WAG
Wearable Action Guidance System

A Major Qualifying Project

Submitted to the faculty of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

April 25, 2016

team suitup

Amanda A. Adkins	aaadkins@wpi.edu
William S. Barnard	wsbarnard@wpi.edu
Matthew J. Beardsley	mjbeardsley@wpi.edu
Charles J. Frick	cjfrick@wpi.edu
Samantha L. Swartz	slswartz@wpi.edu

suitup-all@wpi.edu

Project Advisors:

Professor Michael A. Gennert

Professor Fred J. Looft

Professor Hugh C. Lauer

This page was intentionally left blank

Abstract

The Wearable Action Guidance (WAG) System is a training tool designed to improve the efficiency and convenience of teaching and learning new physical skills, while matching or exceeding the quality of feedback received from an in-person trainer. The system consists of a computer application and a set of wearable bands; each band comprises a 3D printed case, an inertial measurement unit, a battery, a ring of vibration motors, and a secure strap. Trainers can use the WAG System to record and save motions for distribution to trainees, while Trainees can use the system to play back those motions with directed vibratory feedback. Initial prototypes have attracted potential partners interested in introducing the technology to markets including athletic training and physical rehabilitation.

Table of Contents

Table of Figures

Table of Tables

This page was intentionally left blank

Challenge Definition

Over a lifetime, an individual will occasionally need to undergo training – for a new job, a hobby, recovery from an injury, etc. With current technology, this process can be inefficient and time consuming, often involving doing a task wrong multiple times before learning to do it right. Imagine T'ai Chi students learning a new form. The master must first demonstrate the form being taught. The students can then try to replicate this, but usually need some personal guidance. The master may need to physically and verbally instruct each student to correct the student's mistakes. While this process may work, it is not very efficient, and demands lots of personal attention, which is hard to give to a room full of students.

Industrial operations provide very different teaching environments from T'ai Chi. In this setting, instead of improving athletic ability, teaching is used to explain safe techniques for hazardous or complex physical activities. Currently, the most common ways to train individuals are through videos or directly with instructors, followed in both cases by extensive practice. Unfortunately, neither approach can guarantee that the action is being practiced correctly – something that can have serious consequences in the future.

What if there was a device that provided feedback about how far the user was straying from the desired motion? In the first example above, such a system could record one user's sweeping movements and later guide another user through the action. In the work environment, the device could be used to ensure that the employee was correctly practicing safe techniques. This device could also catch persistent errors more frequently than a trainer.

Currently, motion capture devices (used in the film industry) and haptic, or tactile, feedback devices (often used for immersive gaming) are commercially available. The few devices that combine wearable motion capture and tactile feedback capabilities are currently in research and development and are primarily designed for physical rehabilitation or simple demonstrations of matching single poses, rather than complex motions. This leaves a need for such a device to be developed for the purpose of training someone in a new physical skill.

A system capable of capturing a motion and conveying corrective feedback during practice, has the potential to be used in markets such as physical therapy and rehabilitation, athletic training, and military conditioning. In each of these fields, the technology would help to prevent injuries as a result of incorrectly executed physical actions.

Project Statement

The goal of this project is to create a training tool that makes learning a new physical skill easier. The solution is a Wearable Action Guidance (WAG) System consisting of seven wearable bands (the "Suit") and an accompanying computer application. This system helps to increase user autonomy and independence for learning a physical motion, while attempting to match the quality of feedback from in-person trainer. To accomplish this the WAG System uses motion tracking and haptic feedback. Using haptic feedback to indicate error in the user's motion, the system provides targeted real-time instructions to the user. With real-time feedback, users could immediately correct their errors and learn much faster as a result. As a complement to the WAG Band hardware, an easy-to-use graphical user interface (GUI) allows the user to find, download, and learn new motion and activities. By combining motion tracking and haptic feedback, the WAG System could make athletic training and physical therapy faster and easier.

Customer Value Proposition

Traditionally, to learn a new motion, an individual will hire a personal trainer or watch training videos online. The trainer will be paid hourly to coach the trainee through the motion. The training sessions will be spread out over days or weeks, during which time the trainer will keep checking in to make sure the trainee is performing the action correctly. In this scenario, the trainer is a significant expense and the trainee is entirely dependent on the trainer's schedule, location, and ability to teach. In addition, the feedback given by the trainer is limited to the errors that s/he can visually detect during practice. A trainer would likely miss small errors, particularly at the beginning of the training when the user is more error prone due to lack of practice and experience. If the trainer misses these errors, the trainee may learn the motion incorrectly, resulting in injury or more training to fix the problem later. A trainer also may not consistently point out persistent errors that the trainee should address.

To learn a new motion using the WAG System, Trainees will purchase the system, choose the motion file they want to learn from a motion library, put on and calibrate the WAG System, and play back the motion training file. Trainees will then practice the motion by playing back the file numerous times over the course of days or weeks. Using the WAG System, Trainees will make a one-time payment for the system, and (possibly) a one-time payment for the motion file. Trainees can then learn at their own pace, can practice (and be corrected) according to their own schedule, and could become as accurate as the motion file they are using to practice.

As these examples illustrate, once a WAG System has been purchased, it is thereafter completely reprogrammable for any desired motion or activity. It is possible that trainers would charge a fee to purchase a motion file but that cost will likely be significantly less than the cost of a personal trainer. In addition, the WAG System, once purchased, will always be available to the Trainee, unlike a trainer who may be too busy to meet or not easily accessible due to weather or lack of transportation. The WAG System has the potential to provide higher quality training since it is designed to detect real-time deviations through accurate sensor readings rather than waiting for a trainer to notice an error. The sensor readings will also notice much more subtle deviations than a human typically would. Overall, the WAG System has the potential to be more effective, efficient, and cost effective than traditional training methods.

Background Research

This chapter presents an overview of the current research for a number of relevant technologies and is used as a basis for the WAG System design and development.

Exoskeleton Applications

In 2013, two different research groups published papers describing separate exoskeletal-like systems designed to assist the user's walking gait. The exosuits, which were not rigid and did not impede movement, are of interest to this project because they focused on creating an exoskeleton with actuation while minimizing the strain and impedance on the body. Both papers discussed two important design considerations: matching the human's degrees of freedom and keeping the weight of the suit relatively low. The first paper, written by Wehner *et al,* is titled *A Lightweight Soft Exosuit for Gait Assistance* [1] and the second, written by Ashbek *et al,* is titled *Biologically-Inspired Soft Exosuit* [2]. Pictures of the designs can be seen in Figure 1 and Figure 2, respectively.

Wehner analyzed which parts of the body move the least during normal movements and used those points to help assert torque to the joints. In contrast, Ashbek explained a design of fabric "webbing" that used the body's movement to create moments around the various joints that assisted the user. Both had unique and interesting designs for a lightweight exosuit used on the legs.

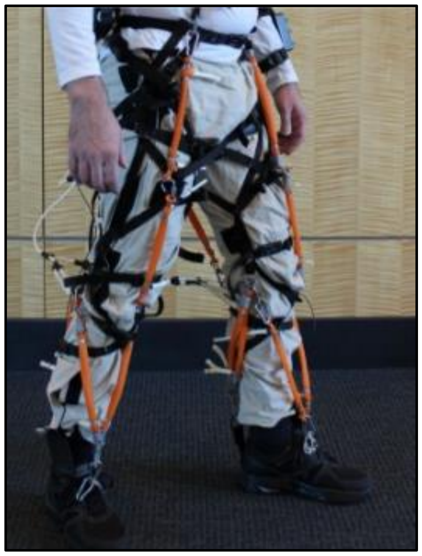


Figure 1. A light exoskeleton for gait assistance [1]

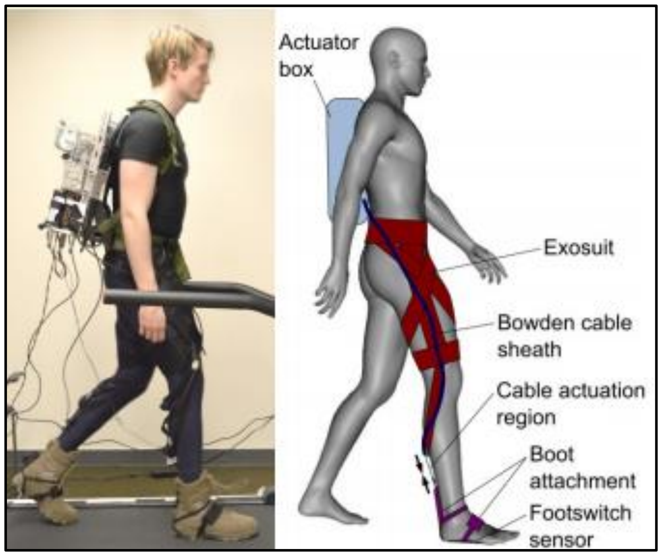


Figure 2. A biologically inspired soft exosuit [2]

Common areas for research in rehabilitation exoskeletons focus on the arms and hands. The EMG[1]-driven Exoskeleton Hand, presented by Ho, et al, at IEEE's International Conference on Rehabilitation Robotics, was developed to help stroke patients recover the use of their hands [3]. The system has a linear actuator for each finger along with a motor control box and an external, wireless remote control

[1] *Electromyographic* signals. EMG signals are those formed by the body such as heart beats or other electrical nerve signals controlling muscles.

system. This device uses EMG signals from the wearer to determine the desired action and guides the hand through that action. Another example of a rehabilitating exoskeleton hand was described in a paper by Cahn, *et al.* entitled *Finger Grip Rehabilitation Using Exoskeleton with Grip Force Feedback* [4]. This system, shown in Figure 3, consists of two exoskeletons, a slave and master, which incorporate two different modes of operation. The first mode allows the user to control the slave system through the movements of the master system. The second mode allows the user to record a movement using the master system and then play it back again using assistance from actuators.

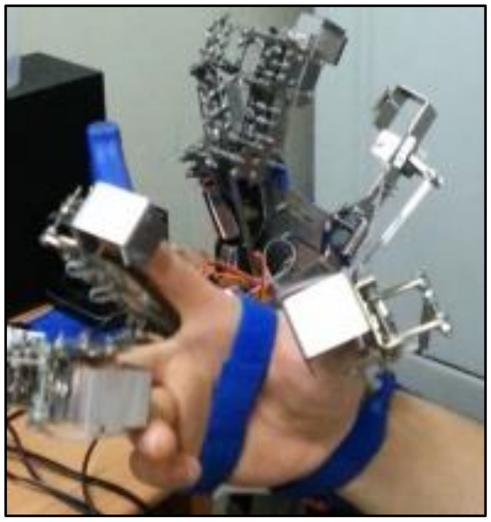


Figure 3. The master of a master/slave wearable robot system [4]

Wearables

Outside the realm of rehabilitation, Dexta Robotics created an exoskeletal glove called Dexta F2 (Figure 4) that allows the wearer to either interact with virtual objects or control a robot [5]. The device uses rotational sensors to track movement and has force feedback units on the index finger and thumb. A signal can be sent to the glove to lock the feedback unit in place and not allow those fingers to close any further. This system is of interest to this project because of its simplicity and low cost.

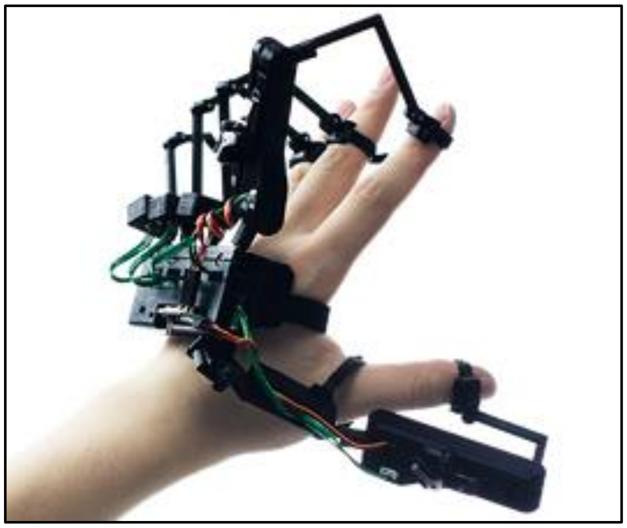


Figure 4. The Dexta F2 glove [5]

The realm of haptic feedback as applied to wearable exoskeletons has been growing rapidly in recent years due to interest in immersive gaming. These systems range from small components like the gloves created by CyberGlove Systems [6] (Figure 5) and NeuroDigital Technologies [7] to full body suits like those developed by PrioVR [10] (Figure 6) and Tesla Studios [8] (Figure 7). Some of these systems, such as KOR-FX [9], provide only haptic feedback in the form of vibration or force feedback, while others, like PrioVR [10], also provide game control. Most of these systems are either under development or for commercial use, so limited information about the specific technology or software is publically available.

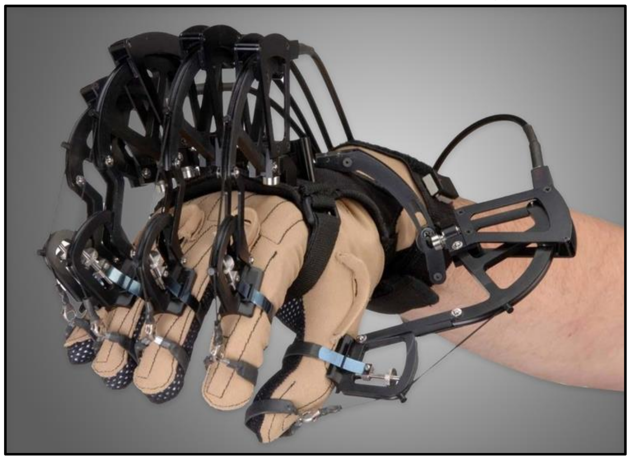


Figure 5. Cyberglove glove with force feedback [6]

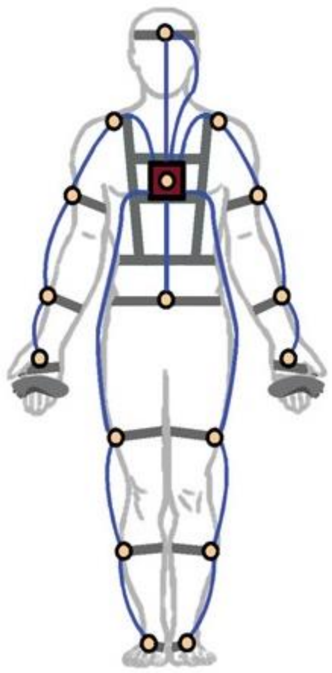


Figure 6. PrioVR motion capture and game control system [10]

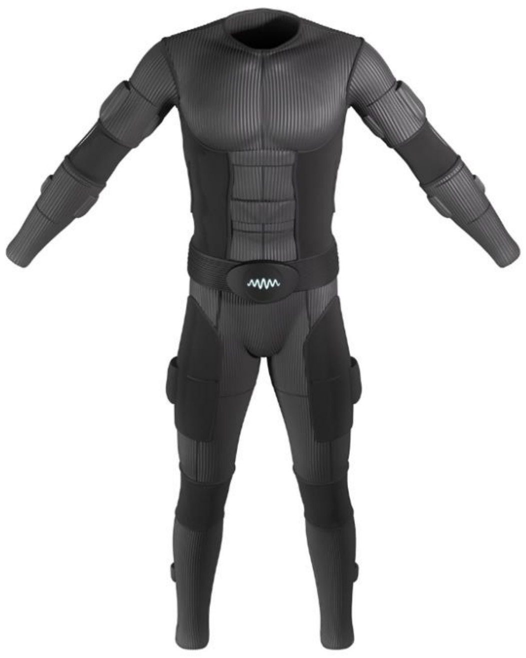


Figure 7. Tesla Suit for in game motion capture and haptic feedback [8]

An example of an open source wearable robotic controller was described by Cele, Ybes, *et al* [11]. The goal is to translate human motion into actions mimicked by the robot. This project uses a wearable sensor array consisting of eight sensors – six linear potentiometers on the legs and hips and two accelerometers on the arms. Data is transferred directly to the robot from the sensors over the ZigBee communication protocol. A fuzzy logic component filters the signals and converts them into usable drive values for the motors. This project is useful because it describes exactly how user motions were tracked, smoothed, and translated into drive values.

Motion Capture

Motion Capture Systems

There are currently two main types of motion capture systems: optical and mechanical. Optical recording uses multiple cameras to detect either markers on a user's body or features in the images. This type of motion capture is typically very accurate and capable of fast update rates. Although, these systems have a limited range of applications due to their cost, lack of portability, and time required for processing image data, they are commonly used in the animation and film industries. Optical motion capture systems can cost $50,000 for systems with low capture area and over $1 million for more precise systems [12], while image-based methods, which use computer vision techniques to discern motion without wearing special markers, are both less expensive and less accurate.

The second type of motion capture, called a mechanical system, uses an "exoskeleton" of sensors on the user's body. These sensors can include accelerometers, gyroscopes, and magnetometers, or potentiometers and flex sensors, to determine location in 3D space. These systems are more portable than optical or image-based systems, but must have many degrees of freedom, fit properly, and enable smooth movement to avoid impeding movement. Magnetic systems detect the position of different locations on the body using magnetic fields, either generated by the Earth or specifically for the motion capture system. They are fairly accurate and fast, but cost more than accelerometers or gyroscopes, consume higher amounts of power, and are more sensitive to metallic objects in the environment. Inertial motion capture systems use accelerometers and gyroscopes to estimate movements but these sensors are prone to drift over time, making them less accurate. These different types of systems are often combined to improve performance. For example, Xsens produces both individual sensors and full suits using gyroscopes, accelerometers, and magnetometers; however they cost up to $12,200 for the hardware components alone [13].

Motion Capture Software

Although many motion capture companies develop proprietary software to accompany their hardware, independent software, such as Autodesk MotionBuilder [14], and open source packages, such as OpenMoCap [15], are also available. Proprietary software is often expensive – a one-year subscription to Xsens's motion capture software can range from $5,400 to $9,500 [16] – and lack of support and functionality could make the open source options undesirable. Many motion capture applications are also designed for animation or film rather than motion analysis or training, making their architecture not suited for playback.

An important component to motion capture software is how the motion data is stored. One encoding method is to maintain all raw sensor data [12, 17], while another contains representations of points in space with respect to a global coordinate frame. The latter, called translational files, allows for more complex analysis of the motion capture data. A third category of file encodings represents data based on segments or limbs rather than free floating points in space. Files of this type, referred to as rotational files, contain information about the rotation of segments and are typically easier to use because the work of relating points to a skeleton has been done. Many companies have special file formats for use within their motion capture software, but there are several standards commonly used within the industry, including **.C3D**, **.bvh**, **.amc**, **.asf**, and **.fbx** file formats [19, 12].

Databases for captured motion are currently available on the internet, some of which support the file formats listed above. Academically supported databases include the CMU Graphics Lab *Motion Capture Database* [18], while motion capture community databases include the Motion Capture Society's *Motion Capture Library* [19]. Many of these allow free download of recorded content, but do not support sharing of original content.

Position Determination

Motion capture systems must be able to interpret human position based on sensor readings. The method for determining a user's position depends on the type of system used. Mechanical systems, which use sensors to directly measure joint angles, determine the user's position by calculating joint angles from sensor data. After this, kinematics can be used to determine the location of a point along a body segment. Multiplying transformation matrices containing joint angles and body segment lengths results in position information. This information is then as useful as the output of accelerometers or gyroscopes.

Inertial sensors are also used in motion capture and give information about the sensor position rather than joint angles. Unfortunately, inertial sensors tend to have significant error and drift over time. Therefore, techniques are needed to combine and smooth the sensor data. These are commonly referred to as *sensor fusion techniques*. Two well-known examples are Kalman filters and alpha-beta filters. Kalman filtering combines control inputs with sensor readings in a two-step recursive process that yields an estimate of the resulting state or position. Alpha-beta filtering works similarly to Kalman filtering. Although it is slightly less accurate than Kalman filtering, it is less computationally expensive and does not require a model of the system [20]. Some companies develop custom sensor fusion algorithms for specific sensors or applications. For example, MTi 1-series IMU from Xsens features its own sensor fusion algorithm [34].

Feedback

Some sort of physical feedback is necessary to convey how the user's body position varies from the target position. Two ways to achieve this are *force feedback*, and *haptic feedback*.

Force Feedback

Force feedback is the simulation of physical forces such that a person can interact with and "touch" virtual objects and experience a force pushing back from an object [21]. When applied to an exoskeleton, force feedback can simulate an environment or motion paths. Because force feedback systems actually apply forces to the user, they are very helpful for applications like robotic surgery, where visual feedback alone does not provided the surgeon with enough tactile information.

Haptic Feedback

Haptic feedback uses the sense of touch without actually pushing on the person. There are several forms of haptic feedback including cutaneous, kinesthetic, vibratory, or tactile [22]. *Cutaneous feedback* and *tactile feedback* both involve direct skin contact; the difference is that *cutaneous* is typically on the arms while *tactile* is typically on the fingertips. *Kinesthetic feedback* involves guidance of a limb or digit through external structures. *Vibratory feedback* is the easiest and cheapest to implement, and can be found in many common devices, such as cellphones. The two prominent types of haptic vibrators are *linear resonant actuators* (LRAs), as seen in Figure *8*, and *electric rotary vibrators* (ERVs), shown in Figure 9. Both are very small and lightweight, making them ideal for wearable applications. The tradeoff, is that

LRAs can be more efficient than ERVs, but they cost more and require special driver circuitry to control the vibrations [23].

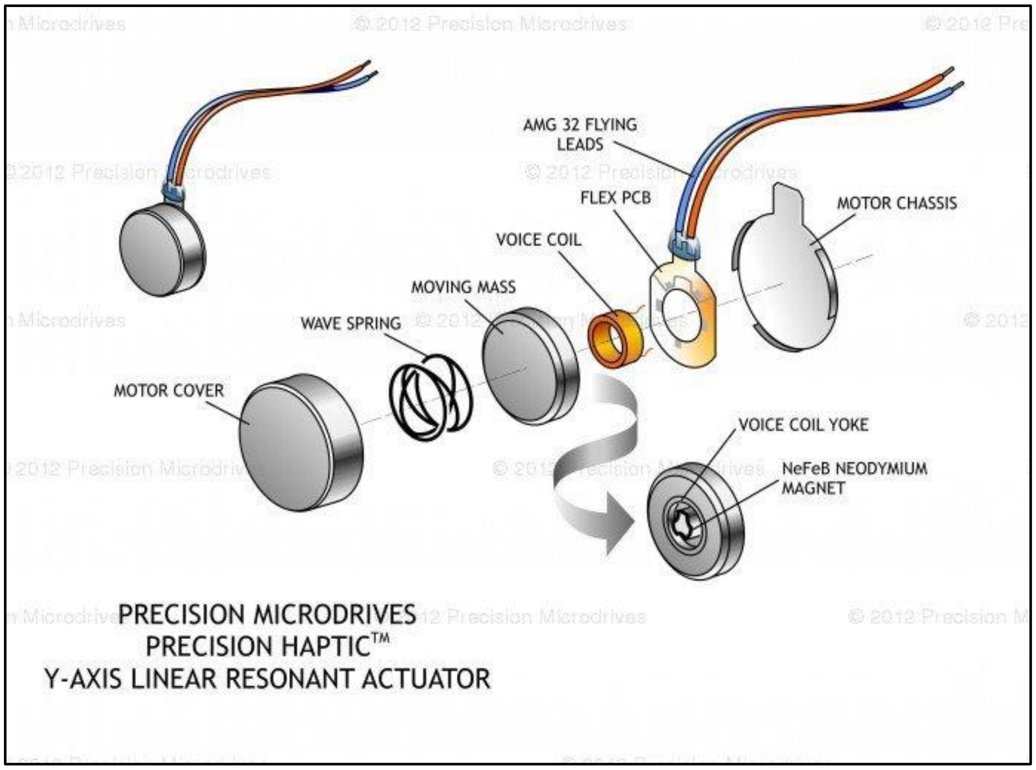


Figure 8. Basic principle of a Linear Resonant Actuator (LRA) [24]

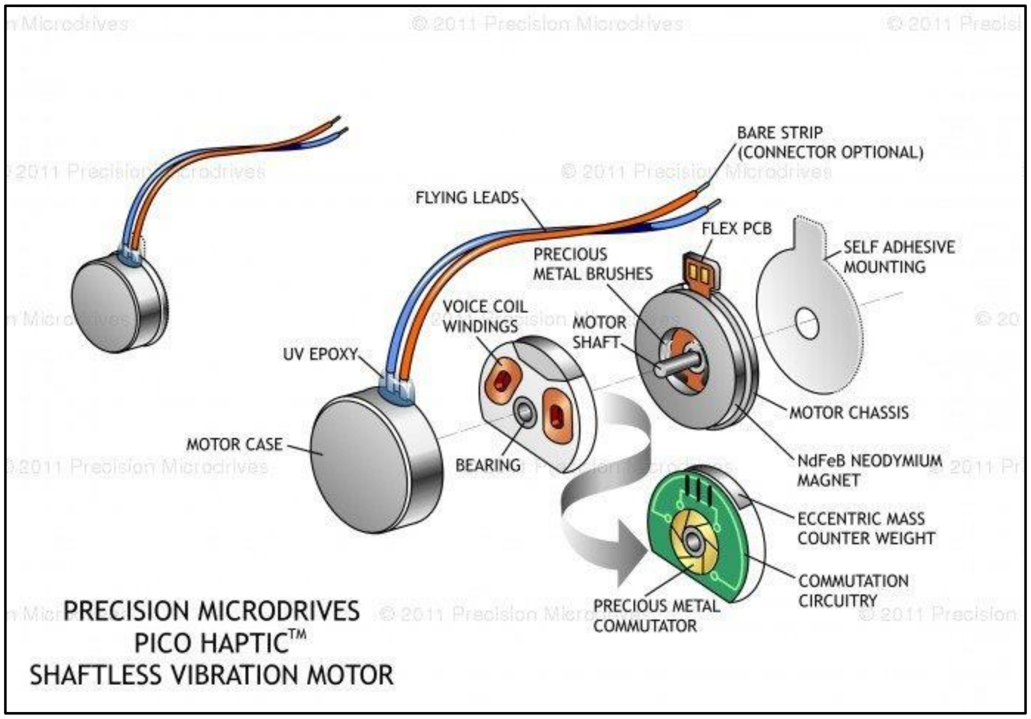


Figure 9. Basic operation of an Electric Rotor Vibrator (ERV) [25]

Sensor Types

The main functionality of this project rests in the accuracy of the sensors and their ability to represent the world. This section describes several types of sensors and how they can be used to capture various types of motion for applications in this project.

Accelerometers

Accelerometers are devices designed to measure acceleration along one or more axes at a specific point. Several types of accelerometers are suitable for this project. *Piezoelectric accelerometers* use the movement of piezoelectric crystals[2] mounted to a moving mass to generate a voltage related to acceleration [27]. *Piezoresistive accelerometers* use a similar moving feature to the capacitive accelerometers, but the movement of the feature changes the device's resistance [26]. This resistance change can be converted to a voltage and digitized. *Hall Effect accelerometers* measure motion using a changing magnetic field [27]. *Magnetoresistive accelerometers* use a magnetic field to change resistivity of a material and ultimately cause a noticeable change in voltage when placed in a voltage divider circuit. *Capacitive accelerometers* measure a change in capacitance caused by the movement of small features within the device [26].

The most commonly available type of accelerometer is a capacitive accelerometer which is classified under the Micro Electro Mechanical (MEMS) category of accelerometers [27]. Figure 10 illustrates the principle of operation of a MEMS accelerometer.

[2] Piezoelectric crystals generate electricity based on their deflection from an equilibrium position

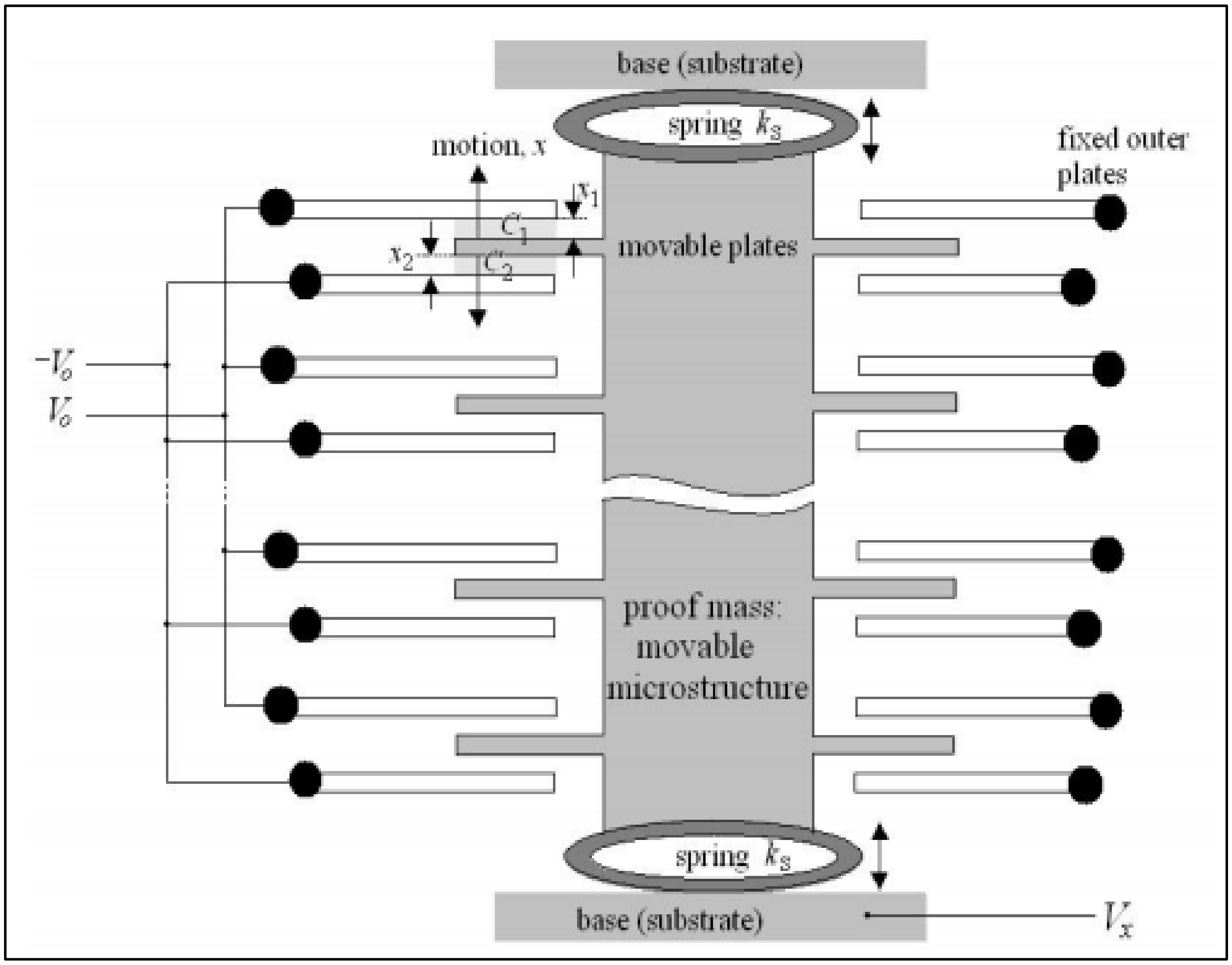


Figure 10. Basic outline of a capacitive MEMS accelerometer [27]

To sense acceleration, the movable plates shown in Figure 10 translate along the axis of the sensor. As the plates move, the capacitance between the fixed plates (C1 and C2 in the top left of the figure) change. When a reference voltage is applied, the changes in the capacitance create a varying analog output voltage that can be converted into a usable digital reading [27].

In order to determine a sensor's suitability to an application, it is important to consider its cost, supply voltage, sensitivity[3], range of the readings, and if it is ratiometric[4] [27]. The range of readings is the range of possible accelerations that the sensor can accurately measure. The range of these readings is from 0 up to 16g (16 * 9.8 m/s^2).

For proper functionality, users should be careful not to induce large accelerations/decelerations usually caused by dropping the devices as this can damage the unit. Users also must mount accelerometers at the proper angle, as this can have a drastic effect on sensitivity [27]. Some typical applications for accelerometers are measuring tilt/roll, vibration, accelerations along various axes, and position and speed. The cost of an accelerometer also can range from several dollars to hundreds of dollars based on the precision needed and the output features of the accelerometers.

[3] Sensitivity indicates how quickly the output of the sensor changes with respect to the input voltage and is measured in Volts/g for accelerometers and mV/^{0} for gyros

[4] Ratiometric sensors are those that scale the output voltage proportionally with change supply voltage

Gyroscopes

Gyroscopes, also known as gyros, today measure the angular velocity of an object rotating around a point. Most gyros are designed using MEMS technology (see Figure 10) and in contrast to MEMS accelerometers, measure angular rotation linear motion. Inside a MEMS gyro, a mass moves producing a change in the MEMS device's capacitance and a measurable voltage (see Figure 10); this is the output of the gyroscope [28]. The output may be either digital or analog and can be used to tell how quickly and what direction the center axis of the gyroscope sensor is moving around the axis. As such, the gyro has important characteristics for this project. Similar to the accelerometer, the gyro has a sensor resolution and an output sensitivity[3] [28]. The cost of gyros is comparable to most accelerometers. The limitation to this technology is the lack of accuracy due to noise from small vibrations.

Instead of using mechanical motion to measure rotational speed, some types of gyros use the Sagnac Effect for measurement. The Sagnac effect results after two beams of light are sent around a path and then measured at a detector. This principle is demonstrated in Figure 11. If the device is not moving, the beams will travel equal distances and reach the detector at the same time. If the device is moving, however, one of the paths of the light will be shorter and the interference pattern between the different paths can be measured and related to the direction of rotation [29].

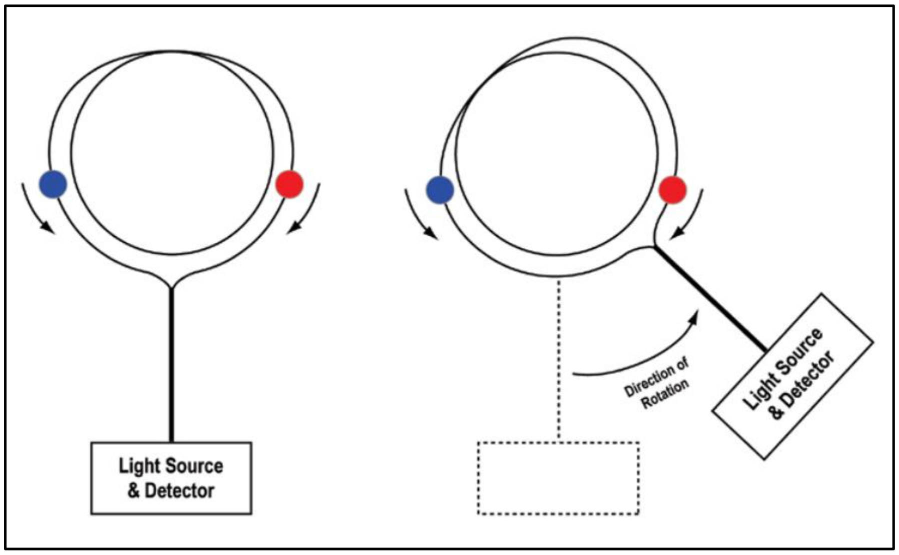


Figure 11. Basic principle of operation of a gyro with the Sagnac Effect [29]

One type of gyro that uses this effect and fiber optic technology is called a ring laser gyro (RLG). RLGs use a single laser beam that is split and reflected around a glass chamber containing inert gases such as helium and neon. Fiber Optic Gyros (FOGs), another type of gyro, use a beam of light produced from a laser diode or a photodiode, but instead of a glass chamber, light passes through a single fiber optic cable. The two basic categories of FOGs are closed loop, which use an optical chip, and open loop, which use a piezo-electric modulator to modulate the light in the circuit. The open loop tend to be much cheaper because of the cost associated with the integrated optical chip. A commercial example of a FOG is the DSP-1750 FOG produced by KVH Industries [29].

Two main advantages of FOGs over MEMS gyros are:

1. FOGs are immune to vibration noise as they track interference of light patterns instead of changing mass positions

2. FOGs tend to be more accurate due to the precision of the components and the resistance to changes in temperature.

However, FOGs are typically much more costly, and as such MEMS gyros are preferred for this project [30].

Bias or error in measurements of gyro systems tends to accumulate over time. This error is also influenced by temperature but can be reduced by using lookup tables for various temperature ranges [30]. Figure 12 below shows a comparison of different gyro systems and the error that builds up over time. The scale factor corresponds to how much the output reading differs from the actual velocity. Overall, the most accurate sensors tend to be in the bottom left corner of Figure 12, but these are also the largest and most costly. Low-cost calibration techniques, which usually involve data collection of gyro readings followed by software offsets, can reduce accumulated error in gyroscopes [31].

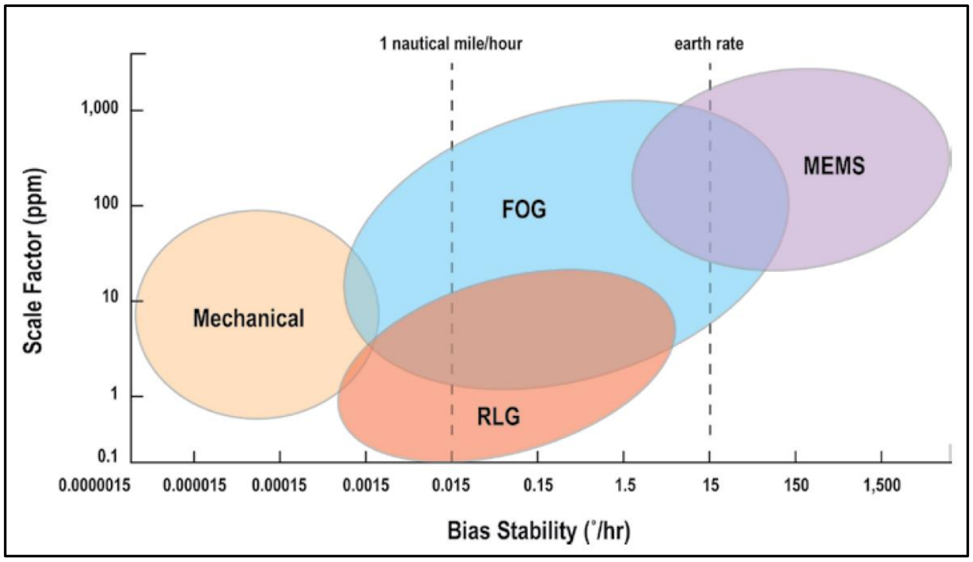


Figure 12. Chart describing the bias of gyros over time [32]

Inertial Measurement Units (IMUs)

An inertial measurement unit (IMU) consists of a collection of gyros, accelerometers and magnetometers to make a system for tracking position. One advantage is that IMUs tend to have an overall lower chip footprint than the component sensors combined. Calibration of the IMU is required to reduce the effect of the bias from gyros and accelerometers [33]. Some IMUs have microcontrollers (MCUs) that can combine gyroscope and accelerometer data to useful values such as position angles and quaternions using sensor fusion algorithms [34]. IMUs tend to be more expensive than single sensors, but are more accurate than the individual sensors.

Communication Systems for Sensors

In 2007, Lee *et al* evaluated the features and uses of various radio frequency (RF) communications protocols, Wi-Fi, Bluetooth, ZigBee, and ultra-wideband (UWB), with the desire to help engineers choose an appropriate protocol for their system communications [35]. All of these technologies operate around 2.4GHz in the unlicensed radio spectrum band for industrial, commercial and medical (ICM) purposes [35]. These protocols also revolve around the creation of a wireless personal area network (WPAN), which acts as the realm of information transfer between devices using these communications [35]. While the paper discusses many of the specific power consumptions, security strategies, transmission efficiencies, and data rate limits of the various protocols, the main theme is specifying engineering factors key to choosing the best communication system. For low data rates, small data sizes (less than about a hundred bytes of information), ease of use and power consumption, the ZigBee protocol tends to dominate [35]. A benefit of ZigBee over the other protocols is the number of sensors that can be added to a single ZigBee network. Zigbee can support approximately 65000 sensors per network while other protocols usually support fewer than 10. ZigBee also has the added benefit of separating its networks by a specific channel number. The channel number allows for smaller subnetworks of devices to be formed.

Wi-Fi protocols are capable of send much more data than ZigBee. With the recent innovations in Wi-Fi technology, the cost of connecting devices to the internet has dramatically decreased, resulting in a huge boom in online-integrated technology known as the Internet of Things (IoT). Examples of these devices include wearable fitness bands, smart meters for real-time energy monitoring in homes, and mobile-connected thermostats and home automation systems produced by Google, IBM and GE [36]. However, one of the big concerns with all new devices is the security of data transmissions between devices and the Internet. Currently, no well-defined standards exist from either the IEEE or a governmental body dictating the security requirements for IoT devices. Additional analysis could be used to help guide in developing additional security layers to any communications protocol selected for the overall sensor framework, however this falls outside the scope of this project.

Concept of Operations

This project followed systems engineering practices[5] and identified stakeholders, determined their needs, and translated the needs into functional and nonfunctional requirements. The project also established use cases to identify additional requirements and to specify the operation of the software.[6] These requirements and use cases formed the basis for the WAG System design and are described in detail below.

Stakeholders

Table 1 lists the project's key stakeholders along with their descriptions, roles, representations, priorities, and associated needs (Table 2). Stakeholders were identified by identifying individuals, groups and entities that would have an impact on or stake in the project at any point during its lifecycle.

Table 1 - Stakeholders and Relevant Information

ID	Title	Description	Role	Representation	Priority	Needs (see Table 2)
SH.01	Students	Developers	Directly involved	Self	1	N.04, N.08, N.09
SH.02	Advisors	Advise/grade	Directly involved	Self	1	N.09
SH.03	WPI	Sponsoring organization	Graduation requirements	Registrar personnel	1	N.09
SH.04	RBE/ ECE/CS Departments	Sponsoring departments	Funding/ MQP requirements	Advisors	1	N.09
SH.05	End users	Use the suit	Use the suit	Test subjects	2	N.01, N.02, N.03, N.04, N.05, N.06, N.07, N.08, N.10
SH.06	Future students	Continue work on this project	Continue work on this project	Proxy or not represented	3	N.10, N.12

Needs

Table 2 lists the key stakeholder needs, and their associated compliance metrics, priorities, and stakeholder traceability (validation)[7]. The traceability column references Table 1 above. These key needs were identified by brainstorming use cases, analyzing software operational needs, and investigating the roles of the stakeholders.

[5] For a general review of SE methods, practices and standard forms, see the CA Department of Transportation web site.

[6] For example, see this Bridging the Gap article.

[7] See: https://www.captechconsulting.com/blogs/validate-vs-verify-a-traceability-matrix-gift-for-you for clarity between validation and verification, and the need for a traceability column in a requirements matrix.

Table 2 - Stakeholder needs

ID	Title	Description	Verification	Priority	Validation (Traceability)
N.01	Ease of Use	Should be easy to use/wear/interact with	User test	1	SH.05
N.02	Save Recording	Should be able to save a recorded motion for later use	User test, prototype	1	SH.05
N.03	Edit recording	System should allow for basic editing of recordings before being published/up-loaded to the database	User test, prototype	2	SH.05
N.04	Download/open saved recordings	System should be able to load existing recording to get feedback	User test, prototype	2	SH.05
N.05	Playback control	User should be able to set ‘breakpoints’, jump to places, and pause/play re-cordings	User test, prototype	1	SH.05
N.06	Safety	System will be designed to applicable engineering and human interface practices that limit injury risk	Standards	1	SH.05, SH.01
N.07	Visualization/ Simulation	User should be able to visu-alize the motion after re-cording it and before per-forming it	User test, prototype	2	SH.05
N.08	Adjustable Fit	Should be configurable for different body types	User test, prototype	2	SH.05
N.09	System Feedback	Should provide haptic sys-tem feedback (vibratory)	Model, simulation	1	SH.05
N.10	Documentation	Should be well docu-mented. Should provide component details and a “how-to” of using the soft-ware and hardware	User review	2	SH.05, SH.06
N.11	Response Time	Should have a non-noticea-ble response time for hap-tic vibrations	Prototype, simulation	3	SH.05
N.12	Portability[8]	Should be portable when on and off the body	Prototype test	3	SH.05

[8] Portability refers to how easily the entire system can be transported when on or off of a person. This includes the host computer, the bands and the accompanying charging equipment and Wi-Fi router.

Use Cases and User Stories

This system is intended to perform two primary actions: recording a user's motion and playing back the recorded motion in order to guide another user along the recorded motion trajectory. The use cases and user stories below illustrate the interactions between the system, the software, and the user. They also identify features and components that may require additional research. Finally, the use cases detail typical software usage and operation.

UC.01 below outlines a process in which a user wishes to learn a motion by using the system with a pre-recorded motion file. The user can view a simulation of the motion file to verify the motion is the one s/he specified by the motion file. After the user starts using the system, s/he will experience vibration along her or his limbs indicating directions to the correct location of the motion.

UC Identifier	**UC-01**
UC Name	Playback Downloaded Action
Primary Actor(s)	Baseball player who wants to practice a pitch
Initiating Conditions(s):	1. Athlete identifies a pitch he wants to practice 2. Athlete downloads the action he wants to practice onto the computer
UC Description	1. Athlete puts on the WAG Bands and turns the system on 2. Athlete pairs the system with his computer using a simple GUI 3. Athlete calibrates the system 4. Athlete plays the desired action on the GUI's motion viewer window to verify the motion 5. Athlete presses a button on computer to start the motion playback 6. Athlete moves through the action following the haptic vibrations 7. Athlete restarts the action using the GUI to play it through again 8. Athlete finishes the action 9. Athlete presses a button on the GUI to stop playback capabilities 10. Athlete removes the WAG Bands and connects charger
Alternative(s) and corresponding step number	5. Athlete issues simple voice command to start the motion playback 9. Athlete issues simple voice command to stop playback capabilities
Exit Conditions	• Athlete turns off WAG Bands and closes GUI • Battery dies on computer or bands • Athlete moves out of range of Wi-Fi router
Needs/Requirements Discovered	• User needs to be able to interface with library of actions • User should be able to stop/start the system via GUI on the computer and via simple voice commands • System should include basic controls for action playback • User needs to be able to connect/disconnect the WAG Bands from the computer • User needs to be able to charge the system or tether it for control
Models/Studies Needed	• Response time for haptics • Capabilities of wireless communication with computer (ranges) • Battery life

UC.02 describes the process by which a user's own motion is recorded. The user can start or stop the motion recording via verbal commands or GUI input. Once the motion is recorded, the user can crop the motion recording to narrow it down to a specific selection, and play it back as a simulation in order to verify his/her recording. This motion recording can then be saved to a file and kept for personal review, or distributed to others as a teaching tool for that particular recorded action.

UC Identifier	**UC-02**
UC Name	Record Action
Primary Actor(s)	T'ai Chi trainer
Initiating Conditions(s):	A T'ai Chi trainer wants to teach others T'ai chi
UC Description	1. T'ai Chi trainer puts on WAG Bands and turns the system on 2. T'ai Chi trainer pairs the system with her computer using a simple GUI 3. T'ai Chi trainer calibrates the system 4. T'ai Chi trainer selects the recording interface on the GUI 5. T'ai Chi trainer issues simple voice command to start the recording 6. T'ai Chi trainer performs and records a T'ai Chi form 7. T'ai Chi trainer issues simple voice command to stop recording 8. T'ai Chi trainer plays the action on the GUI's simulation to verify motion 9. T'ai Chi trainer uses the GUI to complete small edits on the action 10. T'ai Chi trainer removes the suit and connects charger 11. T'ai Chi trainer uploads the complete motion file to the library
Alternative(s) and corresponding step number	5. T'ai Chi trainer uses GUI button to start the motion playback 7. T'ai Chi trainer uses GUI button to stop the motion playback
Exit Conditions	• T'ai Chi trainer stops the recording through simple voice commands or the GUI • The maximum recording time is reached • Suit battery dies or computer battery dies
Needs/Requirements Discovered	• User needs to be able to see the action recorded • User needs to be able to edit the action (trim it to critical parts)
Models/Studies Needed	• Study of visualization tools of motion • Study of recording process for sensors to minimize data file size

Each of the user stories below highlights a different part of the intended software functionality. These stories specify what needs to be available to the user, and intend to provide specific usage examples for the system.

- As a Trainer, I can create a new motion file with a unique name and description.
- As a Trainer, I can record myself moving through a motion and save it to a file.
- As a Trainer, I can crop off the first 3 seconds of my motion.
- As a Trainee, I can search through a library of motion files and select one to learn.
- As a Trainee, I can play back a motion at half speed.
- As a Trainer, I can connect my bands to the application with one click.
- As a Trainer, I can calibrate my bands by mimicking the pose shown on the application and saying "calibrate."
- As a Trainer, I can disable my wrist bands.
- As a Trainee, the application will notify me when my bands disconnect from the application.
- As a Trainee, the application will notify me when the battery in a band needs charging.
- As a Trainee, I can step-through frames of a motion.
- As a Trainer, I can review the motion I recorded.
- As a Trainee, I can preview a motion before I play it.

The user stories and use cases are used to develop the functional requirements which are detailed in the following section.

Requirements

Table 3 and Table 4 present the functional and nonfunctional project requirements for this project, respectively. Functional requirements detail how the system should behave while the non-functional requirements describe auxiliary aspects of the system not necessarily critical to operation. The requirements were developed by analyzing the stakeholder needs and use cases, and then developing "good" requirements that were traceable, unambiguous, measurable, testable, and feasible.[9] Each requirement has a short description, validation, a priority, and a verification process [37].

The requirements are individually testable, and encompass concrete subsections of the project that must be realized to be considered a success.

Functional Requirements

Functional requirements are presented in Table 3 on the following page. In this table, the requirements FR.01, FR.02, FR.03, and FR.06, refer to the ability of a user to record motions; preview or review the recorded motions in a GUI visualization; control playback and recording with controls such as play, stop, fast-forward, or rewind; make basic edits to the recorded motions; and save the motions for future use.

The requirements FR.04 and FR.05 involve the ability of users to control settings on the motion they are about to learn. These settings allow the users to learn at their preferred speed, and enable users to repeatedly practice specific sections of the motion without needing to play through the entirety of the motion every time.

Finally, the last two requirements, FR.07 and FR.08, involve haptic feedback. These requirements ensure that the system helps to guide the user through a motion. The response time between when the suit senses positional error and when the user feels the suit's haptic feedback should not be noticed by the user. FR.08 relates to a human's response time to recognize a visual change.

[9] For example, see: Slide Share Good Requirements or IBM Good Requirements

Table 3 - Functional requirements for this project

No.	Title	Description	Validation (See Table 2)[10]	Priority	Verification[11]
FR.1	Record Time	The system shall be able to record at least 10 min. of continuous movement	N.02	1	Prototype, user tests
FR.2	Edit Motion Recordings	The system shall allow the user to perform basic editing on a motion, including cropping recorded actions	N.03	2	Unit tests, Prototype
FR.3	Transferrable Motion Recordings	The system shall be able to save motion recordings to a file format	N.04	1	Unit tests, Prototype
FR.4	Playback Speed	The system shall allow the user to specify a range of motion playback speeds	N.05	2	Unit tests, Prototype
FR.5	Motion Playback Controls	The system shall allow the user to control motion playback via setting breakpoints, skip forward/backward, play/pause, stop	N.05	3	Unit tests, Prototype
FR.6	Simulation	The system shall allow the user to view a simulation of the movement	N.07	2	Unit tests, Prototype
FR.7	Haptic Feedback	The system shall provide vibratory feedback relative to positional errors made by the user	N.09	1	Prototype, Modeling
FR.8	Haptic Response Time	The system shall provide haptic feedback within 100ms of a positional error made by the user [38]	N.11	2	Unit tests, Prototype
FR.9	Visual Playback Ghosting	The system shall visually overlay the user's current location with the intended position	N.01, N.07	3	Unit tests, Prototype

Non-Functional Requirements

Non-functional requirements are presented in Table 4 on the following page. Many of the non-functional requirements listed are self-explanatory, but a few require more information. NFR.02 and NFR.04, though intangible, can be measured by surveying test subjects to determine how well they felt that the system adhered to the requirements. NFR.03 refers to safety – although there are currently no OSHA standards for exoskeletal robots, ISO 13482 covers "personal care robots," which includes wearable exoskeletons.

[10] Validation refers to requirement traceability. Essentially this asks what need each requirement comes from.

[11] Verification refers to how the requirement will be tested for compliance.

Table 4 - Non-functional requirements for this project

No.	Title	Description	Validation	Priority	Verification
NFR.1	Degrees of Freedom	The sensors of the system shall allow for up to 9 degrees of freedom per arm	N.01	1	Prototype, modeling
NFR.2	Non- Restrictive	On a Likert scale of 1 to 5, at least 80% of the users shall rank the system as 4 (good) or 5 (very good) in terms of motions not being impeded by the system	N.01	2	Prototype, User studies
NFR.3	Safe Operation	The system shall adhere to ISO 13482:2014(en) safety standards. *NOTE: According to the FDA classification of devices, this sort of device falls into class II.*	N.06	1	Prototype, Modeling
NFR.4	Fits Variety of Hu-mans	On a Likert scale from 1 to 5, 60% of users should rank the system adjustability and fit as 4 (good) or 5 (very good)	N.08	2	Prototype, Modeling
NFR.5	Documentation	Documentation is included for all parts of the system for developers to continue the project and for users to understand the system	N.10	2	User study, Prototype
NFR.6	Lightweight	The weight of the entire wearable system shall be less than 10 pounds	N.13	1	Prototype
NFR.7	Intuitive UI	The system interface shall be ranked as either a 4 or 5 on a Likert scale by at least 90% of users for ease of navigability.	N.01	2	Prototype, User studies
NFR.8	Playback Battery Life	The system shall support at least an hour of continuous playback which involves the use of vibratory feedback	N.13	1	Prototype
NFR.9	Recording Battery Life	The system shall support at least 2 hours of motion recording which does not in-volve the use of vibratory feedback	N.13	1	Prototype

System Design

This project's deliverables include wearable action guidance (WAG) bands for the arms, shoulders, and chest, and a software application to accompany the bands. Each band consists of a hook and loop strap, a secure elastic cord, a wireless electronic control board, and vibratory modules for feedback. The WAG Bands for the arms, shoulders and chest make up a “Suit.” Functionally, the system supports two types of users who correspond to different modes of use: a Trainer, who will primarily record motions, and a Trainee, who can play back motions. During motion playback, haptic feedback indicates the Trainee's deviations from the motion in real-time so the Trainee can correct himself/herself as the motion is executed. The main benefit of a system with these capabilities is the potential to train individuals in a more efficient, more effective, and overall less expensive manner. Due to time and financial limitations the project is limited to the upper body only, however the WAG System could easily be expanded to include lower body. T’ai Chi is applicable for system validation because it consists of large sweeping motions and is relatively slow.

“Suit” Overview

The wearable component of the system consists of a set of seven bands that the wearer places on his/her wrists, biceps, shoulders, and on the front of the torso, as illustrated in Figure 13. Each shoulder, wrist, and bicep band looks similar to the prototype in Figure 13 with the overall design. Each band consists of a 3D printed case, a strap, a 1000mAh battery, a Teensy 3.2 microcontroller, a Wi-Fi module (ESP8266), an accelerometer/gyroscope sensor package (MPU6050), and six vibration motors. The final result of this design can also be seen in Figure 14. The bands use the accelerometer and gyroscope sensors to calculate their orientations.

The bands communicate wirelessly and have individual batteries, allowing them to be completely self-contained. Each band is securely affixed to the user to prevent the sensors from shifting while in use to maintain sensor accuracy. The bands use six vibration motors to indicate motion error to the wearer.

The chest band is distinct from the other bands in two key ways: it does not have any motors and it has a voice control module. The chest band is the central control and reference unit; thus it is the only band that the wearer is required to use. This module uses voice control to allow the user to give hands-free commands to activate various functions of the WAG System.

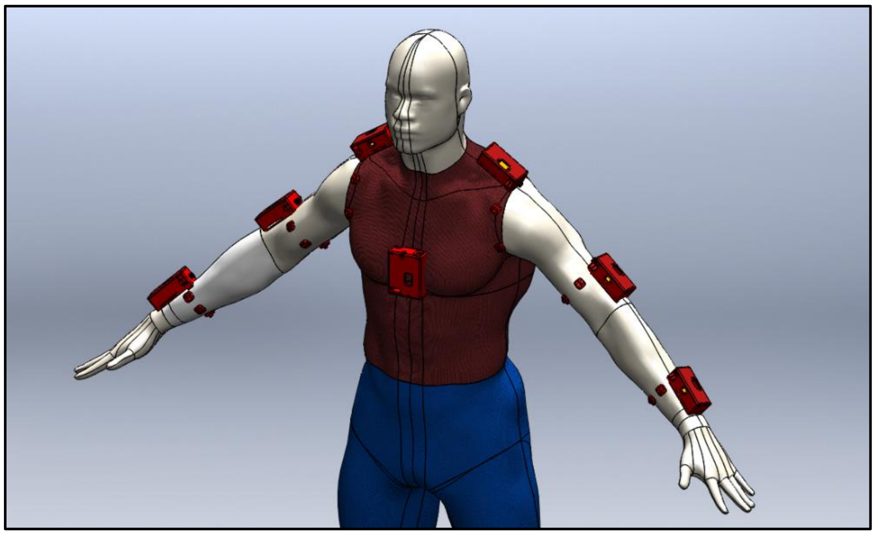


Figure 13. Band concept layout

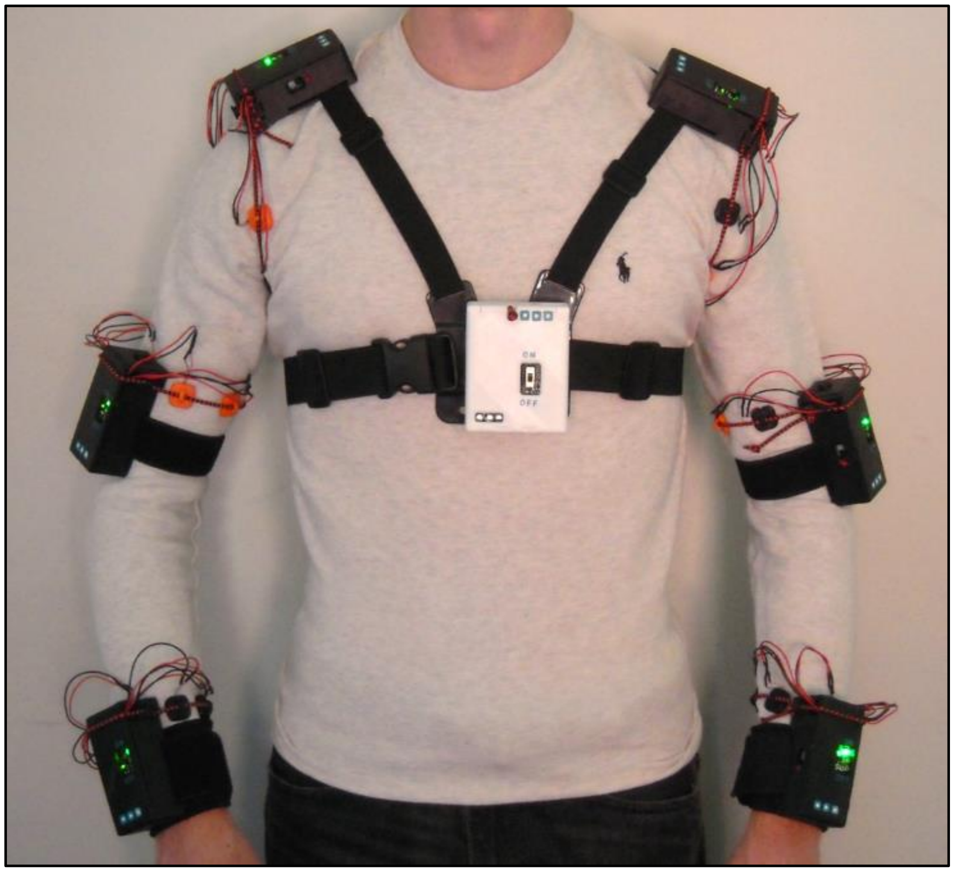


Figure 14. Physical realization of band layout on a person

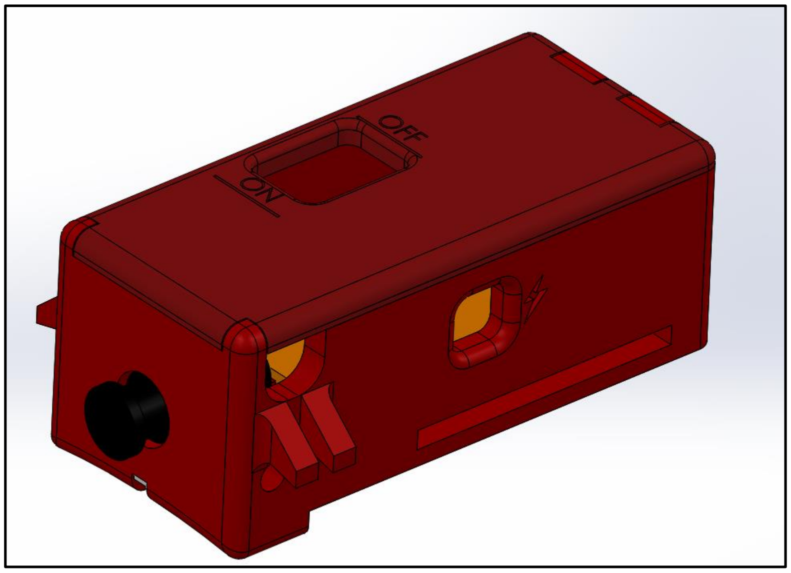


Figure 15. Band prototype design

The physical realization of the band prototype in Figure 15 can be seen in Figure 16. The Initial Band Prototype section within Appendix 8 – Precursor Testing Before System Design Stages includes a progression of all prototypes that led up to the final version shown here.

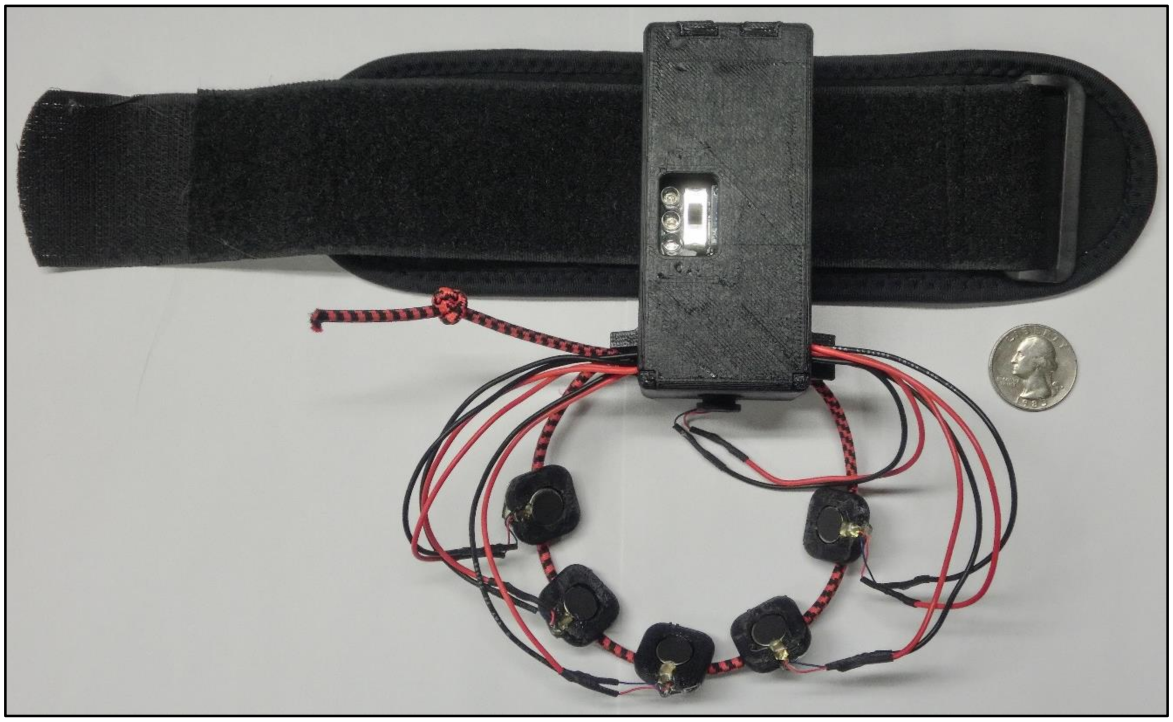


Figure 16. Physical band prototype

The band for the center of the chest is the reference point for the rest of the suit and can be seen in Figure 17.

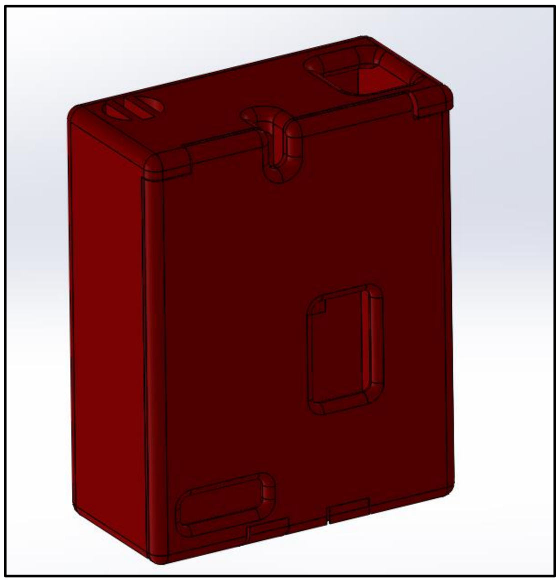


Figure 17. Chest band prototype design

The physical realization of this chest piece band attached to the chest straps can be seen in Figure 18.

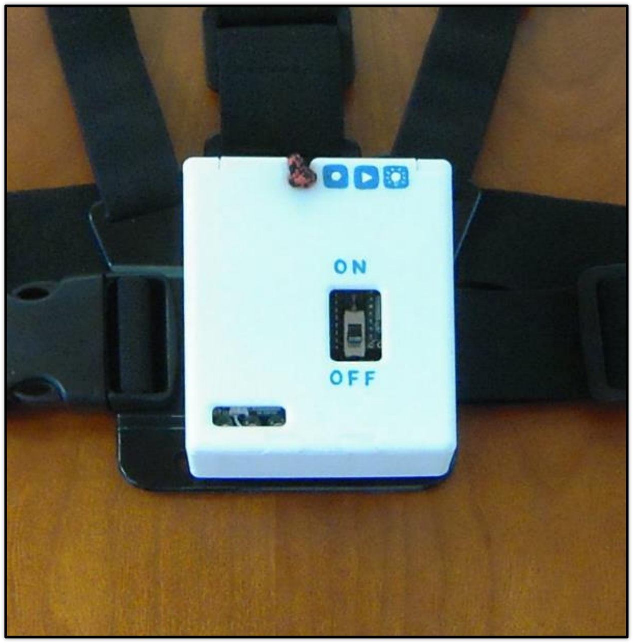


Figure 18. Physical chest band prototype

The final component of the system is the host computer that runs the computer application. This can be the Intel Atom or any other computer capable of running the control interface where the user controls the suit and playback settings. Motion files can be stored to the local computer and a cloud library will be implemented later for file sharing.

Requirements and Specifications

Detailed below are the requirements relating to the full, wearable system. The wearable components were designed to meet the requirements specifications. The applicable requirements are FR.07, NFR.01, NFR.02, NFR.03, NFR.04, NFR.06, NFR.07, NFR.08, and NFR.09.

The design process of the band casing was primarily driven by the non-functional requirements. These requirements focused on making the bands user friendly, comfortable, and safe. The band prototypes were designed so that no tools are needed to open and close them. They are highly adjustable to allow differently sized users to adjust them for a more comfortable fit. The bands need to be tightly secured to the user to prevent the sensors from shifting, but not so tight that they would hinder his/her motion. The arm bands use hook and loop straps to hold the main part of the band in place and elastic cord to hold the vibration motors in place. Both the hook and loop fasteners and the elastic are adjustable. The elastic cord allows the user to move each motor to be appropriately and evenly spaced for his/her body. A padded harness holds the shoulder and chest bands in place. The harness is adjustable, lightweight, and unrestrictive. To ensure that the bands are safe to operate, all electronics are completely encased, except the vibration motors that have to be in contact with the user's skin to effectively indicate motion error.

The internal design of the bands is such that the components, particularly the accelerometer and gyroscope sensors, are securely held in place in order to maintain the accuracy of the bands' motion data. Additionally, because the bands are independent of each other, any one band can be easily replaced if a component is broken. There is no hardware limit to the number of bands that can be added; only software changes are required to add additional bands.

The functional and non-functional design requirements drove the design process to ensure that the system meets the needs of the challenge specification.

Software Overview

The WAG System includes an application with a graphical user interface (GUI) that handles user inputs and the system's modes and settings. This application runs on the user's computer and allows the user to easily interact with the system.

Requirements and Specifications

The WAG System software design came directly from the requirements detailed in Table 3 and Table 4, as well as the user stories and use cases developed in the Concept of Operations section. The software-related requirements are FR.01, FR.02, FR.03, FR.04, FR.05, FR.06, FR.07, FR.08, FR.09, NFR.05, and NFR.7. These requirements fall into a few categories: playback, recording, editing, and opening/saving a motion. A state diagram detailing a user's interactions with the GUI, shown in Figure 19, describes the control flow of the software. Each arrow represents a button that changes the content of the primary screen. This diagram only shows interactions that involve major screen switches for the user; interactions changing only the state of the current screen are not shown.

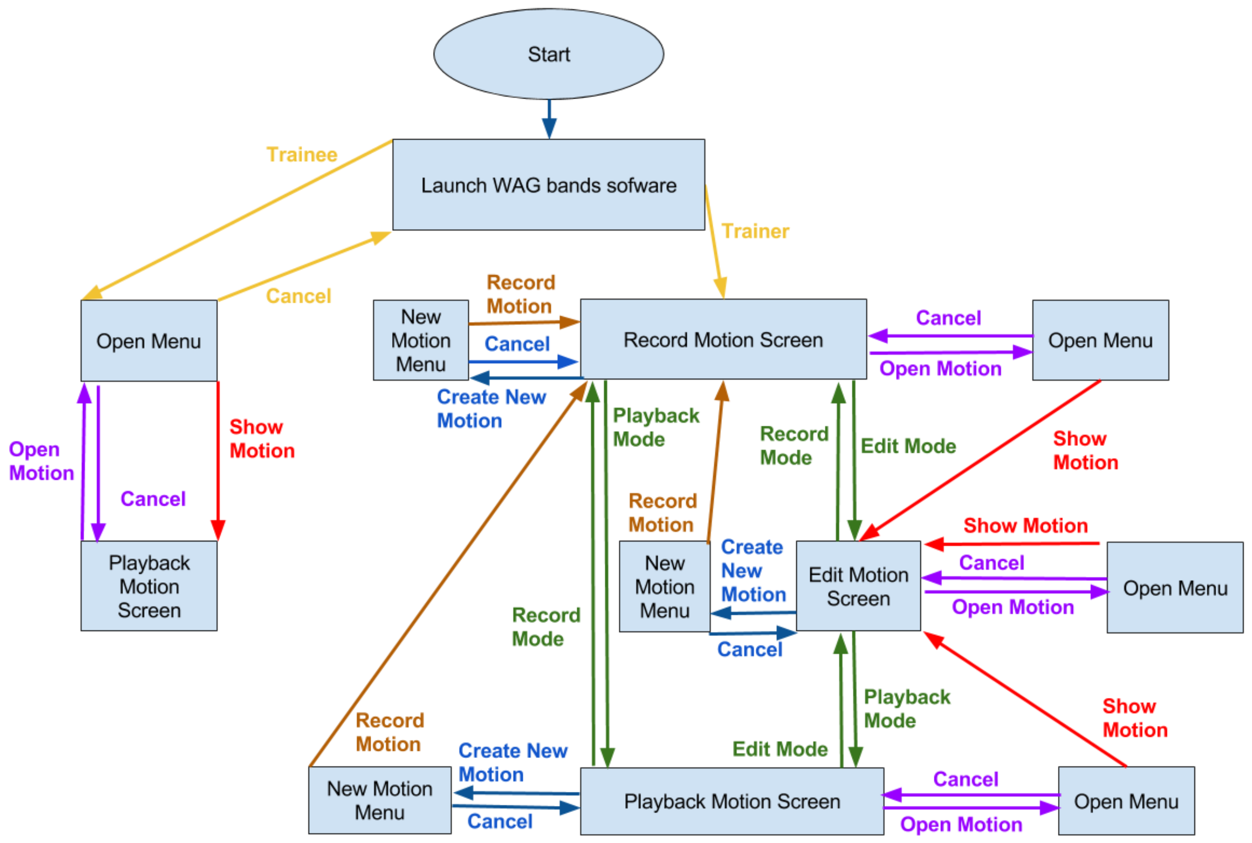


Figure 19. User interface flow diagram

The WAG System has two types of users: Trainers and Trainees. Since these users have different needs and can interact with the software differently, the application first prompts users to select their type (Trainer/Trainee) on the program's welcome window. The program then opens up windows according to the user type, as illustrated in Figure 19. Trainees only have the ability to open and playback motions recorded by Trainers, while Trainers can record, edit, and/or playback motions that they have created. As a result, Trainees only have access to the open menu while Trainers can open or save their motions from any window. Each of the three modes (Recording, Editing, Playback) are on separate screens so Trainers will only see settings pertaining to the mode they currently have open. The full descriptions of the modes can be seen in Table 5.

Table 5 - Software modes and descriptions

Mode	Selectable by	Description
Record	Trainers	Trainers can adjust relevant settings and start/stop recording a motion. Trainers also have the ability to record over an existing motion. Trainers can switch to edit or playback mode.
Edit	Trainers	Trainers can crop their motion and edit the motion's name, description, and keywords. This mode includes the motion viewer so Trainers can see their motion performed on a 3D model. From here, Trainers can return to recording mode to rerecord the motion, or they can enter playback mode to see how a Trainee would learn the motion.
Playback	Trainers and Trainees	Contains playback settings and the motion viewer. This mode allows Trainers/Trainees to playback the current motion file on either the viewer window's 3D model only or both the 3D model and the WAG bands.

Saving and opening a motion file can be done either from the user's local file directory or through the motion library. When opening a motion from the motion library, users search for specific motion names, descriptions, or associated keywords.

Hardware Overview

The WAG Bands are designed to be compact and lightweight. The high level design diagram in Figure 20 shows the general outline of all the hardware for the band. Each band uses a wireless link to send data from the accelerometer/gyroscope to the main computer. The microcontroller transmits data between the sensors and the wireless link, and controls the vibration feedback motors. The design changes slightly for each different type of band, but the general architecture is still the same.

Hardware Requirements

This section pertains to hardware requirements that come from functional and nonfunctional requirements established by the systems engineering design process. The hardware requirements FR.01, FR.07, FR.08, NFR.01, NFR.03, NFR.11, and NFR.12 come from Table 3 and Table 4 in the Requirements section. The bands must be able to provide haptic feedback within 100ms of error detection (10Hz). Each band also must be able to support 2 hours of recording usage, and 1 hour of vibration playback. The system hardware was designed to meet each of these requirements so that the system meets the needs of the challenge specification.

Technical Documentation

Band Design

Each band is responsible for communicating with the off board computer by sending its position data. The various types of bands and their functions can be seen in Table 6.

Table 6 - Band tasks

Band type	Quantity of band in system	Function
Wrist	2	Measure orientation of wrist Provide position information to computer Receive rotational and translational feedback from computer
Bicep	2	Measure orientation of bicep Provide position information to computer Receive rotational and translational feedback from computer
Shoulder	2	Measure orientation of shoulder Provide position information to computer Receive rotational and translational feedback from computer
Chest piece	1	Measure reference orientation at center of chest Manage voice commands (for speech recognition)

Band Electronics

The high level overview of the band hardware can be seen in Figure 20. The core functionality of the band is its ability to determine its orientation in 3D space using a six degree of freedom accelerometer and gyroscope sensor package over I2C (2 wire)[12] communications. The band's primary processor is a Teensy 3.2 which runs at a clock speed of 96MHz, and features 256k bytes of flash memory and 64k bytes of RAM [39].[13] The band communicates with the software application over Wi-Fi using an ESP8266 Wi-Fi chip. The Teensy relays the orientation updates it receives from the accelerometer/gyroscope sensor over UART to the Wi-Fi module, which packages the data into TCP packets and sends those packets to the host computer via Wi-Fi.

In order to provide vibratory feedback, the system uses six electric rotor vibrators (ERVs). These are about half the size of a dime and provide a sensation similar to the buzz of a cell phone on vibrate mode. Each band contains its own battery which supplies the band with 3.7V and 1000mAH capacity. A 3.3V regulator outputs constant voltage to the ERVs, Teensy 3.2 and ESP8266, which require the lower voltage level. Each of the band's components and their functionalities are listed in Table 7.

[12] I2C is a communication protocol developed by Phillips Semiconductor to communicate between integrated circuits using a simple two wire serial communication. The benefit of using this approach is that a simple communications bus between an arbitrary number of integrated circuits can be developed.

[13] The Teensy is an embedded microcontroller with more memory and functionality than the Arduino Uno platform that runs at 16MHz and has 32k bytes of flash and 2k bytes of RAM. The Arduino platform was considered for this project due to its low-cost, but it could not maintain the refresh rate required by the platform.

Table 7 - Hardware component functionalities

Hardware component	Description
Teensy 3.2	Run communication to ESP8266, receive position and rotation information from MPU6050
ESP8266	Implement and run Wi-Fi communications over UART at 115200 baud
Low battery detection circuit	Determine when battery voltage has dropped below threshold of acceptable operation
N-channel MOSFETs	Increase current drive capability of Teensy PWM pin from 10mA to 300mA for ERVs
ERV (Electric Rotor Vibrator) Motors (shaft-less motors)	Vibratory feedback

The high-level diagram for the bands on the arms, wrists, and shoulders shown in Figure 20 describes the hardware for each of the shoulder, bicep and wrist bands. This was then implemented into the Printed Circuit Board (PCB) shown in Figure 21.

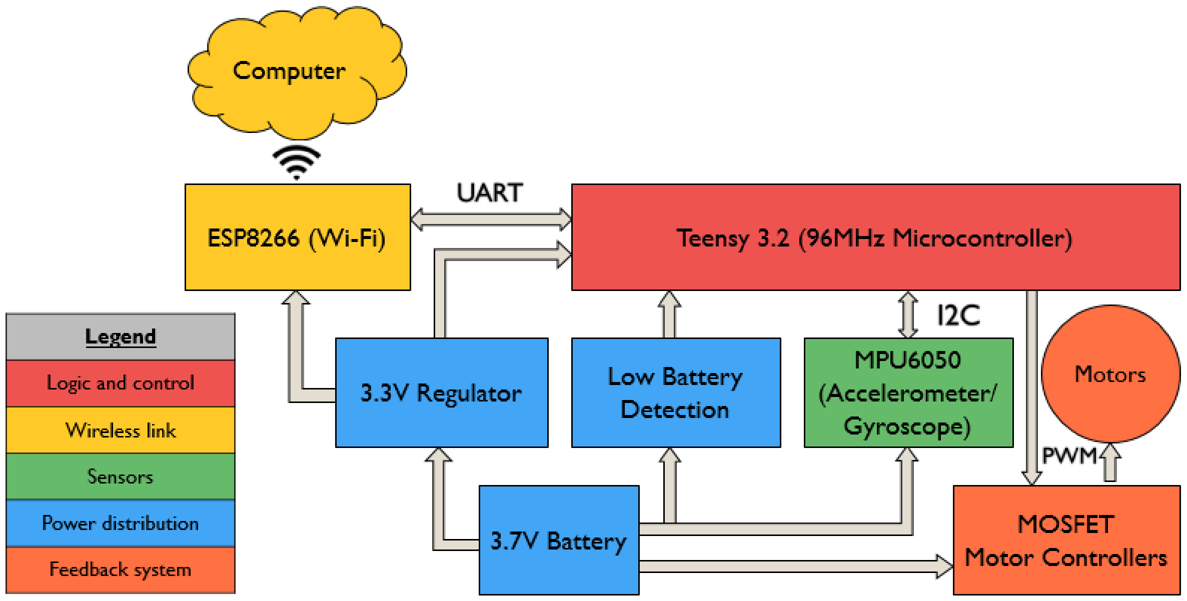


Figure 20. High level band design

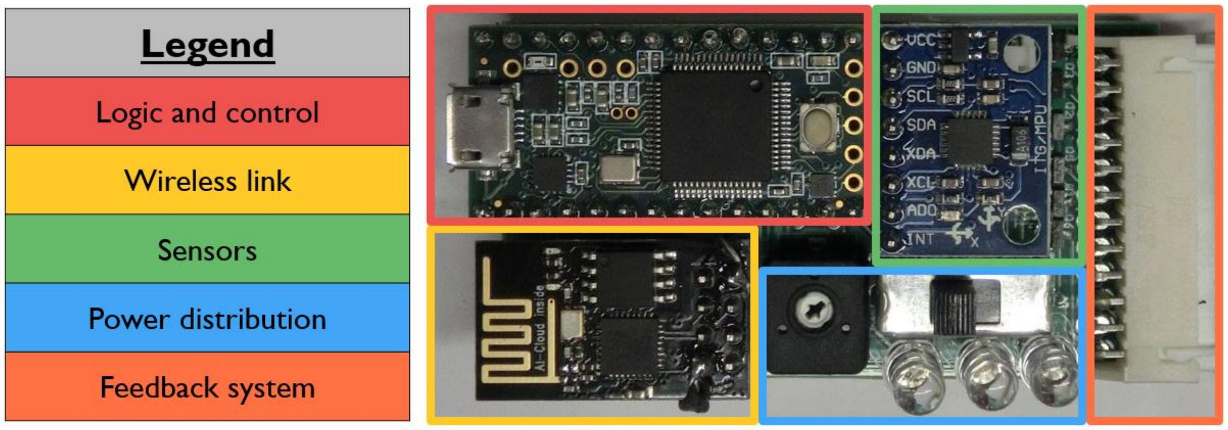


Figure 21. Printed circuit board of WAG Band

The high-level diagram for the center chest module seen in Figure 22 shows a slight variation in the hardware: the EasyVR 3.0 speech recognition module[14] [40]. The chest piece also does not have any haptic motor feedback; it is intended to be a central reference frame for the suit, and does not provide vibratory feedback.

Appendix 4 – Assembly & Construction includes additional images of the completed printed circuit board for both the bands and the chest module.

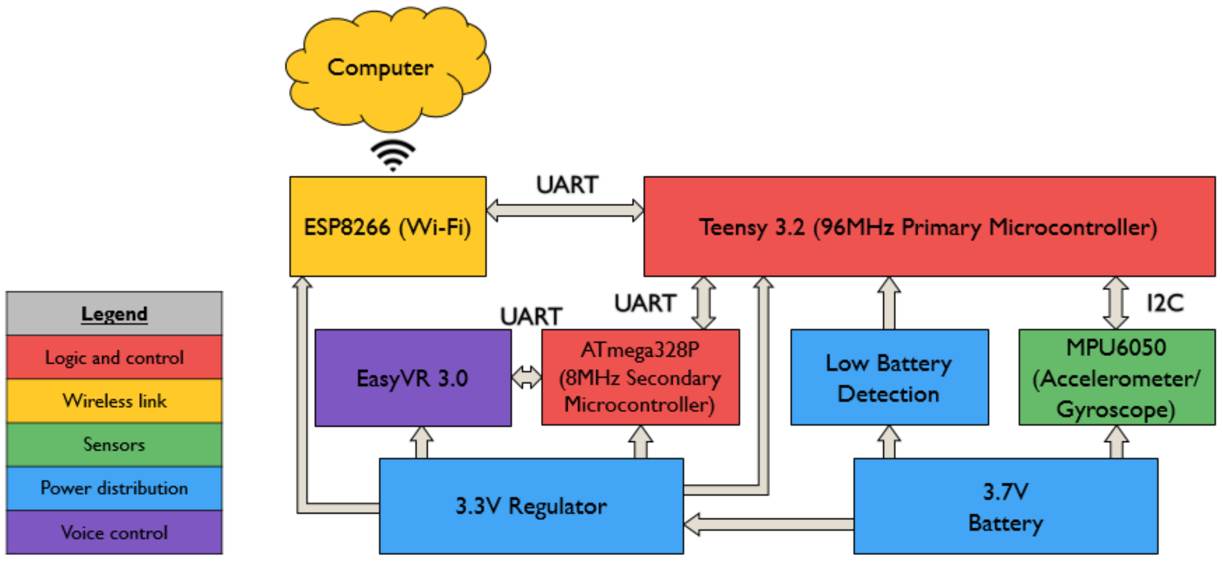


Figure 22. Chest piece high level design

[14] The EasyVR 3.0 module is produced by VeeaR and is intended to be a plug-and-play speech recognition module for Arduino platforms. The module comes with 26 built in person-independent speech commands it can recognize along with the option to add more custom user-specific words.

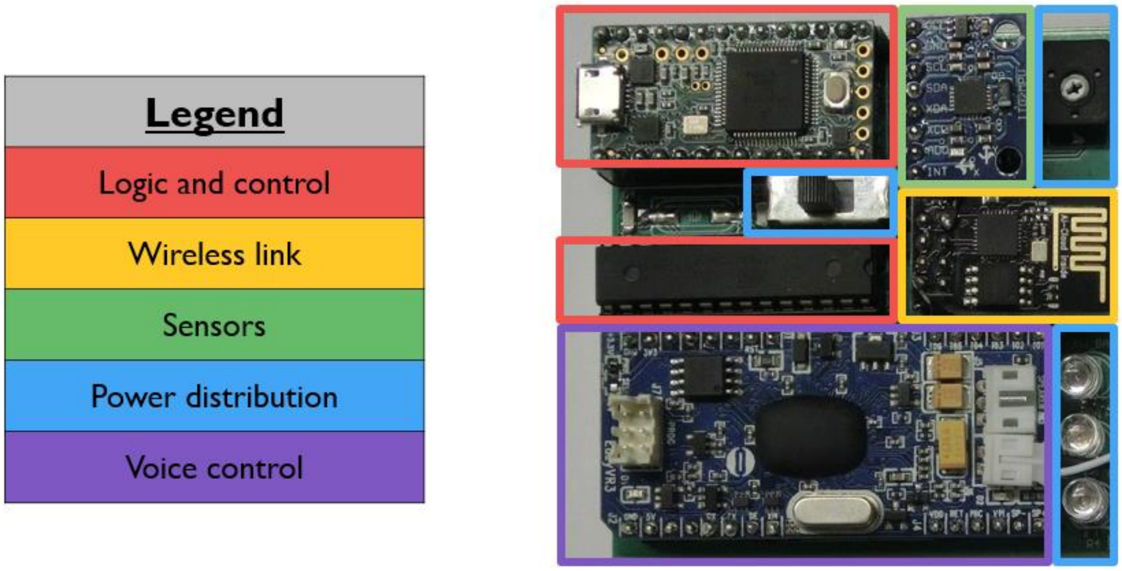


Figure 23. Printed circuit board for WAG Chestpiece

Component Selection Justification

A trade study of different accelerometers, gyroscopes, and microcontrollers was used to select components that would allow the WAG System to meet the system requirements. These trade studies were conducted using decision matrices, which weight important features of the components to generate a final score (see Appendix 6 – Initial Design Steps: Trade Study). The MPU6050 was selected from the accelerometer/gyroscope trade study due to its small size, low cost, and high accuracy. The MPU6050 also uses built-in motion processing that fuses its accelerometer and gyroscope data into one filtered sensor orientation, making it easy to use.

A key component of the WAG System is the communication between each band and the host computer. Three main options exist for wireless communications in this range: ZigBee, Bluetooth, and Wi-Fi. ZigBee, however is typically used for applications with very low data rates, and therefore, would not be suitable for the WAG System. After conducting hardware tests with Bluetooth and Wi-Fi, Wi-Fi was determined to be significantly simpler to develop with, while greatly exceeding the data rate requirements of the project.

A decision matrix-based trade study also determined that the easiest to use and least expensive Wi-Fi chip available was the ESP8266, while also being able to meet all of the system's data rate requirements. This chip includes an easy to use UART interface and a built-in Wi-Fi stack inside the microcontroller, and it can be programmed from the Arduino environment.

The Teensy 3.2 was selected from the microcontroller decision matrix because it features several hardware UARTs to debug the module and to communicate with peripherals (Wi-Fi chip), it has a small form factor suitable for wearable applications, and it is capable of using the breakout board within the final band. Despite being relatively expensive, the Teensy 3.2's performance, form factor, and debugging capabilities made it the best choice as the primary band processor.

The shaft-less vibration motors were selected due to their cost, size, the intensity of vibration they could provide and the simplicity of their controller circuitry. Since the shaft-less rotor motors run on a DC voltage, they can be run using a PWM (pulse-width modulation) signal from a microcontroller. This signal can be applied using a simple MOSFET circuit, as opposed to the more complex driver circuitry required by AC vibration motors.

The EasyVR3.0 voice control module was selected based on its pre-configured intuitive control and command set. The software libraries available for this board make the module particularly easy to integrate into a pre-existing microcontroller project with an additional UART connection. This module is fairly expensive ($50), but the benefit of having built-in recognition of 26 commands in several languages, along with its general ease-of-use, justifies its cost.

Sensors and Sensor Fusion

Each band uses an MPU6050, an accelerometer and gyroscope sensor package, to measure its orientation in three dimensional space. The bands use a closed-source motion determination algorithm developed by InvenSense to precisely estimate their rotational poses. Each band's sensors are capable of generating pose estimates at rates of up to 200Hz. The MPU6050 is the bands' means of tracking the wearer's motion.

Haptic Motor Control

Each band includes 6 vibration motors distributed in a circle around the band to indicate pose error to a user through vibratory stimuli. The bands use these vibration motors to indicate spatial error to a user in two primary ways: by guiding the user to rotate the band in-place (a motion similar to turning a key), or to guiding the user to move the band perpendicular to its in-place rotation axis (a motion similar to moving an arm vertically without rotation). Each band receives error updates comprising three numerical values from the computer application over the Wi-Fi link. These three numerical values describe the rotational and perpendicular errors: (1) the in-place rotation error magnitude, (2) the angle describing the axis of perpendicular motion error, and (3) the magnitude of the perpendicular motion error. The rotational and perpendicular errors are calculated using a swing-twist decomposition of the error rotation, shown in the following equation:

$$R_{error} = R_{twist} = R_{twist}R_{swing}$$
$$\vec{p} = (axis(R_{error}) \cdot \vec{x}_{twist}) = (axis(R_{error}) \cdot \vec{x}_{twist})\vec{x}_{twist}$$
$$R_{twist} = Quaternion(scalar(R_{error}), \vec{p})$$
$$R_{swing} = (R_{twist})^{-1} = (R_{twist})^{-1}R_{error}$$

where $\vec{p}$ is the projection of the axis of the error R_{error} represented in axis-angle form onto the unit-axis about which the twist rotation will occur, $\vec{x}_{twist}$. The twist rotation component of the total error is constructed by constructing a quaternion rotation representation using the scalar component of the original error (from axis-angle format) and the projection vector $\vec{p}$. The swing rotation component of the total error is all that remains of the error after removing the twist component. The magnitude of the in-place error (1) is the angle of the twist error from its axis-angle representation, and the angle of the perpendicular motion error (2) is represented by the angle between the z-axis of the band and the swing axis. The magnitude of the perpendicular motion error (3) is the angle of the swing error is calculated from its axis-angle representation.

The band indicates the in-place rotational (twist) error to the user by vibrating the motors sequentially around the band in the direction of the user's error. If the user needs to rotate a band clockwise to correct an error, for example, the vibration pattern will move clockwise about the band. The band uses the magnitude of the in-place rotational error it receives over the Wi-Fi link to control the amplitude of the rotational vibration pattern.

The band uses the perpendicular motion (swing) error axis it receives over the Wi-Fi link to place a vibration stimulus behind where the user needs to move in order to correct the perpendicular motion (swing) error. The band uses the perpendicular motion error magnitude to control the amplitude of this stationary vibration stimulus.

The rotational vibration pattern and stationary vibration stimuli are superimposed so that both the in-place rotational error and the perpendicular motion error are evident to the user. If both the in-place rotational error and the perpendicular motion error are large and the vibration signals to the motors become saturated, the maximum allowed vibration amplitude is divided between the rotational vibration pattern and the stationary vibration proportional to the in-place rotation error magnitude and the perpendicular motion error magnitude.

Communication

Each band uses an ESP8266 chip (seen in Figure 24) communicate with the WAG System software application. The ESP8266 can be programmed using a custom Arduino core by placing the device into bootloader mode and then sending program data over an FTDI cable to the chip's UART. This core allows a developer to program code using Arduino syntax using the Wi-Fi capabilities. The ESP8266 exchanges data with the Teensy microcontroller via a UART link operating at 9600 baud. Further details on how the ESP8266 interfaces with the computer application are provided in the Software Design section.

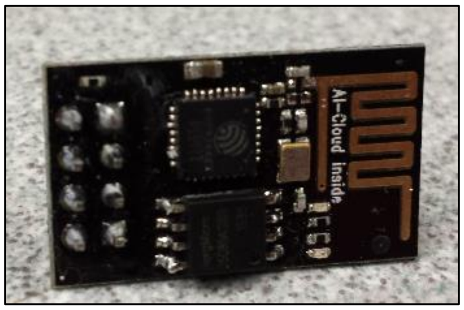


Figure 24. ESP8266 Wi-Fi chip

Software Design

The WAG System software application is written in C++ and uses key libraries including Boost[15], OpenGL[16], and Qt[17][41][42][43]. C++ offered easy communication with the hardware components, object-oriented design capabilities, fast operation, and its OpenGL and Qt libraries. The Qt Library offered a 'slots-and-signals' message passing mechanism. This mechanism is used extensively throughout the application to pass data between objects. A 'signal' is emitted when a specific event occurs. Signals can be parameterized to carry data related to the event or can have no parameters and simply indicate that an event has occurred. Slots are similar to functions. A slot and signal can be linked by using the 'connect' function. If this occurs, then a slot is called when a signal it is connected to is emitted. Signals and slots must have the same signature, or number and type of inputs, to correctly connect. Multiple signals can be connected to multiple slots. This allows data to be transferred between objects without having to maintain instances of an object within another object [44].

The WAG System software application runs on the Intel Atom, which serves as the central communication hub for all of the WAG Bands.

[15] Boost provides a number of free peer-reviewed portable C++ source libraries. We are using libraries from Boost to interface with the user's file system.

[16] OpenGL is a widely used graphics application programming interface. OpenGL supports 2D and 3D graphics.

[17] Qt is an IDE for cross-platform C++ development. Qt also has numerous libraries that extend the core functionality in C++.

Class Interactions

The core interactions between the main classes are shown in Figure 25 below. Each of the three modes (recording, editing, and playback) have a central controller that processes user input to maintain the control flow of the application and routes and packages data. The suit class is a container for all of the band objects, which consumes messages received over Wi-Fi and routes them to the appropriate software band object. Encapsulating the band objects in this way allows the application to communicate with and maintain the state of the user's WAG Bands. The WAGFile object represents a single motion and contains its name, location, description, and searchable keywords, as well as the mapping from times to PositionSnapshot objects. A PositionSnapshot object represent the full state of a user's body at a given time, and contains mappings from a band to a pose.

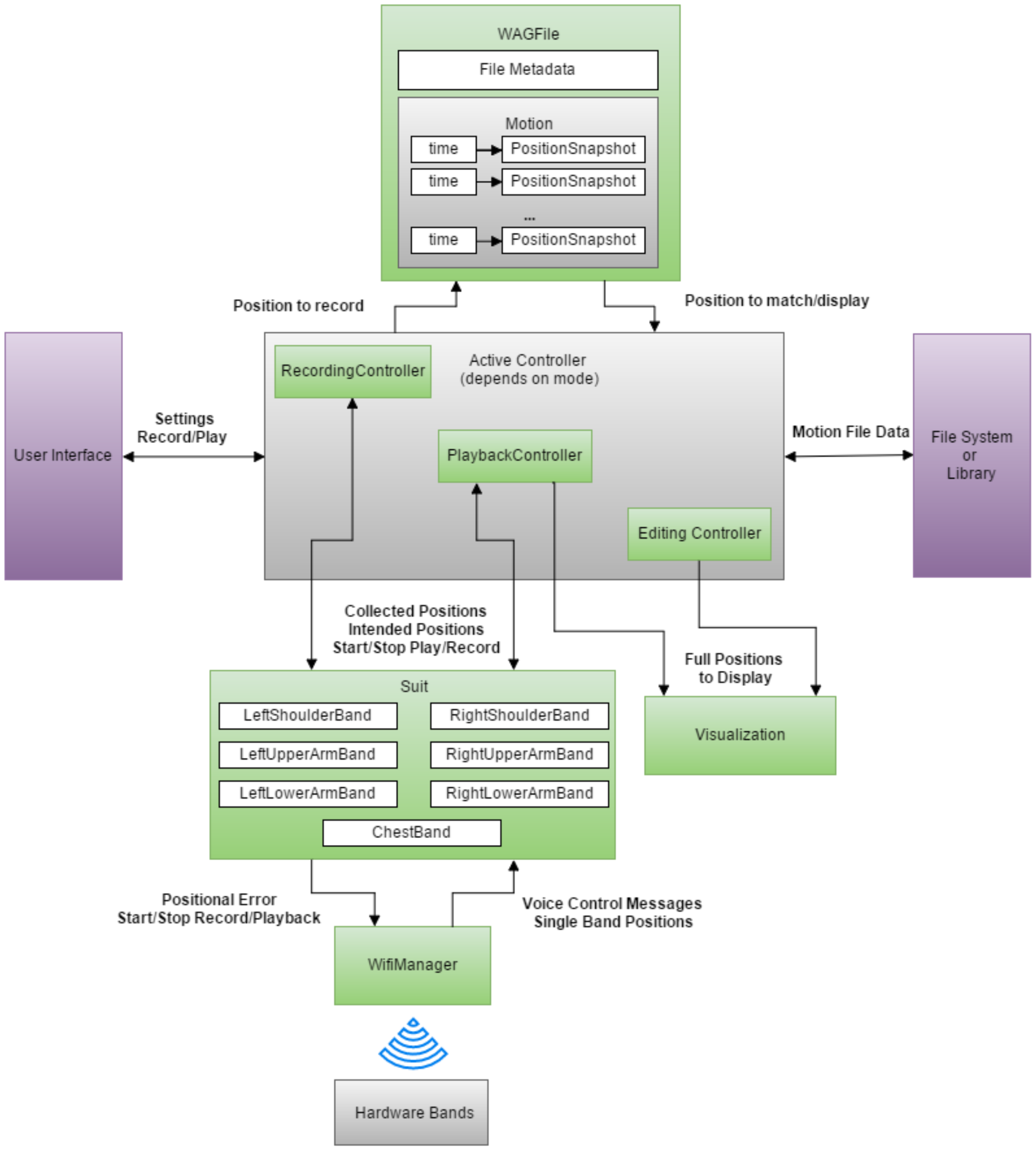


Figure 25. High-level software diagram

Class Functionality

To accomplish the required software functionality, a number of C++ classes were developed as part of the application. The most important of these are shown in Figure 25. Several other classes were developed to support and encapsulate data required by the key classes. The classes, their descriptions, and their fields and methods are shown in Table 10 in Appendix 2 – Software Core Classes and Functionality.

Communications

Within the application, the WifiManager class acts as a hub for Wi-Fi communications. To send a message, a band object calls a method in the WifiManager with the message it would like to send to its respective hardware band. The WifiManager class also receives and routes messages from hardware bands to the suit object, which then passes messages to the proper software band objects. All Wi-Fi communications are done using TCP connections and all inner-application communications use Qt's signals and slots mechanism. This interaction is detailed in Figure 26.

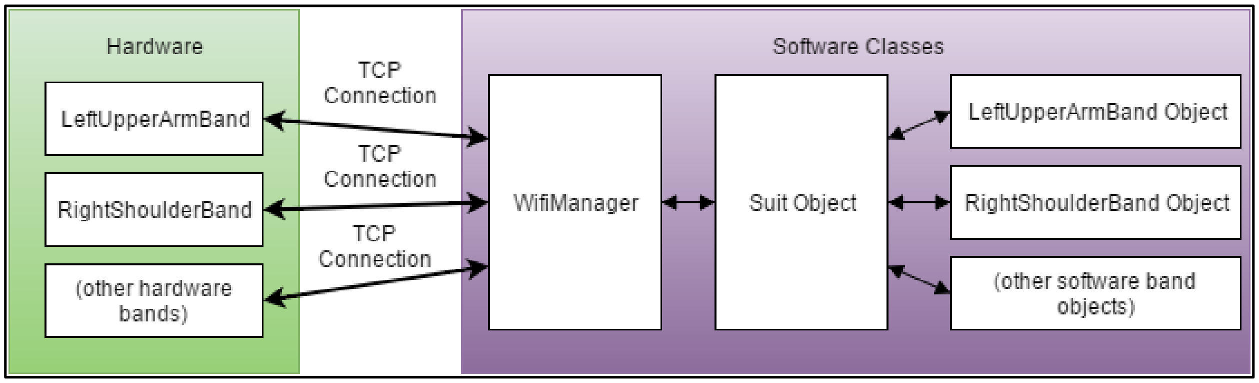


Figure 26. Network interface diagram

Table 8 details the required update frequency from the project requirements, the actual update frequencies measured with the hardware, the bandwidths, communication partners, and required input processing capabilities for each component of the system.

Table 8 - Band communication implementation

Device	Required update frequency	Actual update frequency	Bandwidth	Communication Partner	Processing
Computer	10 Hz	16Hz	~49kbps	Hardware bands (positional error information)	All incoming accelerometer and voice control data from the bands
Arm and Shoulder Bands	10 Hz	16Hz	~7kbps	Computer (sensor data)	Read sensor data Received data from computer into motor command signals
Chest Band	10 Hz	16Hz	~7kbps	Computer (sensor data, voice control messages)	Reads voice control module data Read sensor data

Motion Visualization

The visualization module for displaying a motion for the user is constructed using OpenGL. The Qt library provides an OpenGL widget that can be embedded in an application and extended for customization. The software classes that run the visualization extend the provided OpenGL widget class to make use of the built-in OpenGL support while adding custom functionality. This includes the ability to set a camera location with the camera always looking at the origin, the ability to add a planar convex shape to the scene by specifying the corner points of the plane as well as the translation and rotation from the origin point, and functionality for registering mouse click and drag events for moving the camera.

A kinematic human model was developed in Blender, a 3D modeling application, and included individual links for each movable segment of the body. This Blender model was exported as a Wavefront object file, which preserved the identities of each individual body segment (bicep, wrist, chest, etc.) are all recognized as distinct meshes in the Wavefront object file). The C++ library Assimp imported the Wavefront object into the software application and loaded the meshes, materials and transformations between each of these meshes into structs. The OpenGL widget used the information from these structs to display the model, which can be seen in Figure 26. Each link in this model can be individually colored, and independently moved via standard linear transformation operations.

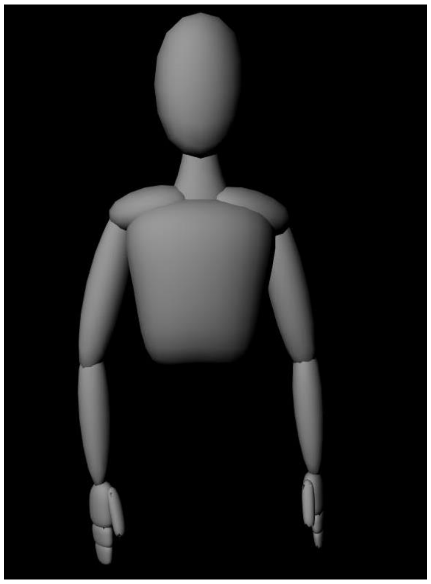


Figure 27. 3D human model rendered in an OpenGL widget

Each segment of the mesh is independently movable by a single orientation update with respect to the visualization's world frame. However, the original Blender model defines the positions and orientations of each mesh with respect to their parent frames. Therefore, the software calculates the transformation of each mesh from the world frame to their final position so that they can be easily updated by the bands' reported orientations. Each limb's mesh is initialized using the following calculation:

$$R_i^0 = \prod_{i=1}^{n} R_i^{i-1}$$

R_i^0 is the rotation from the Blender coordinate frame of the *i*th mesh to align with the world coordinate frame, and gets calculated by successively pre-multiplying the rotation from itself to its parent, R_i^{i-1}.

$$T_i^0 = \begin{bmatrix} R_i^0 & -\vec{h}_0^i \\ \vec{0} & 1 \end{bmatrix}$$

T_i^0 is the homogeneous transformation from the default Blender frame to the OpenGL world frame. h_0^i is the translation from the origin to the mesh, and is extracted directly from the Blender model using the Python API. This transformation is used to rotate and translate every mesh into the shared world coordinate frame.

Once each mesh is aligned with the world coordinate frame, each mesh can be updated with orientations specified in that world coordinate frame. The following equations define the recursive update calculations that the software uses to correctly track the WAG Bands user's pose:

$$\vec{h}_0^{i\,\prime} = \vec{t}_0^{i-1\,\prime} + R_0^{i-1\,\prime} R_{i-1}^0 \left(\vec{h}_0^i - \vec{t}_0^{i-1}\right)$$
$$\vec{t}_0^i = \vec{h}_0^{i\,\prime} + R_0^{i\,\prime} R_i^0 \left(\vec{t}_0^{i\,\prime} - \vec{h}_0^{i\,\prime}\right)$$

$\vec{h}_0^{i\,\prime}$ represents the rotated head vector of the ith mesh in its parent chain. This vector places the base of the mesh in the correct position in the visualization based on the world rotation update of the mesh, $R_0^{i-1\,\prime}$. $\vec{t}_0^{i\,\prime}$ is the rotated tail vector of the ith mesh, and is used to correctly place child meshes. $\vec{h}_0^i$ and $\vec{t}_0^i$ are both pulled directly from the Blender model using the Python API, and represent the transformations of the ith head and tail vectors in the model's default pose. The ith band's mesh is transformed into its updated position using the following equation:

$$T_0^{i\,\prime} = \begin{bmatrix} R_0^{i\,\prime} & R_0^{i\,\prime}\vec{h}_0^{i\,\prime} \\ \vec{0} & 1 \end{bmatrix} T_i^0$$

$T_o^{i\,\prime}$ is used to correctly place and orient the ith mesh in the visualization, and is updated with one variable input – the updated world rotation, $R_0^{i-1\,\prime}$. The rest of the parameters in the previous equations are constants collected from the original Blender model.

Through the wireless link discussed in the previous section, the bands send orientation updates to the software application. These orientation updates are used to rotate and translate the meshes of the links of the human model to match the actual pose of the wearer – for example, as the wearer bends his or her elbow, the OpenGL simulation updates, bending at its elbow joint as well.

Motion Calibration

The orientation updates sent to the software from each band are defined in a semi-arbitrary reference frame that is dependent on the configuration of the band when the user turns it on. Also, the coordinate frames defined by the Wavefront meshes exported from the Blender model do not necessarily align with the sensors in the bands. Therefore, calibration has two stages – an axis alignment stage and a user pose-matching stage.

The axis-alignment stage is simple and occurs completely behind the scenes in the software. The software uses pre-computed coordinate frame transformations to align orientation updates from the bands with the Blender axes. These transformations are precomputed by moving a band, and observing how the model responds in order to define a transformation that re-aligns the mesh model with the motion of that particular band.

The user pose-matching stage must be executed by the user each time any new bands get powered on, because each time a band gets powered on its arbitrary reference frame changes. The pose-matching stage requires a user to match a pose shown on-screen while wearing the WAG Bands. The user then signals the software to calibrate, and the software generates a rotational conversion to the shown pose (which the user is now matching) from what each band is reporting as its orientation. Then, each rotational conversion is applied to all future orientation updates that the software receives from the associated band, mapping the arbitrary band reference frames to the mesh model's coordinate frame. The following equation represents how the band orientation updates get processed for calibration:

$$R_{Band} = R_{PoseMap} R_{RawBandUpdate} R_{FrameAlignment}$$

The R_{Band} output represents the final processed band orientation that gets used as the band's actual orientation. $R_{RawBandUpdate}$ represents the raw orientation data generated by the band that neither matches the axes of the mesh model, and is with respect to an arbitrary coordinate frame. $R_{PoseMap}$ represents the orientation map that gets generated when the user matches the pose shown on the GUI, and gets pre-multiplied by $R_{RawBandUpdate}$ to map the raw band update into the mesh model coordinates. $R_{FrameAlignment}$ aligns the axes of the band's accelerometer and gyroscope sensors with the mesh model's coordinate frame for the associated limb. The final R_{Band} value gets used for all further spatial computations, including visualization updates, feedback calculations. R_{Band} is consistent for all users, since it has been processed by both frame alignment and by pose mapping. R_{Band} represents the orientation of the band with respect to the base frame of the visualization, which is defined by the OpenGL widget. Each band stores and uses their own independent R_{Band} values. R_{Band} is equivalent to the $R_0^{i\,\prime}$ parameter used for world orientation updates in the Motion Visualization section.

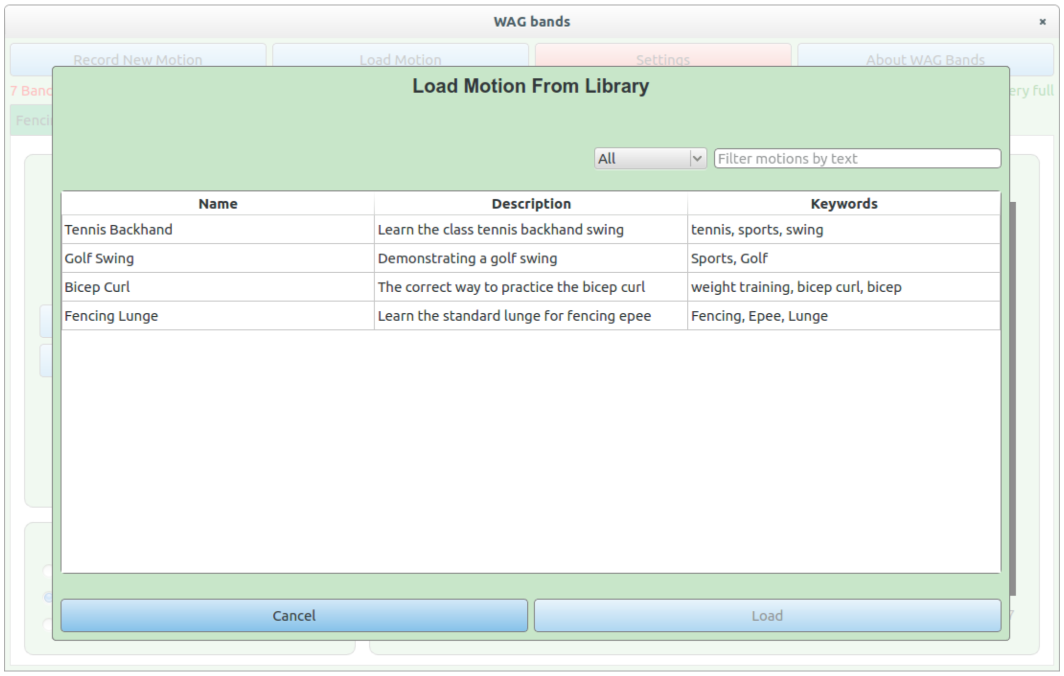
Playback Controls

A PlaybackController class was developed to maintain playback parameters and to issue updates for which frame to display. This class is integrated with the user interface, so any playback options that are selected are reflected in this class. There are two mode options for playback: timed, in which the user matches motions using timing, and step-through, in which the user tries to match a position from a discretized version of the motion before moving on to a new position in the motion. When the user presses play in timed mode, the PlaybackController class issues frame updates reflective of the user chosen speed. Playback stops when a pre-determined end frame number has been reached. In step-through mode the PlaybackController issues frame updates only when the error from the last frame is within the user specified tolerance threshold.

Saving and Opening Motions

One key feature of the WAG System software is the ability for a motion file to be saved and loaded to/from the user's local computer or the motion library. More specifically, Trainers have the ability to save and load recorded files and Trainees can load files from their computer or the motion library. When Trainers want to record a new motion, they must first give the motion a name and a description, optionally giving the motion some relevant keywords, and choose a save location – either a directory on their local computer or the motion library. This allows the motion to be automatically saved as the Trainers record and then edit and/or play back the motion. When a motion is saved, all of its data, including its name, description, keywords, and PositionSnapshots, is saved to the designated save location. When Trainers/Trainees want to load a file from their local computer, a generic file browser is launched. This file browser lets the Trainers/Trainees navigate around their computer, but only shows files with the extension '.wagz.'

To abstract the motion library, the file '.WAGConfig' was created to specify where the motion library is. For testing, the configuration file points to a MotionLibrary directory, allowing for easy access to any motions saved to the library. In the future, the configuration file may be updated if the motion library is implemented as a central database. When users want to load a file from the library, they are shown a window with a table where each rows contains information for one motion file, shown in Figure 28. This window also contains a search bar for searching through motion names, descriptions, and/or keywords. Once users select a row and press the ‘Load’ button, the chosen motion is loaded into the application.

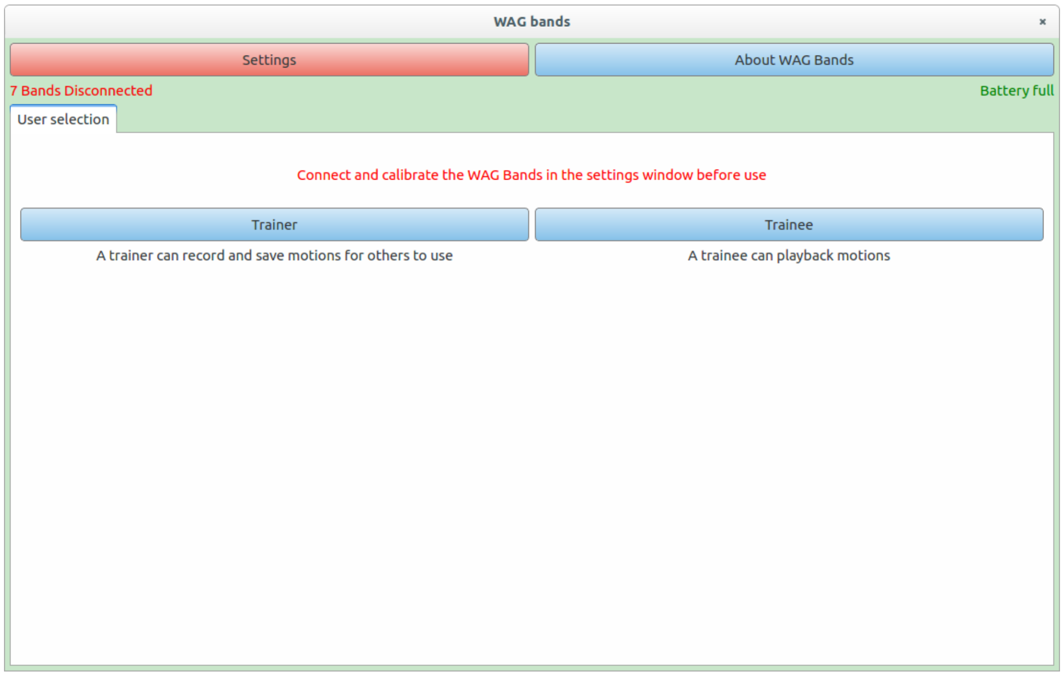


Name	Description	Keywords
Tennis Backhand	Learn the class tennis backhand swing	tennis, sports, swing
Golf Swing	Demonstrating a golf swing	Sports, Golf
Bicep Curl	The correct way to practice the bicep curl	weight training, bicep curl, bicep
Fencing Lunge	Learn the standard lunge for fencing epee	Fencing, Epee, Lunge

Figure 28. “Load Motion From Library” window in the host computer application

Graphical User Interface Design

The design of the GUI came from analyzing the user interface flow chart shown in Figure 20. This chart made it clear that the two types of users (Trainers and Trainees) had different needs. To address this, the application prompted the user to choose which type of user they were, and this decision dictated which windows they would be presented. Below are screenshot images of the final GUI design along with an explanation of how the user would access the given window.


WAG bands

Settings

About WAG Bands

7 Bands Disconnected

Battery full

User selection

Connect and calibrate the WAG Bands in the settings window before use

Trainer

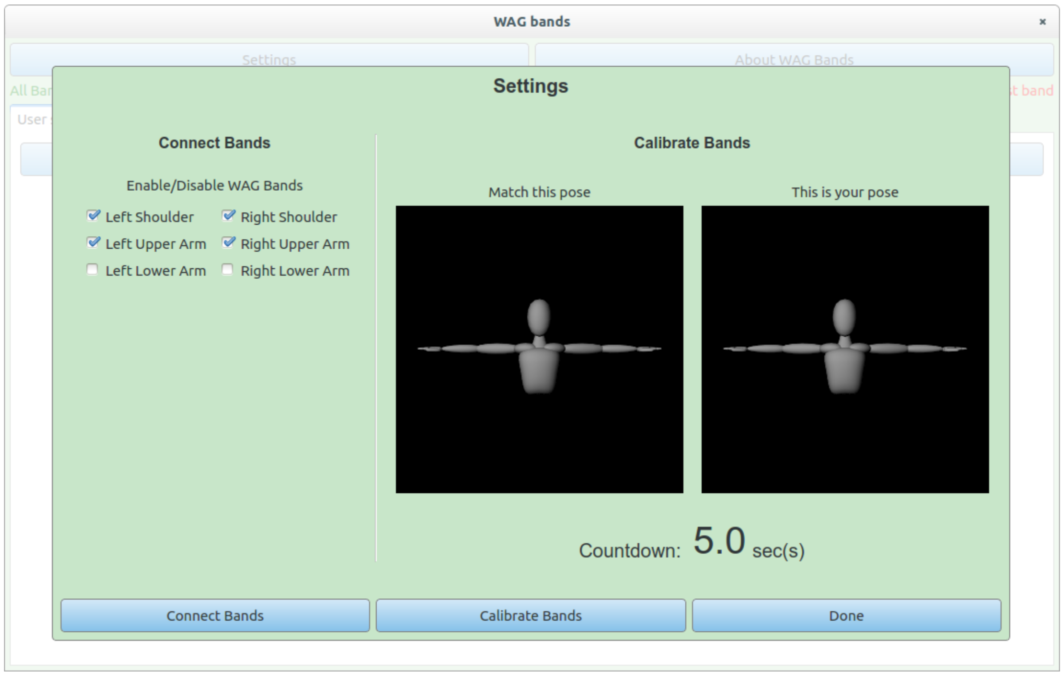
A trainer can record and save motions for others to use

Trainee

A trainee can playback motions

Figure 29. User interface – user selection

When a user first opens the software application, they are presented with the window shown in Figure 29. At the top of this window is the ‘Settings’ button and the ‘About’ button. These are available to all users at any time and remain in the menu bar. Below the menu bar, there is a status bar that displays how many bands (out of 7) are disconnected and which bands have low battery. The status bar updates in real time. In addition, if any bands are disconnected, the settings button changes from blue to red, to signify to the user that they should enter the settings panel to reconnect the bands. The majority of the window is taken up by a white tab titled ‘User selection’. This tab prompts the user connect and calibrate the bands and to select a type of user. Once the user has chosen, this tab is replaced with the content for the mode the user is in. The red connect and calibrate message was added as a result of analyzing the user test results, as discussed in the Product Performance Evaluation section.



☑ Left Shoulder ☑ Right Shoulder
☑ Left Upper Arm ☑ Right Upper Arm
☐ Left Lower Arm ☐ Right Lower Arm

Countdown: 5.0 sec(s)

Figure 30. User interface – settings screen

When the user clicks on the 'Settings' button, the overlay shown in Figure 30 is displayed. This overlay is the same regardless of user type or current window. The first thing a user should do upon launching this window is choose which bands to work with, by selecting/deselecting the checkboxes, and clicking the 'Connect Bands' button. The application tries to initiate a connection with each of the selected WAG Bands and updates the application's status bar accordingly. Next, the user needs to calibrate the bands using the two windows displaying 3D models of a human torso. The top model is in the position that the user needs to mimic for calibrating the bands, while the bottom model shows what position the system believes the user to be in. After matching the top pose, the user can click the 'Calibrate Bands' button and the system zeroes the bands' positions to those shown by the first model.

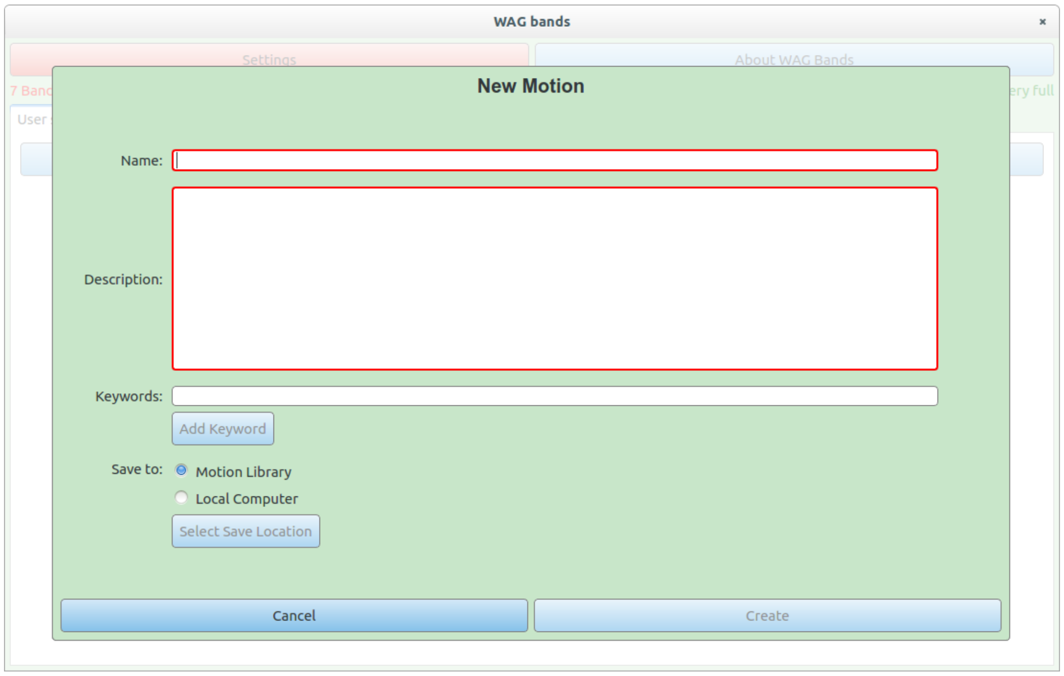


Figure 31. User interface – new motion window

If a user has chosen to be a Trainer, s/he is given the options of creating a new motion or opening an existing one. Selecting the 'Record a New Motion' option, leads them to the overlay shown in Figure 31. This window requires the user to give the motion a name and description, and optionally give the motion keywords that can help someone find the motion later. Finally, the user must choose a save location, which defaults to the motion library but can also be a directory on the user's local computer. This window has basic verification and only enables the 'Create' button if all required fields are filled.

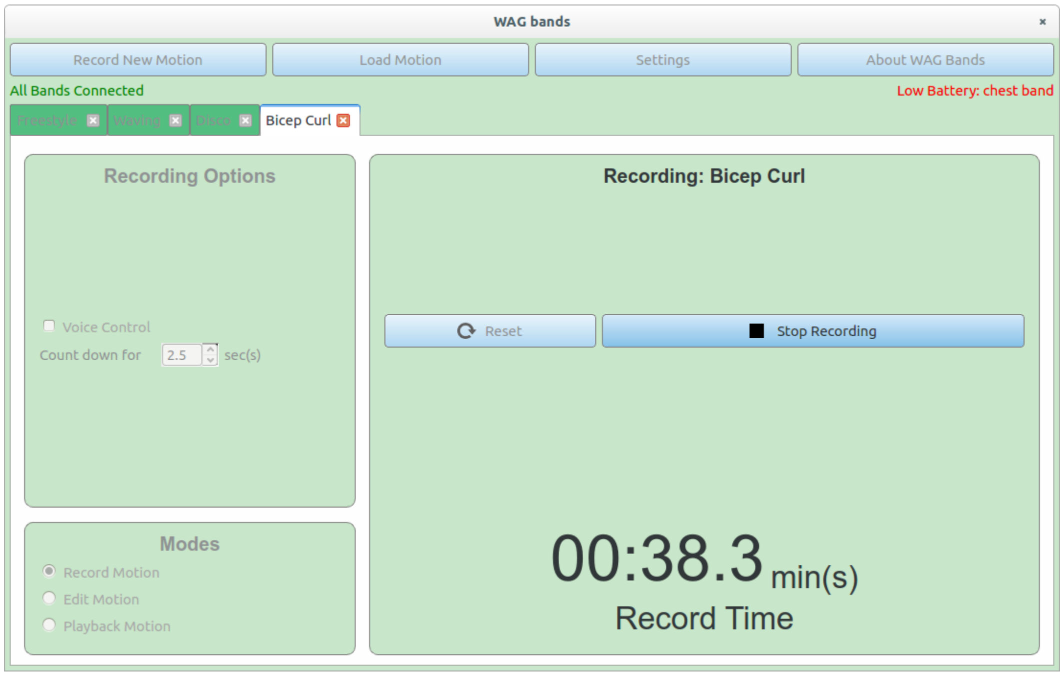


Figure 32. User interface – Trainer's record motion window

After creating a new motion, a Trainer is shown the window in Figure 32. This window signifies that the user is done “setting up” the system and is ready to begin using the WAG Bands. The Trainer now has two new buttons in the menu bar allowing them to easily create a new motion or load a previously recorded motion. Figure 32 also illustrates that a user can have multiple motion files open at once, laid out in the form of tabs under the status bar. The open tab above is displaying the application in record mode. There are a few ‘Recording Options’ available to the user, and an enabled ‘Start Recording’ button. Because the user must record a motion before editing or playing it back, the ‘Modes’ at the bottom left are initially disabled. Once the user clicks the ‘Start Recording’ button, that button changes to say ‘Stop Recording’ and the countdown runs to zero and then keeps track of the time passed during the recording (shown in Figure 32). Once the user stops recording, the other modes and reset button are enabled.

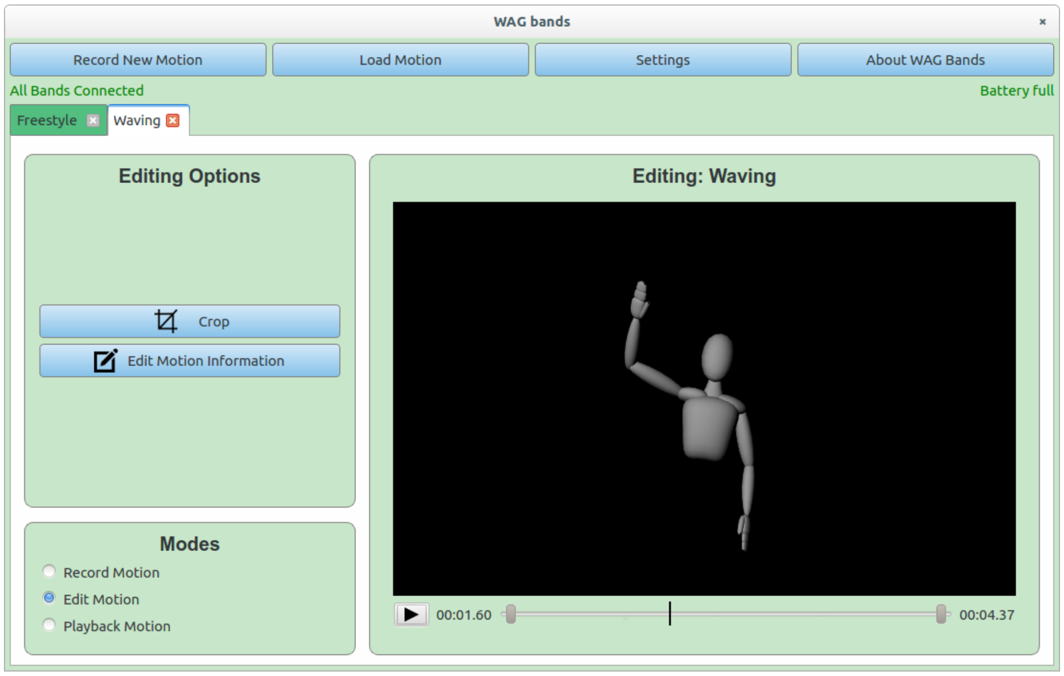


Figure 33. User interface – Trainer’s edit motion window

Once the Trainer has finished recording a motion and selected the ‘Edit Motion’ mode, the window in Figure 33 is displayed. This window has the same layout as the recording mode window to maintain consistency across the application. The ‘Editing Options’ include ‘Crop’, which allows the user to cut off the beginning or end of the motion by moving the two grey handles across the video slider at the bottom of the visualization window, and ‘Edit Motion Information’, which displays an overlay identical to Figure 31 and allows the user to edit the previously entered information. Also available in edit mode, is the option to play the motion on the 3D model, without the WAG Bands, and see what was previously recorded.

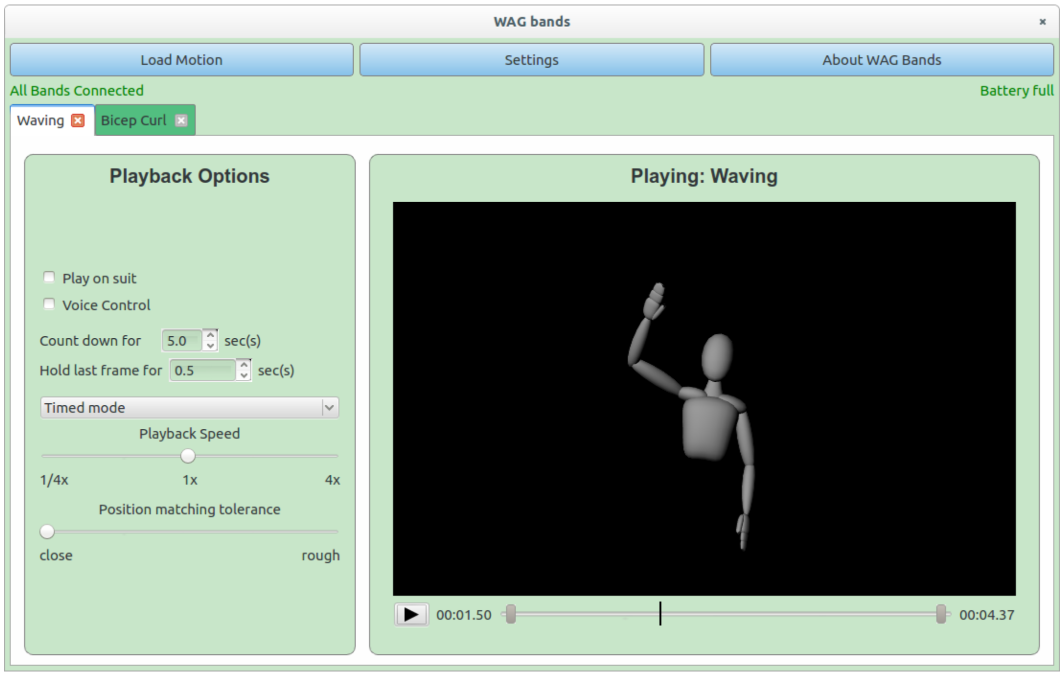


Figure 34. User interface – Trainee's playback window

The last window available for users is the playback mode window shown in Figure 34. For a Trainer this window would have the same layout – menu bar and mode selection – as displayed in Figure 33 and Figure 32. Instead, Figure 34 shows what a Trainee would see after selecting the user type. A Trainee does not have the 'Record New Motion' button in the menu bar, and does not have the ability to switch modes within the application. A user who wants to play back a motion is given the various playback options shown above. These options allow users to customize their training. To being playback mode the user must click the play button at the bottom of the visualization window, at which point the application counts down for the designated number of seconds, and begins playing the motion on the 3D model and the WAG Bands, if that option was selected.

Product Performance Evaluation

To successfully evaluate the WAG System, the project was split into three types of requirements: Hardware, Software, and Full System. Each of these categories are explained and evaluated below.

Complete System

The complete system refers to the completed set of WAG Bands as a whole. The requirements for the complete system are non-functional requirements NFR.1 - NFR.6 presented above in Table 4. This document fulfils requirement NFR.5 for comprehensive documentation.

The requirements NFR.1 and NFR.6, which involve the system's degrees of freedom and its weight limit respectively, have both been met due to the nature of the system. By using three wireless bands per arm (each with 3DOF of rotation measurement), the system can track 9 degrees of freedom per arm. The bands each weigh ~0.5lbs, multiplied by 7 bands, for a total of ~3.5lbs of wearable components, which is well under the requirement of less than 10lbs.

User Studies

NFR.2 and NFR.4, which involve a non-restrictive system and adjustability respectively, were hard to design around because of their intangibility. Both of these non-functional requirements were evaluated with user tests to determine how well test participants felt that the system adhered to these requirements. The responses collected from user tests indicated that while the WAG Bands do not inhibit the user's motion and are capable of fitting on most bodies, putting on the bands takes a significant amount of time.

Next, user tests were used to determine the effectiveness of the WAG Band vibration patterns, and the effectiveness of learning a new motion with the WAG System. The test participants were subjected to three different vibration patterns in order to determine their natural response to the translational and rotational impulses. Test subjects consistently determined that they should move away from a localized vibrational impulse, which are currently used to indicate translational error, without hesitation. Unfortunately, the rotational vibration patterns were less intuitive to the test subjects. Test subjects took longer to comprehend the rotational pattern than the translational pattern, and even longer for them to decide how to react once they identified the rotation pattern. Finally, the test subjects were unsuccessful in identifying a natural response to the combined rotational and translational vibration patterns. Some test subjects were able to clearly identify the rotational pattern within the combination, while other test subjects were able to only identify the translational pattern. No test subjects had a natural response to both components of the combined signals. These results indicate that a more natural rotation pattern should be researched and tested.

The test subjects also evaluated the effectiveness of the WAG System for learning new physical skills, in order to compare video-based learning methods with using both the haptic guidance and the built-in motion visualization. The motion learning test measured the effectiveness of the 3D visualization and vibration feedback for a simple hand-wave motion. The test results indicated that the built-in 3D visualization alone served as an effective motion teaching tool, with survey respondents rating it 4.375 on average on a 1 to 5 Likert scale for motion teaching effectiveness. When test subjects tried to learn the same motion using the vibration feedback, they generally found that the vibration was distracting, and made the vibration feedback and visualization combination a less effective teaching tool than the 3D motion visualization alone. Because the test subjects responded very well to the translational vibration patterns and not as well to the rotational vibration patterns, these results may be improved by researching better means for indicating rotational error to the wearer. Additionally, the bands vibrate with increasing intensity as a user's error increases, but the bands do not stop vibrating unless their measured errors are very small. Many test users suggested including adjustable vibration thresholds to change how closely a user needs to mimic a motion for it to be considered 'correct'. These tolerances may better indicate correct motion following to the user.

Safety Standards

Requirement NFR.3 involves adhering to safety regulations. Although there are currently no OSHA standards for exoskeletal robots, the International Standards Organization standard ISO 13482, which covers "personal care robots" including wearable exoskeletons, was referenced. According to ISO 13482, a robot is an "actuated mechanism programmable in two or more axes with a degree of autonomy moving within its environment to perform intended tasks." Consequently, the WAG System does not fall under the category of devices covered by ISO 13482. Instead, the WAG System more closely fits with the classification of robotic devices, defined as "actuated mechanism fulfilling the characteristics of an industrial robot or a service robot but lacking either the number of programmable axes or the degree of autonomy." ISO 13482 does provide guidelines for restraint-type physical assistant robot such as wearable exoskeletons, which more closely matches the WAG System. Guidelines for these robots that are relevant to the WAG System include battery charging procedures, electrical levels and component insulation requirements, and important safety and usage information. Most of these guidelines are important for consumer safety and were not strictly followed for this project prototype. Future iterations of this project geared toward introducing this system as a consumer product will strictly follow these safety guidelines.

Hardware

The hardware requirements for this project are functional requirements FR.7 and NFR.8, presented above in Table 3, and non-functional requirements NFR.8 and NFR.9, shown in Table 4. The requirements FR.7 and FR.8 involve haptic feedback and its response time. FR.7 is met through the design of a WAG Band, which includes a ring of vibration motors which encircles the user's arm or shoulder. FR.8 requires the system to have a haptic response time of 100ms (10Hz), meaning that the system needs to respond to an error within 100ms. The WAG System response frequency was found to be within ~16Hz, as calculated using Wireshark. This corresponds to a ~62ms response time, satisfying this requirement (see the ESP8266 Testing section for details).

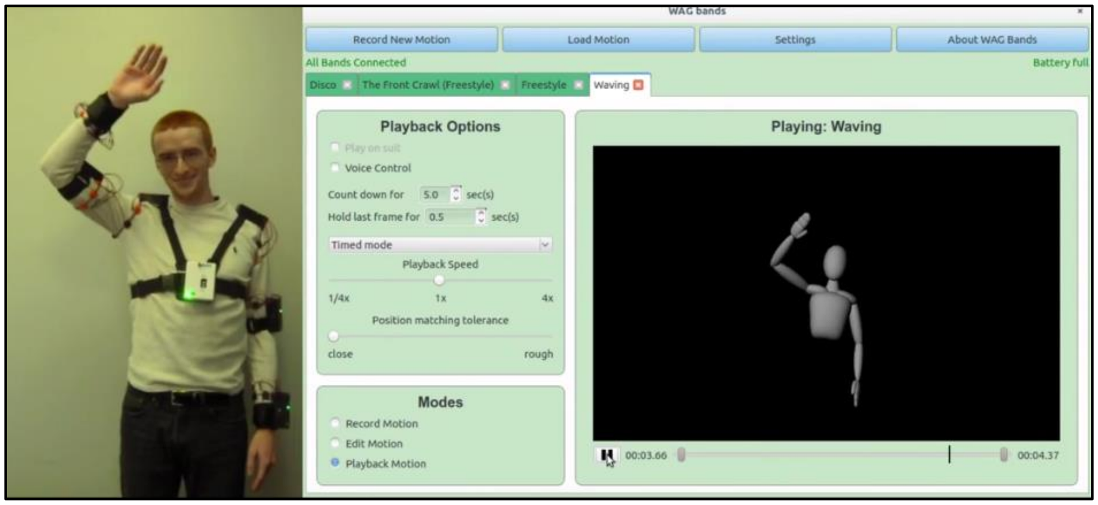
The requirements NFR.8 and NFR.9 involve the battery life and how long a user can play a motion and record a motion respectively. NFR.8 refers to the playback battery life which should be at least an hour. This is different than NFR.9 the recording battery life requirement of 2 hours, as playback draws much more current than recording. The system draws about 350mA in playback and about 200mA in recording (as measured from a power supply), and as such the battery lives are ~2.8hrs and ~5 hours respectively; which fulfill both requirements. More detailed information on testing the hardware can be found in the Hardware Testing section of Appendix 5 – Test Plans).

Software

The key software requirements include FR.1 - FR.6, FR.9, and NFR.7 from Table 3 and Table 4. These include maximum allowed record time, editing capabilities, save/load capabilities, playback controls, simulation, ghosting, and an intuitive user interface (UI). Each of these features, excluding ghosting and the intuitive UI, underwent unit testing and integration testing to verify that they were met. The ability for the software application to show ghosting, a priority 3 requirement, was not completed due to time constraints. The user stories provided more detailed functionality to include in the software application. Over the course of development, each of the functionalities described in the user stories was addressed and added to the final application.

Integration Testing

Integration tests were completed to verify that all components worked together. The most extensive test completed involved recording a motion with the WAG Bands, editing it, and playing it back with the 3D model. This test verified that the system could record the positions of the bands over time, save them to a file, and play them back on the 3D model in the application. This test, which was successfully completed, was videotaped and is shown in Figure 35.



Count down for 5.0 sec(s)
Hold last frame for 0.5 sec(s)
Timed mode

Figure 35. Integration test with recording (on the left) and playback (on the right)

User Studies

To meet the intuitive UI requirement, a Human Computer Interactions professor reviewed an early version of the WAG System graphical user interface and provided suggestions to help make the application more user friendly. User studies were also conducted to determine the usability of the interface. The feedback from both of these sources were used in the final design of the software application's interface.

User studies conducted with a focus on GUI intuitiveness indicated that the user interface was well-designed and easy to use overall. Test subjects were assigned specific tasks in the interface with no other guidance (e.g. record a new motion, play back a motion, etc.). Test subjects consistently rated the simplicity of completing these tasks between 4 and 5 on a scale of 1 to 5 for simplicity and intuitiveness, with 5 being the most intuitive. However, users consistently failed to connect and calibrate the bands when they tried to record motions, indicating that the software's existing visual cues were not clear or substantial enough for users to take those necessary action. As a result, the calibration and connection options were made much more prominent in the GUI, and the software will not allow the user to record or play back motions without first connecting and calibrating the bands.

Risk Management

Risks for this project included a limited time frame, budget, and previous experience. To mitigate these, systems engineering practices were followed and a concept of operations was created before starting development of the WAG System. This kept the project under budget and helped to ensure milestones were reached on time. To mitigate a lack of previous experience, the team met with the project advisors every week for updates and suggestions.

Project Execution Performance Evaluation

Execution Summary

This project explored the fields of wearable devices, haptic feedback, and motion capture. The WAG System has the potential to influence athletic training and rehabilitation by providing accurate, direct, and exact feedback to the wearer. The research and development leading to this achievement faced many challenged and endured several setbacks. Wireless communication was one key component of the system design which proved to be perhaps the hardest challenge. Bluetooth was originally intended for the communications system, but after much testing, it was found to be insufficient for the project purpose due to the complexity of library dependencies for the computer software, and a low slave device limit, and insufficient data rates. This setback was costly and time consuming but lead to the implementation of Wi-Fi for system communication.

Wi-Fi hardware also experienced a few issues including stability of the Wi-Fi chip (ESP8266) and the code base for the ESP8266. It took several weeks to develop a stable and usable code base after throwing out the original firmware and choosing a community-supported Arduino code base. Part of increasing the stability of the platform came in changing the hardware to have an additional capacitor due to the chip drawing current relatively erratically. An additional circuit board was designed specifically to program the module to increase its ease of use for testing.

The first order of printed circuit boards was completed and assembled in early January. However, the voltage rails for several key components had been combined resulting in incoming battery and regulated voltages being merged. While originally this seemed to be passable for the project, testing with batteries showed this to be a critical error that required redesign. The final printed circuit boards were designed in late February and re-ordered. This took around a week, but only cost around $20.

Another setback found later in development was the presence of memory leaks in the software application. These leaks caused segmentation faults while running the application. The leaks were identified using the open source program Valgrind and patched manually. This delayed the development software application and shifted testing much later in the project.

Timeline Adjustments

The final timeline has shifted quite substantially from initial estimates. The first major reason is that the project completion date was extended from early March to early May. This was done as a result of the balance of team workloads around the project. Many tasks are lengthy and also interdependent on each other. Consequently, it made more sense to accomplish small pieces in parallel so that subtasks could be tested with each other, rather than completing a major task quickly and not being able to thoroughly test how it interacts with other subsystems. Another major change was the shift of communications earlier in the schedule and other tasks later. After developing the core functionality of the GUI, fully implementing communications was necessary for meaningful testing and development of other subsystems. Once Wi-Fi communications functionality was implemented, it was much easier to test other subsystems as well as determine how the remaining portions of the system would interface with each other. The final timeline can be seen in Appendix 1 – Timeline.

Budget and Expenditure Justification

The budget for the project was set at $1000 with university departmental funding at $200 per student (5 students). The actual spending was $978.31 out of the total. This can be broken down into large categories within Table 9. Each item shows the cost associated with it and how it factored into the project. For some of the costs under communications testing and sensor testing, these were for parts that served no role in the final project (such as the flex sensors). These parts were considered for a hand module that never was developed due to time restrictions. The Bluetooth modules were also not used as mentioned above and served as part of the research costs associated with this project. A full listing of materials in each final WAG Band and the WAG Chestpiece can be seen in Table 12 and Table 13. Note that the costs in Table 9 do not include any human costs such as labor, since this is a student project.

Table 9 - High level budget breakdown

Part description	Company	Cost	Justification
BTLE USB Dongle	Amazon	$12.95	Communications testing
Adafruit nRF8001 module	Adafruit	$32	Communications testing
Adafruit Bluefruit LE UART Friend	Adafruit	$29	Communications testing
CC3000 Wi-Fi Board	SparkFun	$39.08	Communications testing
Bluetooth 4.1 chips with antenna	DigiKey	$24.77	Communications testing
Kinivo Bluetooth 4.0 Adapter	Amazon	$13.99	Communications testing
ESP8266 Wi-Fi modules	Zou Ting	$30.72	Communication modules
ESP8266 Wi-Fi module	DigiKey	$13.90	Communication modules
MPU6050 Modules	Amazon	$19.92	Sensors for project
30 ERV motors	Polulu	$89.15	Feedback motors for project
Shaftless vibration motors	Pololu	$35.35	Feedback motors for project
Stranded Wire for Motors	SparkFun	$10.50	Feedback motors for project
5 Teensy 3.2 boards	PJRC	$102.22	Microcontrollers for project
Teensy 3.2 Modules (x2)	PJRC	$62.34	Microcontrollers for project
Connectors for PCBs	DigiKey	$8.56	PCB parts for motors
Newark PCB parts (round 1)	Newark	$29.24	PCB parts for boards
PCBs from advanced circuits (round 1)	Advanced Circuits	$23.05	PCBs
PCBs (round 2)	Advanced Circuits	$24.35	PCBs
Newark PCB parts (round 2)	Newark	$26.65	PCB parts for boards
Chest harness	Amazon	$43.03	Band straps
1 1000mAh Battery	HobbyKing	$8.72	Batteries for project
8 1000mAh Batteries	HobbyKing	$28.00	Batteries for project
Battery charger and splitter	HobbyKing	$50.30	Batteries for project
Easy VR Shield 3.0 (speech recognition)	SparkFun	$54.17	Speech recognition
Flex sensors	SparkFun	$25.90	Sensor testing
Force sensors	SparkFun	$18.03	Sensor testing
Freescale Freedom board Kinetis K22F	DigiKey	$37.21	Sensor testing
Freedom Board FXAS21002	DigiKey	$15.95	Sensor testing
4A power supply for Intel Atom	Adafruit	$24.53	Cornell Cup Competition
60GB SSD for Intel Atom	Amazon	$44.94	Cornell Cup Competition
	Total costs:	$978.31	
	Budget total:	$1000	
	Under budget:	$21.69	

Project Reviews

This section addresses the reviewer feedback from advisors during the two review sessions. There were two major design reviews called the Preliminary Design Review (PDR) and the Critical Design Review (CDR) which are included in the university curriculum for capstone projects. These sessions are designed to give advisors and general university students an opportunity to critical and evaluate a design by listening to a 20 minute presentation and then providing questions and feedback. Typically the PDR occurs early on in the beginning of the design phase and seeks to refine the project proposal. The CDR occurs later in the design phase to verify the design for the project is feasible and applicable to the original design. In addition to the PDR and CDR, the project advisors were consulted weekly for advice and feedback. These meetings helped the team follow better systems engineering practices, overcome many of the project challenges, and produce a patentable technology. The advice and feedback of the advisors led the team to meet with industry professionals, technical advisors, and the campus patent lawyer. These connections gave the team access to valuable resources and advice which the team was able to integrate into the design.

By the PDR, the team had developed a full project proposal that outlined the details of the WAG System. The feedback provided by the advisors at this meeting helped to define the scope of the project. Due to the budget and time limitation, it was decided that the project would be limited to upper body only. At the time of the PDR, plans for adding flex sensors for hand motion tracking were a major component of the project, but it was decided that hand motion tracking should be a low priority in order to make sure torso and arm tracking was completed first. The overall outcome of the PDR was to begin taking this theoretical system and turning it into a realistic design.

Prior to the CDR, a preliminary design was complete and most parts were ordered to allow for assembly to begin in January. The feedback from this presentation was primarily on presentation quality rather than content and technical details. While the team had made good progress on the technical side of the project, the presentation was very technical and failed to address some of the larger goals of the project.

The team improved non-technical aspects of the presentation the following week. The team presented to Rita Vasquez-Torres, a Senior Technology and Programs Strategist who was asked to come evaluate the WAG System. Working with the feedback from the CDR, the team kept in mind that the potential lack of familiarity of audience members with the WAG System. The presentation went very well and Mrs. Torres provided lots of valuable information about preparing a product for market, which would later lead to the team filing for a provisional patent.

Recommendations and Future Research

A useful investigation following development of the WAG System would be determining how useful it is for teaching people new physical skills. This research could examine stimuli patterns that could be used with the WAG Bands to indicate error in motion, as well as how the WAG System compares to teaching motion using a trainer or video instruction. This would require longer and more numerous tests on human subjects as well as researchers with more background in experiments involving learning.

In addition to investigating the usefulness of the WAG System, it would also be useful to investigate different vibration schemes and thresholds. Test users reported that translational feedback was intuitive, but were typically confused by rotational feedback and the combined feedback. If other rotations schemes were tested, a more intuitive and consequently more useful scheme could be implemented instead. In addition, users noted confusion at the degree of vibration that was present during most of their attempted motions. Testing different ranges for the acceptable margin of error for the user, as well as trying different scale factors relating error and vibration could make the feedback more easily interpreted by the user. Additional user tests with various vibrational schemes would be necessary to determine these margins of error and scale factors. Developers would also have to implement ways to vary the vibration scheme for different trials. This could involve settings to select the vibration scheme or different implementations that would be switched between trials. Depending on the approach, it may be necessary to send the vibration signal to the bands instead of the angular error to simplify modification of the vibration scheme. This change would require modifying the playback message structure in the software application, and the processing done on received packets in the code for the WAG Bands.

One potential future addition to the project would be the ability to record motion for other portions of the body, including the abdomen and legs. Adding motion capture for these components would increase the number and utility of motions that could be recorded. This would likely require more biomechanics knowledge due to increased complexity of the additional body segments. The physical band layout would also need to be heavily revised to be able to capture these different body segments. Developing wearable units for measuring hand movement would allow users to learn more actions involving hand movements.

Another expansion to this project could involve developing a series of modules that could interface similarly with the software that could support force feedback. Force feedback would be useful as it would help to physically guide users through the correct paths, rather than simply alerting them to errors in their motion. This force feedback could be implemented through motors driving parts of the module or soft actuators. This would also likely require greater knowledge of biomechanics, as precision and accuracy needed in applying force to a user would need to be greater than for haptic feedback as to prevent injury. This expansion would also require significant knowledge of the actuation mechanisms used to implement force feedback. Adding actuation would also be a costly and time consuming addition, as most hardware components would have to be redesigned, and actuators are much more expensive than vibration motors. Additional time and effort would be needed to ensure that the safety issues raised by applying force to humans are mitigated and managed appropriately.

In addition, implementing visual playback ghosting, where the users can see their actual path and intended path into the motion visualization window could improve the usefulness of the WAG System. Currently, the system conveys positional error to users through vibration. Extending the system so that users could also visually review their motion path alongside the intended motion path could help them remove errors more quickly. This would require that PositionSnapshot objects also be aggregated into a timestamped data structure during playback as well as during recording and that the visualization would be capable of displaying and updating two body models as a motion is played. This feature was frequently requested in the open-ended feedback section at the end of the user studies.

Enabling compatibility with a wider variety of motion capture file formats could also be a valuable extension to this project. This greater compatibility could either be through direct compatibility with the application, through which users could directly play or save motions with other motion capture file formats, or through development of file converters, which could be used outside of the main flow of the application to convert files between the standard used by the WAG software and other file available file formats. Compatibility with other file types could allow users to pull motions to learn from existing databases as well as to use the WAG software for other motion capture purposes. Having a greater range of databases to pull motions from would likely increase the usefulness of the application, as motions would not need to be created specifically through the WAG software to be learned.

Enabling live recording and playback between a Trainer and a Trainee could also be a valuable extension to the project. This feature could allow the WAG System to enhance in-person teaching sessions by providing tactile feedback in addition to the verbal and visual feedback that an instructor typically provides. Adding live playback would require that the system draw PositionSnapshots to match from a secondary suit object, corresponding to the set of bands worn by the Trainer, instead of from a file. It would also likely require improved communication and processing speeds in the application, as the software would have to receive position data for two sets of WAG Bands instead of just one. This could eventually be expanded to have multiple students receiving live feedback from one instructor in a group class environment.

Reducing the cost of the WAG System would also be a useful next step in development of this project. The current cost of materials per band is between $70 and $80. To be more available and affordable for potential consumers, this cost should be below $10 per assembled device. However, this would require a redesign the breakout boards to use just the components of the breakouts rather than the full breakout. This will increase development time but will make the overall footprint of the device smaller and will lower the cost. Buying circuit component items in bulk would also decrease the cost.

Another improvement for this project would be reducing the overall latency in the system (to increase the duplex communication rate between the computer and the bands). Next year, the ESP32 chip (an improvement on the ESP8266 hardware) is being released and could potentially increase the communication. The ESP32 chip is also set to use less power than the ESP8266 chip and could help increase the potential battery lifetime of the bands. Alternatively, better communication chips such as Bluetooth could be used if better libraries (embedded and host computer side) were available. The use of Bluetooth could decrease the power usage of the circuit as well.

Should this project be developed for consumer use, miniaturizing the hardware would allow the WAG System to be less intrusive and obstructing to motions of the user. The existing components could be replaced with smaller models. The hardware size could also be reduced without switching components by soldering components together rather than allowing them to be removable and reprogrammable.

Other sensors that monitor body functions could be added to form a more complete picture of the user's body while they are performing a motion. These could include blood-oxygen sensors, breathing monitors, and heart rate monitors. Adding these would involve either developing more bands or adding components to existing bands. Additional message types would be needed to process the data from these sensors. The GUI would also need to be modified to display this information to users while they are executing motions or after the execution of motions.

Incorporating long term performance data for motions could also allow the user to better track their progress over time. If performance data was stored, the software could be extended to analyze this data to look for habitual errors or suggest other exercises or stretches based on individual progress. Simply storing historical data would require adding specific storage for each user and saving either each run of an exercise or analysis data calculated from each exercise. Knowledge of artificial intelligence and exercise science would be needed to identify persistent errors or suggest potential exercises or stretches for the user to improve their performance.

Nomenclature Glossary

- ADC – Analog to Digital Converter
- AES – Advanced Encryption Standard
- Breakout board – A small printed circuit board designed to give users easy access to circuit functionality and onboard peripherals
- Bootloader – A section of code installed into low memory of a microcontroller to run user programmed code
- BJT – Bipolar Junction Transistor; An electrical component used commonly as an amplifier
- C++ – A middle level programming language good for hardware communication and object oriented programming
- CDR – Critical Design Review; a design review with advisors completed mid-way through the design phase to verify system design and concepts
- Crimp connector – A style of metal contact that is affixed to the end of a wire with a combination of pressure and solder
- CPU – Central Processing Unit
- DAC – Digital to Analog Converter
- DIP Socket – Dual Inline Package; A style of connector to enable integrated circuits to be connected to a printed circuit board without solder
- EAGLE – A PCB design software developed by CadSoft
- EasyVR3.0 – A commercial breakout board for speech recognition
- ERV – Electric Rotary Vibrator
- Eval Board – Evaluation Board. A PCB designed for testing electrical hardware, such as microcontrollers and active sensors.
- FOGS – Fiber Optic Gyroscope
- FR – Functional Requirement
- GND – Electrical Ground
- GPIO – General Purpose Input/Output pins on a microcontroller
- GUI – Graphical User Interface
- I2C – I^2C, Inter-Integrated Circuit communication protocol
- IC – Integrated circuit; An electrical component designed for a generic function
- ICM – Industrial, Commercial, and Medical
- IDE – Integrated Development Environment used for writing code for an application
- IoT – Internet of Things
- ISO – International Standardization Organization
- JTAG – Joint Test Action Group
- LED – Light Emitting Diode; a device which produces light given a voltage of ~2V to ~3V
- Li-Po Battery – Lithium Polymer battery; This style of battery features high current capabilities and fast recharge rates with a large batter capacity
- LM358 – An amplifier IC used with a single supply (only a positive voltage and 0V).
- LPF – Low Pass Filter
- LRA – Linear Resonant Actuator
- MATLAB – Numerical computing environment and programming language
- MCU – Microcontroller
- MEMS – Micro Electro Mechanical System
- MOSFET – Metal-oxide-semiconductor field-effect Transistor, basically a digital switch

- MPU6050 – Accelerometer and Gyroscope sensor breakout board available for around $4
- NFR – Non-Functional Requirement
- NPN – A type of BJT transistor
- OpenGL – Open Graphics Library for rendering 2D and 3D vector graphics
- OSHA – Occupational Safety and Health Administration
- PCB – Printed Circuit Board
- PLA – Polylactic Acid; material used for 3D printing
- PositionSnapshot – mapping of band type to band position; represents full body position measurable by bands at a given time
- Potentiometer – An electrical component which varies its resistance as a knob is turned
- PDR – Preliminary Design Review; a design review with advisors completed at the beginning of the design phase to explain and refine the project proposal
- PWM – Pulse Width Modulation
- Quaternion – A four-dimensional representation for expressing rotational position
- Qt – Cross-platform application development framework for C++ with supporting libraries
- RAM – Random Access Memory; used to store variables at run time in an application
- Regulator – A voltage regulator takes in a higher voltage and provides a constant steady voltage output
- RF – Radio Frequency
- RTC – Real Time Clock
- Sensor Fusion – a technique used to combine sensory data from separate sources
- Software application/the application – The software application (GUI) responsible for collecting information from the WAG Bands and processing user input and preferences
- SPDT Switch – Single Pole Double Throw; A style of switch that has three connection pins and two positions of the switch (the middle of the three and one of the outer connection pins are always connected in the states)
- SPI – Serial Peripheral Interface Bus
- Suit – Another term for the collection of wearable bands
- TCP – Transmission Control Protocol
- TLV1117 – A 3.3V voltage regulator produced by Texas Instruments
- Trainee – Person using the WAG System to learn a new skill
- Trainer – Person using the WAG System primarily for recording actions
- UART – Universal Asynchronous Receiver/Transmitter
- USART – Universal Synchronous/Asynchronous Receiver/Transmitter
- User – Person using WAG System – either a Trainee or a Trainer
- Vcc – Supply Voltage
- WAG – Wearable Action Guidance
- WAG System – Full system including seven wearable bands (WAG Bands) and the accompanying software application
- WAG Bands – Wearable bands containing position measurement devices and vibration motors
- WAGFile – Software object containing motion file data; file containing motion file data
- WPAN – Wireless Personal Area Network
- XML – Extensible Markup Language; human and machine readable markup language

Appendix 1 – Timeline

This appendix contains the timeline for the project split up into chronological subsections. Each subsection contains additional subtasks, a start time and a duration.

Research

This was the initial research phase of the project.

- Subtasks: background research, trade study
- Duration: 16 weeks (through November 2015)
- Start Time: Late August 2015

Systems Engineering Analysis

This included the concept of operations and resulted in the delivery of the initial systems engineering content.

- Subtasks: stakeholders analysis, needs analysis, use case development, requirements analysis
- Duration: 5 weeks (through mid-October 2015).
- Start Time: Mid-September 2015

High Level Design

This involved the creation of the high level design of the system including the concepts of bands and the computer application.

- Duration: 2 weeks (through late October 2015).
- Start Time: early October 2015

Basic Hardware Block Diagrams

This included laying out the block diagrams for the hardware functionality. The deliverables included diagrams for both the bands and the chest piece as well as interaction diagrams between hardware subsystems.

- Subtasks: Laying out hardware functionality, determining hardware subsystems (microcontroller, wireless interface, motor controllers etc.), determining interaction medium between subsystems
- Duration: 1 week (through mid-October 2015).
- Start Time: early October 2015

Core Software Class Design Document

This involved the creation of a document containing core classes for the software application. The document contains functionality of each class and how the classes communicate with other classes

- Duration: 2 weeks (mid-October 2015)
- Start Time: early October 2015

User Interface Development

This involved creating and implementing a UI design using Qt's widget library of buttons, tabs, sliders, text boxes, etc. so that the user can navigate and access the functionality of the WAG System.

- Subtasks: Settings display, open motion display, motion library display, recording display, editing display, playback display, new motion display

- Steps:
    - Design: November 2015
    - Build: Mid-November 2015 through December 2015
    - Preliminary Review: Early January 2016
    - Build: January 2016 through March 2016
    - Test: January 2016 through April 2016
    - Document: Mid-February 2016 through April 2016
    - Review: March 2016 through April 2016

Accelerometer/Gyro Integration

This task included testing and integrating the MPU6050 into the WAG Band and chest piece hardware.

- Subtasks: MPU6050 integration
- Dependencies: This task was independent from other hardware subsystems.
- Steps:
    - Design: started October 2015, took 2 weeks (through mid-October 2015)
    - Ordering: two rounds of ordering sensors (early October and November 2015)
    - Build: early October 2015 (testing), November 2015 (assembly with first round of PCBs), February 2016 (final PCB integration)
    - Test: early October 2015 (testing), November 2015 (assembly with first round of PCBs), February 2016 (final PCB integration)
    - Document: from October 2015 through March 2016
    - Review: from October 2015 through March 2016

Haptic Feedback

This task involved the design, integration and testing of the motor controllers along with the vibration pattern.

- Subtasks: Scheme determination, mapping error to vibration signal, vibration motor attachment
- Dependencies: This task was independent from other hardware subsystems
- Steps:
    - Design: October 2015, took 8 weeks through November 2015
    - Ordering: October 2015 and November 2015
    - Build: November 2015
    - Test: October 2015 through November 2015. March 2016 through May 2016
    - Document: November 2015 through April 2016
    - Review: January 2016 through May 2016

Hardware Low Battery Detection

This task involved the testing and installation of the low battery detection circuit.

- Subtasks: Battery installation, low battery circuit design, testing and calibration of circuit at desired low battery voltage
- Dependencies: The task has only dependent on the operating voltage of the system which was decided during design in October 2015
- Steps:
    - Design: Early November 2015 through mid-November 2015
    - Ordering: Mid-November 2015 (1 week)

  - Build: Late November 2015 through December 2015
  - Test: Mid-November through late November 2015
  - Document: Late November 2015 through April 2016
  - Review: Late November 2015 and January 2016 through May 2016

Hardware Voice Control

This task involved the development of the embedded hardware and software to enable the voice control on the WAG Chestpiece. The deliverable was the hardware setup that integrated this into the chest piece.

- Subtasks: Voice control module setup, interface voice control module with other hardware, determine and implement appropriate voice commands for stopping/starting motion and calibration
- Dependencies: The development of this was dependent on the microcontroller interface i.e. which microcontroller communicated with the voice control module.
- Steps:
  - Design: Early November 2015
  - Ordering: Mid November 2015
  - Build: Prototype in mid-November 2015, final system integration in March 2016
  - Test: Mid-November through March 2016
  - Document: February 2016 through April 2016
  - Review: March 2016 through May 2016

Wireless Communications

This task involved testing Bluetooth communications initially and then involved the switch to Wi-Fi and the integration of the Wi-Fi communication hardware into the WAG Bands. On the software application, this involved including Wi-Fi processing code.

- Subtasks: Bluetooth feasibility (connection management on computer and device, data rates), connection setup, multiple connection maintenance, message parsing, message encoding, state machine setup within the EPS8266, integration of UART with Teensy, increasing stability of the module
- Dependencies: This task was dependent on the microcontroller choice.
- Steps:
  - Design: Mid-October 2015 through December 2015
  - Ordering: Mid-October 2015 (1 week)
  - Build: Mid-December 2015 through February 2016
  - Test: Mid-January 2016 through March 2016
  - Document: Mid-January 2016 through late January 2016
  - Review: Late January 2016 through early February 2016

Printed Circuit Board Design

This task involved the creation of the PCBs for the project to integrate all of the hardware components. This task featured one original design period and a design revision where a second round of PCBs were ordered to correct several errors.

- Subtasks: Designing PCBs, Ordering PCBs, Revising PCBs, Assembling PCBs, Testing PCBs
- Dependencies: This task was dependent on all of the hardware testing tasks to determine the components needed for the WAG Bands and the chest piece
- Steps:

    - Design: Mid-December 2015 (1 week), 2nd round: Late February 2016
    - Ordering: Mid-December 2015 (1 week), 2nd round: Early March 2016
    - Build: Early January 2016 through late March 2016
    - Test: Mid-January 2016 through April 2016
    - Document: Mid-December 2016 through April 2016
    - Review: Late January 2016 through late February 2016

Wearable Device Design and Construction

This task involved the physical construction of the WAG Bands and the chest piece.

- Subtasks: Prototyping band designs in SolidWorks, 3D printing of band cases, strap development, vibration motor housing design and printing, PCB design
- Dependencies: This step depended on the final choice for vibration motors and the PCB sizing
- Steps:
    - Design: November 2015 – February 2016
    - Build: Mid-December 2015 – March 2016
    - Test: January 2016 – March 2015
    - Document: January 2016 – February 2016
    - Review: February 2016 – march 2016

Recording

This task involved getting data from the wearable bands and storing it into a motion file.

- Subtasks: parse position data, aggregate into snapshots, store snapshots, signal recording start, signal recording stop
- Dependencies: Position Determination, Wi-Fi Communications
- Steps:
    - Design: Mid-December 2015 through January 2016
    - Build: January 2016 through late January 2016
    - Test: Late January 2016 through late February 2016
    - Document: Late January 2016 through late February 2016
    - Review: February 2016 through March 2016

Motion Saving and Loading

This step involved being able to save and load motion files from the motion library or the user's computer.

- Subtasks: serialize motion metadata, serialize position state, serialize mapping of bands to position states (serialize a position snapshot), serialize mapping of times to position snapshots, deserialize metadata, deserialize position state, deserialize position snapshot, deserialize mapping of times to position snapshots
- Dependencies: Recording
- Steps:
    - Design: February 2016 through mid-February 2016
    - Build: Mid-February 2016 through Early March 2016
    - Test: March 2016 through late March 2016
    - Document: March 2016 through late-March 2016
    - Review: March 2016 through April 2016

Visualization

This step involved being able to review a motion performed using a 3D human model in playback and editing modes.

- Subtasks: create human model, export model, load extra data from Blender model, load model into software, configure OpenGL, configure nodes to receive world orientation updates while staying attached to parent
- Dependencies: User Interface Development
- Steps:
  - Design: February 2016
  - Build: February 2016 through March 2016
  - Test: March 2016
  - Document: early March 2016
  - Review: late March 2016

Playback

This involves being able to play a collected motion and trigger seeing the motion in the visualization and sending appropriate error messages to each of the bands.

- Subtasks: determine desired snapshot, compare desired snapshot with current position to determine error, serialize error message, send error message, signal playback start, signal playback stop
- Dependencies: Haptic Feedback Scheme Development, Recording, Position Determination, Wi-Fi Communications
- Steps:
  - Design: January 2016 through late January 2016
  - Build: Late January 2016 through mid-February 2016
  - Test: Mid-February 2016 through March 2016
  - Document: Mid-February 2016 through late March 2016
  - Review: Late February 2016 through March 2016

Calibration

This involves calibrating the accelerometer/gyroscope sensors themselves so their filtering algorithm generates stable, non-drifting orientation measurements, defining coordinate frame transformations from the sensor axes to the visualization model axes, and implementing the pose-matching calibration algorithm to align orientation updates based on fixed rotational offsets.

- Subtasks: calibrate MPU6050 sensors, calculate coordinate frame transformations from sensor axes to model axes, develop pose-matching math
- Dependencies: Recording
- Steps:
  - Design: December 2015
  - Build: February 2016
  - Test: March 2016
  - Document: March 2016
  - Review: late March 2016

Motion Library

This involves saving and loading files to a motion library and allowing users to easily interface with the library.

- Subtasks: setup motion library directory, design interface, implement interface, connect to relevant application windows, implement search bar
- Dependencies: Motion Saving and Loading
- Steps:
  - Design: Late February 2016
  - Build: March 2016
  - Test: Late March 2016
  - Document: March 2016
  - Review: Late March 2016

Edit Motion

This involves allowing users to crop their motion and edit the motion metadata, including name, description, keywords, and save location.

- Subtasks: add buttons, re-implement the new motion window for editing, implement crop by interfacing with the editingController and WAGFile
- Dependencies: Recording, Motion Saving and Loading
- Steps:
  - Design: Mid-February 2016
  - Build: March 2016
  - Test: Late March 2016 through Early April 2016
  - Document: Early April 2016
  - Review: Mid-April 2016

Software Voice Control Integration

This task involves configuring the software to parse the voice control messages and trigger calibration or start and stop playback or recording.

- Subtasks: send voice control data to computer, parse voice control messages sent to application, voice control trigger computer action
- Dependencies: Hardware Voice Control, Wi-Fi integration
- Steps:
  - Design: February 2016 through mid-February 2016
  - Build: Mid-February 2016 through late February 2016
  - Test: Mid-February 2016 through mid-March 2016
  - Document: Late February 2016 through late March 2016
  - Review: March 2016 through late March 2016

Software Low Battery Notification

This involves getting low battery signals from the wearable bands and displaying and removing low battery notifications from the GUI.

- Subtasks: low battery message parsing, software application notification on low battery
- Dependencies: Hardware Low Battery Detection, Wi-Fi integration,
- Steps:
  - Design: Mid-January 2016 through late January 2016

- Build: Late January 2016 through February 2016
- Test: February 2016 through mid-February 2016
- Document: February 2016 through late February 2016
- Review: Mid-February 2016 through March 2016

Appendix 2 – Software Core Classes and Functionality

Since a provisional patent was filed partially based on the software developed for this project, the full software for replication was withheld from this report. This includes both the software for the embedded hardware (band hardware including the Wi-Fi chip, the Teensy 3.2 and the ATMega328P) and the C++ code to run the Qt application on the user's computer. This appendix includes tables outlining classes and their functionality, for both the software application and the embedded software.

Table 10 - Core Software Classes and Functionality

Class Name	**Description**	**Fields & Methods**
WifiManager	Handles all communications with hardware bands	Fields: HashMap<BandType, socket> socketMap Methods: void sendMessageToBand(BandType destinationBand, BandMessage msg) void initiateConnection(QList<BandType> bandsToConnect) void sendRawDataToBand(BandType destinationBand, QByteArray data) Signals: void dataAvailable(BandType recvdFrom, BandMessage data, QElapsedTimer timestamp)
BandMessage	Class for parsing and serializing data for Wi-Fi transmission	Fields: MessageType msgType QByteArray msgData Methods: BandMessage(MessageType, QByteArray msgData) BandMessage() void parseFromByteArray(QByteArray recvdPacket) QByteArray getSerializedMessage()
PositionSnapshot	Contains position information for a given time for multiple bands in a suit Represents the user's full position at a given time	Fields: map<BandType, AbsState> positions Methods: void addMapping(BandType, AbsState) QSet<BandType> getRecordedBands() QHash<BandType, AbsState> getSnapshot()

Class Name	Description	Fields & Methods
WAGFile	Contains information about a single motion recording.	Fields: map<int, PositionSnapshot> positions QString name QString description QString author QVector<QString> tags Methods: int getLastFrameNum() PositionSnapshot getSnapshot(int frameNumber) QHash<int, PositionSnapshot> getSnapshotsInFrameRange(int startFrame, int endFrame)
AbsPose (abstract)	Maintains calibration and can return an AbsState object representing the pose of the band at the current time	Fields: AbsState current AbsState calibration Methods: void calibrate(AbsState calibrationPose) AbsError error(AbsState desiredPose) void update(AbsState newPosition)
IError (interface)	Defines functions for error in a band (difference between two states)	Methods: QByteArray serialize() bool withinTolerance(int tolerance)
AbsState (abstract class)	Position for a band Currently, only implementation is quaternion	
AbsBand (abstract)	Class for bands 3 subclasses (Arm, Shoulder, Chest) Responsible for processing messages sent to the application and for determining messages to send during playback Each Band class communicates with its hardware counterpart	Fields: BandType typeOfBand bool connected AbsPose pose Methods: void updateState(AbsState, stateTime) bool moveTo(AbsState) void handleMessage(messageTime, BandMessage) void sendIfConnected(BandMessage) Signals: Void sendMessage(BandType, BandMessage)

Class Name	**Description**	**Fields & Methods**
Suit	Keeps track of a single set of Bands Serves as interface between bands and the rest of the applica-tion	Fields: QHash<BandType, AbsBand> bands Methods: void startOrStopMode(StartOrStopType) AbsBand getBand(BandType) void processNewPose(AbsState pose, BandType, int msgTime) void playSnapshot(PositionSnapshot) Signals: void positionSnapshotReady(int msgTime, Posi-tionSnapshot) void voiceControlCommandReady(StartOrStop-Type) void bandHasLowBattery(BandType) void positionMet()
PlaybackController	Handles Playback mode. Main-tains playback settings and sends playback signals to the Suit class	Fields: bool playing bool suitActive bool stepThroughMode float framerate int currentFrame int endFramePointer int beginningFramePointer Suit suitObj Methods: void togglePlay() void setActiveMotion(WAGFile) void updateSpeed(int newSpeed) void positionMet() void changePlaybackMode(bool isStepThrough) void toggleSuitActive() Signals: void frameChanged(int newFrame) void goToSnapshot(PositionSnapshot)

Class Name	Description	Fields & Methods
EditingController	Handles Edit mode settings and user input Updates suit visualization window for reviewing recorded motion Issues signals to update GUI elements	Fields: int beginningFramePointer int endFramePointer int currentFrame WAGFile activeMotion Methods: void setActiveMotion(WAGFile) WAGFile cropMotion(int startFrame, int endFrame) void togglePlay() Signals: void goToSnapshot()
RecordingController	Handles Record mode	Fields: bool voiceControlEnabled WAGFile activeMotion Methods: RecordingController(Suit) void startRecording() Motion stopRecording() void setActiveMotion(WAGFile) void addSnapshotToMotion(int snapshotTime, PositionSnapshot) void handleVoiceControlCommand(StartOrStopType)
Model	Stores motion simulation states	Methods: void updatePose(PositionSnapshot)
ModelLoader	Loads the data needed to construct the motion simulation model	Methods: Model* load()
GLWidget	Draws and displays the motion visualization	Methods: void initializeGL() void paintGL() void resizeGL(int,int)
ModelGL	Stores visualization meshes and coloring/lighting data	Methods: void updatePose(QHash<QString,NodeState>) Signals: void modelGLChanged()

Class Name	Description	Fields & Methods
ModelGLLoader	Reads and parses the human model object files, loading into memory	Methods: bool Load(QString,PathType) ModelGL* toModel()
enum BandType	List of possible band types	Types: leftLowerArmBand rightLowerArmBand leftUpperArmBand rightUpperArmBand leftShoulderBand rightShoulderBand chestBand

Table 11 describes the basic breakdown of classes within the embedded hardware.

Table 11 – Core classes and functionality of embedded hardware

Class Name	Description	Fields & Methods
WifiStateMachine	Contains all the ESP8266 code	Fields: char recordingMsg[11] char playbackMsg[11] Methods: void readTeensySerialSendPkt(boolean printStuff) boolean listenForSpecificPacket(char specificPacket, boolean printInfo)
WiFiMsgTypes	Contains the definitions of the Wi-Fi message types	Fields: typedef enum MessageType

pfodESP8266WiFi	Contains non-blocking implementation of ESP8266 Wi-Fi client [45]	Methods:
		WiFiClient.write(...)
		WiFiClient.isSendWaiting()
		pfodESP8266BufferedClient
		pfodESP8266BufferedClient()
		pfodESP8266BufferedClient.connect(WiFiClient*)
		pfodESP8266BufferedClient.write(..)
		pfodESP8266BufferedClient.isSendWaiting()
		pfodESP8266BufferedClient.flush()
		pfodESP8266BufferedClient.setDebugStream(Stream* debugOut)

BatteryMonitor	Contains the code to check on the low battery circuitry	Fields: NUM_LOW_BATT_CYCLES LOW_BATT_PIN_ACTIVE_LOW Methods: BatteryMonitor() void initLowBatteryInfo() void checkBattery() int hasLowBat()

Class Name	Description	Fields & Methods
ESP8266Comms	Contains the code within the Teensy to enable communications with the ESP8266	Fields: uint8_t msgToESP8266[12] Methods: void ESP8266Comms() void zeroErrorCalculations() void setCommand(char cmd) void sendMsgToESP8266(char cmd)
I2Cdev	Contains the I2C library code [46]	Method: Wire.begin()
MPU6050	Contains the I2C development for the MPU6050 [46]	Methods: void initialize() void dmpInitialize() int testConnection() void getFIFOBytes(uint8_t buffer, int pack-etSize)
MPU6050WAGWrapper	Contains the code to abstract the MPU6050 interface for the WAG Band	Methods: MPU6050WAGWrapper(uint8_t bandNum) void extractMPU6050Vals(uint8_t* packet) void loadAccelGyroOffsets(int xAccel, int yAc-cel, int zAccel, int xCyro, int yGyro, int zGyro)
VibrationPattern	Contains the code necessary to control the motors from the Teensy 3.2	Methods: void performMotorCalculationsAndRunMo-tors() void writeMotorSpeeds(int* motorSpeeds, int nAngles) void stopAllMotors()
WAGBandCommon	Contains global definitions of shared information such as UARTs	Fields: ESP8266_SERIAL DEBUG_SERIAL
WAGBandTeensyCode	Contains the code to connect all embedded hardware with a WAG Band	Methods: boolean readESP8266SerialSendPkt(boolean printStuff) void loop()
WAGBand-ChestPieceTeensyCode	Contains the code to enable all embedded hardware within the chest piece	Methods: void listenToArduino() void enableVoiceControl() void loop()
easyVR_arduino_simple	Contains all the code running on the ATMega328 within the chest piece (based on the EasyVR code base [47])	Methods: void checkMonitorInput() void loop()

Appendix 3 – Full Bill of Materials for Project

This appendix contains a listing of materials for each band and what would be required to build another band.

WAG Band

The full bill of materials is seen in Table 12. This shows the part along with the manufacturer part number, the cost, quantity on the board and the cost for the component(s).

Table 12 - Bill of materials for WAG Band

Part	Manufacturer number	Quantity	Cost per	Cost total
Panasonic 1000µF Capacitor	EEU-FR1A102L	1	$0.19	$0.19
MPU6050 Breakout board	-	1	$3.50	$3.50
Teensy 3.2	-	1	$20.00	$20.00
1kOhm (SM)	ERJ-U02F1001X	3	$0.01	$0.04
10kOhm (SM)	CRCW120610K0FKEAHP	1	$0.10	$0.10
1uF capacitor (SM)	0805ZC105KAT2A	2	$0.01	$0.02
TLV1117 3.3V regulator	TLV1117LV33DCYR	1	$0.52	$0.52
NPN BJT (SM)	MMBT3904LT1G	2	$0.01	$0.02
N-Channel MOSFET (SM)	BSS138LT1G	6	$0.24	$1.46
10kOhm potentiometer	CB10LV103M	1	$0.20	$0.20
5mm Red LED	-	1	$0.20	$0.20
5mm Yellow LED	-	1	$0.20	$0.20
5mm Green LED	-	1	$0.20	$0.20
Female 40 pin header 0.1" spacing	-	2	$0.40	$0.80
LM358 (SM)	LM358DR	1	$0.10	$0.10
30kOhm (SM)	CRCW120630K0FKEA	2	$0.01	$0.01
ESP8266 Wi-Fi Board	-	1	$4.00	$4.00
PCB	-	1	$4.00	$4.00
SPDT power switch	MS12ASW13	1	$1.79	$1.79
JST-PH connector for battery and charger (female)	S2B-PH-K-S(LF)(SN)	2	$0.17	$0.34
Motor crimp connectors	SXH-002T-P0.6	12	$0.10	$1.20
Stranded wire for motors	-	1	$1.50	$1.50
JST-PH connector for battery and charger (male)	PHR-2	2	$0.10	$0.20
Wire for battery	-	1	$0.30	$0.30
Battery 3.7V Li-Po	T1000.1S.20	1	$2.60	$2.60
Motor connector (male)	XHP-12	1	$0.24	$0.24
Motor (3V) 10x2mm	Polulu 1636	6	$3.49	$20.94
Motor connector (female)	S12B-XH-A(LF)(SN)	1	$0.68	$0.68
3D Printed Case and motor mounts	-	1	$4.00	$4.00
		Total cost per band:		$73.86

WAG Chestpiece

The full bill of materials is seen in Table 13. This shows the part along with the manufacturer part number, the cost, quantity on the board and the cost for the component(s).

Table 13 - Bill of materials for WAG Chestpiece

Part	Manufacturer number	Quantity	Cost per	Cost total
Panasonic 1000µF Capacitor	EEU-FR1A102L	1	$0.19	$0.19
MPU6050 Breakout board	-	1	$3.50	$3.50
Teensy 3.2	-	1	$20.00	$20.00
1kOhm (SM)	ERJ-U02F1001X	3	$0.01	$0.04
10kOhm (SM)	CRCW120610K0FKEAHP	1	$0.10	$0.10
1uF capacitor (SM)	0805ZC105KAT2A	2	$0.01	$0.02
TLV1117 3.3V regulator	TLV1117LV33DCYR	1	$0.52	$0.52
NPN BJT (SM)	MMBT3904LT1G	2	$0.01	$0.02
10kOhm potentiometer	CB10LV103M	1	$0.20	$0.20
Red LED	-	1	$0.20	$0.20
Yellow LED	-	1	$0.20	$0.20
Green LED	-	1	$0.20	$0.20
Female 40 pin header 0.1" spacing	-	3	$0.40	$1.20
LM358 (SM)	LM358DR	1	$0.10	$0.10
30kOhm (SM)	CRCW120630K0FKEA	2	$0.01	$0.01
ESP8266 Wi-Fi Board	-	1	$4.00	$4.00
PCB	-	1	$5.00	$5.00
SPDT power switch	MS12ASW13	1	$1.79	$1.79
JST-PH connector for battery and charger (female)	S2B-PH-K-S(LF)(SN)	2	$0.17	$0.34
JST-PH connector for battery and charger (male)	PHR-2	2	$0.10	$0.20
Wire for battery	-	1	$0.30	$0.30
Battery 3.7V Li-Po	T1000.1S.20	1	$2.60	$2.60
EasyVR3.0 Speech Recognition Module	COM-13316	1	$49.95	$49.95
ATMega328-PU (with internal 8MHz oscilla-tor bootloader)	ATMEGA328-PU-ND	1	$3.38	$3.38
DIP Socket 28 position	ED3050-5-ND	1	$0.33	$0.33
3D Printed Case	-	1	$4.00	$4.00
Chest strap	-	1	$8.00	$8.00
		Total cost		$106.40

Appendix 4 – Assembly & Construction

In Table 12 and Table 13, the full listings of parts required for a WAG Band and WAG Chestpiece are detailed and various manufacturer part numbers are included where applicable. These parts could be ordered either directly from the manufacturer or from a distributor. The part that is not detailed there is the PCB board and schematic files. This appendix describes these files and the process of constructing a band. Each PCB was designed in EAGLE 7.5.0 Light produced by CadSoft. EAGLE is a schematic capture (design) and PCB layout software. The Light version of EAGLE is a limited feature free version of the program (with one of the main restrictions being the size of PCB board of 100mmx80mm and the number of layers of the board to only top and bottom layer routing). The free version was fine for project as the board was simple with only a few components needing to share space. One challenge was the spacing of all the through-hole modules (ESP8266, Teensy 3.2, MPU6050 etc.) which required larger clearances than using surface mount parts only. To layout the board, a number of virtual "parts" needed to be created. These "parts" define the specific physical footprint of a component so the designer knows how to properly layout the board. For a through-hole component such as a resistor with metal leads, this package is usually two holes with some spacing between them. For more advanced surface mount components these packages can have much smaller spacing between the pads (metal rectangles) where the part is soldered to. With the plethora of components available, each one can have a unique footprint. To aid in the design process, EAGLE includes a number of common components in virtual form in libraries. For this project a number of the pieces were specialized and required custom virtual parts to be created (for schematics and for the board layout) for objects such as the potentiometer for the low battery threshold, the EasyVR3.0 module, the motor controller MOSFETs, and the motor JST connector. The remainder of this appendix covers the schematics, PCBs, band construction and assembly.

WAG Band: Schematics and PCB design

The full schematic for the WAG Band can be seen in Figure 36. The whole schematic is broken down into smaller subsections in the following sections.

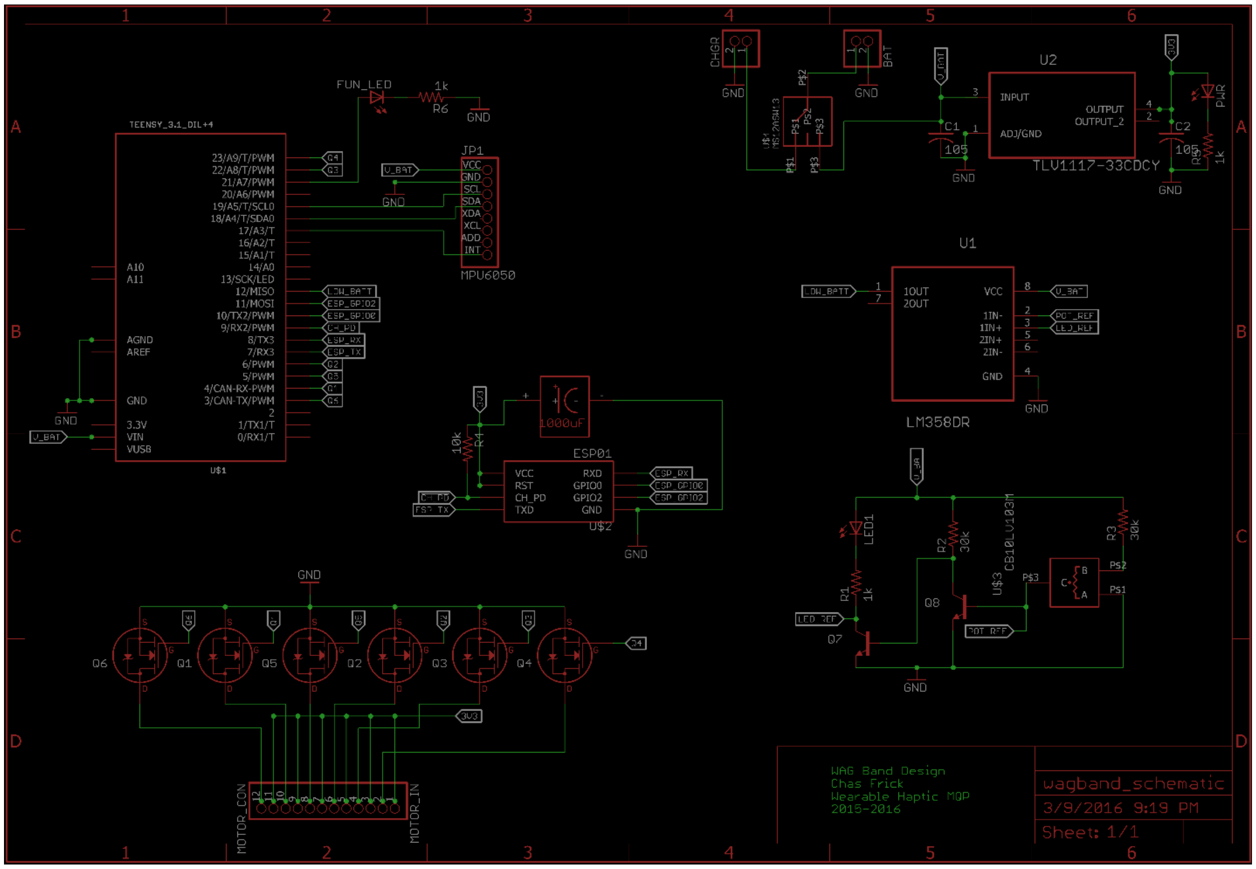


Figure 36. WAG Band schematic

The full board layout can be seen in Figure 37.

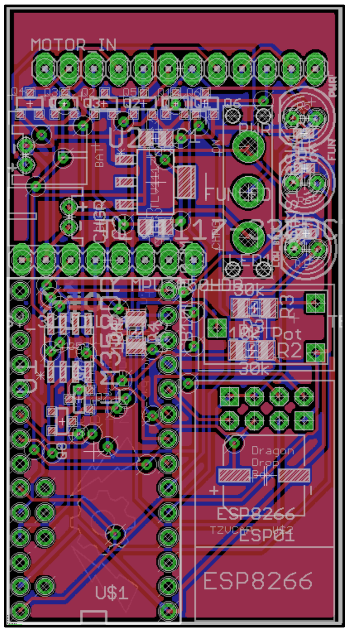


Figure 37. Board design for WAG Band

The final PCB for the WAG Band can be seen in Figure 38.

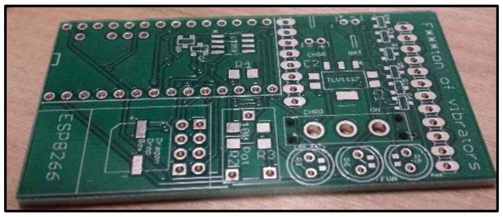


Figure 38. Final PCB for WAG Band

An intermediate step can be seen in Figure 39. This shows where all of the larger components would go as well.

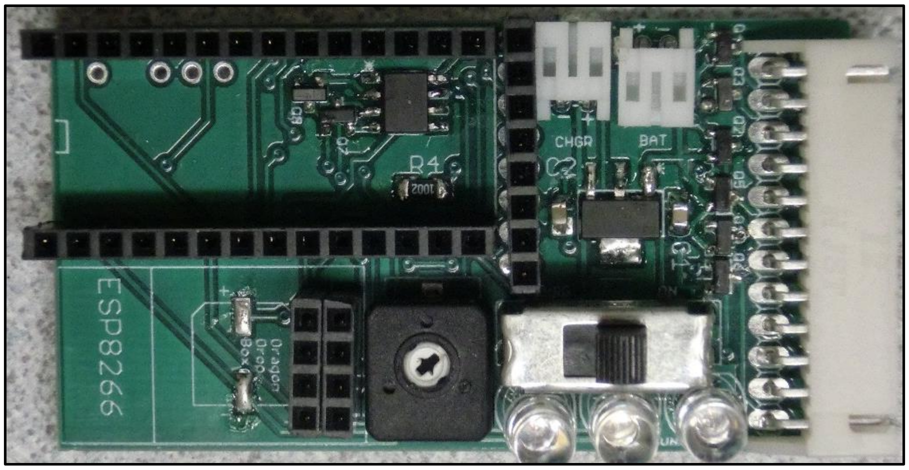


Figure 39. Final WAG Band PCB with surface mount components and female headers

The populated final WAG Band PCB can be seen in Figure 40. This shows the various modules placed in the female headers from Figure 39.

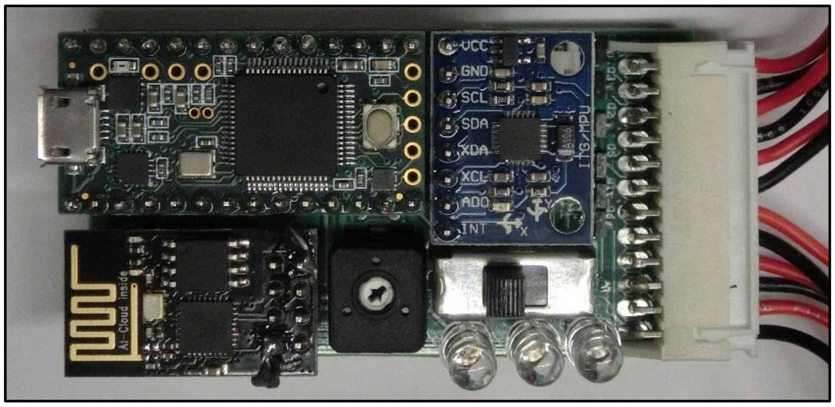


Figure 40. Final PCB with components for WAG Band

Teensy 3.2 Microcontroller and MPU6050

For this part of the schematic, the only important part is shown in Figure 41. The main portion of the image shows the Teensy 3.2 schematic symbol with nets going to LOW_BATT (battery monitoring circuitry), ESP_GPIO2 (GPIO2 pin on the ESP8266), ESP_GPIO0 (GPIO0 pin on the ESP8266), CH_PD (CH_PD pin on the ESP8266), ESP_RX (the ESP8266 RX UART pin), ESP_TX (the ESP8266 TX UART pin), Q1, Q2, Q3, Q4, Q5 (all the 6 motor MOSFET gates) and the V_BAT (battery voltage input). There also exist a few unlabeled connections going to a MPU6050 part in the image. These lines are the interrupt line (on Teensy pin 17) and the I2C connection lines (Teensy Pins 19 and 18) to the I2C pins on the MPU6050 header JP1 (SCL and SDA). On the actual PCB, both the MPU6050 and Teensy 3.2 are held in place in 0.1" spacing female through-hole headers. These aid in debugging and allow for modularity where components can be easily swapped out in case of failure.

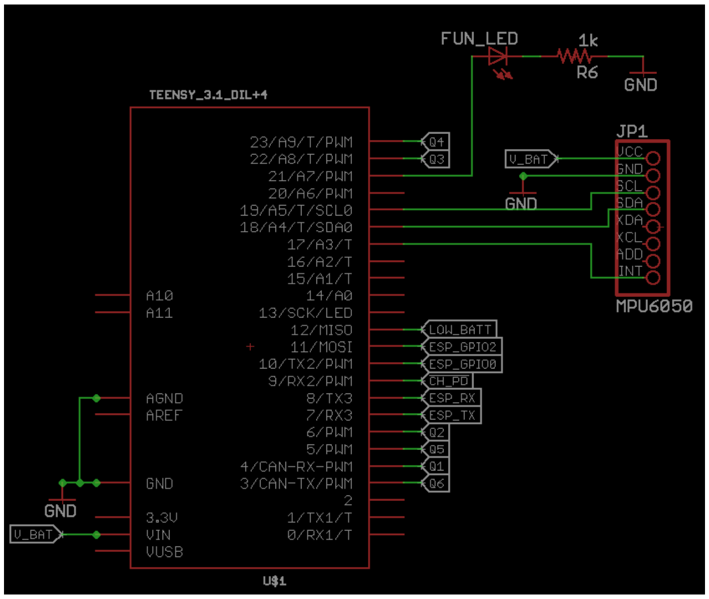


Figure 41. WAG Band microcontroller schematic

This part of the final PCB can be seen in Figure 42. This shows the 14 pin female headers for the Teensy in addition to the 8 pin vertical header for the MPU6050. Below the MPU6050 are the connections for the charger and the battery in addition to the voltage regulator. Beneath the Teensy 3.2 is the low battery detection circuitry including the two BJT transistors and an LM358 op-amp.

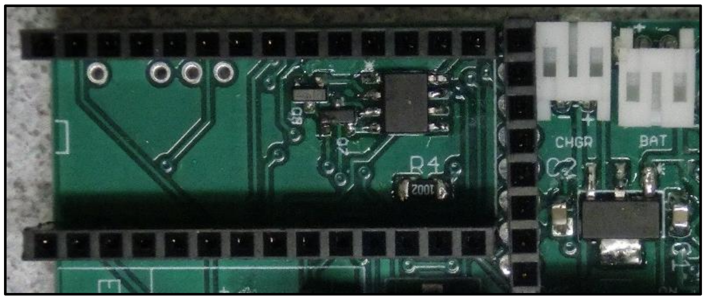


Figure 42. Microcontroller portion of final PCB

ESP8266 Wi-Fi Chip

The schematic for the ESP8266 Wi-Fi module can be seen in Figure 43. This shows the ESP_GPIO2 (GPIO2 pin on the ESP8266), ESP_GPIO0 (GPIO0 pin on the ESP8266), CH_PD (used to pull the module out of sleep mode), ESP_RX (the ESP8266 RX UART pin) and ESP_TX (the ESP8266 TX UART pin). Additionally the 3V3 line is connected to the VCC and RST connections (and CH_PD through a pull-up resistor). This 3V3 line comes from the regulated 3.3V power provided by the onboard regulator.

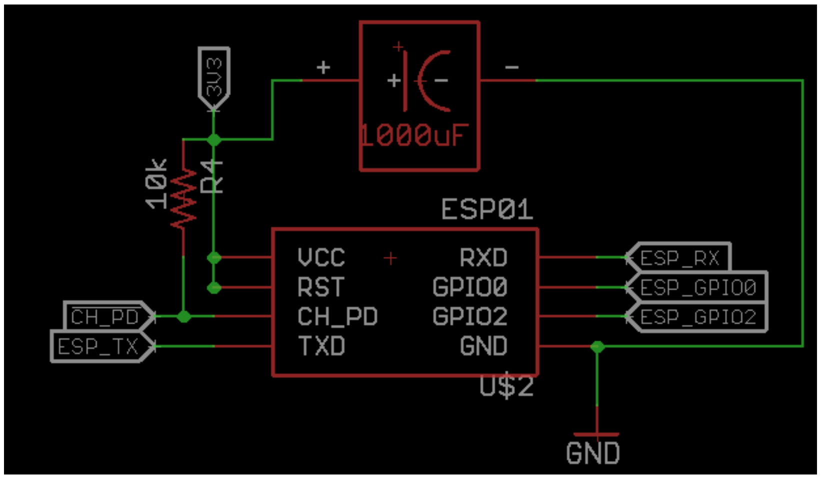


Figure 43. WAG Band ESP8266 schematic

The 1000µF capacitor shown in the top of Figure 43 was originally a surface mount capacitor to stabilize the operation of the ESP8266 module, however it was found this did not solve the issue. Instead a 1000µF “backpack” capacitor soldered directly onto the VCC and GND pins of the ESP8266 provided the required stability (see Figure 44). Similarly to the Teensy 3.2 and MPU6050, the ESP8266 symbol U$2 in Figure 43 is represented on the PCB with two 4 pin 0.1” spacing female through-hole headers. This allows for easy programming and swapping of ESP8266 chips. This also provides the required spacing between the PCB and the ESP8266 for the “backpack” capacitor.

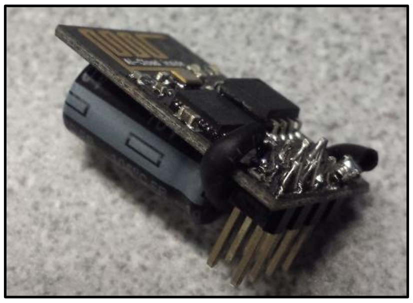


Figure 44. ESP8266 with "backpack" capacitor for stability

The physical representation of the ESP8266 socket can be seen in Figure 45. The PCB also contains the surface mount capacitor pads, but this was replaced with the "backpack" capacitor.

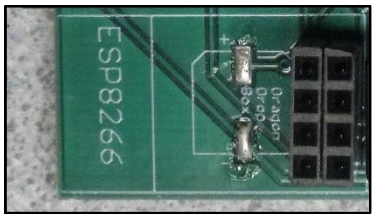


Figure 45. ESP8266 Socket on final WAG Band PCB

To make programming of the ESP8266 module easier (after electing to use a custom Arduino-core developed by the ESP8266 community), a simple ESP8266 breakout board was used. This can be seen in schematic form in Figure 46.

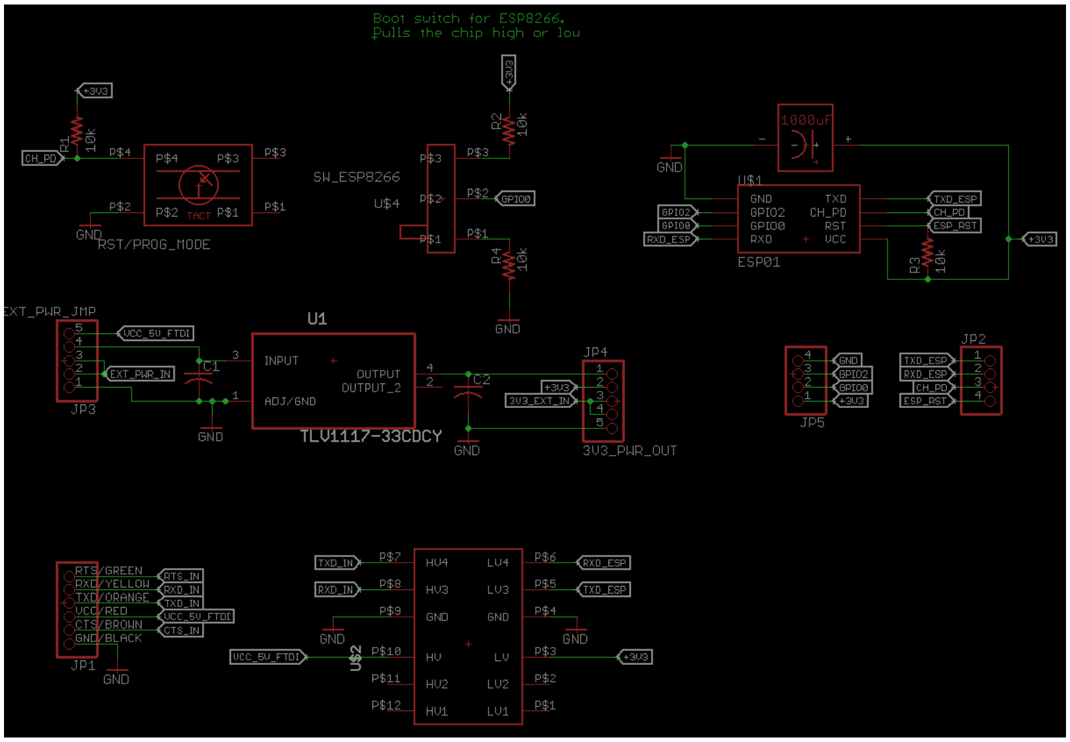


Figure 46. Schematic for simple ESP8266 programming breakout

The resulting board layout can be seen in Figure 47. This breakout board includes the ESP8266 headers on the right side and brings the UART pins out to an FTDI header on the far left. The switch in the upper right corner allows the module to be placed in normal operating mode or programming mode (for loading code through the FTDI header). Using 0.1” male headers, two other 5 pin connections allow for a selectable voltage input to the ESP8266. An on board voltage regulator also handles voltage filtering. Additionally since the ESP8266 is a 3.3V system, an external level shifter is included on the board to translate the voltage thresholds between the 5V and 3.3V UARTs of the FTDI cable and the ESP8266. A simple pushbutton switch is also included to reset the module. Lastly 4 pin headers on the top and bottom of Figure 47 are included to allow the device to be slotted into a breadboard for ease of debugging.

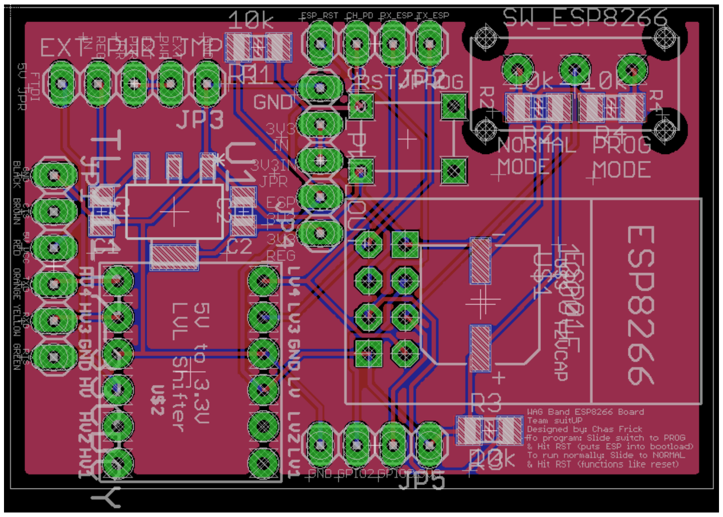


Figure 47. PCB layout for ESP8266 breakout board

The final product of the ESP8266 breakout board can be seen in Figure 48.

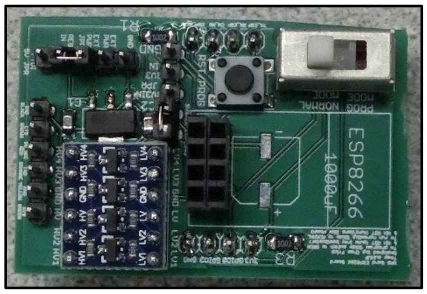


Figure 48. Finished ESP8266 breakout board

Motor Controllers

For this portion of the WAG Band, the schematic is shown in Figure 49. Each one of the devices named Q1 through Q6 is an n-channel MOSFET with the source connected to ground and the drain connected to one lead of the motor (through the MOTOR_IN connector). The gate is connected to a wire which leads back to a Teensy PWM pin. The motors connect to the MOTOR_IN connector through a JST-XH 12-pin connector with one lead going to a connection to 3.3V regulated and the other to the drain of a MOSFET.

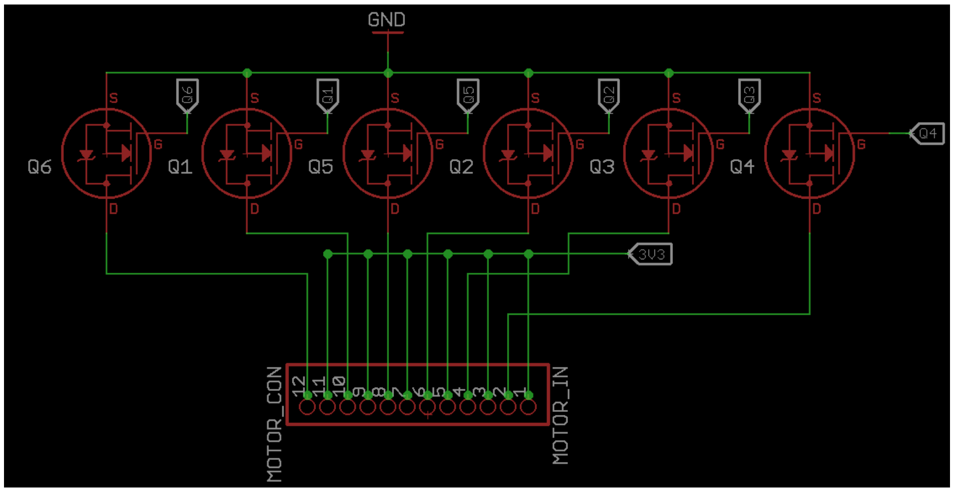


Figure 49. WAG Band motor controller schematic

The physical representation of these MOSFETs on the PCB can be seen in Figure 50. The JST-XH connector was chosen to allow the motors to be disconnected from the PCB if necessary (to replace a component or battery). With the ability to disconnect the motors from the PCB, other circuit components such as LEDs could be substituted for testing the WAG Bands.

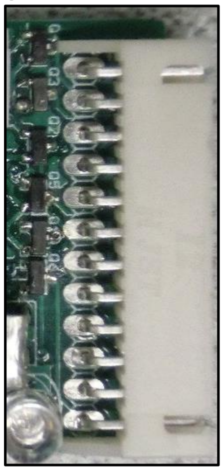


Figure 50. MOSFET motor controllers on the final PCB

Low Battery Circuit

The circuit schematic for the low-battery schematic can be seen below in Figure 51. The top portion of this circuit is a surface mount LM358 single supply op-amp to take in the voltage levels of the potentiometer and the LED from the detection circuit and provide a single digital signal indicating if the battery voltage supplied through (V_BAT) has dropped below a certain threshold. This threshold is set manually by adjusting the potentiometer (in a voltage divider with the 30kΩ resistor). The BJTs used in the detection circuit are biased at by the potentiometer voltage and conduct when this voltage changes as the voltage drops. This design is based off the original design by Swagatam Majumdar which was for 9V batteries [48], but was modified for 3.7V battery systems and converted into a digital signal.

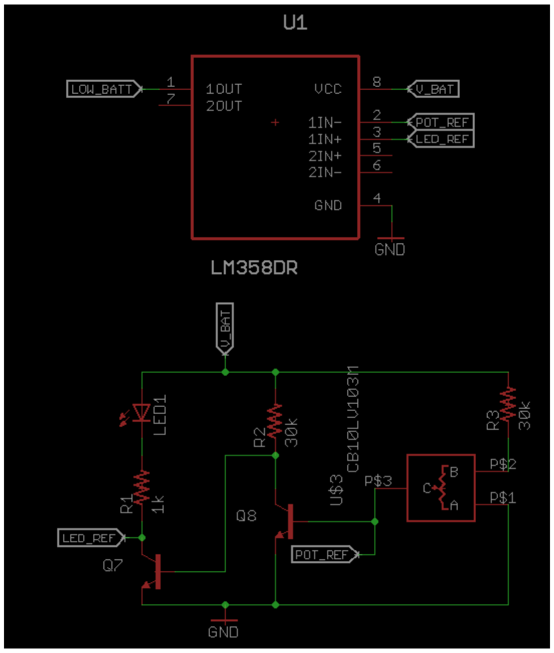


Figure 51. WAG Band low battery schematic

The final version of this battery monitor circuit can be seen in Figure 52. This shows the LM358 along with the required NPN BJT transistors. The potentiometer can also be seen with the LED for displaying the LOW battery notification. This information is also sent over Wi-Fi to the computer application.

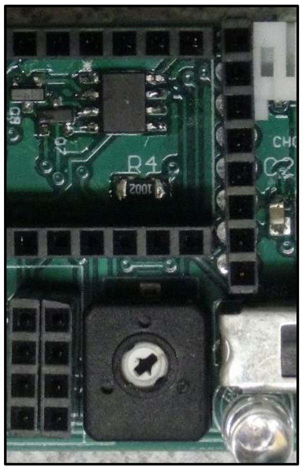


Figure 52. WAG Band PCB low battery circuit

Power Distribution

The power distribution subsystem can be seen in the schematic snippet in Figure 53. The biggest part of this circuit is the 3.3V voltage regulator (TLV1117). The connections for powering the rest of the board (the V_BAT and 3V3 connections) can also be seen here. The CHGR and BAT connections are to JST-PH connectors on the PCB. The U$4 module is a SPDT power switch which enables charging the battery when the band is not in use. On the 3V3 power side, there is a green PWR LED to indicate the band is powered on.

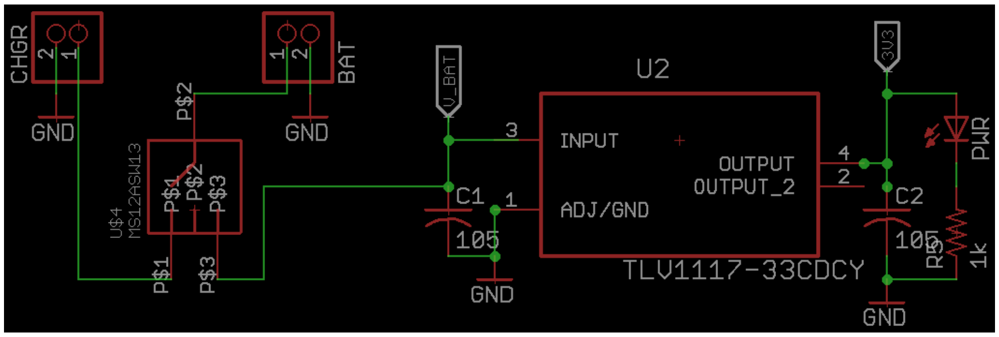


Figure 53. WAG Band power distribution schematic

The final PCB result of this power distribution module can be seen in Figure 54. This shows the SPDT switch along with the LEDs on the bottom of the image. The voltage regulator and the JST-PH connectors are on the top of the image.

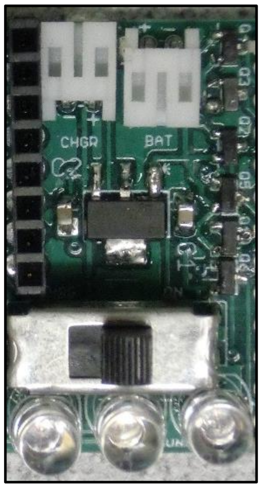


Figure 54. WAG Band power distribution module PCB

Motor Connector

For this part of the hardware, the motors were soldered to stranded wires which were fed into a male JST-XH 12-pin connector. They were then placed into 3D printed motor housings and placed on a nylon rope outside of the band. This can be seen in Figure 55. Heat shrink was used on the motor wires to reduce the chance of snapping a motor lead. The male JST-XH connector was slotted into the PCB within the band. Fastener straps were also fed through the bottom of the 3D printed case to attach the band casing.

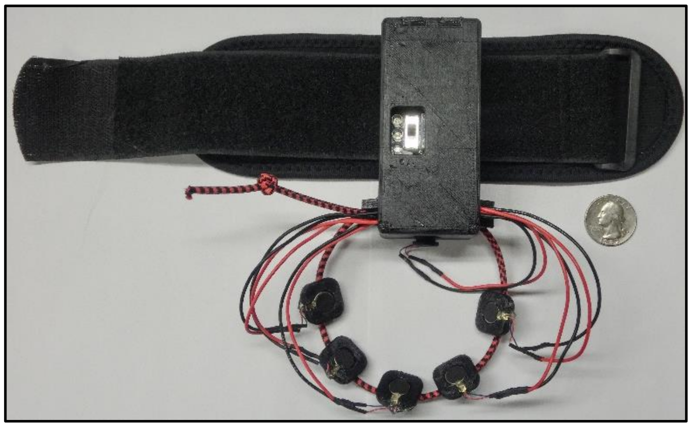


Figure 55. WAG Band motors outside 3D printed case

WAG Band: Programming and Construction

With an understanding of the hardware behind the WAG Band, each of the bands was constructed with all the surface mount components soldered first. Next the boards were populated with the various through-hole components (potentiometer, LEDs, power switch, JST connectors). Once the boards were completed, these were tested for complete functionality and were populated with the various modules (Teensy, MPU6050, ESP8266 etc.).

After testing the PCBs and functionality of the bands, the physical construction of the bands with additional Fastener and harnesses was completed. Each band casing was 3D printed in PLA. The casings were designed to house the battery and custom PCB. The casing attaches to the user with a hook and loop strap and an elastic band is used to hold the ring of ERVs. 5 of each band's ERVs was glued into a 3D printed slider which was attached to the elastic band so that the user could adjust the ERVs' locations. The 6th ERV's was located on the underside of the main band casing. Each ERV had leads of appropriate length soldered on the connector plug for attachment to the PCB. The lid of the band casing was designed to securely hold the PCB in place inside of the band so that the IMU would not move relative to the limb it was tracking.

Programming of the Teensy 3.2 was done via the USB port on the device and code was written in the Arduino 1.6.5 IDE. Code was also stored on a private Github repository for version control. The ESP8266 module was programmed using a special plugin for the Arduino 1.6.5 IDE that enabled programming of the module through the breakout board (and the ESP8266 community Arduino-core). This was programmed with the use of an FTDI cable and the ESP8266 breakout board seen in Figure 48. Once the ESP8266 was programmed, the board was slotted into the band and the band hardware was complete.

WAG Chestpiece: Schematics and PCB design

The schematic of the WAG Chestpiece can be seen in Figure 56. The biggest difference from the normal WAG Band schematic is the addition of an ATMega328P (the chip behind the popular Arduino Uno) and an EasyVR3.0 module and the removal of the motor MOSFETs.

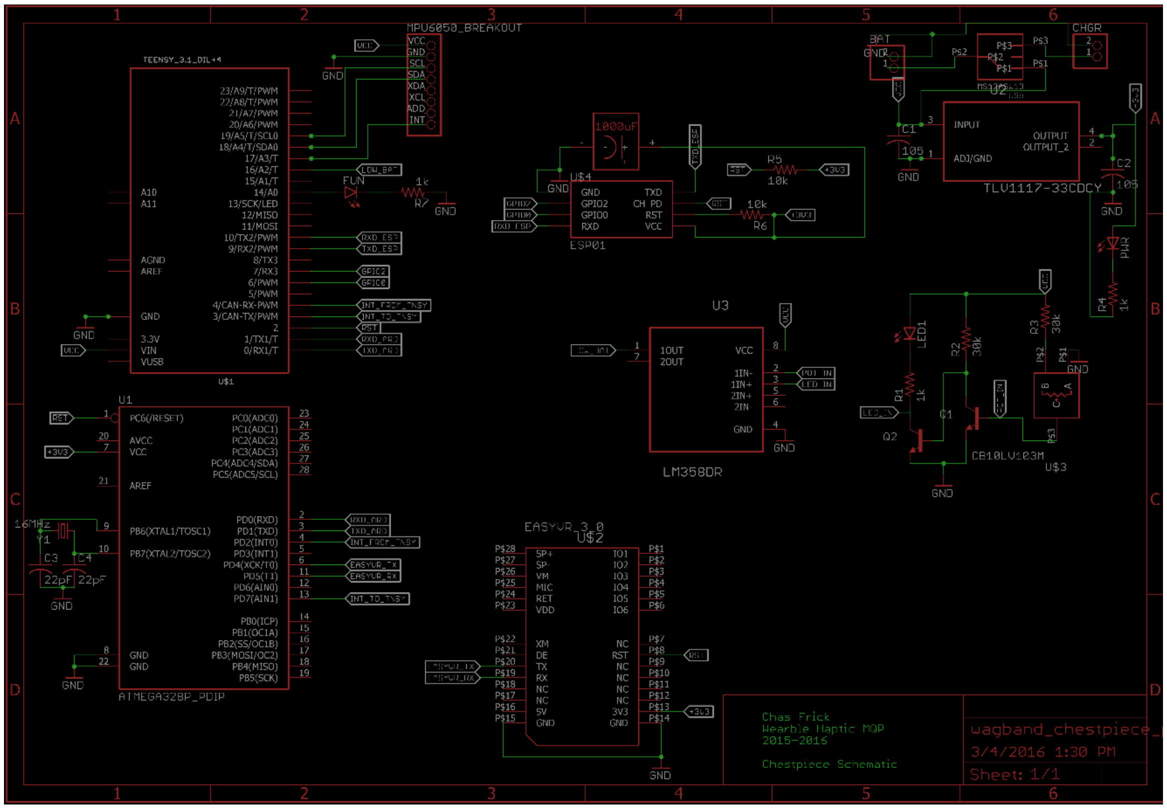


Figure 56. WAG Chestpiece schematic

Since the chest piece shares similarities to the WAG Band, see the WAG Band: Schematics and PCB design section for information about the Teensy 3.2, MPU6050, low battery detection circuitry, ESP8266 or the power distribution circuitry. The resulting board layout for the WAG Chestpiece can be seen in Figure 57.

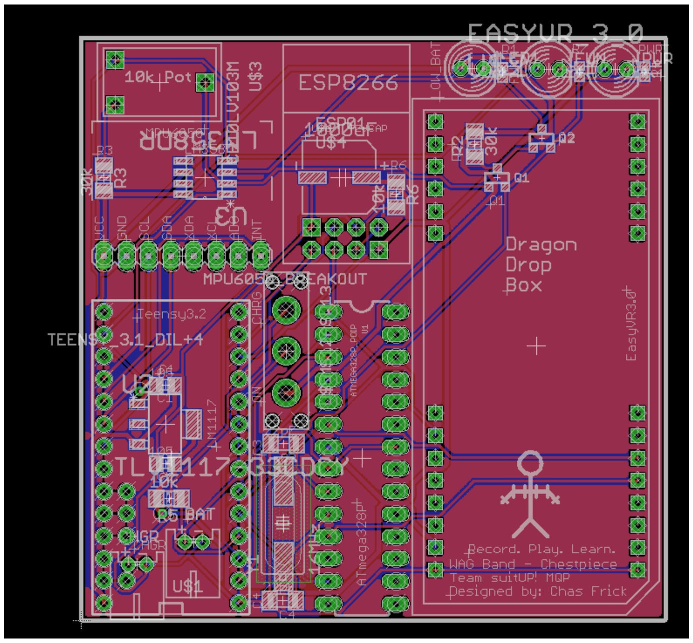


Figure 57. WAG Chestpiece board layout

The PCB produced from this layout, without the microcontroller and speech recognition modules, can be seen in Figure 58.

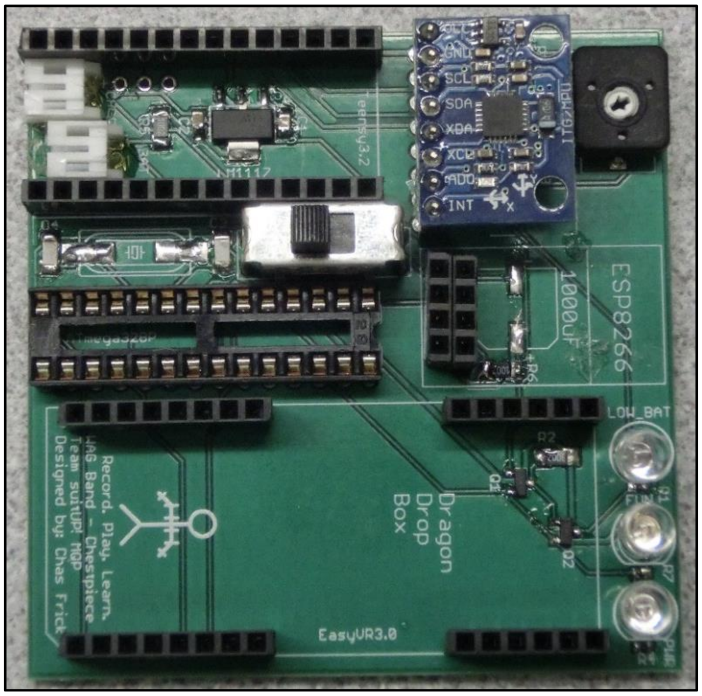


Figure 58. Finished PCB without microcontroller and speech recognition modules inserted

The final populated PCB with all modules installed can be seen in Figure 59. This also includes a quarter for size reference. This board is then installed in the harness for the chest as seen in Figure 14.

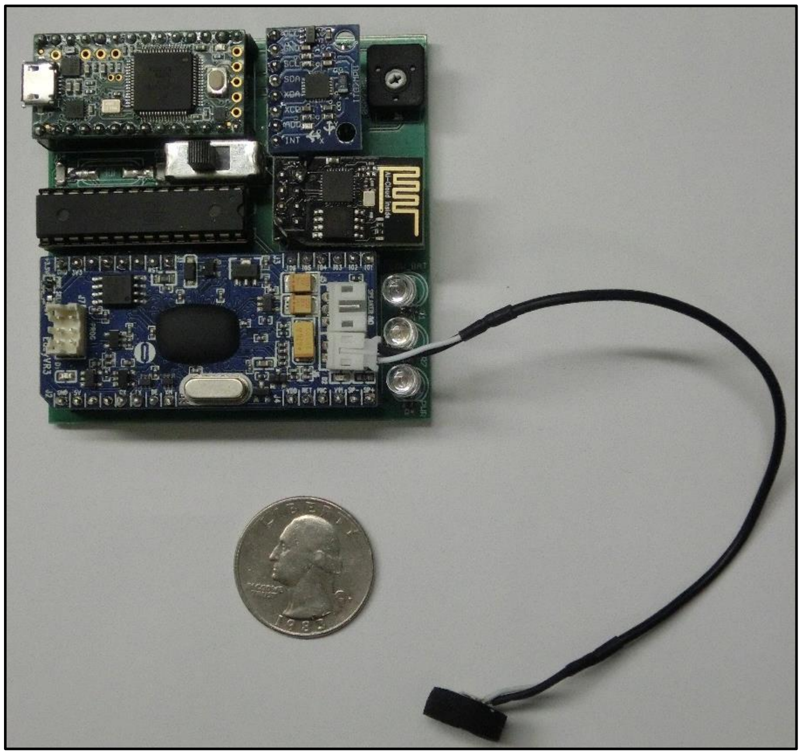


Figure 59. WAG Chestpiece PCB with all components on board

Secondary Microcontroller: ATMega328P

One difference was the ATMega328P chip. This circuitry can be seen in Figure 60. This chip was used because of the low cost associated with the microcontroller (around ~$4). Additionally, this chip is easily programmable using the Arduino IDE and bootloader. While Figure 60 indicates the presence of a 16MHz oscillator crystal, this was used to burn the 8MHz internal oscillator Arduino bootloader into the chip. It was not used on the final chest piece PCB.

The main connections are the UART connections (RXD_ARD and TXD_ARD) to the Teensy's second UART. There was an included interrupt line from the Teensy to the Arduino (INT_FROM_TNSY) to indicate data was ready, although this was not needed in the final version of the hardware and was not used. The other connection was another UART link to the EasyVR3.0 module (on lines EASYVR_TX/RX performed with a software defined UART called SoftSerial in the Arduino Libraries). There is also a !RST (active low) rest line coming in from the Teensy to reset the Arduino). Lastly, is an interrupt line to the Teensy (INT_TO_TNSY) which is used to tell the Teensy that voice command data has been transferred from the Arduino UART to the Teensy.

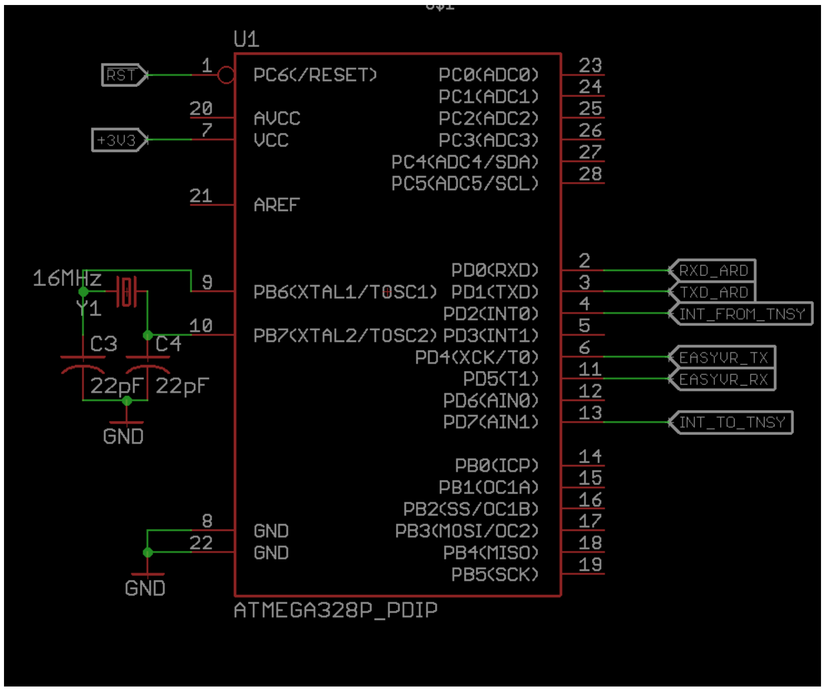


Figure 60. Chest piece ATMega328 schematic

The physical representation of the ATMega328 can be seen in Figure 61. This shows a 28 pin DIP socket used to include the IC. The reason behind this was the ability to easily remove and reprogram the module (using an Arduino Uno board) for the final system.

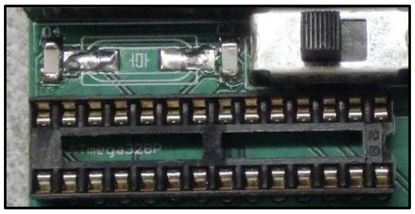


Figure 61. ATMega328 interface on chest piece PCB

Voice recognition Module: EasyVR3.0

The EasyVR3.0 module schematic can be seen in Figure 62. The main connections are the UART connection to the ATMega328 (see Figure 60). The only other connections needed are 3.3V power, GND and an active-low reset signal.

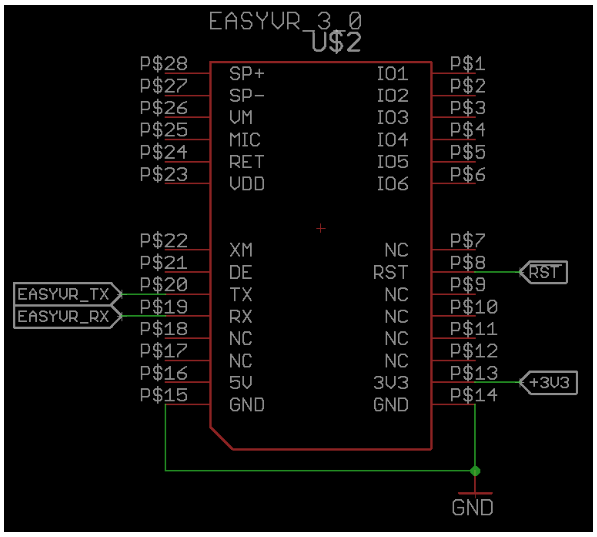


Figure 62. EasyVR3.0 interface schematic

The physical representation of this part of the chest piece PCB can be seen in Figure 63. The actual module sits on top of this and uses the female 0.1” spacing headers. Figure 63 also features a few components from the low-battery circuit.

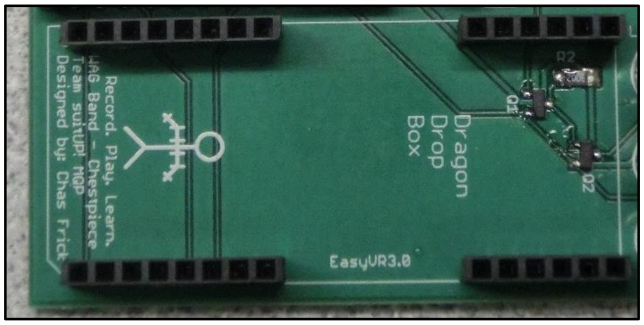


Figure 63. Chest piece EasyVR3.0 interface

WAG Chestpiece: Programming and Construction

The chest piece went through the same construction as described in the WAG Band: Programming and Construction section, although the EasyVR module had to be broken out and tested separately before being included in the final design.

The physical box and mounting of the chest piece to the harness around the user’s torso was completed in early April. The harness used is a repurposed GoPro chest harness. The shoulder bands and chest band were attached to the harness in their appropriate locations. The harness is adjustable and form fitting so it is perfect for holding the bands in place on the user.

Similar to the WAG Band: Programming and Construction section, programming of the Teensy 3.2 was done via the USB port on the device in the Arduino 1.6.5 IDE. The ATMega328P was also programmed through the 1.6.5 IDE, however the DIP chip was placed on a breadboard rather than a standard Arduino Uno breakout board. This was done because a 3V bootloader was burned into the device that relies on the internal 8MHz oscillator rather than the traditional external 16MHz oscillator crystal. This “bread-board” Arduino setup is described on the Arduino CC website. The ESP8266 chip was also programmed using a 5V FTDI cable and the ESP8266 breakout PCB.

After the Teensy, Arduino and ESP8266 were programmed, they were placed into their headers on the board and the band was complete. The battery was placed in the chest piece below the PCB and a protective layer was placed above that. The PCB was then slotted into the casing and the lid was replaced on the top of the open case. Both the chest piece and the WAG Band case lids were painted with the correct ON/OFF locations and the logo of the team. The bands could then be tested and connected with the computer. For the chest piece this included verifying that the voice commands of “run,” “stop” and “action” worked as they should.

Software Development

GUI

To meet the project requirements, a list of features and functions that the GUI would need was made. After the list was complete, some preliminary designs were made using moqups.com. Designs were implemented after gaining stakeholder approval. Qt creator was used to design the GUI using C/C++ because it is well documented and supported, and some team members had prior experience with it. Figure 65, Figure 66, Figure 67, and Figure 68 show the final GUI design next to the original moqups.com design. The edit recording and playback recording windows were implemented first, followed by the settings and save as overlay windows.

The hardest feature to implement was the double handled slider shown in Figure 64. Qt’s generic slider object does not support two handles so the application used a custom SuperSlider class. The base implementation was taken from a Stack Overflow answer and modified to meet the needs [49]. The final result is a slider with two handles that cannot cross.

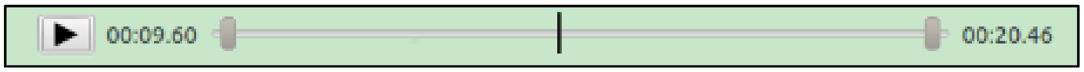


Figure 64. The double handled slider, used in editing and playback mode

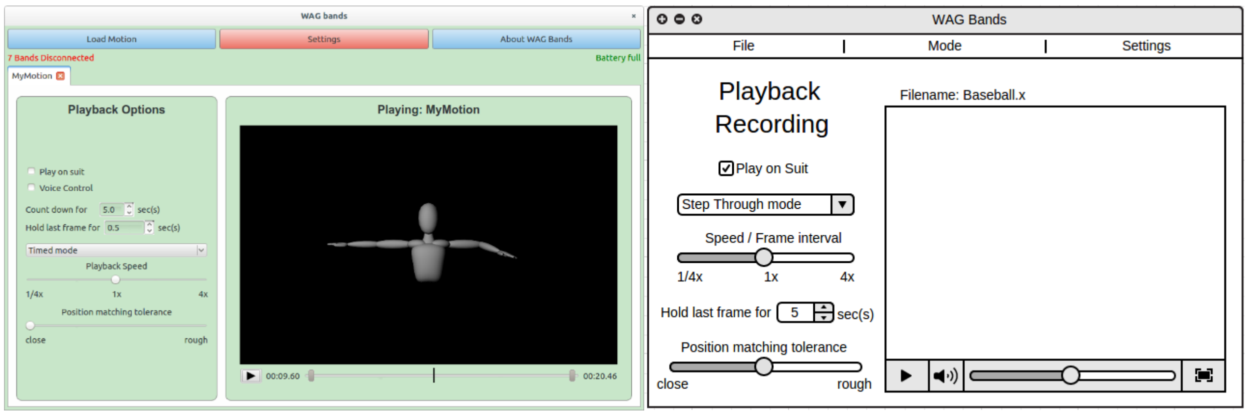


Figure 65. Playback motion GUI. Qt Creator on the left, moqups on the right

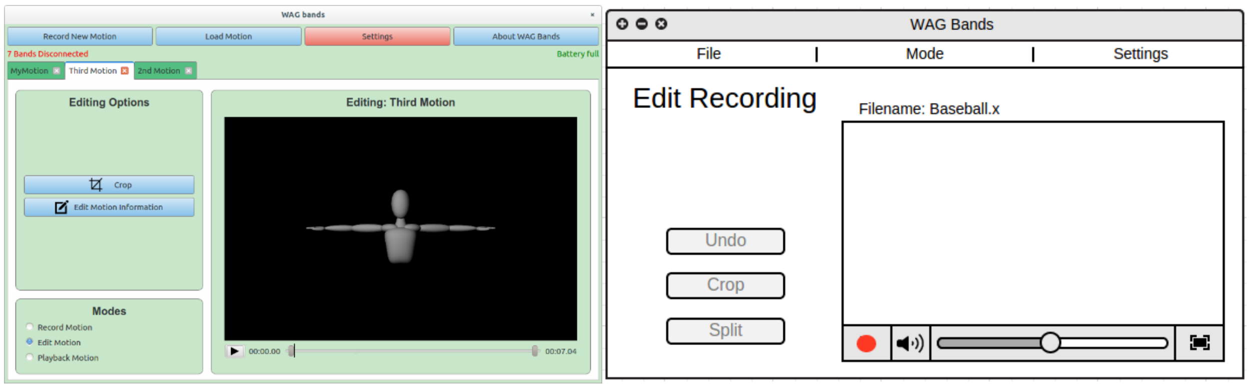


Figure 66. Record/Edit motion GUI. Qt creator on the left, moqups on the right

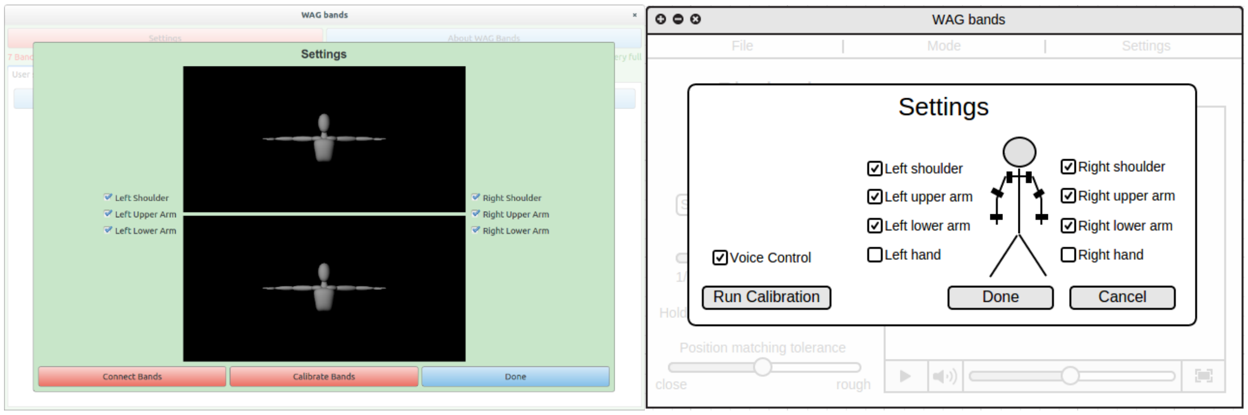


Figure 67. Settings window. Qt Creator on the left, moqups on the right

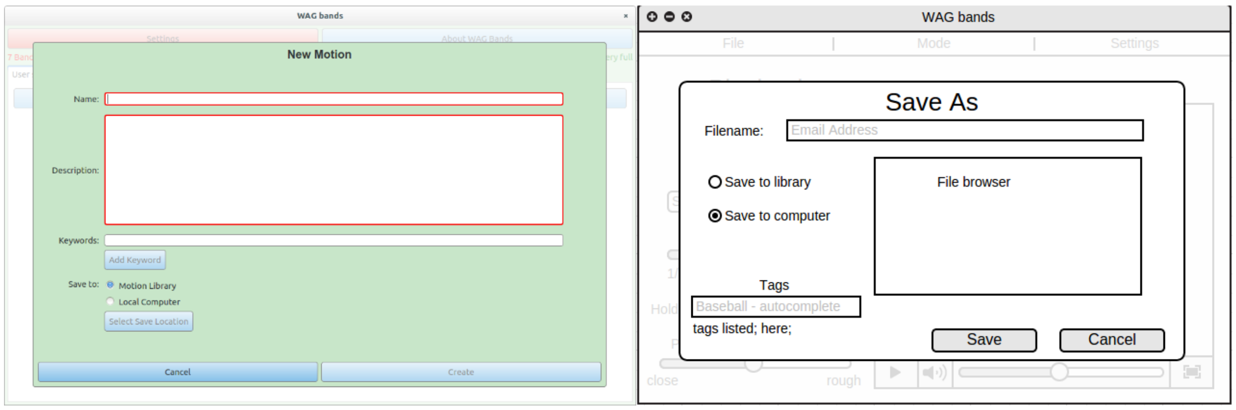


Figure 68. Save as window. Qt Creator on the left, moqups on the right

Appendix 5 – Test Plans

Software Testing

Communications

Testing communications capabilities of the software took place in a few stages. The first was developing a basic server in within the application and using a TCP command line utility to connect and send messages to the IP address and port number associated with the messages. The next step in testing was to create a simple C program that would wait for data, and when received, would reverse the string, and then send it back to the application. This was used to test basic communication capabilities, and that messages were fired on certain button presses.

Message Processing

After significantly more application development, the application message handling and propagation was tested. To accomplish this, a C program was developed that simulates behavior of the physical bands. This enabled testing of communications and application functionality before the physical bands were complete. When the user pressed connect, record, or playback in the application, messages would be sent to the C program. The program would then respond accordingly. In the case of starting recording or playback, the program would send mock IMU data. Print statements embedded in the computer application would then display how the information was processed and propagated through the application. This confirmed that messages were being received, parsed, and saved properly. In playback mode, the C program would print out the returned error data that would be converted to motor vibrations. The bytes received by the mock band program could be examined to verify that this error was being calculated and transmitted correctly.

Kinematics

The visualization and human model software classes include several methods that track the user's pose in three-dimensional space based on the relationships between the user's limb orientations and the model's limb lengths. These relationships are referred to as the kinematics of the system. Unit tests were created using QtTestLib to test the functionality of the kinematic models. These tests verified the functionality of the operations implemented to track pose updates and error calculations. These unit tests tested each method of the AbsState, AbsError, and AbsPose classes, and each of their subclasses (e.g. QuatState, QuatPose, and QuatError). The test procedure included developing a series of test inputs and expected outputs for each method, and evaluating that method with those inputs to verify that that method's output matches the expected output. This test procedure identified multiple errors and missed edge cases in the software, which were then fixed. Also, with these test cases in place, any future updates to the kinematics methods can be immediately comprehensively tested using the same core tests.

Hardware Testing

Testing of the hardware has been ongoing since early October. Since that time, subsystems such as the communications Wi-Fi chips, IMU, feedback motor controllers and the low battery monitoring system have been tested independently and verified. For each subsystem embedded C code was developed for integrating the system with the Teensy 3.2 microcontroller. In mid-December, an initial run of PCBs was created to integrate all of the subsystems in a single board. In early January, these boards were assembled and tested using the subsystem code that was developed. Connections on the PCBs were also verified to enable the functionality of each subsystem. To see the detailed testing process for each subsystem see Table 14.

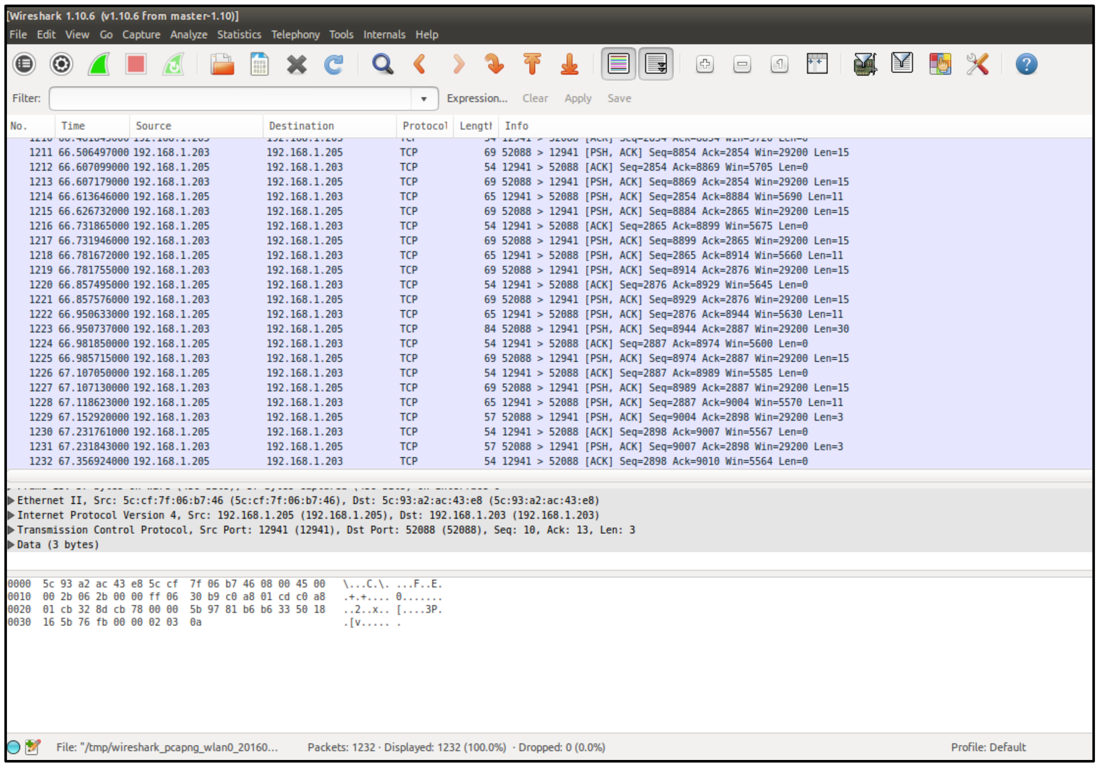
Table 14 - Subsystem testing for hardware

Hardware tested	Subsystem	Testing strategy
MPU6050 (Accelerometer and gyroscope)	Sensors	Check connections to Teensy Read position information from internal FIFO using Teensy Send position information over Wi-Fi to computer and check data integrity
ESP8266 (Wi-Fi)	Communications	Check connections to Teensy Check received Wi-Fi packets for data integrity from computer application Receive accelerometer data from Teensy via UART Send data received from computer to Teensy via UART Connect to Wi-Fi access point and wait for computer Check connect/reconnect to computer Check current usage to be under 250mA
Low battery monitor circuit	Power distribution	Check connections to Teensy Check can send interrupt to Teensy Check can be set low battery threshold at varying voltage between battery and 1V
Motor controllers	Feedback	Check connections to Teensy Check each motor can be driven with PWM signal with duty cycle ~0% to ~100% Check motor vibration works correctly
3.3V Regulator	Power distribution	Check connections to supply power to motors and ESP8266 Check dropout voltage below 3.3V input voltage
3.7V battery	Power distribution	Check connections to Teensy and MPU6050 Check can be charged using external charger by changing power switch

ESP8266 Testing

After changing to the custom Arduino Core for the ESP8266 Wi-Fi device, the stability of the chip began to degrade. Occasionally the chip would crash or trigger a system reset as the watchdog timer (a module designed to reset the board in case of a failure) would catch a failure. Various solutions were proposed online, but the only solution that worked was a 1000μF capacitor on the back of the ESP8266 (see Figure 44). With the final code running on the ESP8266 it was found that the module draws around 80mA during normal operation and draws up to 120mA of current during sending of Wi-Fi packets. This was checked with the power supply providing current to the board. The ESP8266 did emit small spikes in current which were detected in the low battery monitoring circuitry, but this was accounted for by setting the threshold a bit above the desired low-battery voltage setting.

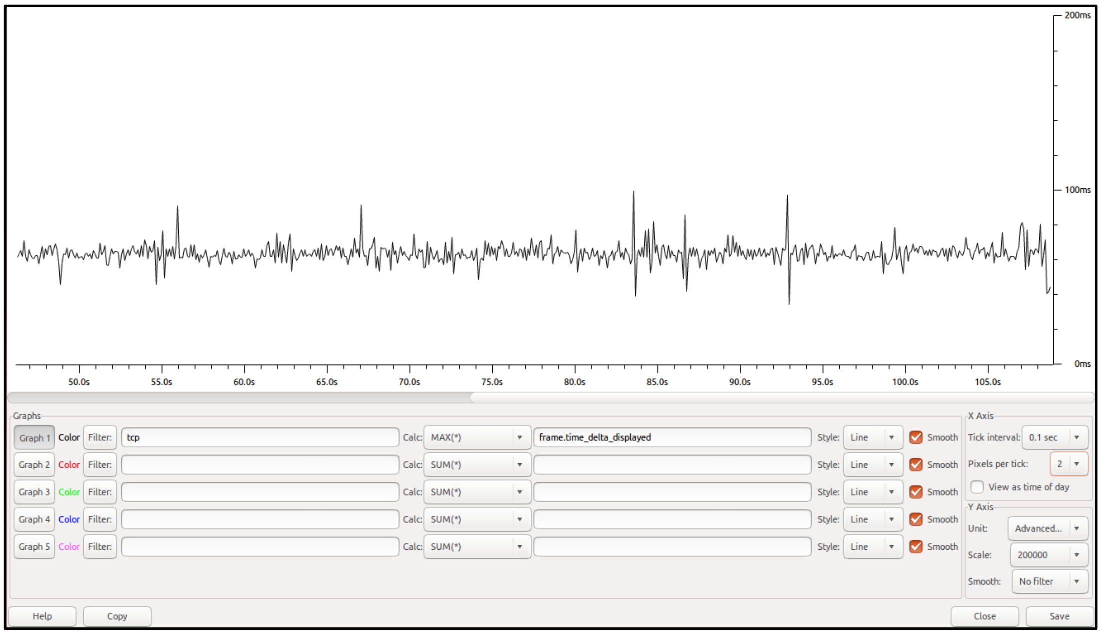
To test the Wi-Fi speed of the ESP8266 and the computer, Wireshark, an open-source application for measuring various types of wired and wireless network traffic, was used to capture and verify the contents of messages between the computer and Wi-Fi chip. A capture of several packets from a single band and the computer can be seen in Figure 69. The main body of the program shows TCP traffic between 192.168.1.205 (a band) and 192.168.1.203 (a laptop computer).


[Wireshark 1.10.6 (v1.10.6 from master-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.   Time          Source          Destination     Protocol Length Info
 1211 66.506497000 192.168.1.203   192.168.1.205   TCP        69 52088 > 12941 [PSH, ACK] Seq=8854 Ack=2854 Win=29200 Len=15
 1212 66.607099000 192.168.1.205   192.168.1.203   TCP        54 12941 > 52088 [ACK] Seq=2854 Ack=8869 Win=5705 Len=0
 1213 66.607179000 192.168.1.203   192.168.1.205   TCP        69 52088 > 12941 [PSH, ACK] Seq=8869 Ack=2854 Win=29200 Len=15
 1214 66.613646000 192.168.1.205   192.168.1.203   TCP        65 12941 > 52088 [PSH, ACK] Seq=2854 Ack=8884 Win=5690 Len=11
 1215 66.626732000 192.168.1.203   192.168.1.205   TCP        69 52088 > 12941 [PSH, ACK] Seq=8884 Ack=2865 Win=29200 Len=15
 1216 66.731865000 192.168.1.205   192.168.1.203   TCP        54 12941 > 52088 [ACK] Seq=2865 Ack=8899 Win=5675 Len=0
 1217 66.731946000 192.168.1.203   192.168.1.205   TCP        69 52088 > 12941 [PSH, ACK] Seq=8899 Ack=2865 Win=29200 Len=15
 1218 66.781672000 192.168.1.205   192.168.1.203   TCP        65 12941 > 52088 [PSH, ACK] Seq=2865 Ack=8914 Win=5660 Len=11
 1219 66.781755000 192.168.1.203   192.168.1.205   TCP        69 52088 > 12941 [PSH, ACK] Seq=8914 Ack=2876 Win=29200 Len=15
 1220 66.857495000 192.168.1.205   192.168.1.203   TCP        54 12941 > 52088 [ACK] Seq=2876 Ack=8929 Win=5645 Len=0
 1221 66.857576000 192.168.1.203   192.168.1.205   TCP        69 52088 > 12941 [PSH, ACK] Seq=8929 Ack=2876 Win=29200 Len=15
 1222 66.950633000 192.168.1.205   192.168.1.203   TCP        65 12941 > 52088 [PSH, ACK] Seq=2876 Ack=8944 Win=5630 Len=11
 1223 66.950737000 192.168.1.203   192.168.1.205   TCP        84 52088 > 12941 [PSH, ACK] Seq=8944 Ack=2887 Win=29200 Len=30
 1224 66.981850000 192.168.1.205   192.168.1.203   TCP        54 12941 > 52088 [ACK] Seq=2887 Ack=8974 Win=5600 Len=0
 1225 66.985715000 192.168.1.203   192.168.1.205   TCP        69 52088 > 12941 [PSH, ACK] Seq=8974 Ack=2887 Win=29200 Len=15
 1226 67.107050000 192.168.1.205   192.168.1.203   TCP        54 12941 > 52088 [ACK] Seq=2887 Ack=8989 Win=5585 Len=0
 1227 67.107130000 192.168.1.203   192.168.1.205   TCP        69 52088 > 12941 [PSH, ACK] Seq=8989 Ack=2887 Win=29200 Len=15
 1228 67.118623000 192.168.1.205   192.168.1.203   TCP        65 12941 > 52088 [PSH, ACK] Seq=2887 Ack=9004 Win=5570 Len=11
 1229 67.152920000 192.168.1.203   192.168.1.205   TCP        57 52088 > 12941 [PSH, ACK] Seq=9004 Ack=2898 Win=29200 Len=3
 1230 67.231761000 192.168.1.205   192.168.1.203   TCP        54 12941 > 52088 [ACK] Seq=2898 Ack=9007 Win=5567 Len=0
 1231 67.231843000 192.168.1.203   192.168.1.205   TCP        57 52088 > 12941 [PSH, ACK] Seq=9007 Ack=2898 Win=29200 Len=3
 1232 67.356924000 192.168.1.205   192.168.1.203   TCP        54 12941 > 52088 [ACK] Seq=2898 Ack=9010 Win=5564 Len=0

▶Ethernet II, Src: 5c:cf:7f:06:b7:46 (5c:cf:7f:06:b7:46), Dst: 5c:93:a2:ac:43:e8 (5c:93:a2:ac:43:e8)
▶Internet Protocol Version 4, Src: 192.168.1.205 (192.168.1.205), Dst: 192.168.1.203 (192.168.1.203)
▶Transmission Control Protocol, Src Port: 12941 (12941), Dst Port: 52088 (52088), Seq: 10, Ack: 13, Len: 3
▶Data (3 bytes)

0000  5c 93 a2 ac 43 e8 5c cf  7f 06 b7 46 08 00 45 00   \...C.\. ...F..E.
0010  00 2b 06 2b 00 00 ff 06  30 b9 c0 a8 01 cd c0 a8   .+.+.... 0.......
0020  01 cb 32 8d cb 78 00 00  5b 97 81 b6 b6 33 50 18   ..2..x.. [....3P.
0030  16 5b 76 fb 00 00 02 03  0a                        .[v..... .

File: "/tmp/wireshark_pcapng_wlan0_20160...   Packets: 1232 · Displayed: 1232 (100.0%) · Dropped: 0 (0.0%)   Profile: Default
```

Figure 69. A capture of TCP packets during playback mode using Wireshark

By using Wireshark, an "IO Graph" of the difference in arrival times (to tell how quickly the packets are being sent and received) can be constructed to give an idea of how quickly data is being transferred in the system. This sort of plot is shown in Figure 70 for recording mode and Figure 71 for playback mode. From visual inspection, the majority of the peaks in recording mode occur between 60 ms and 80 ms.

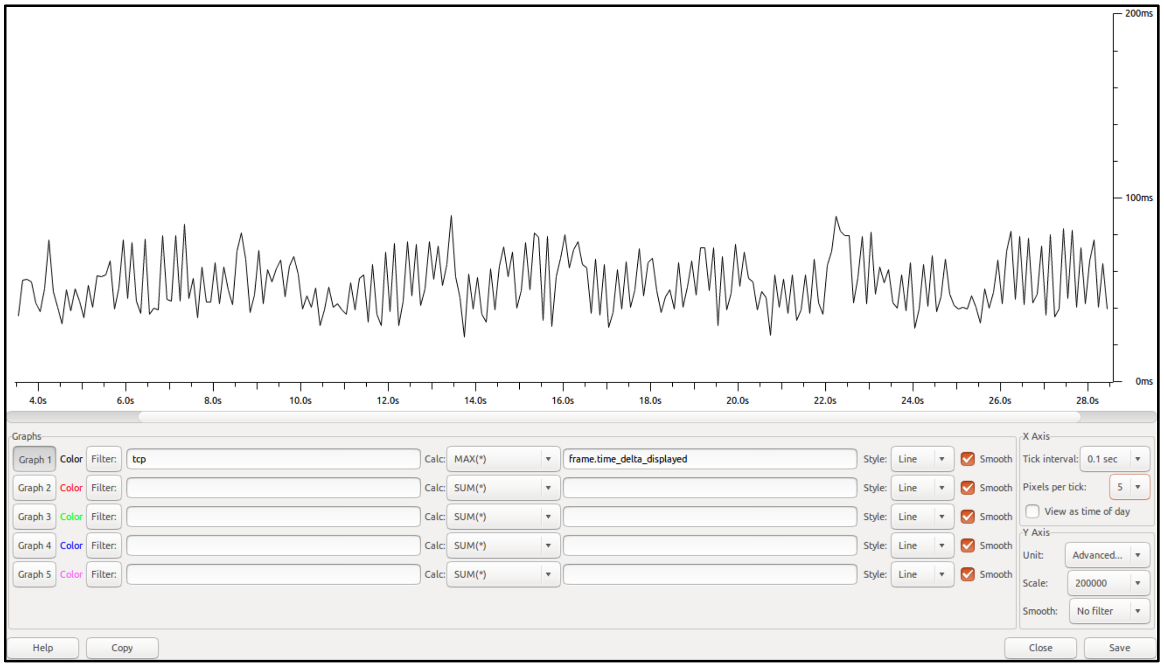


Graph 1	Filter: tcp	Calc: MAX(*)	frame.time_delta_displayed	Style: Line	✓ Smooth
Graph 2	Filter:	Calc: SUM(*)		Style: Line	✓ Smooth
Graph 3	Filter:	Calc: SUM(*)		Style: Line	✓ Smooth
Graph 4	Filter:	Calc: SUM(*)		Style: Line	✓ Smooth
Graph 5	Filter:	Calc: SUM(*)		Style: Line	✓ Smooth

X Axis: Tick interval: 0.1 sec; Pixels per tick: 2; ☐ View as time of day
Y Axis: Unit: Advanced...; Scale: 200000; Smooth: No filter

Figure 70. Plot of time differences between arrival times of packets for recording mode

Looking at Figure 70, the average arrival time is around 60ms between packets (meaning the frequency is around 16.7Hz) for recording. For playback, the results of Figure 71 show that the average arrival time for packets is about the same and is around 60ms or so on average. This means the frequency for playback is around 16.7Hz.



Graphs

Graph 1	Color	Filter:	tcp	Calc: MAX(*)	frame.time_delta_displayed	Style: Line	✓ Smooth
Graph 2	Color	Filter:		Calc: SUM(*)		Style: Line	✓ Smooth
Graph 3	Color	Filter:		Calc: SUM(*)		Style: Line	✓ Smooth
Graph 4	Color	Filter:		Calc: SUM(*)		Style: Line	✓ Smooth
Graph 5	Color	Filter:		Calc: SUM(*)		Style: Line	✓ Smooth

X Axis
Tick interval: 0.1 sec
Pixels per tick: 5
☐ View as time of day

Y Axis
Unit: Advanced...
Scale: 200000
Smooth: No filter

Figure 71. Plot of time differences between arrival times of packets for playback mode

Teensy 3.2 Testing

Since the code inside the Teensy 3.2 and the ESP8266 is not included in this report (see Appendix 2 – Software Core Classes and Functionality), the testing is not covered as deeply. Testing results indicated the Teensy 3.2 was able to successfully integrate floating point math to process error signals coming from the computer, while running UART communications between the ESP8266 and the Teensy. The data transfer rate between the ESP8266 and the Teensy (set at 115200 bits/second) was high enough to minimize latency from reading the current motion position to sending this to the computer via Wi-Fi. The Teensy 3.2's I2C libraries were also fast enough to read data from the MPU6050 at around 116 Hz as measured within the loop of the Teensy code.

PCB Testing

After checking the power distribution of on the PCBs, it was found that the regulated 3.3 V rail and been combined with the raw input voltage rail on all the boards. This necessesitated a design change given the voltage tolerances on the ESP8266. The chest piece design also had the JST connections mirrored onto the underside of the board and as such needed to be changed. Given these two problems, a second round of PCBs was ordered that included the fixes for the voltage rails and the connector issue. This second round of boards became the final boards used for the project. The second round of boards were tested upon arrival and verified to fix the issues previously encountered on the first boards.

This second round of boards also included the ESP8266 breakout board (see Figure 47) that greatly increased testing of the embedded code using the FTDI breakout. The code used the UART on the ESP8266 to print debug messages to the Arduino IDE for testing and verification of the Wi-Fi code. After fixing the issues on the board, the operation of the PCBs was as expected.

Full System Testing

Full system testing of the system indicates that all the core functionality exists. Testing has indicated the system captures motion during recording with minimal latency on the Intel Atom (with dedicated Wi-Fi hardware). However, certain USB Wi-Fi dongles cause packet loss; dedicated Wi-Fi hardware is preferable for optimal WAG System performance.

User tests include evaluating the performance of the software application and the set of bands together. The user tests evaluate the full system to determine its effectiveness as a motion training tool – particularly, to identify the effectiveness of real-time vibrational haptics as a teaching and training mechanism. This is evaluated by surveying test subjects to determine how well they felt that the system adhered to these requirements. The test subjects evaluate the effectiveness of the WAG System for learning new physical skills, in order to compare video-based learning methods with using both the haptic guidance and the built-in motion visualization. The tester reports the effectiveness of the system for motion training with just the visualization, and with the visualization and haptic feedback. The motion training effectiveness tests also have a tester perform the same motion multiple times with haptic feedback to try to learn that motion. The software measures improvements based on the magnitude of the angular errors of each band over the course of the motion. The tests compare the haptic-driven motion learning to the same procedure, except the tester only has access to the visualization as a control. These results are compared to determine if the haptic feedback-based training shows significant improvements in motion training compared to video or visualization-based training. These tests also evaluate the test subjects' natural reactions to three primary vibration patterns – a localized impulse, a sequential rotation, and a superposition of the impulse and sequential patterns – in order to determine if users have natural reactions to these impulses.

The following surveys are used to evaluate the intuitiveness of the WAG System software application, and the functionality and the effectiveness of the WAG System as a physical training tool:

Survey 1: GUI Evaluation

Section 1: Task Intuition

This survey will ask you to complete a few tasks in the WAG System computer application, and will ask you to rate the software's intuitiveness, and to provide open-ended feedback on how you think we can improve the software. If you have any problems along the way, please make note of it in the open-ended feedback section at the end!

1. Record a new motion. Then please rate how intuitive or unintuitive you found the software to use. (1-5 scale, 1 being very unintuitive, 5 being very intuitive).
2. Next, crop the motion you just recorded. How intuitive was the software? (1-5 scale, 1 being very unintuitive, 5 being very intuitive).
3. Now, play back the motion you just recorded. How intuitive was it to complete this action? (1-5 scale, 1 being very unintuitive, 5 being very intuitive).

Section 2: GUI Color Cues

This will evaluate whether the color cues used to encourage you to take a specific action were effective.

1. Did you open the "Settings" window and connect/calibrate the bands before trying to record a motion? (Yes/No/Other)

Section 3: Demographics

1. What is your gender? (Male/Female/Prefer not to answer/Other)
2. What is your age (in years)? (number)

Section 4: Final Thoughts

1. What other thoughts do you have regarding the usefulness and intuitiveness of the software application? How do you think the software could be improved? Are there any features that you think need to be added? (Open response)

Survey 2: WAG System Evaluation

Section 1: Feedback Patterns – Natural Reactions

In this section, the bands will provide different vibrational stimuli patterns to determine the user's natural reactions. (it is unknown to the test subject that the first pattern is a localized vibration impulse, the second pattern is a sequential rotational vibration pattern, and that the third pattern is a superposition of feedback patterns #1 and #2)

1. Feedback Pattern #1 - Don't move in reaction - say out loud how you want to move as a response to the vibration feedback. (Open response)
2. Feedback Pattern #2 - Don't move in reaction - say out loud how you want to move as a response to the vibration feedback. (Open response)
3. Feedback Pattern #3 - Don't move in reaction - say out loud how you want to move as a response to the vibration feedback. (Open response)

Section 2: Training Improvement

In this section, we will have you try to learn a motion by a) watching a visualization of the motion with no feedback from the bands, and b) watching the visualization with feedback from the bands.

1. How effective was solely watching the visualization as a teaching tool? (1-5 scale, 1 being very ineffective, 5 being very effective)
2. How effective was visualization + vibration feedback as a teaching tool? (1-5 scale, 1 being less effective than with only the visualization, 5 being more effective than with only the visualization)

Section 3: Demographics

1. What is your gender? (Male/Female/Prefer not to answer/Other)
2. What is your age (in years)? (number)

Section 4: Final Thoughts

1. What other thoughts or feedback do you have regarding vibration feedback as a motion training tool? Do you have any suggestions for how to improve the vibration patterns in order to better correct motion errors? Is the vibration feedback actually helpful? (Open response)

User Testing Results

Most users found the user interface very intuitive. Figure 72, Figure 73, and Figure 74 show the ratings for intuitiveness of main tasks within the software application, with 1 representing very unintuitive and 5 representing very intuitive. Most responses for these primary tasks were either intuitive or very intuitive.

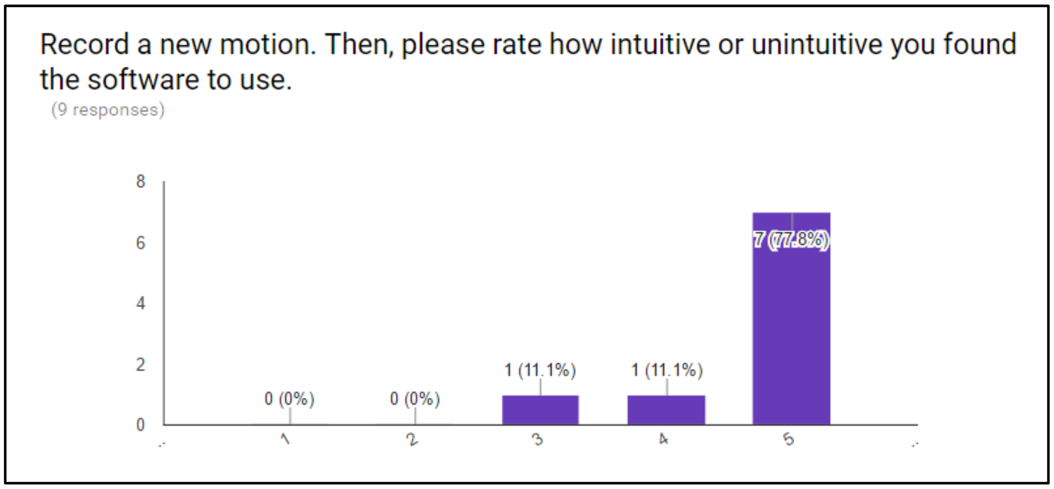


Figure 72. Motion Recoridng Intuitiveness

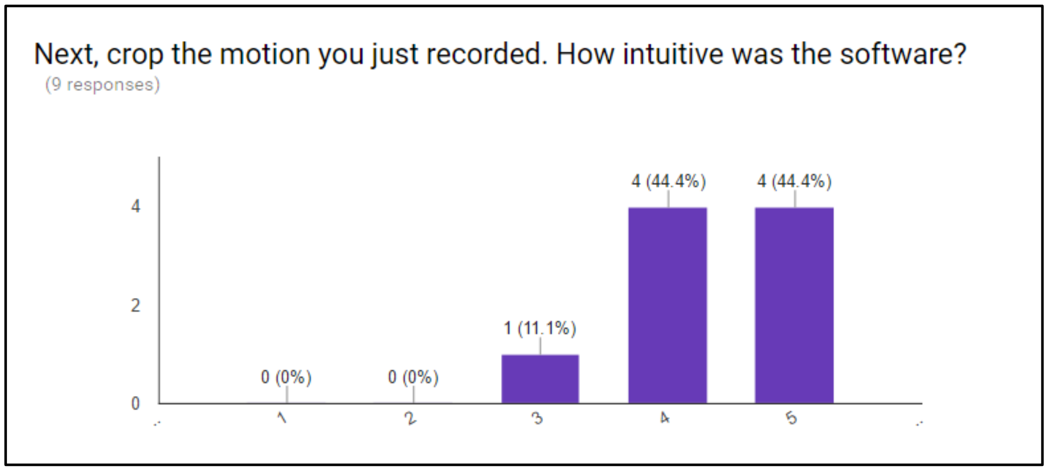


Figure 73. Motion Editing Intuitiveness

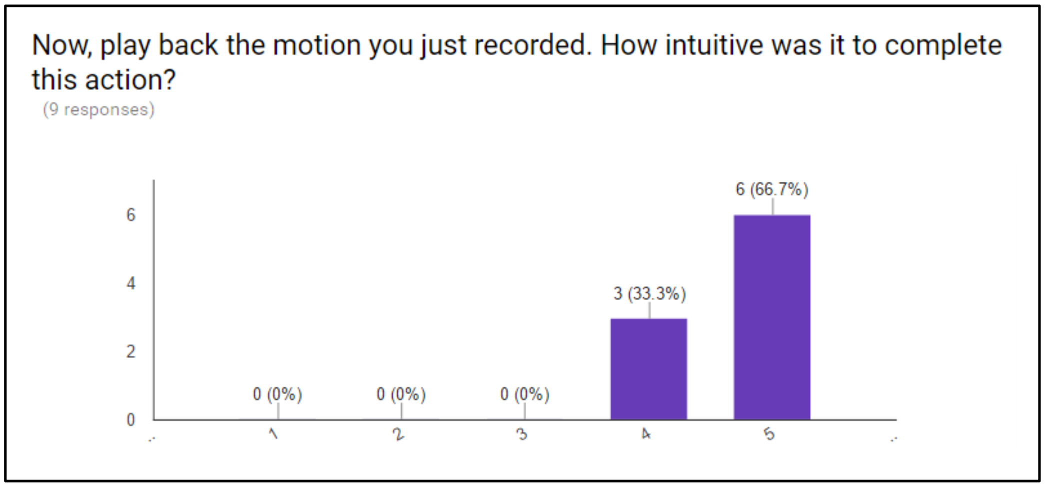


Figure 74. Motion Playback Intuitiveness

However, all users failed to connect and calibrate the bands. The results of this are shown in Figure 75, with the single 'Other selection' indicating that they did not connect and calibrate the bands, plus an additional comment.

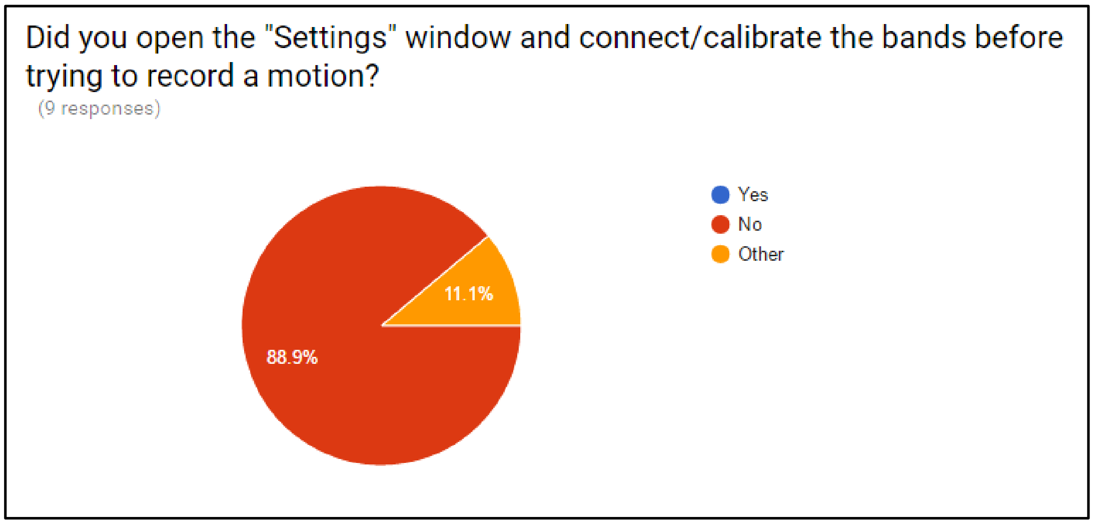


Figure 75. Percentage of users that followed color cues to connect and calibrate bands

Seven out of eight users who responded to the vibrational feedback schemes said that the translational feedback scheme was intuitive. Most users took longer to recognize the rotational vibration scheme, and only five out of eight interpreted the scheme to indicate that they should rotate their limb. Of these five, the intuitive direction to rotate was inconsistent, with some users indicating that they should move their arms with the vibration and others in the opposite direction of the vibration. When subjected to the superimposed translational and rotational signals, only one user correctly interpreted the vibrational signals. Three other users were able to perceive one signal and react to it. The remaining four users indicated that it was clear that they were not correctly performing the motion, but that the vibration scheme did not intuitively convey how to correctly perform the motion.

Users also compared the vibrational and visual components of the WAG System with solely the visual component. Most users found that, in their initial trials, the vibration feedback was distracting. Figure 76 shows how effective the visualization and vibration feedback is compared to solely visual feedback, with 1 representing significantly less effective, 3 being equally effective, 5 being significantly more effective.

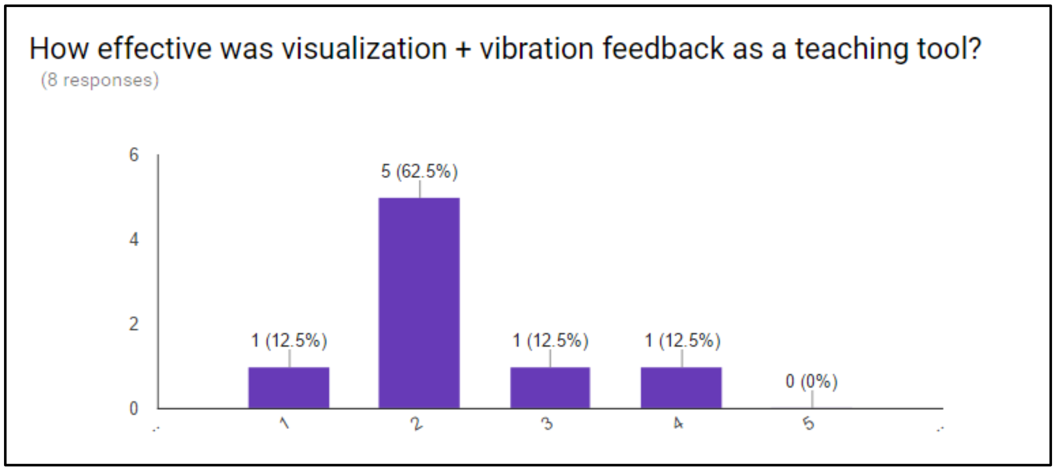


Figure 76. Visualization and vibration feedback effectiveness vs. only visualization

Appendix 6 – Initial Design Steps: Trade Study

This appendix contains a number of trade studies that were performed for different hardware platforms available. These studies guided selection of a MCU and accelerometer/gyroscope during the design phase of the project. Additionally, these studies helped to identify a number of key features to focus on for the selection of the final hardware.

32-Bit MCU Comparison

Initially in the selection of a 32-bit MCU, several hardware options were compared using a scoring matrix appropriate for that hardware. Each option was scored based on the features desired for the system and the Teensy 3.2, powered by the MK20DX256, was selected as the final MCU for the system.

The listing of 32 bit MCUs can be seen in Table 16 and Table 15. Each MCU was ranked based on a feature of the hardware and a grading scheme. The CPU speed categories were <=50MHz (1), 50-100MHz (2) and >=100MHz (3). The RAM in each MCU was about the same and as such was not included. The RAM is sufficient enough to run the small amount of software needed to interface to sensors over SPI. The ADC resolution categories were 12-bit (1), 14-bit (2) and 16 bit (3). The I2C categories were based on the number of I2C connections: 2 (1), 3 (2) and 6 (3). The UARTs section was based on the number of UARTs on the board: 3 (1), 4 (2) and 6 (3). The ease of programming feature was scored on how easy the software required to program the MCU would be to learn (based on previous experience with it).

The same sort of ranking: low (1), medium (2) and high (3) was used to rank the number of resources available for the chip. This included online forum support, any reference designs or schematics, datasheets available and other technical support. The breakout board cost metric is based on the cost of the evaluation board that is available. The rating was high cost (1), medium cost (2) and low cost (3). The MSP432 was the cheapest evaluation board at $12.99 for a functional board, but the Teensy 3.2 breakout was close at only $20 per breakout. The extra features category was scored based on the usefulness of other features of the chip. The Teensy 3.2 scored a 3 because it had all the features of the other MCUs but had the smallest breakout, a touch sensor module and a USB debugger built into the board. The last criteria for scoring was the chip cost. The grading was cost for 1 unit > $14 (1), $9-$14 (2) and < $9 (3). The MK20DX256 (used in the Teensy 3.2) came up as the cheapest at only $7 per unit (without factoring in the volume discount). The package areas were also included to show that the MK20DX256, while being the best by the scoring, does not have the smallest footprint on the board. Despite this minor drawback, the MK20DX256 scored the best on the matrix (as seen in Table 15).

Table 15 - Microcontroller trade study scoring matrix

Chip	**TI F28M35H22C** [50]	**TI MSP432P401R** [51]	**STM32F411CE** [52]	**STM32F205RGT6** [53]	**Atmel AT32UC3C264C** [54]	**MK20DX256** [55]
CPU Speed	3	1	2	3	2	2
RAM	2	2	2	2	2	3
ADC resolution	1	2	1	1	1	3
I2C	3	2	2	2	1	1
UARTS	3	1	1	2	2	1
Ease of programming	1	1	2	1	1	3
Resources available	1	2	2	2	2	3
Breakout board cost	1	3	2	3	1	3
Extra features	2	2	2	2	2	2
Chip Cost	1	3	2	2	2	3
Package area (mm^2)	144	256	49	100	81	100
Total points from column:	18	19	18	20	16	24

The purpose of the trade study was to determine the requirements for various hardware components in a project design. For the study several key hardware features of the system were developed based on the requirements and needs. These features ranged from power supply constraints to number of SPI interfaces to programming interfaces. Soft features such as breakout board availability or ease of programming were also included. A small list of appropriate microcontrollers was then constructed as seen in Table 16 and each device was scored on specific features and functionalities as seen in Table 15. Note that all microcontrollers in Table 16 have a temperature range of -45^{o}C to 85^{o}C and support USB interfaces.

Table 16 - 32-Bit MCU comparison

Chip name	**TI F28M35H22C** [56]	**TI MSP432P401R** [57]	**STM32F411CE** [58]	**STM32F205RGT6** [59]	**Atmel AT32UC3C264C** [60]	**Freescale MK20DX256 (used in Teensy 3.2)** [61]
Company	Texas Instruments	Texas Instruments	STM	STM	Atmel	Freescale
Family	F28M3x	MSP432X	ARM	ARM	AVR UC3	K20
CPU	C28x, Cortex-M3	Arm Cortex-M4F	ARM-Cortex-M4	ARM-Cortex-M3	32-bit AVR	ARM-Cortex-M4
Clock Frequency	150 MHz 100	48 MHz	100 MHz	120 MHz	66 MHz	96MHz
RAM KB	136	64 KB	128 KB, 512 KB Flash	128 KB RAM, 1024 KB Flash	64 KB Flash, 20 KB SRAM	256 KB Flash, 64 KB SRAM
PWM Channels	24	-	-	-	14	12
ADC Resolution	12-bit	14 bit	12 bit	12 bit	12 bit	16 bit
ADC Channels	20	12,16,24	12	16	11	21
Number I2C	3	3,4,4	3, 5x I2S	3, 2x I2S	2, I2S also	2, I2S also
Number of UARTs	6	3,3,4	3x USARTs	4x USARTs, 2x UART	4	3
Number of SPIs	5	6,7,8	5	3	5	2
GPIO	64	48,64,84	36	51	45	40
Timers	4x 16bit	4x 16bit timers, 2x 32 bit timers	6x 16bit timers, 2x 32 bit timers	12x 16 bit timers, 2x 32 bit timers	3	4
Power Rails	2.97V to 3.63V	1.6V to 3.7V	1.7V to 3.6V	1.7V to 3.6V	3.0V to 3.6V or 4.5V to 5.5V	1.71V to 3.6V
Current draw (min/max)	150MHz: 2mA/325 mA 100MHz: 2mA 295mA	850nA (Min) LPM, 90uA/MHz * 48MHz	Current: 1.8uA LPM, 100uA*100MHz max	2.5uA LPM, 188uA*120MHz	48uA/MHz * 66 MHz (Min),512uA /MHz * 66 MHz (Max)	39mA (all peripherals running)
Package Type/Size	HTQFP 144	LQFP 100 16.2x16.2x1.4mm	UFQFPN 48 7x7x0.55mm	LQFP 64 10x10mm	QFN64_V 9x9mm	LQFP 64 10x10mm
DACs	6	0	2	2	1x 12 bit (2 channels)	1x 12 bit
Programming option/Debug	JTAG emulator	JTAG/Serial Wire Debug (SWD)	JTAG/Serial Wire Debug (SWD)	JTAG	JTAG	USB OTG, JTAG
Other Features	IEEE-754 single-precision floating point	AES encryption, floating point unit	RTC, 2.4 MSPS ADC	RTC	Floating point unit, 1.5MSps ADC	RTC module, CAN controller, touch sensor module

Chip name	**TI F28M35H22C** [56]	**TI MSP432P401R** [57]	**STM32F411CE** [58]	**STM32F205RGT6** [59]	**Atmel AT32UC3C264C** [60]	**Freescale MK20DX256 (used in Teensy 3.2)** [61]
Cost each	$22 from Arrow [62]	$7 from TI [63]	$7.35 each from AVNET [64]	$13.08 each from Digikey [65]	$9.5 for AT32UC3C264C-Z2UT from ATMEL	$7 for MK20DX256VLH7-ND from DigiKey
Breakouts	$185 from TI [66]	$13 eval board [67]	$22 [68] or another board for $8 [69]	Simple Wi-Fi board [70] or $330 for full eval board [71]	$330 eval board [72]	$20 Teensy 3.2 Module breakout [73]
Cons:	Have to design RF layout a bit or copy reference diagram, expensive Breakout/ dev board	No DACs	Have to learn the new programming environment, have to buy separate modules and solder together for eval board	Have to learn programming environment	Have to get avr32program Have to learn AVR32 Studio 2.6 Expensive dev board	Cost of breakout board
Pros:	Lots of resources from TI and reference designs	Very easy eval board Easy to program from Code Composer (CCS)	Integrated Wi-Fi Support Easy drag and drop module chip	Programming could be simplified via eval board	Free software Lots of forums/ support RTC counter	Free software loader, Low cost with multiple libraries availble, Programmable using Arduino IDE

Accelerometers

This section contains a trade study performed in a similar manner to that of the 32-bit MCUs. While the final accelerometer used in the project was not on this list, Table 17 shows the original work involved in considering an accelerometer for the project.

Table 17 - Accelerometer Specifications and Pricing

Manufacturer	**ADI**	**ST**	**ADI**	**InvenSense**	**Freescale**
Chip type	Triple axis accelerometer	Triple axis accelerometer	Triple axis accelerometer	Triple Axis Accelerometer/Triple Axis Gyroscope	Triple accelerometer and triple magnetometer
Part	ADXL345	LIS331	ADXL362	MPU6050	FXOS8700CQ
Supply voltage	2.0 - 3.6V	2.1V to 3.6V	1.6V to 3.5V	2.375V to 3.46V	1.9V to 3.6V
Current	0.1uA to 40uA	10uA to 250uA	0.01uA to 3.3uA	10uA to 500uA	2uA to 575uA
Interface	I2C, SPI (3 and 4 wire)	I2C, SPI (3 and 4 wire)	SPI (4 wire)	I2C	I2C, SPI (3 and 4 wire)
Range	3, 4, 8, 16g	6, 12, 24g	2, 4, 8g	2,4,8,16g and 250,500,1000, 2000 °/s	2, 4, 8g/1200uT
Data resolution	13 bit	16 bit	12 bit, 8 bit	16 bit for both	14 bit accelerometer, 16 bit magnetometer
Other pins	2x interrupt pins, measure down to 1.0 degrees	2x interrupt pins	2x interrupt pins, Low noise modes	1x interrupt, alternate address lines interface	1x interrupts
Cost	$18 (breakout) [74]	$28 (breakout) [75]	$15 (breakout) [76]	$5.45/chip [77] $3 a breakout [78]	$2.63/chip [79] $16 (breakout) [87]
Development	Libraries available	Libraries available	Libraries available	Small and compact cheap breakouts available ($3), Many libraries available	Breakout available

Gyroscopes

This section contains a trade study performed for gyroscopes in a similar manner to that of the 32-bit MCUs. While the final gyro used in the project was integrated into the MPU6050 (above in Table 17), Table 18 shows the other gyro considered for the project.

Table 18 - Gyroscope Specifications and Pricing

Part	**LPY503AL**	**L3G4200D**	**ITG-3200**	**MPU-3050**	**MPU-6500**	**FXAS21002C**
Manu-facturer	ST	ST	InvenSense	InvenSense	InvenSense	Freescale
Chip type	2 axis gyro	3 axis gyro	3 axis gyro	6-axis, gyro with I2C link to external accel	6 axis ac-cel/gyro	3 axis gyro
Supply voltage	2.7V to 3.6V	2.4V to 3.6V	2.1V to 3.6V	2.1V to 3.6V	1.71 to 3.6V	1.95 to 3.7V
Current	1uA to 6.8mA	5uA to 6.1mA	5uA to 6.5mA	6.1mA	6.37uA to 3.5mA	2.7mA
Inter-face	Analog	I2C, SPI (3 wire)	I2C (400kHz)	I2C (30kHz)	I2C (400kHz), SPI (1MHz)	I2C (100kHz/40kHz), SPI (3 and 4 wire 2MHz)
Range	30 (°/s), 120 (°/s)	250,500,2000 (°/s)	2000 (°/s)	250, 500, 1000, 20000 (°/s)	250, 500, 1000, 2000 (°/s) 2, 4, 8, 16g	250, 500, 1000, 2000, 4000 (°/s)
Data resolu-tion	8.3mV/(°/s), 33.3mV/(°/s)	16 bit data	14.375 LSB/(°/s) 16 bit	131LSB/(°/s), 65.5LSB/(°/s), 32.8LSB/(°/s), 16.4LSB/(°/s)	16 bit	16 bit data, 0.0625 (°/s) at 2000 (°/s)
Other pins	Amplifier (4x) pins/non-am-plified pins, external filter connections	2 interrupt pins	1 interrupt	1 programma-ble interrupt	1 programma-ble interrupt	1 programma-ble interrupt
Fea-tures	Zeroing, low pass filters	Embedded temp sen-sor, external filter specs	Digitally controlled LPF	programmable LPF, HPF	Digital motion processor for gestures programmable filters, temp sensor	LPF, temp sen-sor
Cost	$30 dev board [80]	$50 dev board [81]	$25 dev board [82]	$9 chip $70 dev board [83, 84]	$10 chip [85]	$3.5 chip, $15 dev board [86, 87]
Devel-opment	Reference layout in datasheet, li-braries avail-able	Reference layout in datasheet, libraries available	Example wiring guides for mBed MCU, schematics	Not much in-formation	No eval boards	Breakout board and reference designs
Package	LGA16, 5x5x1.6mm	LGA16, 4x4x1.1mm	QFN20, 4x4x0.9mm	24-QFN 4x4mm	24-QFN 4x4mm	24-QFN 4x4mm

Appendix 7 – Gap Analysis (Desired Features Analysis)

A gap analysis involves comparing the current state of the art of technology to the desired capabilities of a project, to understand the "gap" and the project's feasibility. Table 19 outlines the desired features of this project, relevant information about the current state of the art and its limitations, topics that need further research and development, and possible risks associated with meeting each desired feature. As the table illustrates, the current state of the art is very close to the features and technologies identified in this project proposal, which suggests that this project is feasible.

Table 19 - Gap Analysis chart containing desired features and additional information

Desired Feature	**Current state of the art (and cost)**	**Limitations to the state of the art**	**Research and development topics**	**Possible risks**
Portable system	Exoskeletal hand with wireless remote control [88] Wireless motion capture gloves [89]	Exoskeleton's power consumption Batteries increase size and weight of suit Sensors and haptic modules increase system weight	Suitable wireless communication protocols How much power does the system need How much power do sensor/comms need	Power supply is too big to comfortably make portable Too expensive Too many batteries
Implements haptic feedback	DexMo glove freezes when robot hand senses that it can't move [**5**] Using the sensAble stylus to control and receive haptic feedback from Baxter [90]	Force feedback system too large and bulky Limitations of the controlling hardware using actuators and AC voltages for LRA motors	Various feedback types and characteristics How sensitive they are Easiest ways to build haptic feedback systems (low cost options) [91]	Feedback mechanisms are too expensive Feedback is not detailed enough to guide user to correct position
Real-time response (<1 second) latency	LCH is 500 ms [92] Telekyb[93] framework of 0.3 seconds Human body imitation at 1.5 seconds [94] MVN Link - 240Hz refresh MVN Awinda 60Hz	LCH uses Arduino system, but recommends ARM for better math Wireless transmission time (data size) MVN Awinda - 30ms MVN Link - 20ms	Optimal data size for transmission Data rates of wireless protocols Data rates of wired protocols Latency for motion capture ADC sampling rates MCU clock speeds	Latency from wireless system could higher than response rate given overhead Response rate of MCUs might not be high enough (clock speeds)
Cost	Market Costs: Araig - $500 TeslaSuit $600 Xsens suits: The MVN suits [**16**] MVN Awinda $7.4K MVN Link $12.2K	Grant money Sponsorship money	Costs of sensors Costs of hardware Costs of feedback (haptics) Costs of software	Overspending the budget

Desired Feature	**Current state of the art (and cost)**	**Limitations to the state of the art**	**Research and development topics**	**Possible risks**
Battery life >= 2 hours	Motion capture suits: MVN Awinda – 6 hours [**16**] MVN Link – 9 hours [**16**] IGS-190 runs for 3 hours on NiMHs Hulc from Lockheed Martin with 72 hour batteries	Current battery capacities Motors draw use too much power More dense batteries = more power = more weight Interchanging batteries means more hardware Battery charging adds more complexity to design	Current battery tech Motors for feedback General power consumption for sensors Costs of batteries Best rechargeable batteries Weight of batteries	Batteries not good enough (small enough) Vibrators take too much current
Ability to power over tether	Most current exoskeletons are tethered Gypsy is USB tethered (~500mA) 5DT gloves use up to 150mA per glove powered from 9V power supply	Wire size not suitable for current load Cable length creates too much loss of power Power supply not able to support power Tethering adds complexity to design (managing of wires)	Powering over tether for distributed power systems Wire sizes and current limitations Power over Ethernet	Power distribution system makes system more rigid

Desired Feature	Current state of the art (and cost)	Limitations to the state of the art	Research and development topics	Possible risks
Ability to capture motion & save motion path	Optical motion capture using motion capture dots or manual or automatic feature detection in video, ex. Vicon, Phase Space ex. Vicon, Phasespace Inertial motion capture using combination of accelerometers, gyros, and magnetometers, ex. Synertial (gloves and body suits), Xsens (individual sensors > $1000; full body suits: above)Existing file formats require costly editing software (Xsens Studio, 1 yr - $5400), Autodesk MotionBuilder, ~$4000 Audio Gypsy system (IGS-190) This system uses sonar sensors and mics to record the sensor outputs Need more mics to record system and need external costly setup	Optical Many cameras require large space and minimal view obstructions Expensive and difficult to set up Difficult to track features if person's limb obstructed Inertial High drift unless many sensors and advanced filtering used Many systems restricted to major limbs due to error (support arms and legs, not fingers, etc)	Formatting data so that it can be cropped/paused Existing motion capture file formats Filtering sensor readings for smooth motion paths	Incompatibility with existing file formats limiting usability of recorded motions proprietary/restricted motion capture file formats Sensor drift in inertial sensors could be difficult to limit, leading to inaccurate position estimates

Desired Feature	**Current state of the art (and cost)**	**Limitations to the state of the art**	**Research and development topics**	**Possible risks**
Non-intrusive, low-profile, non-restricting	Optical Unrestricted user motion within the configured space [95] Inertial User is much more globally mobile (not confined to a single configured mocap space) due to the motion capture system being attached to the body The hexoskin doesn't do motion capture, but it collects other biometric data with a relatively unrestrictive form factor ($400) [96] Actuating exoskeletons Vanderbilt rehabilitation exoskeleton weighs 27 lbs [97]	Optical User is confined to a single room or setup area Haptic feedback is impossible with a purely optical system - need some sort of worn device Inertial User's limbs have added bulk due to sensors being attached directly to the body rather than using an external imaging system, but this limitation can be as minimal to what feels like a tight-fitting outfit Actuating exoskeletons Having actuation to directly aid user motion adds an additional degree of rigidity to the system, which prevents the user from recording natural movements	Need to create a system that balances user's freedom to move naturally with ability to move around in different locations Need to develop system that can guide a user to move in a certain manner without adding substantial rigidity/motion limitations to the system	Increasing user's ability to move naturally and incorporating motion guidance may be direct trade-offs
Control through speech commands	Google web speech API sends audio to external servers and gets transcribed offsite CMU Sphinx/PocketSphinx are standalone speech recognition systems. PocketSphinx is intended for embedded platforms	Voice recognition requires additional microphone hardware (cost and additional processing)	Libraries and necessary hardware	Processing time for speech recognition may introduce delay in process of suit operation incompatibility of the hardware with the system

Desired Feature	**Current state of the art (and cost)**	**Limitations to the state of the art**	**Research and development topics**	**Possible risks**
Database of motion capture	mocap.cs.cmu.edu Library of motion capture data All recorded at CMU using Vicon optical motion capture system Supports several motion capture file formats. mocapclub.com Motion Capture Data library large variety of motions recorded	Neither seem to allow 3rd party upload of motion capture	Existing motion capture file formats Database management Allowing uploads and downloads to database Suitability of existing file formats for haptics	Managing potentially large database Managing categories of motion capture data (want large amount of data to be available, but still find desired content without digging through all of the data)

Appendix 8 – Precursor Testing Before System Design Stages

This section describes a few ideas and tests that were conducted before settling on the final hardware design for the project. The reason for these being different from the normal report is due to the fact that these were not used in the final project, but served as useful information to assist the project.

Ubuntu Bluetooth Libraries

Bluez is the default Bluetooth protocol stack for Linux machines. Bluez has a low level API for discovering Bluetooth devices and sending and receiving messages as well as a higher level API for pairing and Bluetooth profiles and services. Both APIs have little documentation available. The low level API has direct C/C++ support, whereas the higher level API must be accessed through DBus calls. DBus is an interprocess communication protocol. The Bluez DBus API allows function calls to be sent to the core Bluetooth libraries from a program through the *DBus* protocol. However, there is little documentation on a high-level C/C++ binding for the DBus protocol or the Bluez DBus APIs.

In addition, the Bluez version on a machine varies with the Linux distribution. Bluez 5 is the latest version and is the only version with significant support for Bluetooth LE and Bluetooth Smart devices. Bluez 5 is only officially compatible with Ubuntu 15.10 (an open-source Linux operating system), which at the time of this draft, has been available for only a few months.

The tools tested with Bluez were the low level APIs, which were found could scan and find Bluetooth devices, and the DBus API. Although the DBus API was not particularly useful due to its complex nature.

Qt also has a Bluetooth API that provides support for ClassicBluetooth, Bluetooth Smart, and Bluetooth Low Energy devices. The Qt Bluetooth API includes functions for scanning for devices, pairing with devices, connecting to devices, and sending data through sockets or Bluetooth profiles and services. The Qt APIs use the Bluetooth protocol stack of the operating system running the application. There are several examples for both classic and low energy Bluetooth within Qt. Using the Qt Bluetooth libraries, the software application was able to scan for external Bluetooth devices, and connection and pairing attempts with Bluetooth devices were tried. However, this approach was ultimately abandoned in favor of Wi-Fi options due to the complex nature of the libraries and system specific dependencies that could not be resolved.

Along with the libraries, a number of Bluetooth hardware modules were tried. The hardware modules tested included the Adafruit Bluefruit LE UART Friend ($20), the Adafruit nRF8001 Bluetooth LE breakout ($20), several JY-MCU HC-06 BT 2.0 modules and the Roving Networks RN4020 BTLE device. The accompanying embedded software libraries for these devices were found to perform at max capacity of around 10Hz duplex communications with the computer.

Initial Band Prototype

Throughout the system development many hardware changes were made. Below is a picture of all the generations of the band prototypes as different features were added to accomidate increasing functionality. The current prototype can be seen on the left with the electrical hardware installed.

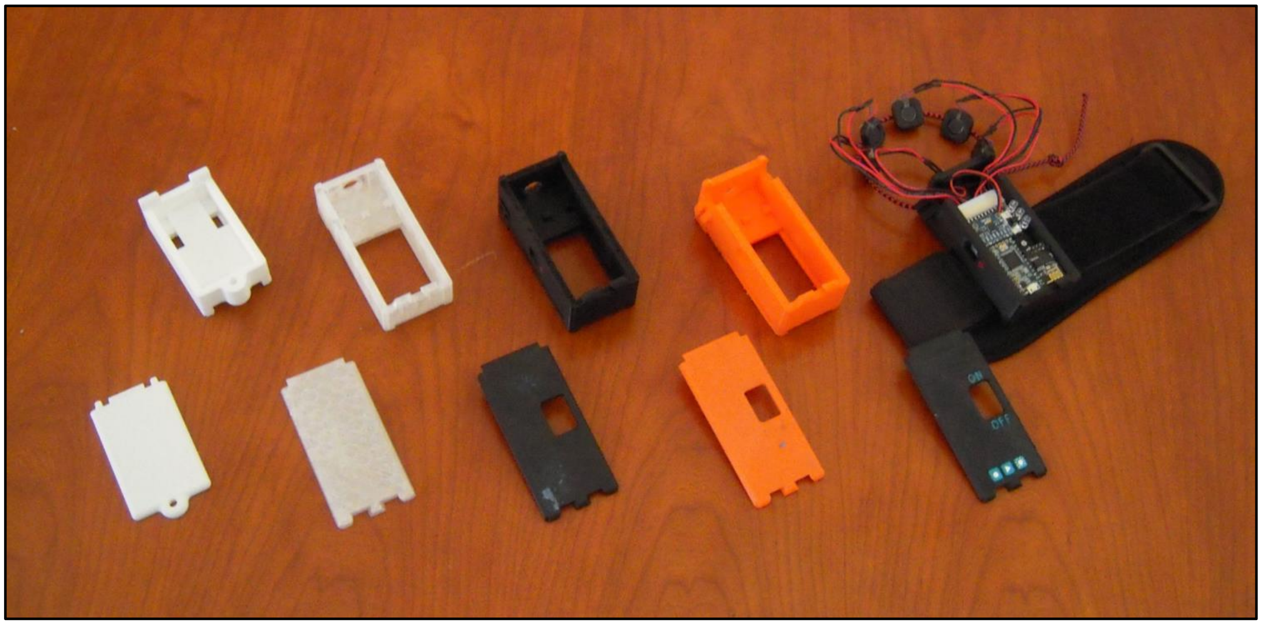


Figure 77. Band prototype progression

FRDM-K22F MBed and Sensor Fusion Toolbox

The FRDM-K22F MBed microcontroller (MCU) was purchased to support a simple demo of sensor fusion tools within the project (at the start of the project). To complement the MCU, a simple 9 degree of freedom (DOF) sensor board, called the FRDM-STBC-AGM01 and produced by Freescale, which features an accelerometer and magnetometer combo and a gyroscope was used to demonstrate and evaluate the effectiveness of sensor fusion techniques. To this end, Freescale provides all the C code (extensible to other microcontroller platforms) to run and support the sensor fusion on the sensor board. The setup can be seen in Figure 78.

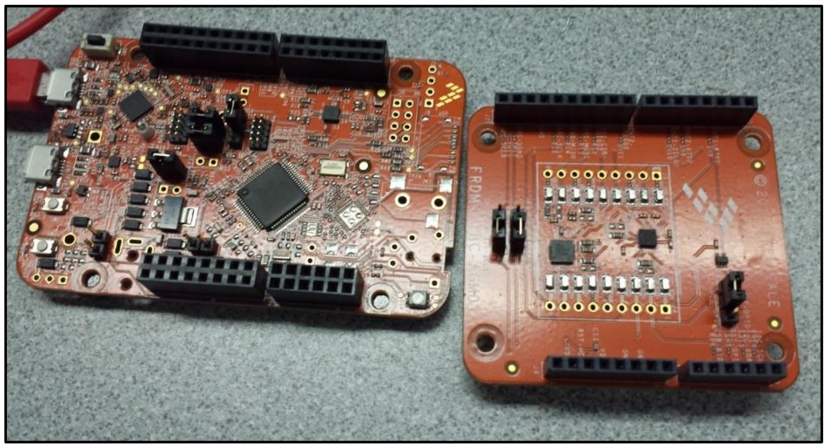
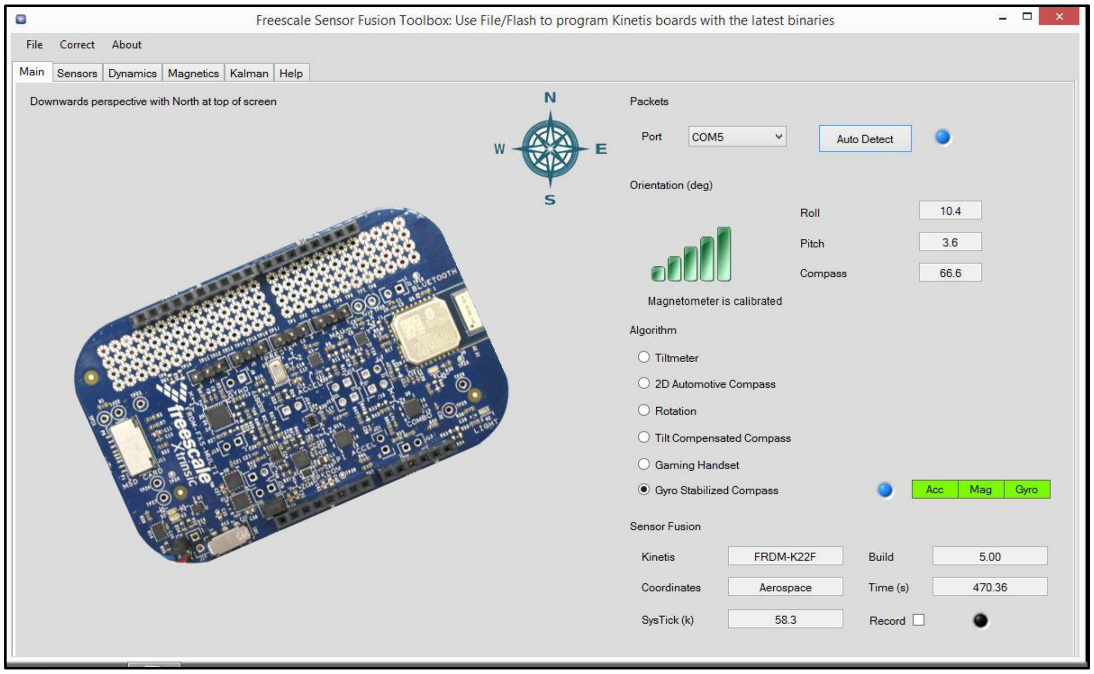


Figure 78. MBed (FRDM-K22F) on left and 9DOF sensor board on right

Testing of this device relied on the use of the sensor fusion toolbox program developed for Windows by Freescale (seen in Figure 79). The benefit of using this program is that it provides an easy way to enable and disable various sensors. On the left portion of the window is a view of the FRDM MCU board that will move correspondingly to the 9 DOF sensor board being moved. The movement is identical to the real life position of the board as seen in Figure 80. This shows the board and the onscreen orientation. Within the GUI, each sensor can be turned off independently. By comparing combinations of various sensors within the program, it was determined that the combination of the accelerometer and gyro afforded the greatest accuracy and least drift over time of the board's position. From this testing, the next step was to find a suitable accelerometer and gyro combination that could be easily interfaced with the MCU system. This was determined to be the MPU6050 (described in the Hardware Overview section) based on the wide available of the low-cost breakout board.


Freescale Sensor Fusion Toolbox: Use File/Flash to program Kinetis boards with the latest binaries
File
Correct
About
Main
Sensors
Dynamics
Magnetics
Kalman
Help
N
W
E
S
Auto Detect
Acc
Mag
Gyro

Downwards perspective with North at top of screen

Packets

Port COM5

Orientation (deg)

Roll 10.4

Pitch 3.6

Compass 66.6

Magnetometer is calibrated

Algorithm

- Tiltmeter
- 2D Automotive Compass
- Rotation
- Tilt Compensated Compass
- Gaming Handset
- [✓ Gyro Stabilized Compass]

Sensor Fusion

Kinetis FRDM-K22F

Build 5.00

Coordinates Aerospace

Time (s) 470.36

SysTick (k) 58.3

Record ☐

Figure 79. Sensor fusion toolbox program

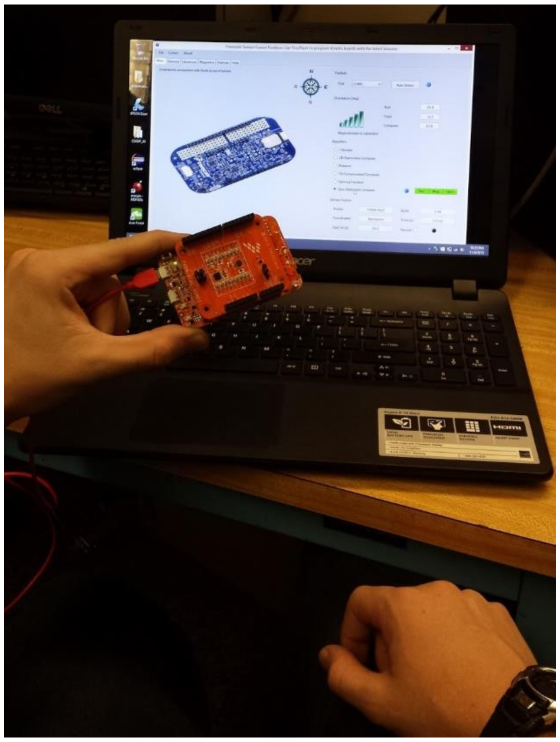
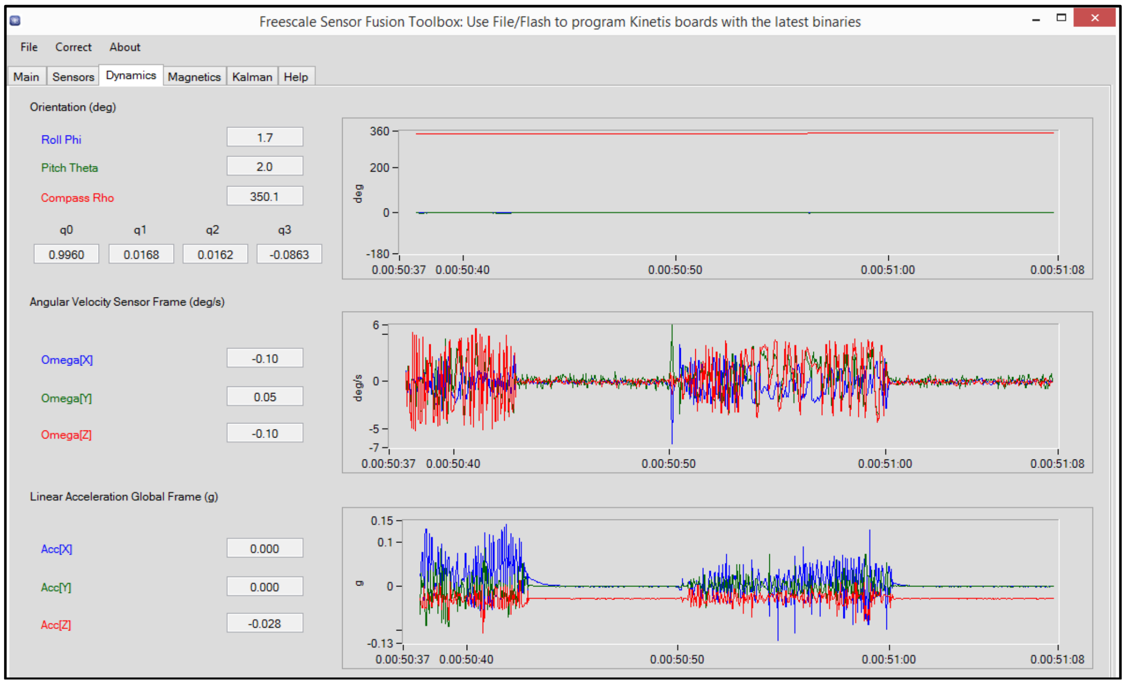


Figure 80. Viewing the FRDM-K22F board and Sensor Fusion Toolbox concurrently

Another benefit of testing out sensor fusion techniques with this board was the ability to visualize the raw sensor readings and the output of Kalman filters. These can be seen in Figure 81 and Figure 82. These readings provided a greater insight into the sensor fusion algorithms.


Freescale Sensor Fusion Toolbox: Use File/Flash to program Kinetis boards with the latest binaries
File
Correct
About
Main
Sensors
Dynamics
Magnetics
Kalman
Help
deg
deg/s
g

Orientation (deg)

Roll Phi	1.7
Pitch Theta	2.0
Compass Rho	350.1

q0	q1	q2	q3
0.9960	0.0168	0.0162	-0.0863

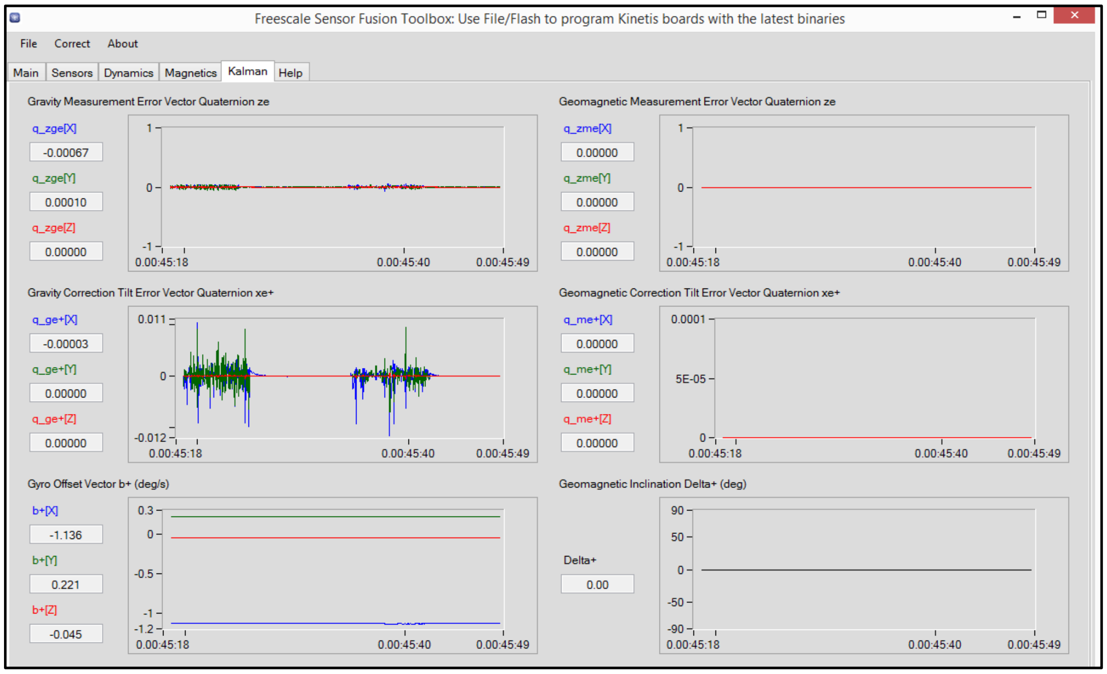
Angular Velocity Sensor Frame (deg/s)

Omega[X]	-0.10
Omega[Y]	0.05
Omega[Z]	-0.10

Linear Acceleration Global Frame (g)

Acc[X]	0.000
Acc[Y]	0.000
Acc[Z]	-0.028

Figure 81. Sensor Fusion Toolbox Dynamics section (for looking at raw sensor values)


Freescale Sensor Fusion Toolbox: Use File/Flash to program Kinetis boards with the latest binaries
File
Correct
About
Main
Sensors
Dynamics
Magnetics
Kalman
Help
0.00:45:18
0.00:45:40
0.00:45:49

Gravity Measurement Error Vector Quaternion ze

q_zge[X] -0.00067

q_zge[Y] 0.00010

q_zge[Z] 0.00000

Geomagnetic Measurement Error Vector Quaternion ze

q_zme[X] 0.00000

q_zme[Y] 0.00000

q_zme[Z] 0.00000

Gravity Correction Tilt Error Vector Quaternion xe+

q_ge+[X] -0.00003

q_ge+[Y] 0.00000

q_ge+[Z] 0.00000

Geomagnetic Correction Tilt Error Vector Quaternion xe+

q_me+[X] 0.00000

q_me+[Y] 0.00000

q_me+[Z] 0.00000

Gyro Offset Vector b+ (deg/s)

b+[X] -1.136

b+[Y] 0.221

b+[Z] -0.045

Geomagnetic Inclination Delta+ (deg)

Delta+ 0.00

Figure 82. Sensor Fusion Toolbox Kalman filtering tab for seeing filter stages and outputs

The last testing this board enabled was checking if vibrations from the vibration motors would disrupt the measurements and sensor readings. The motors were affixed directly to the sensor board using secure adhesives and then motors were run at full speed while the sensor fusion toolbox was open. While the raw sensor data showed the vibrations due to the motors, the Kalman filtering stages removed the effect caused by the vibrations. As such the use of an accelerometer and gyro with Kalman filtering is immune to motor noise that would be experienced in this application.

Appendix 9 – Reviewer Evaluation Summaries

This section contains a summary of the reviewer feedback from advisors during the two review sessions. There were two major design reviews called the Preliminary Design Review (PDR) and the Critical Design Review (CDR) which are included in the university curriculum for capstone projects. These sessions are designed to give advisors and general university students an opportunity to critical and evaluate a design by listening to a 20 minute presentation and then providing questions and feedback. Typically the PDR occurs early on in the beginning of the design phase and seeks to refine the project proposal. The CDR occurs later in the design phase to verify the design for the project is feasible and applicable to the original design.

Preliminary Design Review Presentation (9/29/15)

The following listing shows the advisor feedback provided during the PDR.

Project Goal

- Potential applications to consider
    - Physical therapy + reporting progress to therapist
    - Telemedicine
    - Sports training
    - Physical activity training within workplace
- Objectives of project
    - Refer to motion database as library
- Suit configuration: jacket
    - 'Tendons' for actuation (piezoelectric material for force feedback, could get additional funding)
- Hardware - hand sensor
    - Flex sensors on fingers won't be as precise, won't be able to get definite determination of finger location
- Add functional diagrams of system to project
    - Look at motor vibration patterns for indicating error to the user
    - Specify the interface between subsystems
- Potential risks
    - Consider weight of bands in addition to overall weight
    - Communications
        - Test Bluetooth
            - Limited channels, but could have multiple modules
            - Lower power
        - Test Wi-Fi
            - Interference from other 2.4GHz sources
            - Higher power than Bluetooth
            - Campus IT services might have issues with personal Wi-Fi router use
        - Analysis on bandwidth usage
- Budget
    - Perform a 'reasonable first pass at a budget'
    - Reevaluate suit material costs
    - Check if MSP 432 can work for project
    - Use h-bridge rather than transistors
    - Add PCB costs

- Project timeline
    - Put Project Presentation Day (for university) on timeline
    - Look into Atom board usage requirements for Intel Cornell Cup
    - Put Intel Cornell Cup on project timeline

Critical Design Review Presentation (11/20/15)

This was the information gathered from advisors after the CDR.

General:
- Make sure CDR slides reflect talking points
- Incorporate pictures to back up talking points
- Make sure to use professional language
- Speak slowly and clearly
- Add general use case to provide outline/context to presentation

Problem and goal presentation slide:
- Enumerate benefits of project more clearly

Band Design slide:
- Talk about band and then electronics (high level and then low level)
- Include amp-hours of battery along with relevant electrical characteristics

Software slides:
- Make clear that this is a functional diagram rather than class hierarchy
- Add class hierarchy diagram as well
- Try saving/exporting data to XML

Current Testing:
- Explain acronyms (IMU)
- Speak in terms of functionality and then relate to technical (don't say technical and then relate back to functionality, ex. Quaternions)
- MATLAB diagram of initial visualization not clear
- Explain more specifically what we mean by sensor fusion

Future testing slide:
- Specify that testing against requirements/what requirements were
- Specify testing with users

Risks and challenges:
- Don't use precision and accuracy interchangeably
- More clearly state precision/cost trade off
- Characterize error (order of magnitude)
- Specify difference between cost of parts and potential market/development costs

Information to add to future presentations:
- Future implications
- Suggest additional research questions

Intel Cornell Cup Proposal:
- Tell as a story (incorporate problem, background, design and testing, potential benefits & future applications)
- Emphasize systems engineering

Appendix 10 – Acknowledgements

We would like to thank three supporters to our project. The first is Advanced Circuits who did an excellent job creating and providing us with high-quality printed circuit boards for out circuits. Without the special student sponsorship deal they gave us on the PCBs, we would not have been able to afford making PCBs for our project.

Another supporter for our project is Pololu Corporation which gave us a huge discount on our order of vibration motors. This helped us purchase enough motors for all seven bands.

Another supporter was Tin Can Tools (the manufacturer of the SilverJaw Lure board that holds the SSD and Wi-Fi chip for the Intel Atom). When our Lure board failed, Tin Can Tools sent us a new board completely free, which we thank them for.

We would also like to thank WPI as their project-based curriculum allowed for the conception and completion of this project as an MQP. WPI also provided the budget for purchasing electrical components and materials.

Appendix 11 – Authorship

Section	**Primary Author(s)**
Abstract	Team
Challenge Definition	Team
Background Research	Team
Concept of Operations	Swartz
System Design	Team
1. Suit Overview	Barnard
2. Software Overview	Adkins, Beardsley, Swartz
3. Hardware Overview	Barnard, Beardsley, Frick
Technical Documentation	Team
1. Band Design	Barnard, Frick
2. Sensors and Sensor Fusion	Beardsley
3. Haptic Motor Control	Beardsley
4. Communication	Adkins, Frick
5. Software Design	Adkins, Beardsley, Swartz
Product Performance Evaluation	Swartz
Project Execution Performance Evaluation	Adkins, Frick
Recommendations and Next Steps – Conclusions and Future Research	Adkins
Nomenclature Glossary	Team
Timeline Appendix	Team
Software Core Classes and Functionality Appendix	Adkins, Beardsley, Swartz, Frick
Full Bill of Materials for Project Appendix	Frick
Assembly & Construction Appendix	Team
Test Plans Appendix	Team
Initial Design Steps – Trade Study Appendix	Frick
Gap Analysis (Desired Features Analysis) Appendix	Team
Precursor Testing before System Design Stages Appendix	Team
Reviewer Evaluation Summaries Appendix	Team
Acknowledgements	Team
Authorship	Team
Bibliography	Team

Bibliography

[1] Wehner, M. e. a. (2013). A Lightweight Soft Exosuit for Gait Assistance (Web ed., pp. n. pag). International Conference on Robotics and Automation: IEEE.

[2] Asbeck, A. T. e. a. (2013). Biologically-Inspired Soft Exosuit (Web ed.). 13th International Conference on Rehabilitation Robotics (ICORR): IEEE.

[3] Ho, N. S. K. e. a. (2011). An EMG-Driven Exoskeleton Hand Robotic Training Device on Chronic Stroke Subjects: Task Training System for Stroke Rehabilitation (Web ed., pp. n. pag). International Conference on Rehabilitation Robotics: IEEE.

[4] Chan, C. L., Suresh Gobee, and D. Vickneswari. (2014). Finger Grip Rehabilitation Using Exoskeleton With Grip Force Feedback (Web ed., pp. 520-523). IFMBE Proceedings The 15th International Conference on Biomedical Engineering.

[5] Dexta Robotics. Dexta Robotics. Retrieved from Dexmo – Overview. China. <www.dextarobotics.com/products/dexmo>

[6] CyberGlove Systems LLC. CyberGrasp. – Overview of CyberGrasp.. San Jose, CA. <www.cyberglovesystems.com/cybergrasp/>

[7] NeuroDigital RSS: Fun & Serious VR.Technologies. Homepage. Almeria, Spain. <www.neurodigital.es>

[8] Tesla Studios. Tesla Studios | VR Future.Homepage. United Kingdom. <www.teslastudios.co.uk/index.html>

[9] Immerz, Inc. “KOR-FX 4DFX Haptic Gaming Vest.” KOR-FX 4DFX Haptic Gaming Vest. Web. 2 Sep. 2015. <http://Cambridge, MA <korfx.com/>>

[10] YEI Technology. “About Prio-VR.” PrioVR. Web. 2 Sep. 2015. <http://Portsmouth, Ohio <www.priovr.com/>

[11] Cela, A., Yebes, J. J., Arroyo, R., Bergasa, L. M., Rafael, B., & López, E. (2013). Complete Low-Cost Implementation of a Teleoperated Control System for a Humanoid Robot (Vol. Sensors, pp. 1385-1401). National Polytechnic, EC170135 Quito, Ecuador: Department of Automation and Industrial Control.

[12] Menache, Alberto, I. T. Pro Collection Books24x, and Online Safari Books. Understanding Motion Capture for Computer Animation. Vol. 2nd;2;. Burlington, MA: Morgan Kaufmann, 2011. Print.

[13] Vlasic, D., Adelsberger, R., Vannucci, G., Barnwell, J., Gross, M., Matusik, W., & Popović, J. (2007). Practical motion capture in everyday surroundings.

[14] AutoDesk., Inc. “MotionBuilder – 3D Character Animation Software | MotionBuilder. Retrieved from”. San Rafael, CA. <http://www.autodesk.com/products/motionbuilder/overview>

[15] Flam, David L., Ramos, Thatyene L. A. S., Queriroz, Daniel P., & Araujo, Arnaldo A. (2009). Openmocap: An Open Source Software for Optical Motion Capture. 2009. Universidade Federal de Minas Gerais. Print.

[16] Xsens North America, Inc. (2015). Xsens MVN brochure: The ultimate animator's tool. Culver City, CA. Retrieved from www.xsens.com/wp-content/uploads/2013/12/Xsens-DM-MVN-2.0-brochure_D05112014.pdf

[17] Daniel Roetenberg, H. L., Per Slycke. (2013). Xsens MVN: Full 6DOF Human Motion Tracking Using Miniature Inertial Sensors. Retrieved from https://www.xsens.com/wp-content/uploads/2013/12/MVN_white_paper1.pdf

[18] Carnegie Mellon University. CMU Graphics Lab Motion Capture Database. Pittsburg, PA. Retrieved from http://mocap.cs.cmu.edu

[19] Mocap Club -. Motion Capture Library. Retrieved from http://mocapclub.com

[20] Hall, David Lee, and Sonya A. H. McMullen. (2004). Mathematical Techniques in Multisensor Data Fusion. Artech House, 2004. Print.

[21] Oxford Dictionary. (2015). Retrieved from http://www.oxforddictionaries.com/us/definition/american_english/force-feedback

[22] Introduction to Haptic Feedback. (2015). Precision Microdrives.

[23] Understanding Linear Resonance Actuator Characteristics. Retrieved from http://www.precisionmicrodrives.com/application-notes-technical-guides/application-bulletins/ab-020-understanding-linear-resonant-actuator-characteristics

[24] Linear Resonant Actuator (LRA) Vibration Motors. Haptic Feedback. (n.d.). Retrieved October 12, 2015.

[25] Coin Vibration Motor: Pico Vibe™ Range. (n.d.). Retrieved October 12, 2015.

[26] Texas Instruments. (2005). Accelerometers: US FIRST.

[27] Andrejašic, M. (2008). MEMS Accelerometers. University of Ljubljana Department of physics: University of Ljubljana.

[28] a1ronzo. Gyroscope: SparkFun.

[29] KVH Industries. (2014a). An Update on KVH Fiber Optic Gyros and Their Benefits Relative to Other Gyro Technologies. 50 Enterprise Center, Middletown, RI, 02842: KVH Industries.

[30] KVH Industries. (2014b). Guide to Comparing Gyro and IMU Technologies – Micro-Electro-Mechanical Systems and Fiber Optic Gyros. 50 Enterprise Center, Middletown, RI, 02842: KVH Industries.

[31] Looney, M. (2010). A SIMPLE CALIBRATION FOR MEMS GYROSCOPES. EDN Europe: Analog Devices.

[32] IEEE Sensors Journal, Vol. 1, No. 4, 2001, pp 332-339 (Barbour & Schmidt)

[33] Skog, I. a. P. H. (2006). Calibration of a MEMS inertial measurement
unit. Metrology for a Sustainable Development, Rio de Janeiro, Brazil: ResearchGate.

[34] Xsens North America, Inc. (2015). MTi 1-series. Xsens Culver City, CA.

[35] Lee, J.-S., Su, Y.-W., & Shen, C.-C. (2007). A Comparative Study of Wireless Protocols: Bluetooth, UWB, Zigbee, and Wi-Fi. The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON), Taipei, Taiwan: IEEE.

[36] Thierer, A. (2013). Privacy and Security Implications of the Internet of Things. Mercatus Center 3434 Washington Blvd, 4th Floor, Arlington, VA: George Mason University.

[37] MITRE. (2013, 18 December). Analyzing and Defining Requirements. Available: http://www.mi-tre.org/publications/systems-engineering-guide/se-lifecycle-building-blocks/requirements-engineer-ing/analyzing-and-defining-requirements

[38] Miller, R. B. (1968). Response time in man-computer conversational transactions. Proc. AFIPS Fall Joint Computer Conference Vol. 33, 267-277.

[39] Stoffregen, P. (2015). Teensy 3.2 & 3.1 - New Features. PJRC. Retrieved December 21, 2015, from https://www.pjrc.com/teensy/teensy31.html

[40] EasyVR 3. (2015). ROBOTECH. Retrieved December 21, 2015, from http://www.veear.eu/prod-ucts/easyvr3/

[41] Rivera. (2007). Boost C++ Libraries. Available: http://www.boost.org/

[42] Khronos Group. (2012). OpenGL Overview. Available: https://www.opengl.org/about/

[43] The Qt Company. (2016). Qt - Developers. Available: http://www.qt.io/developers/

[44] The Qt Company. (2016). Signals and Slots. Available: http://doc.qt.io/qt-4.8/signalsandslots.html

[45] Ford, Matthew (2016, February 3). Pfod^{TM} Parser Libraries V2.20. pfod^{TM} Parser Library for Arduino – Forward Computing Control Pty Ltd. Retrieved April 18, 2016, from http://www.for-ward.com.au/pfod/pfodParserLibraries/

[46] Rowberg, Jeff. (n.d.). I2Cdevlib. Retrieved April 18, 2016, from http://www.i2cdevlib.com/

[47] Webmaster. (2015, April 3). Introducing EasyVR 3 & EasyVR Shield 3. Retrieved April 18, 2016, from http://www.veear.eu/introducing-easyvr-3-easyvr-shield-3/

[48] Majumdar, Swagatam. (2013, May 26). Low Battery Indicator Circuit Using Two Transistors Only. Homemade Circuits. Retrieved April 10, 2016, from http://www.homemade-circuits.com/2013/05/low-battery-indicator-circuit-using-two.html

[49] Range Slider in Qt (Two Handles in a QSlider). Retrieved November 28, 2015, from http://stackover-flow.com/questions/17361885/how-to-get-two-handles-in-a-qslider

[50] F28M35H22C. (n.d.). Retrieved October 12, 2015, from http://www.ti.com/product/f28m35h22c

[51] MSP432P401R. (n.d.). Retrieved October 12, 2015, from http://www.ti.com/product/msp432p401r

[52] STM32F411CE. (n.d.). Retrieved October 12, 2015, from http://www.st.com/web/cata-
log/mmc/FM141/SC1169/SS1577/LN1877/PF260148

[53] STM32F205RG. (n.d.). Retrieved October 12, 2015, from http://www.st.com/web/cata-
log/mmc/FM141/SC1169/SS1575/LN1433/PF245094

[54] AT32UC3C264C. (n.d.). Retrieved October 12, 2015, from http://www.atmel.com/de-
vices/AT32UC3C264C.aspx

[55] K20P64M72SF1RM. (n.d.). Retrieved October 12, 2015, from
https://www.pjrc.com/teensy/K20P64M72SF1RM.pdf

[56] F28M35H22C. (n.d.). Retrieved October 12, 2015, from http://www.ti.com/product/f28m35h22c

[57] MSP432P401R. (n.d.). Retrieved October 12, 2015, from http://www.ti.com/product/msp432p401r

[58] STM32F411CE. (n.d.). Retrieved October 12, 2015, from http://www.st.com/web/cata-
log/mmc/FM141/SC1169/SS1577/LN1877/PF260148

[59] STM32F205RG. (n.d.). Retrieved October 12, 2015, from http://www.st.com/web/cata-
log/mmc/FM141/SC1169/SS1575/LN1433/PF245094

[60] AT32UC3C264C. (n.d.). Retrieved October 12, 2015, from http://www.atmel.com/de-
vices/AT32UC3C264C.aspx

[61] AT32UC3L064. (n.d.). Retrieved October 12, 2015, from http://www.atmel.com/de-
vices/at32uc3l064.aspx

[62] K20P64M72SF1. (n.d.). Retrieved October 12, 2015, from
https://www.pjrc.com/teensy/K20P64M72SF1.pdf

[63] XMS432P401RIPZR. (n.d.). Retrieved October 12, 2015, from
https://store.ti.com/XMS432P401RIPZR.aspx

[64] STM32F411CEU6. (n.d.). Retrieved October 12, 2015, from http://avnetex-
press.avnet.com/store/em/EMController?action=products&cata-
logId=500201&storeId=500201&N=0&langId=-
1&slnk=e&term=STM32F411CEU6&mfr=STM&hrf=http://www.st.com/web/cata-
log/mmc/FM141/SC1169/SS1577/LN1877/PF260148

[65] STM32F205RGT6. (n.d.). Retrieved October 12, 2015, from http://www.digikey.com/product-
search/en?WT.z_cid=sp_497_0928_buynow&Enterprise=44&lang=en&Ven-
dor=497&mpart=STM32F205RGT6

[66] H52C1 Concerto Experimenter Kit. (n.d.). Retrieved October 12, 2015, from
http://www.ti.com/tool/tmdsdockh52c1

[67] MSP432P401R LaunchPad. (n.d.). Retrieved October 12, 2015, from http://www.ti.com/tool/msp-exp432p401r

[68] “EMWE - 3165 - A Development Board.” Seeed. Retrieved from http://www.seeedstudio.com/depot/EMWE-3165-A-Development-Board-p-2489.html?ref=newInBazaar%E2%80%9D

[69] “EMW3165 - Cortex-M4 based Wi-Fi SoC Module.” Seeed. Retrieved from http://www.seeedstudio.com/depot/EMW3165-p-2488.html?cPath=19_20

[70] Photon Datasheet. (n.d.). Retrieved October 12, 2015, from https://docs.particle.io/datasheets/photon-datasheet/

[71] STMicroelectronics STM3220G-EVAL. (n.d.). Retrieved October 12, 2015, from http://www.digikey.com/product-detail/en/STM3220G-EVAL/497-11202-ND/2640848

[72] UC3C-EK. (n.d.). Retrieved October 12, 2015, from http://www.atmel.com/tools/uc3c-ek.aspx

[73] Teensy USB Development Board. PJRC. (n.d.). Retrieved October 12, 2015, from https://www.pjrc.com/store/teensy32.html

[74] SparkFun Triple Axis Accelerometer Breakout - ADXL345. (n.d.). Retrieved October 12, 2015, from https://www.sparkfun.com/products/9836

[75] SparkFun Triple Axis Accelerometer Breakout - LIS331. (n.d.). Retrieved October 12, 2015, from https://www.sparkfun.com/products/10345

[76] SparkFun Triple Axis Accelerometer Breakout - ADXL362. (n.d.). Retrieved October 12, 2015, from https://www.sparkfun.com/products/11446

[77] InvenSense MPU-6050 Six-Axis (Gyro + Accelerometer) MEMS MotionTracking^{TM} Devices. (2015). Retrieved October 12, 2015, from http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/

[78] Kootek GY-521 MPU6050. (n.d.) Retrieved October 12, 2015, from http://www.amazon.com/Kootek-MPU-6050-MPU6050-sensors-Accelerometer/dp/B008BOPN40/ref=cm_cr_arp_d_product_top?ie=UTF8

[79] FXOS8700CQR1. (n.d.). Retrieved October 12, 2015, from http://www.digikey.com/product-detail/en/FXOS8700CQR1/FXOS8700CQR1CT-ND/4004929

[80] SparkFun Gyro Breakout - LPY503AL (Dual 30°/s). (n.d.). Retrieved October 12, 2015.

[81] SparkFun Tri-Axis Gyro Breakout - L3G4200D. (n.d.). Retrieved October 12, 2015.

[82] SparkFun Triple-Axis Digital-Output Gyro Breakout - ITG-3200. (n.d.). Retrieved October 12, 2015.

[83] MPU-3050. (n.d.). Retrieved October 12, 2015.

[84] MPU-3050 Triple-Axis Gyroscope with Embedded Digital Motion Processor. (n.d.). Retrieved October 12, 2015.

[85] MPU-6500. (n.d.). Retrieved October 12, 2015.

[86] FXAS21002CQR1. (n.d.). Retrieved October 12, 2015.

[87] FRDM-STBC-AGM01. (n.d.). Retrieved October 12, 2015, from http://www.digikey.com/product-detail/en/FRDM-STBC-AGM01/FRDM-STBC-AGM01-ND/5130169

[88] Ho, N. S. K., Tong, K. Y., Hu, X. L., Fung, K. L., Wei, X. J., Rong, W., & Susanto, E. A. (2011). An EMG-driven Exoskeleton Hand Robotic Training Device on Chronic Stroke Subjects. Hong Kong SAR, China.

[89] Motion., M. Datagloves by 5DT.

[90] Ju, Z., Yang, C., Li, Z., Cheng, L., & Ma, H. Teleoperation of Humanoid Baxter Robot Using Haptic Feedback.

[91] macgyver603. How to Build a Robotic Hand with Haptic Feedback. Instructables.

[92] Cela, A., Yebes, J. J., Arroyo, R., Bergasa, L. M., Rafael, B., & López, E. (2013). Complete Low-Cost Implementation of a Tele operated Control System for a Humanoid Robot (Vol. Sensors, pp. 1385-1401). National Polytechnic, EC170135 Quito, Ecuador: Department of Automation and Industrial Control.

[93] Grabe, V., Riedel, M., Bulthoff, H. H., Giordano, P. R., & Franchi, A. (2013). The TeleKyb framework for a modular and extendible ROS-based quadrotor control.

[94] Koenemann, J., Burget, F., & Bennewitz, M. (2014). Real-time Imitation of Human Whole-Body Motions by Humanoids.

[95] Organic Motion: Markerless Mocap for Animation.

[96] Hexoskin. (2015).

[97] Salisbury, D. (2012). Advanced Exoskeleton Promises More Independence for People with Paraplegia.