Motion Capture For Runners

Sponsor: Air Force Research Laboratory

ECE 480 - Team 8
Blake Frantz
Zhichao Lu
Alex Mazzoni
Nori Wilkins
Chenli Yuan
Dan Zilinskas
Executive Summary (ii)  
The objective of this project is to make a motion capture system using multiple inertial measurement units (IMUs). This system will capture the limb movement of a runner, analyze the data, and provide feedback on the running form. Size and accuracy of the analysis is also an important aspect to this project. Baseline data from a team member is used for comparison.  

By using engineering processes, the team was capable of constructing a body suit that would capture limb movement. Through data processing on the central microcontroller, proper real-time feedback was accomplished. Post-processing software was written as well for the runner to analyze their running form after their workout. This was done through wireless data transfer.  

The direct impact of this product is to improve a runner's form. By improving the form, the runner can run more efficiently. Additionally, this general motion capture and analysis process can be applied to other areas. As stated by the sponsor, “analyzing the relative motion of body parts is analogous to understanding the motion of flexible aircraft or spacecraft structures on which sensors may be placed.”

Acknowledgement (iii)  
Team 8 would like to specifically thank a few people that provided us with help or guidance throughout the semester:

**Dr. Selin Aviyente** - As our team’s facilitator, Dr. Aviyente met with us weekly to provide guidance on the project and give constructive criticism on how to improve our presentations or reports, as well as give suggestions on project implementation.

**Roxanne Peacock** - As the team’s purchasing consultant, Roxanne provided support to the team whenever something needed to be ordered. She was efficient and very helpful, giving the team updates on ordered parts, allowing the team to stay focused on the project design.

**Dr. Andrew Mason** - As the team’s hardware consultant, Dr. Mason provided support to the team when choosing components for the project. He also gave insight on the use of a multiplexer that would allow the microcontroller to communicate with multiple IMUs.

**Greg Motter** - As the team’s business application advisor, Greg provided his informative insight on a widely used business management strategy that was employed in our project.

**Toni, Sparkfun Technical Support** - She served as one of the primary resources for questions regarding the team’s project. This included interfacing between the Arduino Uno and the IMU sensors, as well as initial interpretation of the data.
Table of Contents

Chapter 1: Introduction and Background ................................................................. 5
  Introduction .............................................................................................................. 5
  Motion Capture Background .................................................................................. 6

Chapter 2: Exploring Solutions .................................................................................. 7
  Fast Diagram .......................................................................................................... 7
  Feasibility Matrices for Components ..................................................................... 8
  House of Quality .................................................................................................... 10
  Initial Budget Costs ............................................................................................... 10
  Gantt Chart ............................................................................................................ 11
  Proposed Solution .................................................................................................. 12

Chapter 3: Hardware and Software Design .............................................................. 14
  Component Descriptions ....................................................................................... 14
    Arduino Uno Microcontroller .............................................................................. 14
    IMU - MPU-9150 ............................................................................................... 15
    Multiplexer - CD74HC4067 ............................................................................... 15
    Lithium Ion Battery ............................................................................................. 16
    XBee Wireless Communication - XBee ZB Series 2 ......................................... 16

Final Block Diagram ................................................................................................ 17

Hardware Design and Implementation ..................................................................... 18
  Feedback Controller ............................................................................................... 18
  Bodysuit ................................................................................................................ 19
  Arduino (Pins and Connections) ............................................................................ 20
  IMU (Pins and Connections) ................................................................................ 21
  XBee and XBee Shield ......................................................................................... 22
  Multiplexer ............................................................................................................ 22

Software Design and Implementation ..................................................................... 23
  Arduino .................................................................................................................. 23
  IMU ....................................................................................................................... 23
  XBee Arduino Software ......................................................................................... 23
  Multiplexer ............................................................................................................ 23
  XBee Software ....................................................................................................... 24
  MATLAB .............................................................................................................. 25

Project Obstacles .................................................................................................... 25
  Multiple IMUs ....................................................................................................... 25
  Comparison Software Delay ................................................................................ 26
  Length of Ribbon Cables ..................................................................................... 26
  Enclosure for Motion Capture Circuit .................................................................... 27
  Comparison Software Accuracy and Post-Processing ........................................... 28
Chapter 1: Introduction and Background

Introduction

Many people have little knowledge of the efficiency of their running form. By improving their form, runners can significantly improve their performance. Runners can improve upon the efficiency of their gait by comparing their running form to that of an elite athlete. To achieve the measurements necessary for this analysis, (inertial measurement units) IMUs can be placed along the body to capture limb and joint motion. IMUs contain numerous gyroscopes and accelerometers that electronically measure position and movement of the runner. Different types of IMUs are used for different applications. The data collected by the IMUs must then be analyzed to provide feedback to the runner. Data collected from gait analysis is comparable to data collected from sensors on flexible portions of an aircraft. Therefore, the motion of flexible aircraft can be better understood through this application of running form analysis.

The objective of this project is to design a motion capture device that can be worn by runners in order to improve their running form. The device must be portable and unobtrusive, to not impede the runner. The product must be able to send real time data about the runner’s form to provide accurate feedback with which improvements can be implemented. The product must also be able to function with different sized runners, as not all runners will have the same proportions. The feedback provided must also be easy to interpret, to allow the runner to make alterations in their form at will. Lastly, the product should be as low-cost as possible, making it accessible to many different users.
**Motion Capture Background**

Motion capture is the process of recording the movement of objects or people that can be analyzed or modified using computer software. Motion capture systems are used in video games and film making industries to translate human movements to animated 3D models. These systems have even been developed and used in military applications along with entertainment, sports, medical, and robotics industries.

There are two main types of motion capture technology; optical and non-optical systems. Optical systems involve using a camera or a system of cameras to capture the body's movements. The Microsoft Kinect for the Xbox360 is a prime example of this technology. A built-in camera captures the body's movements and replicates these movements on an animated character. In other systems, small markers are placed on the body and then are continuously captured by the camera and computer software.

There are two main types of optical marker systems. Passive marker systems use markers coated in a retro-reflective coating that reflects light and is then captured by the camera system. Active marker systems use small LED markers placed on the body that emit light for very short periods of time. These systems are more effective due to the increase in distances and volume for capture.

Non-optical systems use inertial sensors that are then wirelessly transmitted to a nearby computer. These systems use sensors that are measured on multiple axes and are often used in a full bodysuit that is worn by an individual. These systems are most popular for video game developers due to the quick and easy setup, resulting in more accurate measurements. These systems require no cameras, are more versatile, and quickly becoming the top choice for motion capture systems.

Currently, there is no widely accepted non-optical motion capture system that can be used for runners. With no base designs for such a product, this project provides a unique approach to provide insight for runners. Currently running form is generally improved using personal intuition or by a personal trainer's guidance. With this motion capture bodysuit, no personal trainer will be necessary. This product will include a detachable feedback controller, multiple IMU sensors, and have post-processing software to improve running form. With quick and accurate feedback that will be provided, the user will be able to change their form during their workout. This product will cut down on the time it takes for feedback and on the potential steep prices associated with personal trainers or running coaches.
Chapter 2: Exploring Solutions

Fast Diagram

The team worked to simplify the overall design by using a FAST diagram to verify the necessary parameters and components of the project. The structure of the diagram is to follow from left to right while asking the question “How”, and in the reverse direction, to ask the question “Why?” Figure 1 above shows the diagram created by the team early on in the semester that was used as a reference to keep track of the team’s overall goals.
Feasibility Matrices for Components

In order to determine which aspects of the project were significantly important to the overall design, feasibility matrices were created to verify each aspect of the project. The matrices were separated based on the different requirements to create a working prototype.

The following matrices are composed of the team’s different conceptual component designs. Each matrix has different criteria to rate each component on multiple aspects which were deemed important. The ratings go from 1 correlating to low importance, to 5 being high importance. The value of each component also follows the same rating system. Below is an explanation of what the team interpreted each criterion as.

**Implementation**: The difficulty of working with and setting up each component

**Accuracy/Consistency**: The overall precision of the data being sent/received by the component

**Usability**: The ease of use for an average user

**Compatibility**: How well the component works with other products/devices

**Functionality**: How well the component functions and its general usefulness

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Importance</th>
<th>Wired Arduino</th>
<th>Completely Wireless</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Implementation</strong></td>
<td>5</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td><strong>Size/Portability</strong></td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td><strong>Accuracy/Consistency</strong></td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td><strong>Usability</strong></td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td><strong>Compatibility</strong></td>
<td>2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td><strong>Functionality</strong></td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

**Overall Weighted Score**: 85 | 79

*Figure 3: Wired vs. Wireless Communication between IMU and Arduino*
<table>
<thead>
<tr>
<th>Criteria</th>
<th>Importance</th>
<th>LED Feedback</th>
<th>LCD Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation</td>
<td>5</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Size/Portability</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Cost</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>User Friendly</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Compatibility</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Functionality</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

**Overall Weighted Score**

**Figure 4: LED vs. LCD Feedback Controller**

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Importance</th>
<th>Wireless</th>
<th>XBee</th>
<th>Bluetooth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Range</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Cost</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Compatibility</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

**Overall Weighted Score**

**Figure 5: Wireless vs. XBee vs. Bluetooth Communication**

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Importance</th>
<th>Accelerometer</th>
<th>Gyroscope</th>
<th>IMU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Size</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Cost</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Sampling Rate</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Compatibility</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Number of Axes</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

**Overall Weighted Score**

**Figure 6: Use of non-optical sensor: Accelerometer vs. Gyroscope vs. IMU**
House of Quality

The house of quality created by the team is located in Appendix -4. The team used this six sigma method to determine the important aspects of functionality vs. customer requirements.

Initial Budget Costs

<table>
<thead>
<tr>
<th>Item</th>
<th>Price</th>
<th>Amount</th>
<th>Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMU MPU-9150</td>
<td>$49.99</td>
<td>5</td>
<td>$249.95</td>
</tr>
<tr>
<td>Arduino UNO</td>
<td>$49.95</td>
<td>1</td>
<td>$49.95</td>
</tr>
<tr>
<td>XBee Explorer USB</td>
<td>$24.95</td>
<td>1</td>
<td>$24.95</td>
</tr>
<tr>
<td>XBee Shield</td>
<td>$24.95</td>
<td>1</td>
<td>$24.95</td>
</tr>
<tr>
<td>XBee Communication Devices (ZB Series 2)</td>
<td>$22.95</td>
<td>2</td>
<td>$45.90</td>
</tr>
<tr>
<td>Compression Long Sleeve Under Armour</td>
<td>$34.90</td>
<td>1</td>
<td>$34.90</td>
</tr>
<tr>
<td>Nike Dri-fit Filament Running Tights</td>
<td>$38.48</td>
<td>1</td>
<td>$38.48</td>
</tr>
<tr>
<td>Analog/Digital MUX</td>
<td>$4.95</td>
<td>1</td>
<td>$4.95</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>$104.97</td>
<td>1</td>
<td>$104.97</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td></td>
<td></td>
<td><strong>$579.00</strong></td>
</tr>
</tbody>
</table>

Figure 7: Initial Budget Costs

The initial budget is shown in Figure 7. This budget shows how much it would cost to implement the bodysuit on a production scale. These values are just an estimate, as buying in bulk may lower costs significantly. Some parts may also be substituted based on the producer, as well as any changes in the overall implementation of the bodysuit. The main sources of cost include the price of the IMUs as well as the bodysuit clothing and the Arduino microcontroller.
The team’s project is broken into four main phases. Phase one was the fabrication phase. In the phase the team researched parts and components for the project and came up with a feasible design. The ordering of all components, wires, and sensors was completed in this phase. The team also completed the initial fabrication of the bodysuit.
Phase two of the project was the software phase. Within this phase, the team wrote all necessary software code for the microcontroller to communicate correctly with the sensors and the nearby computer. The comparison software (comparing an elite runner's form to that of the user) was also completed.

Phase three of the project was the interfacing phase. This is the most critical phase of the project. The team interfaced all of the components of the system. This included the sensors and feedback controller to the Arduino microcontroller and the microcontroller to the computer. The team expected most of the time spent would be in this phase.

The final phase of the project was the testing phase. The team completed testing on all aspects of the project. The team tested every element of the project individually and once all of these tests were complete the final testing period was held.

**Proposed Solution**

With the design requirements set as well as analysis from the feasibility matrix, the team proposed a solution composing of a body suit with multiple IMUs, an Arduino microcontroller, XBee communication, body-worn feedback controller, and an external PC. The IMUs continuously collect data based on the position of each sensor while exercising. The IMUs are placed throughout the body; 1 on the upper back, one above the knee for each leg, and one below the knee for each leg, resulting in 5 IMUs. These IMUs will be hardwired to the microcontroller to decrease data acquisition delay.

By creating the running suit, the product is able to be worn as clothing to decrease issues of running while wearing the device. The clothing will be flexible and tight, to ensure no connections come loose as well as collect accurate data to analyze. The wires will be sewn into the fabric of the clothing through ribbon cables. The team has also decided to use this product with a treadmill due to wireless restrictions.

The Arduino microcontroller is added to the system to ensure that all the IMUs are provided with power, and a synchronized clock signal. The Arduino will be connected with a 7.4Volt, dual-layer lithium ion battery. A synchronized clock signal is necessary to ensure that all the sensors are acquiring data at the same rate and time to prevent conflicting signals from occurring. All of these functions must be initialized using the Arduino programming software. Arduino has its own code format and requires the programs be initialized before it can function correctly.

Once the data is collected on the Arduino, the real-time processing will be done through written code in the Arduino software. The software will compare the locations of the sensors based on obtained measurements from one of the team members. The software will initially be used to compare simpler measurements such as angle, and stride length, as opposed to complicated measurements given time constraints.
After the data is collected, the comparison software will analyze the runner’s form. If the user is running improperly, the corresponding five light emitting diodes (LEDs), one for each sensor, attached to a body worn indicator will light up. The light will turn off once the runner corrects their position.

Post-processing is done through an external PC and MATLAB. The wireless communication is done through XBee communication. XBee communication requires two XBee communicators, as well as a USB device called an explorer. The XBee connected to the PC via USB will transmit data wirelessly to the microcontroller, which will also have an XBee communicator attached. The microcontroller will send the data to MATLAB for post-processing and provide visual plots for the IMUs.
Chapter 3: Hardware and Software Design

Component Descriptions

Figure 9: Arduino Uno Microcontroller

Arduino Uno Microcontroller

The Arduino Uno is the central processing unit for the bodysuit. The Arduino Uno is a very universal microcontroller, widely used for many hobby projects, as well as prototyping because of its universal design as well as its vast amount of online technical support. The Arduino Uno consists of the ATmega328 microcontroller and has an operating voltage of 5V. The Arduino has 14 digital I/O pins, as well as 6 analog input pins, as well as an overall clock speed of 16 MHz. The Arduino Uno will serve as the means of providing power to all the sensors, as well as the feedback controller.
The sensor chosen for the motion capture bodysuit was the MPU-9150 IMU sensor. This IMU gives 9-axes of measurement, as it consists of an accelerometer, a gyroscope, as well as a magnetometer. The accelerometer gives data based on the acceleration felt by the IMU in x, y, and z directions, the gyroscope gives measurements based on the angular acceleration around each of these axes, and the magnetometer gives the heading of the IMU based on the orientation. The MPU-9150 runs on 2.4V - 3.4V, and has an operating current of 4.25mA. The size of the sensor is 1.25” x .625” which is extremely small and works perfectly given the size constraints of the project.

In order to connect multiple sensors on the bodysuit, it is necessary to separate the data lines for each IMU sensor because the data line of the Arduino microcontroller is only one pin on the board. The multiplexer works by connecting pins on the Arduino to the multiplexer, which allows for the Arduino to be programmed to select which sensor to communicate with. The bits that select which sensor to connect to are called the selector bits. Figure 35 in Appendix-3 shows the schematic of the CD74HC4067 multiplexer used for the motion capture bodysuit.
Lithium Ion Battery

The entire body suit is powered by a two-cell 7.4V lithium ion battery. The Arduino microcontroller has a power supply port to allow for connection with a battery, which supplies power to the board as well as all the ports and pins connected to it. The team needed to ensure that the battery selected for the project would be able to supply enough power for an entire workout. By calculating the current draw of each IMU as well as the Arduino, it was determined that a 2100mAh battery would provide enough current to satisfy the necessary requirements. The dimensions of the battery are 4.06 x 1.38 inches.

XBee Wireless Communication - XBee ZB Series 2

XBee is a form of wireless communication that is very commonly used for microcontrollers such as the Arduino. The XBee runs at a frequency of 2.4GHz with an output power of 1.25/2 mW. One of the key features of XBee is its maximum range, which can vary anywhere between 100 meters to 4 km for the higher end products. The only downside of XBee communication could be its data rate, which can range anywhere from 20 kbps to 250 kbps, which sometimes limits its overall use. For the XBee communication to work between the Arduino and the external PC, an XBee shield is needed. The shield fits on top of the Arduino microcontroller by soldering header pins onto the shield.
Figure 14: Final Block Diagram

In figure 14, the final design of the motion capture bodysuit is shown. The five IMU sensors collect and send data to the Arduino microcontroller, and the Arduino sends that data through both the comparison software, as well as the XBee wireless connection. The comparison software analyzes the incoming data and determines which sensor is not experiencing correct running form. The corresponding LED will then light on the feedback controller, indicating to the user which part of their form is incorrect. The data being sent over the XBee wireless communication is then collected with Matlab post-processing software.
Hardware Design and Implementation

Figure 15: Feedback Controller and Strap

Feedback Controller

The feedback controller was designed with simplicity in mind. The goal was to create a feedback system that could provide accurate feedback, without giving too much overbearing information, which could cause difficulty in interpretation. The feedback controller consists of five, 2mm red LEDs that are set in an enclosure resembling a wrist-watch. The controller was completely custom-made, using pieces of G10 board and a mill. Enclosed inside are five 3.3KΩ resistors that act as current limiting resistors to protect the LEDs on the controller from drawing too much current. A port was drilled out the bottom of the controller to allow for wire connections to reach each individual LED. Six wires run out of the port, one for each of the five LEDs, as well as a singular ground wire to connect them all. The wires have quick connectors on them to allow for quick attachment to the body suit. The controller also has a Velcro strap which wraps around the user’s wrist and keeps the controller from moving while the product is in use. Velcro is attached to the back of the feedback controller and it attaches directly on the wrist Velcro strap.
Bodysuit

The team decided on using a body suit in order to provide the least amount of impairment possible for the user. Ribbon fibers were sewn onto the limbs of the body suit to provide connections between the IMU sensors and the Arduino microcontroller. The ribbon cables have four strands, one for ground, one for power, one for the data line, and one for the clock line. There are also four-pin connectors on the ends of each ribbon to provide quick connections between each part of the suit, allowing for easy connection and disconnection. Strips of Velcro were also sewn onto specific areas of the body suit to provide positions where the IMUs would be placed. The Velcro prevents the IMUs from falling off the suit, as well as providing a stable contact to minimize the amount of unnecessary shaking the IMUs endure while the product is in use.
Arduino (Pins and Connections)

The Arduino functions by providing power using either a battery source, power supply, or by powering through the USB port. The Arduino requires a USB connection to a PC in order to receive any software programming to run the Arduino. Analog pins four and five correspond to the data and clock lines respectively, which both set the clock initialization for the connected sensors, as well as collect the data from the sensor. The Arduino also allows for connections with attachments called shields, which can serve purposes such as SD card, wireless, and protoboard capabilities.

For the motion capture bodysuit, the Arduino has five sensors connected over the data and clock lines, as well as connections to control each of the five LEDs for the feedback controller. The Arduino also has an XBee wireless shield to send data over XBee communication to the Matlab post-processing code. Analog pins A4 and A5 serve as the data and clock lines respectively.
IMU (Pins and Connections)

The setup of the IMU required connections on the data, clock, power, and ground lines, which are labeled SDA, SCL, VCC, and GND respectively on the board. The connections were made by soldering headers to each of the IMUs to allow for quick attachment of each of the necessary pins. Figure 18 above shows the placement of the IMU on the bodysuit. The blue connector in the figure is the 4-pin connector that attaches to the ribbon cables on the bodysuit. Each of these can be quickly disconnected to allow for quick maintenance and debugging.
XBee and XBee Shield

The XBees that the team ordered were unable to directly connect to the Arduino Uno. To solve this problem, an XBee Shield was ordered to connect the devices. The XBee Shield is a simple adapter that allows the use of XBee communication with the Arduino microcontroller. The XBee Shield is a Printed Circuit Board (PCB) that mounts on top of the Arduino, allowing the use of XBees while not occupying any of the pins of the microcontroller. The XBee is mounted in the center of the XBee Shield and then needs to be programmed accordingly to ensure proper connection between both XBee devices.

Multiplexer

The multiplexer was setup with connections on channels C0 through C4, corresponding to the five IMUs connected in the design. The pins SIG, GND, and VCC connect to the data line, ground, and power respectively. Pins S3 through S0 correlate to the selector bits that are controlled with the Arduino software. The pins were varied between digital “HIGH” and “LOW” depending on which sensor was being read from in the data acquisition loop of the software.
Software Design and Implementation

Arduino

The Arduino microcontroller has its own programming software, based off the open source software called Processing. Its code is very similar to the language C, using a very similar syntax. In order to communicate with the IMU sensors, it was necessary to include both the MPU-9150 library as well as the I2C bus library to communicate over the necessary ports of the IMUs. The necessary requirements of the Arduino code included initializing each IMU sensor, selecting which sensors to read from using the multiplexer selector bits, as well initializing the XBee wireless connection to send the data to be post-processed.

IMU

The code necessary to initialize each IMU to read data was included in the MPU-9150 library. This code was needed to read the data coming in from each IMU as well as send it to the Arduino’s serial monitor in order for the team to verify the accuracy of the incoming sensor data.

XBee Arduino Software

To get the software portion of the XBee wireless communication working, it was necessary to program the Arduino to recognize not only that the XBee shield was connected to the microcontroller, but also to initialize the XBee communication.

Multiplexer

To get the multiplexer to work, the team needed to write code that would initialize the multiplexer, as well as set up the necessary pins on the Arduino to allow for the selector bits of the multiplexer to be changed. The multiplexer functioned by setting up the digital output pins of the Arduino and assigning those outputs to be either “High” or “Low.” There are a total of four bits that control which channel the multiplexer is communicating with, and the bits are set to allow for up to sixteen sensors to be connected simultaneously to the multiplexer. The team only needed five of these channels, so the bits were set up to read from a binary zero to a binary four otherwise known as “0000”, “0001”, “0010”, “0011”, and “0100”. The channels correlated to channels zero through four on the multiplexer respectively. The enable pin on the multiplexer was always set to a digital “low”.
Before an XBee can communicate and send data to the other Xbee device, both of the XBees need to be set-up and programmed. The setup process of the XBees begins with installing the X-CTU program provided by the manufacturer, Digi. After the installation process is completed, the XBees are ready to be set up. The X-CTU program allows the user to modify the baud rate (transmission rate), transmission mode, and many other features. An important aspect to remember is that the baud rate of both of the XBees and the programs that are being run need to be the same. This allows the data being sent to properly be recovered on the other end of the wireless communication. These features are chosen based on the function of the XBees, the microcontroller, and the overall application of the complete design.
MATLAB

The MATLAB code for the motion capture bodysuit serves as post-processing analysis to give a more in depth overview of the user’s workout. The code works by reading the serial port which is connected to the XBee explorer. The data is read from the XBee in a string, and is able to distinguish between all sensors. The data is then plotted according to each axis read from the serial port.

All code for both the Arduino and Matlab is located in Appendix-4 of the CD report.

Project Obstacles

Multiple IMUs

One of the major issues that the team encountered while working on the motion capture body suit was the implementation of multiple sensors. The ability to connect one IMU sensor to the Arduino microcontroller and read the data was relatively simple. The biggest problem was that each IMU was given a specified address that was not easily changed without adjusting the library code for the IMU. Each IMU also needed its own data line in order separate the incoming signals, so one universal data line could not be used.

The solution to this issue was to use a multiplexer to allow for multiple connections to the Arduino. The multiplexer allowed the team to connect each of the IMUs to a specific channel on the multiplexer which would determine which sensor to read data from. The multiplexer code was written with the Arduino software, initializing each IMU sensor, and then looping the data to read one iteration from each sensor. The selected sensor was specified using the selector bits of the multiplexer, which were connected to the digital outputs of the Arduino.
Comparison Software Delay

The team had initially wanted to use MATLAB as the primary means of comparing the incoming data from the IMU sensors. It was later realized that the data being sent over the XBee wireless connection had a delay ranging from 1 to 3 seconds, which would cause the resulting feedback to the controller to also receive the same amount of delay. This was a serious issue when trying to give real-time feedback to the user.

To solve this problem, the team decided on programming the comparison software for each individual sensor with the Arduino software. This would substantially decrease the amount of delay for the real-time feedback to the feedback controller. The team also decided to keep the MATLAB portion of the software intact and use it as post-processing software to give the user a more in-depth view of their running data.

Length of Ribbon Cables

A problem that was later encountered after the fabrication of the bodysuit was finished was the length of the ribbon cables between the Arduino and each IMU. After the suit was worn by a team member to do testing, it was realized that the ribbon cables were not long enough to reach the desired position on the back of the user. This was a very serious issue as it prevented the team from being able to fully test the functionality of the suit, as well as acquire the necessary data for the baseline comparison data.

The solution that resolved this was to create extension cables that would fit onto the blue quick-connects that had been attached to the end of each ribbon cable. Using custom-made headers and soldering extra wires to each pin, the team was able to create an extension connection to allow the connections of each IMU to reach the desired central focal point of the Arduino.
Enclosure for Motion Capture Circuit

It was necessary to create an enclosure to encompass all the connections between the multiplexer, the IMUs, and the Arduino to prevent any bouncing and disconnecting of any wires while the bodysuit was in use. Originally, the team had decided on using a rather compact pack normally worn by bikers to hold important objects such as phones, but it was later found out that the pack was not large enough to hold all the necessary parts of the bodysuit.

In order to fit all the parts of the circuit inside the pack, the team decided on cutting holes into the pack to allow for wires and connections to poke out through the enclosure. The wires that connected the Arduino, the multiplexer, and the battery were all held together using Velcro straps to keep the parts from moving around too much.
Comparison Software Accuracy and Post-Processing

One of the biggest issues that the team ran into while testing the bodysuit, was the acceleration range of the IMU sensors and the consistency of the incoming data. The default range of each IMU sensor is to measure acceleration from -2g to +2g. While the bodysuit was being tested on the runner, the data would peak at 2g multiple times, meaning that the sensors did not have enough range to encompass the max acceleration being read. The acceleration being read from each sensor was not only the acceleration of Earth’s gravity on the sensor, but also the acceleration that each sensor felt while the runner was hitting the ground during running.

The team was able to change the acceleration range of each sensor by changing the initial value in the IMU library code. This would allow each sensor to measure acceleration up to ±16g, which was more than enough to prevent the sensors from reaching the peak value.

One issue that could not be resolved was the creation of accurate comparison software while the bodysuit was in motion. In order to run the post-processing software of Matlab, the comparison code required a delay of at least 50 milliseconds. This delay means that the sensors were only reading data at a rate just over 3 Hz, which is significantly slower than what is necessary to accurately measure running motion. The team ended up deciding on writing comparison software for static sensors, to show the range of motion that would be allowed for a runner if the sensor data was completely accurate.
Chapter 4: Testing and Results

In order to test the functionality of the IMUs, the team used the Arduino Uno MPU-9150 library to verify the data acquisition of each IMU. Connections for power, ground, clock, and data needed to be made between the IMU and the Arduino. Using the Arduino software, the IMUs would output nine data values based on the specified serial monitor baud rate. The serial monitor shows if the initialization of each component is properly setup. This data is shown in the figure above, verifying a connection was successful between all five IMUs.

Battery Testing

It was necessary to calculate the current draw of all the devices connected to the bodysuit to ensure that the battery would provide sufficient power to all devices for an extended period of time. Based on the current draw of the IMU sensors (4.25mA), the draw of the feedback controller (10mA per LED), and the draw of the Arduino (50mA), the 2100mAh battery was more than sufficient to run for the desired time of 30 minutes. In fact it could run for much longer if necessary, which allowed the team to continue to test over longer periods of time.
Bodysuit Connections

The ribbon fibers of the body suit were very fragile and easy to break, which could cause serious implications if any connections were broken while in use. Using a multimeter, the continuity of each connection was tested for each of the four connectors in the ribbon fiber. Then each pin of the four-pin connector attached to each end of the five ribbon cables was also tested for continuity. This was done by using the multimeter to check the resistance level of each ribbon fiber, as there would be a huge resistance (open circuit) measured on the multimeter should no connection be present. The connections would then be made with the headers that were soldered onto the IMU sensors to allow for easy assembly.
Feedback Controller

Each LED first needed to be tested to ensure that they would light when placed in the controller. Using a power supply set at around 5V, the LEDs were tested individually to ensure that none were defective or burned out. Each LED also required a current limiting resistor to prevent too much current from flowing and burning up the resistor. The controller wires were then connected to the Arduino microcontroller and a simple code was written to test the LEDs. The code would run through each digital output corresponding to each LED and light it for half a second before turning it off. Figure 26 above shows which LEDs correlate to which body parts on the runner.

Final Prototype Testing

Using the comparison software written for the IMU sensors, the team adjusted the comparison to correspond to the baseline data collected for the project. The comparison was different for each sensor since every sensor was placed at a specific point on the body suit, which would cause different data values to be acquired for each sensor. The software was uploaded to the Arduino microcontroller, which would run once a power supply or battery source was connected.

The LED feedback controller was set up so that if any sensor was outside the specified range of the comparison software, then an LED corresponding to that sensor would light. While the sensors are running, data is also simultaneously being sent via wireless connection to a PC for post-processing.
The data for each individual sensor is stored in a Matlab matrix. The matrices then take the data and graph the values provided to indicate the angles, and acceleration of each sensor throughout the workout.

Using a team member as a tester, the body suit was worn to compare the values acquired from each sensor while running. Anytime the member moved the specific sensor out of the range of the comparison software, the LED would light on the controller. The feedback was extremely quick, allowing the user to correct their form almost immediately following a lit LED. The post-data processing provided an accurate representation of the user’s form throughout their workout.

**Results and Data**

| Sensor 1/RK time[µs]/a/g/n: | 5036160 -13236 -2712 -8328 -27 -22 16 8704 -26624 28415 |
| Sensor 2/PA time[µs]/a/g/n: | 5069050 8704 -26624 28415 -27 -22 16 8704 -26624 28415 |
| Sensor 3/LK time[µs]/a/g/n: | 5099152 -14616 -492 -6860 -14 -46 6 1024 19455 23807 |
| Sensor 4/LAtime[µs]/a/g/m: | 5132238 -15764 -3984 1718 3 -20 0 -2551 -17694 30207 |
| Sensor 5/TORtime[µs]/a/g/n: | 5165004 5516 3472 13213 0 0 -20 -22705 10752 4600 |
| Sensor 1/RK time[µs]/a/g/n: | 5197420 -13312 -2630 -8308 -22 -20 11 7424 -26368 29695 |
| Sensor 2/PA time[µs]/a/g/n: | 5230624 7424 -26368 29695 -22 -28 11 7424 -26368 29695 |
| Sensor 3/LK time[µs]/a/g/n: | 5261120 -14668 -524 -6680 -46 -44 -3 -257 18432 22783 |
| Sensor 4/LAtime[µs]/a/g/m: | 5299908 -15620 -4064 1868 -2 -12 8 -2049 -17152 30719 |
| Sensor 5/TORtime[µs]/a/g/n: | 5326756 5506 3560 15196 0 1 -25 -24065 11008 4352 |

**Figure 27: IMU Serial Monitor Data**

**Arduino Serial Monitor on Startup**

When the Arduino is turned on and the bodysuit is ready for use, the feedback controller will first light each LED up once to show that the system is powered. Then data processing will begin and the IMUs will begin measuring the position of each limb. The sensors had comparison data written specifically for each individual sensor, since the locations were unique on the bodysuit.

**Interpreting Serial Monitor Output Data**

From the output data shown above in Figure 27, each sensor is labeled based on its position on the bodysuit. The data from the serial monitor shows a timestamp for the incoming data, as well as raw values for each of the 9-axes of the sensor, which are the accelerometer, the gyroscope, and the magnetometer for the x, y, and z axes.
In order to understand the incoming data from the accelerometer portion of the sensors, it is necessary to first divide the raw values by the scale factor and then apply the Pythagorean Theorem which is shown above in Figure 28. The values each need to be divided by the scale factor, which in the team’s case, was 16384. By taking each variable as one of the axes of the accelerometer, it is possible to calculate the number of g’s being perceived by each sensor. For static sensors, the resulting value should be around 1g because it shows that each sensor is experiencing 1g’s worth of force, which should only be the force of Earth’s gravity on the sensors.

\[
\begin{align*}
\frac{a}{16384} & \quad \frac{b}{16384} & \quad \frac{c}{16384} \\
\end{align*}
\]

\# of g’s felt by sensor \(= \sqrt{a^2 + b^2 + c^2}\)

**Figure 28: Data Conversions for Raw Values**

**Output Data Conversions**

Figure 27 shows the output plots from the post-processing of the data acquisition for Sensor 5, which is the IMU placed on the torso of the bodysuit. The top three plots correlate to the accelerometer, the middle three are the gyroscope, and the last three are the magnetometer. The data shown was plotted during the test run of the team’s test runner. The values were taken...
based on the IMU range of 8 times the normal base sensor scale. This was done because the sensors would peak based on the default settings of the IMUs.

<table>
<thead>
<tr>
<th>Sensor 1</th>
<th>RK</th>
<th>az &lt; -10000</th>
<th>or</th>
<th>az &gt; 2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor 2</td>
<td>RA</td>
<td>ax &lt; -27000</td>
<td>or</td>
<td>ax &gt; -6000</td>
</tr>
<tr>
<td>Sensor 3</td>
<td>LK</td>
<td>ax &lt; -27000</td>
<td>or</td>
<td>ax &gt; -6000</td>
</tr>
<tr>
<td>Sensor 4</td>
<td>LA</td>
<td>az &lt; -12000</td>
<td>or</td>
<td>az &gt; 5000</td>
</tr>
<tr>
<td>Sensor 5</td>
<td>TOR</td>
<td>az &gt; 13000</td>
<td>or</td>
<td>az &lt; 3000</td>
</tr>
</tbody>
</table>

**Figure 30: Ranges for Each Sensor**

**Feedback Comparison Compromise**

Another issue involved the extra movement of the sensors. Unfortunately, the sensors would tend to shake more than desired while the bodysuit was worn, which meant that the sensors would produce inaccurate data results. With that in mind, the team decided to create the feedback so that it would light up for poor form if the sensor values were within the ranges specified in Figure 28. RK is the sensor placed above the right knee, RA is the sensor below the right knee, LK is the sensor above the left knee, LA is the sensor below the left knee, and TOR is the sensor placed on the back of the torso.

**Final Results and Findings**

After much testing and troubleshooting, it was determined that creating a feedback system for running motion was beyond the scope of what the team was capable in the time allotted. Set-backs in fabrication and testing, along with debugging and connectivity issues all added up over time. The end result was a bodysuit that could successfully measure the range of allowable motion for a runner when the sensors were static but not while the user was in motion.

Any quick motions by the user could sometimes set off the LED feedback controller as the sensors did not have a fast enough sampling rate based on the limitations of the Arduino microcontroller. In the bodysuit was capable of measuring data from each IMU sensor, comparing that data, as well as graphing the overall results using post-processing.
Future Improvements

More Robust Design
The overall design of the bodysuit is actually quite fragile. The ribbon cables that connect all the sensors to the Arduino microcontroller are easily frayed, which can cause issues with connectivity. The blue quick connects that are attached to the ends of each of the ribbon cables are also not exactly the right size for the ribbon cables, as the ribbon fibers in each cable are actually offset from the connector pins, which can cause connectivity issues as well should any of the cables be moved.

More Accurate Feedback Comparison
One of the biggest challenges of this project was the creation of accurate feedback comparison software. The post-processing code of the motion capture bodysuit requires that a delay be incorporated into the data acquisition code in order for the post-processing of Matlab to keep up with the sampling rate of the sensors. In turn this causes delays in the data acquisition from each IMU sensor and less accurate overall data collection. With a faster sampling rate to the Matlab post-processing software, it is much easier to write an accurate feedback comparison for the feedback controller because there’s much more data to view and make deductions from. One of the ways around this would be to use an SD card to log data from the Arduino. This would decrease the delay because there would be no wireless components involved in transmission of data.

Enhanced Visualization Software
Currently, the visualization software is built upon an open source model that only incorporates six axes of motion. Alternative software exists that can accurately depict motion using all nine axes from the IMU. Additionally, the current visualization software can only show the movement of one IMU. Future improvements would include displaying all five sensors at a time, as well as creating a 3d human model that moves each limb based on the IMU data.

Higher Processing Powered Microcontroller
It was determined that one of the main reasons the incoming sensor data was inconsistent was because of the processing power of the Arduino microcontroller. The Arduino has a rather low base read speed, much slower than what is feasibly necessary to create accurate motion capture. The trade-off is that a microcontroller with greater processing power and faster read speeds increases the prices substantially compared with the cost of the Arduino Uno.
## Chapter 5: Summation of Design Project

### Final Cost

<table>
<thead>
<tr>
<th>Item</th>
<th>Price</th>
<th>Amount</th>
<th>Totals</th>
<th>Supplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conductive Ribbon Connector</td>
<td>$1.50</td>
<td>16</td>
<td>$24.00</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>IMU MPU-9150</td>
<td>$49.99</td>
<td>5</td>
<td>$249.95</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>Arduino Due</td>
<td>$49.95</td>
<td>1</td>
<td>$49.95</td>
<td>Adafruit</td>
</tr>
<tr>
<td>Arduino UNO R3</td>
<td>$29.95</td>
<td>1</td>
<td>$29.95</td>
<td>Adafruit</td>
</tr>
<tr>
<td>Arduino Headers R3</td>
<td>$1.50</td>
<td>1</td>
<td>$1.50</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>Arduino Headers</td>
<td>$1.50</td>
<td>1</td>
<td>$1.50</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>Arduino UNO USB A-B Connector</td>
<td>$3.95</td>
<td>1</td>
<td>$3.95</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>Arduino Due USBB Cable</td>
<td>$5.95</td>
<td>1</td>
<td>$5.95</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>Conductive Ribbon Wires</td>
<td>$5.25</td>
<td>8</td>
<td>$42.00</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>Micro SD Shield (Only needed if wireless doesn’t work)</td>
<td>$14.95</td>
<td>1</td>
<td>$14.95</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>Class 10 SD Card (8Gb max)</td>
<td>$13.00</td>
<td>2</td>
<td>$26.00</td>
<td>Amazon</td>
</tr>
<tr>
<td>Arduino Wireless Shield</td>
<td>$84.00</td>
<td>1</td>
<td>$84.00</td>
<td>Amazon</td>
</tr>
<tr>
<td>1 XBee explorer dongle (Port Connection)</td>
<td>$24.95</td>
<td>1</td>
<td>$24.95</td>
<td>Maker Shed</td>
</tr>
<tr>
<td>1 XBee explorer USB (Cable Connection)</td>
<td>$24.95</td>
<td>1</td>
<td>$24.95</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>1 XBee Shield for Arduino</td>
<td>$24.95</td>
<td>1</td>
<td>$24.95</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>4 XBee Communication Devices (ZB Series 2)</td>
<td>$22.95</td>
<td>4</td>
<td>$91.80</td>
<td>Adafruit</td>
</tr>
<tr>
<td>Thunder Power RC 2100mAh 2-Cell 7.4V Lithium Battery</td>
<td>$29.99</td>
<td>2</td>
<td>$59.98</td>
<td>Amazon</td>
</tr>
<tr>
<td>6&quot; male 2.1mm plug to female</td>
<td>$6.49</td>
<td>1</td>
<td>$6.49</td>
<td>Amazon</td>
</tr>
<tr>
<td>Silicone Wire - Fine Strand - 24 Gauge- 60&quot;</td>
<td>$5.99</td>
<td>1</td>
<td>$5.99</td>
<td>Amazon</td>
</tr>
<tr>
<td>Force Sensitive Resistor 0.5&quot;</td>
<td>$6.95</td>
<td>6</td>
<td>$41.70</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>IMAX B6 Multifunction Battery Charger</td>
<td>$39.90</td>
<td>1</td>
<td>$39.90</td>
<td>Amazon</td>
</tr>
<tr>
<td>Compression Long Sleeve T-Shirt Tops by Under Armour</td>
<td>$34.90</td>
<td>1</td>
<td>$34.90</td>
<td>Amazon</td>
</tr>
<tr>
<td>SanDisk® microSDHC™ 8GB Memory Card</td>
<td>$8.09</td>
<td>2</td>
<td>$16.18</td>
<td>Amazon</td>
</tr>
<tr>
<td>Amphipod AirFlow Lite Waistpack</td>
<td>$20.99</td>
<td>1</td>
<td>$20.99</td>
<td>Amazon</td>
</tr>
<tr>
<td>10 pair JST plug connector RC Lipo Battery Male/Female</td>
<td>$5.99</td>
<td>1</td>
<td>$5.99</td>
<td>Amazon</td>
</tr>
<tr>
<td>W.S. Dean Micro 4B Plug. 6 pack</td>
<td>$10.20</td>
<td>1</td>
<td>$10.20</td>
<td>Tower Hobbies</td>
</tr>
<tr>
<td>Nike Dri-fit filament running tights, medium black</td>
<td>$38.48</td>
<td>1</td>
<td>$38.48</td>
<td>Amazon</td>
</tr>
<tr>
<td>Analog/Digital MUX Breakout</td>
<td>$4.95</td>
<td>1</td>
<td>$4.95</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>Arduino Stackable Header - 10 Pin</td>
<td>$0.50</td>
<td>5</td>
<td>$2.50</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>Totals</td>
<td></td>
<td></td>
<td>$988.60</td>
<td></td>
</tr>
</tbody>
</table>

### Figure 31: Total Parts List

The final budget is shown in figure 34. This list shows the complete list of all parts ordered for the completion of the bodysuit. The team was given a rather generous budget to work with, which allowed for extra components to be ordered should any complications arise. Some of the biggest expenditures included the costs of the Arduino microcontrollers, as well as the wireless Arduino shield, and the IMU MPU-9150s.
Summary

The purpose of this project was to create a way to improve the overall running efficiency of athletes through the method of motion capture. The project was completely open-ended, allowing for multiple avenues to be explored and giving the team the freedom to design the project as desired.

The way the motion capture bodysuit worked was by collecting data through IMU sensors that would give readings based all 9-axes of the sensors. The data is then sent through ribbon cables that were sewn onto the bodysuit to provide unobtrusive connections between devices. The ribbon cables were then connected to the multiplexer, which determined which sensor was being read from using the Arduino Uno software code. The Arduino served as the main processing unit, time stamping and collecting the data from the sensors and organizing incoming data. Data would then be run through the comparison software that was written in the Arduino software, as well as sent over XBee wireless communication to Matlab for post-processing.

The data that was run through the comparison code would then determine which sensors were out of range and light the according LED on the runner’s feedback controller. The data that was sent over the XBee communication was then collected with Matlab software and plotted based on the number of data points specified.

The team was able to create a motion capture bodysuit that could compare basic movements on any user wearing the bodysuit. Unfortunately accurately capturing data while the user was in any form of running motion caused the data to be inaccurate. This also meant that the comparison code for the LED feedback had to be written for static sensors, instead of while the user was in motion.

Conclusions

The team found this project to be quite challenging as it was necessary to come up with a unique solution. Due to the vagueness of the problem statement, the team had the freedom to approach the problem in many different ways. This became both an advantage and a disadvantage. One of the biggest issues that the team faced was the amount of software coding needed to accomplish this project. No team members were proficient with using Arduino or XBee software. A lot of research and trial runs were needed to properly write and verify the code, which took a lot of the team’s time. Setbacks included issues with programming, as well as unforeseen suit fabrication and functionality issues. Overall, the key goals that the team had established in the beginning were accomplished. Human motion can be captured, analyzed through post-processing software, and compared to a baseline model.
Appendix 1: Technical Roles

Blake Frantz - Suit Fabrication

My technical role for this project was the fabrication of the suit. It was necessary to ensure that all of the components of the suit were placed in a comfortable manner so as not to hinder the running form of the user. It was imperative for the sensors to collect accurate measurements, so the suit was tightly fitted to limit the extra movement that could potentially cause the sensors to receive inaccurate data. Velcro was sewn onto the various positions where the sensors were to be placed to allow for quick attachment of the sensors. The ribbon cables that were sewn onto the suit were an easy way to keep the bodysuit design as functional as possible as well as not cause any impedance for the runner. Each ribbon cable had a connector attached to each end to allow for quick connection of both the IMU sensors as well as to the connector block. Elastic bands were also sewn onto the arm of the upper bodysuit in order to guide wires to the feedback controller. The wires ran from the wrist of the bodysuit all the way up the shoulder of the suit to allow for the connections to be made at the enclosure on the back of the suit.

I was also in charge of creating the enclosure that would encompass all of the circuitry as well as the Arduino microcontroller, and the battery. It was necessary to customize the enclosure in order for all components to fit comfortably and to prevent any unnecessary movement of the circuit.

Lastly, the ribbon cables were sewn to the suit based on previous specifications, and when initial testing of the suit began, it was clear that a couple of the ribbon cables were about 4 inches too short to reach the connector block. I had to come up with a custom extension connector for it to reach the microcontroller.

Zhichao Lu - Testing and Data Analysis

I was responsible for the data acquisition and analysis of the data coming in from each IMU. By looking at the raw data shown in the serial monitor and looking at the data specification of the IMU, I could understand what the data points exactly meant. Since the output was in units of least significant bit (LSB) per g, it was required to convert them into understandable numbers. To determine the net acceleration felt by an IMU, it was necessary to apply the Pythagorean Theorem. The accelerometer data coming from each axis needed to be squared, and then added together. The square root of the resulting number was then taken, verifying the number of g’s being experienced by the sensor. The final result should normally end up around 1g as long as the IMU sensor was static.

I wrote code in MATLAB for post-processing analysis. It was necessary to use Matlab as post-processing software instead of comparison software due to the limitations of the serial ports that connected the XBee wireless communication. The Matlab code consisted of variables that were associated with each individual axis of the IMU sensor. The way the Arduino code was
written required that I use a string to interpret which data was associated with each sensor. In order to get consistent data though, it was necessary to add a delay in the data acquisition loop of the Arduino software in order for the Matlab processing to keep up with the sampling rate of the sensors. When run, the Matlab software would collect a number of samples from the 9-axis IMU sensors, and store them in separate matrices for each axis. The data was then plotted using Matlab in order to view the values acquired by each sensor. The plots would show the raw values acquired by each sensor, and by applying the scale factor of each axis, you could interpret the data and verify the number of g’s, the angular acceleration, as well as the heading for each IMU sensor.

**Alex Mazzoni - Comparison Software Developer and 3D Visualization Software**

I was responsible for developing the comparison software. After the baseline data was measured, it was necessary to see on each sensor which axes had the most movement on them. For example, since the upper back sensor had to only detect if the user was leaning too far forward or backward, only one axis was needed for that comparison. This was done for each sensor by doing different running trials and see what numbers had changed, and their range.

The default setting of each IMU is to measure a range of acceleration between -2g and 2g. A problem that occurred when testing was the saturation of data. This means that while running the IMUs would detect acceleration greater than 2g. To remedy this, I looked in the .h and .cpp library files to find where the accelerometer settings were. I found the setting and changed the range of acceleration to be between -16g and 16g. The gyroscope data was also saturating, so I changed the range of that as well. It was ±250 degrees/sec, and I changed that to ±1000 degrees/sec.

Another piece of the project that I put together was the visualization software. Online we found open source inertial measurement unit visualization software, based on software called Processing. This software took data from the serial port, analyzed the data, and created a 3d block that would move accordingly. This software worked by taking data from three accelerometer axes and three gyroscope axes. The Arduino code, however, was taking three accelerometer, three gyroscope, and three magnetometer values. Therefore, the code had to be modified to only print the first six values to the serial port. Additionally, the visualization software expected the serial information to come as a string with each value separated by a comma. I had to again change the Arduino code to reflect this specification.

**Nori Wilkins - Data Collection and Arduino Software and Feedback Controller**

I was responsible for collecting the baseline data to the system as well as work on the Arduino software. After the baseline data was collected from one of
the team members, I worked with Alex to look at this baseline data and
determine proper threshold values for the comparison software.

Since I had never used an Arduino microcontroller, I watched different
tutorials to do base coding. There was also a lot of support on the internet for
how to communicate between the IMUs and the Arduino.

There were many functions and libraries that were needed for the
communication between the IMUs and Arduino. All of the libraries were already
written and found through the Arduino community. One of the libraries that was
already written had established code to read the values of each axis from the
accelerometer and the gyroscope. The IMU that the team had decided on also
had a magnetometer embedded. Therefore, to read the magnetometer values,
the library had to be slightly modified. The function “Accelgyro.GetMotion9” with
the 9 corresponding axes allowed reading of every axis of the IMU.

Another aspect of my responsibilities was making the feedback controller.
Proper connections to the LED were crucial for them to work properly. An
enclosure was made as well so the only visible part of the feedback controller
was the 5 LEDs. The positions of the LEDs were based on the location of the
sensor. The central LED was allocated for the upper back sensor. By wearing the
controller, the left 2 LEDs corresponded to the upper and lower sensor on the left
leg. The same process was applied for the right leg. Working with Alex with the
comparison software, a testing process was established to ensure the LEDs
were lit when needed. The code was written on the Arduino software to light up
when the values went beyond the threshold of the baseline data.

Chenli Yuan - Wireless Communication Developer

I was in charge of the post-processing wireless communication. By using
the XBee and understanding the technology behind it, I was able to send data
from the Arduino to the external PC wirelessly. The data would be sent to
MATLAB for post processing. I was not aware of how the XBee worked, so first I
had to understand how it worked and download appropriate software programs.

There were many options on wireless communication technology that the
team could use. I had first researched specifically XBee and Wireless (WiFi).
WiFi was first technology to be explored. While working in the engineering
building, I encountered problems connecting to the engineering network. When
using the engineering network for the first time, the network requires you to
register your device by typing in your engineering ID. Since the Arduino did not
have a keyboard to directly register the device, I could not use the engineering
network for data transferring. A solution to this problem is to have your own
hotspot with no security on the network, but I had difficulty setting the external
PC as its own hotspot. Due to the limitations of the network connection, I found it
best to use XBee communication.

Before the final product was established, the team was first thinking of
doing real-time processing on the external PC, which required bidirectional
communication. This meant the XBee would have to transfer raw data from the
Arduino to the PC, and at the same time send appropriate feedback signals for
corresponding LEDs to light up. Code was first written to transfer data. Working
with Nori on the Arduino software, we were able to transfer data. At first, bidirectional communication was challenging. However, it was easily solved after finding the function for reading data directly off the XBee.

**Dan Zilinskas-Interfacing Technician**

I was in charge of the interfacing portion of the project. This means ensuring that all the components were wired correctly and interfacing with one another. The connections to all 5 IMUs and the Arduino contained many wires and connectors that needed to be used. The ribbon connectors on the suit had to be tested to ensure proper connection because they very fragile. Using a multimeter, I verified the continuity of each individual connection to ensure the suit had good connections.

I also was in charge of wiring and programming the multiplexer for applying the 5 IMU sensors to the bodysuit. As shown in our report, this was done by using a multiplexer to separate the data lines for each sensor. I had never used a multiplexer before so it was necessary to research and then test the multiplexer multiple times to ensure that it was working correctly. In order for the multiplexer to function, it was necessary to code using the Arduino software. The multiplexer was controlled with the selector bits on the board, which were connected to the digital output pins on the Arduino microcontroller. These bits would change based on the function that was called in the Arduino software, and therefore, change which sensor was being read from on the bodysuit.

Lastly, I was also in charge of creating the connector that would interface with the multiplexer, the Arduino, as well as the ribbon connectors for each sensor. I created this connector by using microcontroller headers and gluing them all together. I needed four headers to make the connector block, as the bodysuit had 4 pins being used from each sensor. The clock line, ground line, and power line were all shared between the 5 sensors, but the data line needed to be controlled by the multiplexer, which meant separate wires needed to be attached to pins corresponding to the data lines of the IMUs. The end result block was able to connect using the blue quick-connectors of the bodysuit, allowing for easy connecting and disconnecting of the suit.

![Team Photo: (Left to Right) Chenli Yuan, Alex Mazzoni, Zhichao Lu, Nori Wilkins, Blake Frantz, Dan Zilinskas]
Appendix – 2 - References

Arduino Uno Home Page
http://arduino.cc/en/Main/arduinoBoardUno

Arduino Tutorials
http://www.jeremyblum.com/category/arduino-tutorials/

I2C Tutorial
http://www.robot-electronics.co.uk/acatalog/I2C_Tutorial.html

MPU-9150 Data sheet

Visualization Software
http://www.varesano.net/blog/fabio/my-first-6-dof-imu-sensors-fusion-implementation-adxl345-itg3200-arduino-and-processing

Sample IMU Arduino Code

Arduino Website
http://www.arduino.cc/

Processing Website
http://www.processing.org/

Microsoft Kinect- Optical Sensor Example
http://electronics.howstuffworks.com/microsoft-kinect.htm

Inertial Measurement Unit Information
http://celebrating200years.noaa.gov/visions/remote_sensing/imu.html

Measurement of Human Motion
Appendix 3 - Component Schematics

Feedback Controller Schematic

Figure 32: Feedback Controller Circuit Schematic
Arduino Uno Schematic

Figure 33: Arduino Uno Schematic
Figure 34: IMU MPU-9150 Schematic

#2.375V < UCC < 5.465V

*The LSB bit of the 7-bit address is determined by the logic level on pin A0B.
  Default address = 0x08 (pin A0B is logic low)
*Alternative address 0x09 (pin A0B is logic high)
**Optional external reference clock input or synchronization digital input
  Connected to GND by default. Cut trace for external control
Figure 35: CD74HC4067 Multiplexer Schematic
Appendix – 4 – House of Quality

Figure 36: House of Quality