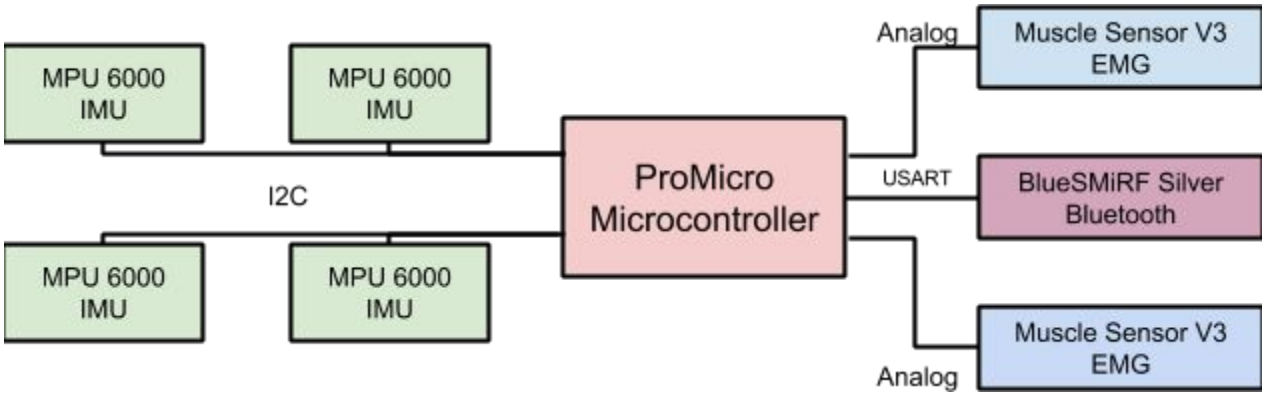
System Level Design

Hardware Requirements

- The device will measure muscle activity in the lower back region.
- The device will measure back orientation from four positions, the upper back on both shoulder blades just below the neck and on each side of the lower back above the waistline.
- The device will be battery operated.
- The device will be embedded in a wearable "harness" to be worn by patients.
- The device will transmit collected data to an android device.

Block Diagram of the Wearable Device

This is a block diagram overview of the system components and the communication protocols between them.



Hardware Components

IMU (MPU 6000)

Inertial measurement units (IMUs) are ideal for measuring orientation and rotation. These measurements are the core of tracking posture. The MPU 6000 is a digital, six degree of freedom IMU. The device features a three axis accelerometer and a three axis gyroscope, each with three 16 bit ADCs to manage the conversion of each axis. The accelerometer can be configured to measure in ranges of +/- 2g, 4g, 8,g, or 16g, with a resolution of 0.0305, 0.061, 0.122, or 0.244 milli-g's per bit respectively. The gyroscope can be configured to measure +/- 250, 500, 1000, or 2000 degrees per second with a resolution of 0.00381, 0.00763, 0.0153, or 0.0305 degrees per bit respectively. These are very fine measurements that should allow us to very closely track the movements and orientation of the patient. The device operates between 2.375 and 3.46 volts, compatible with our 3.3V system. The MPU 6000 can communicate via SPI or I2C, at speeds of 1 Mhz or 400 kHz respectively. A related product, the MPU 6050, matches these specifications except for the SPI interface. The MPU 6050 only supports I2C communication, which is why we chose the MPU 6000 over this device. The other alternative we investigated was using independent accelerometers and gyroscopes. While a feasible idea, this would double the number of peripheral devices in the system, likely increase power draw, not save much in the way of cost, and create a greater mess of wires.

Bluetooth Module (BlueSMiRF Silver)

The device will use a bluetooth module to transfer data from the onboard flash to a paired android device such as a phone or a tablet. The BlueSMiRF is a Bluetooth modem that utilizes a USART interface for communication to the microcontroller and boasts baud rates from 1200 bps to 921 kbps. The device operates on 3.3V, which is compatible with the rest of the system. It is capable of data transfer rates of 3 Mbps for a distance of 20 meters. This is well within the expected range of operation for our system, and the bandwidth is more than capable of meeting our needs..

EMG (Muscle Sensor V3, Sensor Cable, and Electrode Pads)

The Muscle Sensor V3 is a muscle sensor from SparkFun. The device operates between 3V and 30V, compatible with the 3.3V system. This chip also does some initial signal processing by rectifying the raw EMG data and smoothing it so that an ADC can further process the signal.

Microcontroller (Pro Micro from Sparkfun)

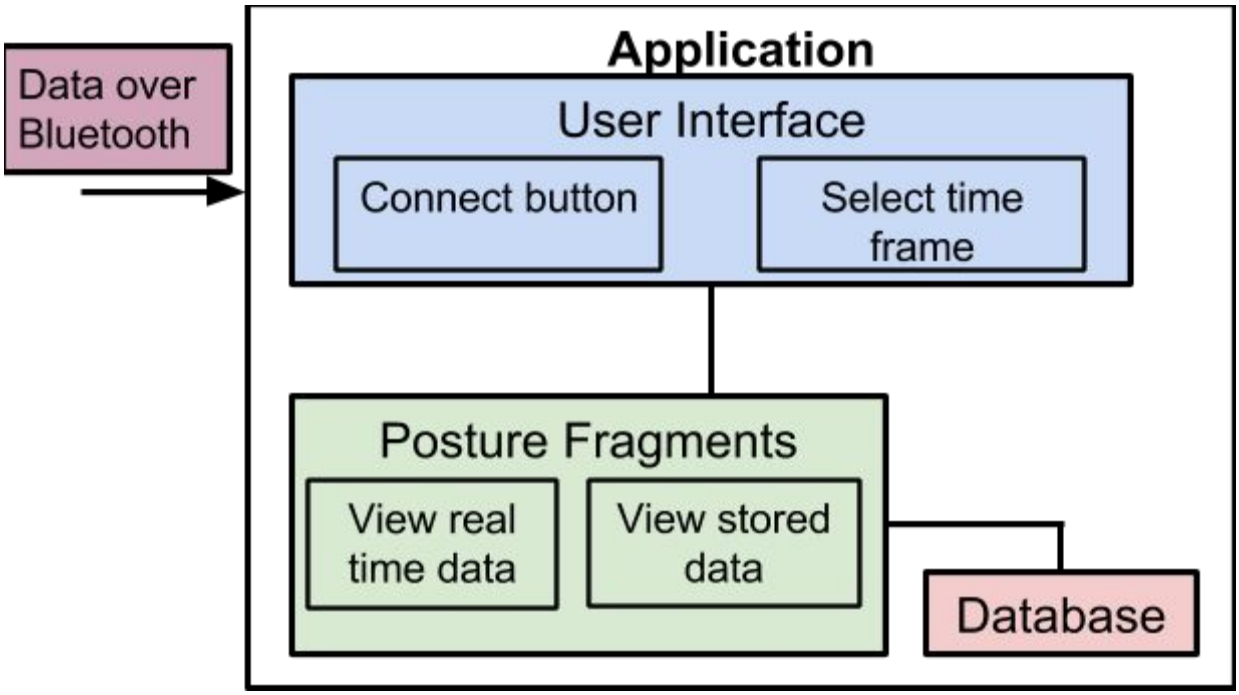
The Pro Micro meets many of the system's requirements. The Pro Micro has built in SPI, I2C, and USART ports which will meet all of the communication needs. The board operates on 3.3V, has a 10-bit 4 channel ADC for interfacing with the EMG chips, 32 kbytes of Flash and 2.5 kbytes of SRAM. This microcontroller is also compatible with the Arduino libraries. There are a variety of libraries for I2C, USART, and interfacing with the ADCs. This makes the embedded side much easier to setup. The downside to this microcontroller is that it only has 1 hardware I2C channel. In order to communicate with all of our IMUs, we need two I2C channels to avoid address clashing. This has been solved using a software I2C library that converts two regular I/O pins into an I2C bus.

Hardware Connections

- IMU
    - The red wire connects the main power(VCC) of the Pro Micro to each IMU
    - The black wire connects the ground(GND) of the Pro Micro to each IMU
    - The green wire is the data line(SDA) for the I2C bus. Two IMUs share each green wire, which then connected them to the same I2C bus.
    - The yellow wire is the clock line(SCL) for the I2C bus. The same IMUs that share a green wire also share the same yellow wire.
- EMG
    - The red wire is the positive bias(VCC+) and connects to the positive lead of the 9V batteries.

    - The black wires connect to ground(GND). The ground associated with the bias voltages is routed to the battery ground and the ground next to the signal is connected to the microcontroller's ground.
    - The yellow wire is the negative bias(VCC-) and connects the the negative lead of the 9V batteries.
    - The green wire is the signal(SIG) and is connected to the Pro Micro's analog to digital converter at either pin A0 or A1.
- The EMG Leads
    - The red cable measures center of muscle activity and connects to the top snap on the back of the shirt.
    - The blue cable measures end of muscle activity and connects to the bottom snap on the back of the shirt.
    - The black cable measures the electrical potential of a bony portion of the body. Use a sticky electrode pad to connect the black probe to the tailbone or hip bone region.

Application Software Diagram



Message Format to be Sent Over Bluetooth

Byte	Type	Measurement
0		Start Bit (Throw away)

1-2	Int	EMG1
3-4	Int	EMG2
5	Int	Degree X Curve of spine (Upper back - lower back)
6	Int	Degree Y Curve of spine (Upper back - lower back)
7	Int	Degree Z Curve of spine (Upper back - lower back)
8	Int	Y angle wrt gravity (bent over forward)
9	Int	Z angle wrt Y (curve of spine forward)
10	Int	X angle wrt Y (curve of spine to the side)
11		End (Throw Away)

Software Components

The microcontroller code is written in C++, because it is a popular embedded system language, the team was familiar with it, and the Arduino libraries are written in C++. We utilized an I2C hardware library from Arduino and one I2C software library for interfacing with the IMUs as well as a UART library from Arduino for interfacing with the bluetooth chip. The Pro Micro collects the data from each sensor and converts the IMU data into measured angles, relative to the direction of gravity. The collected data is packed into a byte array, the layout for which is shown above.

The Android application was developed in Android Studio using Java. After tapping the app icon it opens up into the home page which displays the team, the client and the advisors names. You can then click the button in the top left to open the navigation drawer. This has three panels: home, real time data, and stored data. Selecting the real time data will reveal one of the user interfaces, the connect button. When the connect button is hit, the application uses Android's Bluetooth package to retrieve data transmitted by the wearable device. This data is then displayed in the real time graph fragment and stored in the SQLite database. The other panel on the navigation panel is view stored data. When this is selected it brings up a similar fragment to the real time data except it has text inputs on the bottom and a get button.

The select time frame allows the user to pick a time interval from which to view stored data. Hitting the get button will cause the stored data graph fragment to retrieve the data from the SQLite database and display it on a graph. The application uses an open source graphing library call Graph View to take care of the graphing of the data.

Detail Description

I/O Specification

Hardware

Input

- 3 axis data from each IMU's accelerometer and gyroscope from the upper left, upper right, lower left, and lower right of the back.
- EMG signal from the left and right of the lower spine.

Output

- Angle differences between the top of the spine and the bottom of the spine in degrees and the digitized EMG data

Software

Input

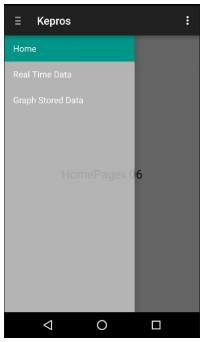
- Data points received from hardware

Output

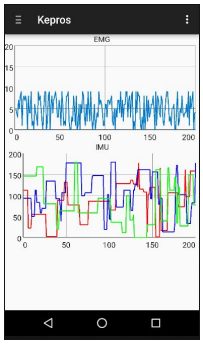
- Degree of curvature of the spine and degree bent forward.

Interface Specification

This screen is the home screen it displays the team information and our clients. On the top is the tool bars. The bar has the name kepros which is the name of the application. On the right side is three vertical dots that open a settings page that currently has a little functionality to support bluetooth settings. On the right of the bar is three vertical lines. This button opens up the navigation bar which is used to get to the screens that do the main functionality of the application as described below.



This image shows the navigation drawer, which is used to navigate around the different part of the application. It has three different panels that navigate different screens. The first is the home panel which navigates to the above screen. The next two are the graphing panels. One goes to the real time graph screen while the other one goes to the stored graphs screen



This image represents the final two panels in the navigation drawer the graphing screens. The first graphing screen has a button you need to hit to connect to the device. After the button it hit it displays the real time graph with a similar look as to this screen.

The other graphing screen has a selection on the bottom where you select the start date and start time as well as the end date and end time. The graph will then display the data between those two selections.

Software Specification

- Mobile application
    - Retrieve data from bluetooth
    - Take retrieve data and possibly further refine/filter it
    - Have ability to dump data to a CSV file
    - Have way to view graph of data real time

Testing, Procedures, and Specifications

We will be using validation and verification to test our product. Both of these are described below.

Validation

We plan to keep our client up to date for the duration of the project to ensure that they are happy with the development of the device. Doing this should ensure that the client is happy with the final result and specifications.

Steps for Validation

We had several meetings with our client to ensure validation of the posture wearable device.

- Meeting 1: Gain a clearer understanding of what the goals of the project were and how to best go about achieving these goals.
- Meeting 2: Review the parts list and go over why each part was selected.
- Meeting 3: Test the initial prototype and review the validity of the data being retrieved.
- Meeting 4: Test and demonstration the second iteration of the product. Verify validity of data that we received.
- Meeting 5: Final demonstration of the product. Get final design opinions of the product

Verification

Coding

For the software and embedded system verification we will have various measures to ensure the quality of the code. We have four people that are experienced with dealing with software. Each person is going to be somewhat familiar with and working on all aspects of the coding. This will ensure that if we have one person gone or busy the others can pick up the slack. This also means that the people are able to double check others work and spot problems early on. We also will have various reviews, checks, and testing of the code so that we can find and document problems early on. We will also be using github for version control and reporting bug issues.

- Hardware
    - Test parts individually to ensure they will work as needed.
    - Add groups of parts together to test ex. EMG and conductive fabric.
    - Connect all the parts to the micro controller to ensure it works as a system
- Embedded System
    - The device is getting the EMG data

    - The device is getting all four of the IMUs' data
    - The device is transmitting over bluetooth
- Android Application
    - The application receiving correctly over bluetooth
    - The application is not missing data
    - The application is correctly parsing and storing data
    - The parsed and stored data is being correctly graphed

Testing Results

The testing results are split up into two categories validation and verification.

Validation

After the fifth meeting we talk with the client's about their opinion of the product and how much we had done. They agreed that they were happy with our prototype and the information we had gained as part of our senior design project was researching into if a device was viable and if it could be a consumer product. They had a few things that we could still do to get to the end like creating a CSV output for the database and including in this documentation a recommendations for a future senior design team.

Verification

- Hardware
    - All components work individually
    - All components work with the microcontroller and bluetooth
    - All components work when connected together as a whole system
- Embedded System
    - The device is correctly receiving the EMG data
    - The device is correctly receiving all the data from the four IMU's
    - The device is transmitting the data it has received and transferred over bluetooth using a byte array.
- Android Application
    - The application is correctly receiving the byte array over data
    - The application failed at receiving all byte arrays.
        - It is infrequently splitting a byte array maybe around 5% of them are split this was addressed by dropping split byte arrays from the stored data as not to improperly store values.
    - The application correctly parses and stores all data from complete byte arrays.
    - All data that is collected is able to be graphed either in a real time graph while collecting or in a stored graph.

Other Documentation

Software/Firmware Development Environment

For the software aspect of the application we have decided to all use Android Development Studio (ADS) as our main integrated development environment. There was some talks about using Eclipse as the IDE, however, we decided that ADS would be the most beneficial IDE. Although it is rather new as an IDE, we are fairly confident that it will suffice for this project.

The application was tested on multiple devices running various versions of android. We had three physical phones (Moto X, Samsung Galaxy Note 3, Samsung Galaxy S3) as well as ADS's virtual emulator of a Nexus 5. The application minimal API SDK pack was set to Android SDK 4.1 (Jelly Bean). During the time of testing, we tested our application on multiple Android versions including JellyBean 4.1, KitKat 4.4, and Lollipop 5.0.

Since multiple people were working on this application at the same time, we made sure that everyone had the same Java Development Kit (JDK 1.8) installed on their respective machines. We also made sure that anytime there was an android update, that everyone would update at the same time starting from Android Studio 1.3.2 to the current version 1.5.1.

We decided to use GitHub as our code sharing portal as it was the most comfortable and convenient way to work upload and work together on the project. There were other alternatives that was considered for SVN use, like our server, however all the members were most familiar with GitHub.

For the graphing in the Android application we used a free software library called Graph View. We decided to go with this library because some of the members of the team has used it in the past. It is very user friendly and easy to follow. It already included many functionalities that the regular graphing library did not include.

We used the Arduino IDE, version 1.0.5, for developing the code for the Pro Micro. Newer versions of the IDE should be usable as well. To use the IDE with the Pro Micro, download the appropriate add-on files from the Sparkfun Pro Micro product page: https://www.sparkfun.com/products/12587. Follow the installation instructions and the Pro Micro should be listed under the Tools->Board menu of the Arduino IDE.

Appendix I. Operation Manual

Step by step instructions on how to use the posture device.

Collecting and Displaying Data

1. Put on compression shirt.
2. Make sure the conductive fabric on the back are both equal distance from the spine.
3. Attach a sticky conductive pad to the hip bone on either side of the spine about at your waist line.
4. Have someone attach all four imu's to your back with '1' being upper left and then the numbers going clockwise around your back. The wires should all be going out toward the spine.
5. The red probes of the EMGs will attach to the button snap on the upper back one on each side.
6. The blue probe of the EMGs will attach to the button snap on the lower back of the same side as their respective red probes.
7. The black probe of the EMGs will attach to the conductive pad on the same side as their respective other probes.
8. Make sure the device has batteries in and if it does turn the device by toggling the switch. A blinking red light should be visible.
9. Open the bluetooth settings on your android
10. Make sure your bluetooth is on.
11. If the device has already been paired with the shirt and not unpaired **skip to step 15.**
12. Click the search for devices.
13. Search for the device "Posture" when it appears.
14. The pairing password is 1234.
15. Open up the Kepros application on your android smartphone device.
16. Touch the three horizontal lines on the top left corner.
17. Select the option that reads Real Time Data
18. It should display two graphs the top one titled EMG and the bottom one titled Degrees and at the bottom should be a button labeled CONNECT AND RUN.

19. Once the device is connected the blinking red light on the wearable device should turn a solid green, the CONNECT AND RUN button should disappear, and the graphs should start displaying data.

Viewing stored data in the Android application

1. Open up the Kepros application on your android smartphone device.
2. Touch the three horizontal lines that are vertically in line on the top left corner.
3. Select the option that reads Graph Stored Data
4. It will open a screen with two graphs one labeled EMG and one Degree at the bottom are some input fields and a Get button.
5. In the From Time input field enter a start time formatted hh:mm in military time for example 2:30 pm would be 14:30.
6. In the From Date input field enter a start date formatted mm.dd.yy for example December 8, 2015 would be 12.08.15.
7. In the To Time input field enter an end time using the above time format.
8. In the To Date input field enter an end date using the above date format.
9. Then click “Get”
10. Depending on the time frame and amount of data it may take a little while for the graphs to load.
11. You can then scroll through the graphs by swiping on them left or right.
12. You can zoom in on the graph by putting two fingers on it and pinching.
13. You can also zoom out from the graph by putting two fingers on it and pulling them apart.
14. Depending on the amount of data the graph may be sluggish to commands.

Appendix II. Initial Versions

In the initial version there was plans to include a SD card storage on the embedded system. This was scrapped after talking to the clients and getting a better understanding that while long term that would be beneficial at the moment they were looking at getting the device to monitor back posture and off load that data so that it could be analyzed and viewed. The best we thought to do this was by not storing it on the device but transferring it to a phone so I could be stored in the phone and viewed real time or later.

A component change that occurred multiple times is our microcontroller. The initial version used the Pro Micro however after we got our four IMU's we realized that the breakout board did not have the ability to choose an address to the SPI pin. This led us to using the IMU's I2C pin but the Pro Micro did not have two I2C ports therefore we looked into getting another micro controller. We decided on the maple since it met above our requirements and had two I2C ports as well as being arduino compatible. The next set of problems with our micro controller was that even though it said it was arduino compatiable a lot of the libraries were not ported including the I2C library and we were unable to establish a connection to our IMU's. In the end and our current design we went back to the pro micro and implemented a software I2C using a library we found.

In the version before our final design we changed two different and major aspects of the software. The first was instead of printing a string to the bluetooth we decided to implement a byte array. This change would ensure we could easily get data out of the bluetooth buffer instead of trying to parse a string; especially since we had problems with splitting up the tokens. The data would sometimes overflow or send over too quickly causing issues when parsing the string. It would be hard to locate at what index and what data point is coming in from the device.

Since we changed to the byte array, it made sure that when a line of data points is all ready to be sent out, it could be sent out in an organised fashion. This ensured that we were retrieving all the data at once. We also then had to try to lower the number of bytes transmitted by the wearable device to ensure it all transmitted quickly and in one piece. This led us to offload the angle calculations from the android side to the embedded system side, decreasing our byte array from fifty eight bytes down to twelve bytes.

Appendix III. Other Considerations

Lessons Learned

From this project there were many things that were learned while developing the prototypes, most notably of which is to first make sure that all of the hardware that you order is compatible with your needs. One of the original problems that developed was the IMU's we ordered were using a different model's breakout board. This led to us not being able to use SPI as the method of communication. Instead we had to use I2C which our micro controller only had one of instead of the two we needed. Because of this we ended up ordering a new micro controller that supported our prefered connection protocol. However after receiving the micro controller, we soon realized that there was little to no software support libraries that had been developed for the micro controller, contrary to what it had stated. Unfortunately because of this we were forced to go back to our original micro controller and implement a more difficult solution by using a software implementation of an I2C bus.

Recommendations for next group

The Shirt

Currently we use a compression shirt which allows the emg sensor to move around on the user's back when the user moves. This results in the emg producing inconsistent and faulty data. Because of this we believe that the best way to go about fixing this is listed below.

- Develop a two part system in which the two EMGs would be strapped separately.
    - This would help ensure that the shoulders or back moving or stretching does not pull the whole system out of alignment.
    - To hold the emg sensor in place, we believe that a setup similar to a heart rate monitor with a strap holding the device in place on the user's back would do the job.
    - The IMUs would then be attached to the these straps in similar locations to where they are now.

The Application

As of right now it gets and stores graphing data at about eleven messages second. Ultimately our client would like to get that number up to twenty. This recommendation is a little harder and below is a list of ways this may be accomplished.

- Speed up processing and decrease wait times on the embedded system side.
    - This will alter the computation of angles from the gyroscope data, because gyroscope conversion uses time increments.
- On the application side, have the bluetooth collection, UI graphing, and storing of data each on separate threads. Possibly have a fourth thread that would process the information, and send it to the storing and UI graphing thread.
    - This would have to take into consideration a kind of race condition in which there could be a backup of information or dropped information between threads.
- It was mentioned that he wanted the application to be more consumer oriented. Currently, the device just records and shows numbers that are raw data points that an average user would not understand completely. For the consumer side of the application, maybe create animations / emotes / notifications to alert the user of their good or bad posture.

Data Representation

Currently you can select data that you want to view by selecting the date and time range. To decrease the excessive time it takes to load it only shows one data point for a little under every second. This is because over a few minutes period there could be thousands of data points and the loading time was getting too long. Below are ways that this could be solved.

- Find ways to speed up the retrieval and graphing without losing data
    - Possibly combining all the sql select statements into one
- Allowing the user to pick which the graphs they want.
    - Each graph would isolate a different axis of motion
    - If a user picks all the graph possibilities the problem will persist
- The best solution may be transferring the data to a web server and having a website or computer application display the data.
    - This would also give the user the ability to have created more powerful tools for manipulating and analyzing the data in the future.

Data Storage

The wearable device could collect all data for a period of time then offload the data to the app.

- This would be for non-real time viewing and would take out the possibility of the Android application slowing down the data being collected.
    - Main application would be in a controlled environment and checking degradation of posture.

Battery

The current battery solution is far from ideal, and there are a number of improvements that can be made to it. The original intention was to use a rechargeable Lithium Ion battery, similar to that you would find in a cell phone. Also, a smaller battery pack would be better, and Li-Ion battery would both occupy less space, and last longer than the disposable 9V batteries currently implemented.

PCB

It would also be beneficial to consolidate some of the wires and circuitry into a printed circuit board and have one manufactured. A PCB would also improve the battery implementation, and the power transmission to the sensors.