# Powered, Programmable Elbow Orthosis

Team 10

Final (Progress) Report

*March 6, 2016*

*ECEN 404 – Capstone (Senior) Design Project*

*Texas A&M University, College Station, TX*

David Cuevas
Nathan Glaser
Joe Loredo
Rafael Salas

# Abstract

Our project explores a powered, programmable elbow orthosis which stabilizes, limits, and assists the movements of a user's elbow. Specifically, this project restores upper arm functionality to users suffering from a range of injuries and disorders, namely ones which weaken muscles and muscular activity. The orthosis intercepts weakened muscle signals, discerns muscular intention, and subsequently drives a motor to help articulate elbow movement. A user interface allows for dynamic manipulation of desired settings, and a battery supplies portable and long-lasting system power.

The project consists of one mechanical superstructure with five embedded electrical subsystems. The custom mechanical structure braces the afflicted joint and mounts the majority of the subsystems. The body-to-sensor interface uses a combination of angle, pressure, and electromyography (EMG) sensors to encode the current and intended elbow movements. Various analog and digital filters help distinguish the relevant muscle signals from noise. Using a microcontroller, the data processing subsystem interprets the filtered signals, manipulates the information per user specifications, and translates the result into a pulse width modulated (PWM) motor signal. The motor-to-body interface takes this signal, actuates the motor, and articulates the elbow. Feedback sensors and a properly tuned transfer function ensure smooth motor movement and corrects any errors sensed. The power subsystem provides portable and sustained power to each component of the orthosis. The user interface allows the user to easily calibrate and set safety limits for the orthosis.

# Table of Contents

# 1 Project Overview

An orthosis is a mechanical aid that controls biomechanical alignment. These devices range from rigid braces (which restrict movement to one position) to more flexible goniometers (which resist and restrict movement between two mechanical limits). Typically following a structural injury, a physician supports the patient's affected limb with an orthosis and, over an extended period of time, adjusts the orthosis until the limb regains function [4]-[5]. The programmable elbow orthosis addresses this procedure of adjustments, offering the option for a programmable rehabilitation regiment. In this scope, the orthosis resists harmful movements with electromechanical motors instead of a goniometer's mechanical springs and stops. The orthosis structure itself serves as a brace.

The powered, programmable elbow orthosis also addresses more serious motor neuron disorders, namely disorders which weaken muscles and muscular activity [6]. To combat these reported muscle weakness symptoms, the orthosis intercepts muscle signals, discerns muscular intention, and subsequently drives a motor to help articulate intended elbow movements. Specifically, the orthosis helps the weakened user with movements normally considered mundane, such as stably lifting a glass of water. Furthermore, the orthosis provides a cost-effective solution to these elbow-related disabilities, costing under $500.

Several solutions already address our concerns for supporting and rehabilitating the elbow. For instance, consumer-grade braces and goniometers offer structural support and a purely mechanical path to rehabilitation [4]. While they are inexpensive, these devices only restrict harmful movement without assisting intended movement. More complex solutions help users recover from neurological degeneration, as associated with a cerebrovascular accident (stroke) or a brachial plexus injury (stretched, compressed, or ripped nerves) [5]-[4]. These orthoses either amplify minute physical movements or act upon intercepted electromyography signals.

People lose functionality of their elbows for numerous reasons. Harmful physical impacts and stresses can degrade the muscles themselves, overworking or tearing the muscle fibers. Physical therapy corrects these injuries by carefully resting and exercising the appropriate muscles. During this procedure, the programmable limits of the orthosis prevent the user from re-aggravating an injured muscle. Motor neuron disorders, body chemistry imbalances, and severe physical impacts can weaken or sever the nerve communication between the limb and the brain. By intercepting, filtering, and amplifying the communicating motor neurons, the orthosis can reestablish and rehabilitate the lost connection between physical movement and mental intention.

The elbow orthosis rests at the interface between two worlds—it connects the biological world of neurons and fat-covered axons to the electronic world of transistors and metallic wires. In fact, the layout of the orthosis circuitry is remarkably similar to the neural structure of the human body. For instance, the brain and its network of transistor-like neurons serves as the "central processing unit" (CPU) for our body, manipulating information and regulating the flow of neurological signals. Furthermore, axons serve as the "transmission lines" of the nervous system. Biologically, to prompt a kinesthetic movement, our brains pulse action potentials along these axons to our muscle fibers which then constrict or relax in response to the stimulus. Electronically, this process mirrors how a CPU transmits a Pulse Width Modulated (PWM) signal along wires to a motor driver which then controls motor movements.

## 1.1 Proposed System

The powered, programmable elbow orthosis consists of a unifying mechanical structure with five embedded electrical subsystems. The mechanical structure supports the afflicted joint and mounts the majority of the subsystems. As shown by the system block diagram in Figure 1, each of the five electrical subsystems connects with others. Specifically, the power subsystem consists of a rechargeable DC battery which distributes power to each of the subsystems. Electronic circuitry regulates this power flow to the processor components (producing 3.3 V for the microcontrollers, sensors, and displays) and electromechanical hardware (providing 12 V for the motor). The body-to-sensor interface captures electric muscle activity through an array of conductive electrodes. It then reduces signal noise through an analog bandpass filter with cutoff frequencies at 10 Hz and 300 Hz, the accepted range for electric muscle signals [1]-[2]-[3]. A dedicated electromyography (EMG) analog-to-digital (ADC) converter, namely the ADS1299, converts these measured analog voltages into a digital value. A digital notch filter then excludes the 60 Hz power line interference and its harmonics. The central processor uses these filtered voltage readings to distinguish muscle activity from inactivity, combines this information with user specifications, decides how to act, and finally transmits a pulse width modulated (PWM) signal to the motor drive. The motor-to-body interface receives the incoming PWM signal and passes it through a DRV8848 motor driver which actuates two DC brushless motor. The motor subsequently articulates the elbow with roughly 1 N • m, the torque required to lift a glass of water. A rotary encoder provides feedback for the motor, enabling accurate movements and smooth transitions. Finally, the user interface allows for dynamic calibration of the sensors and motor, physician customization, and diagnostic data display.

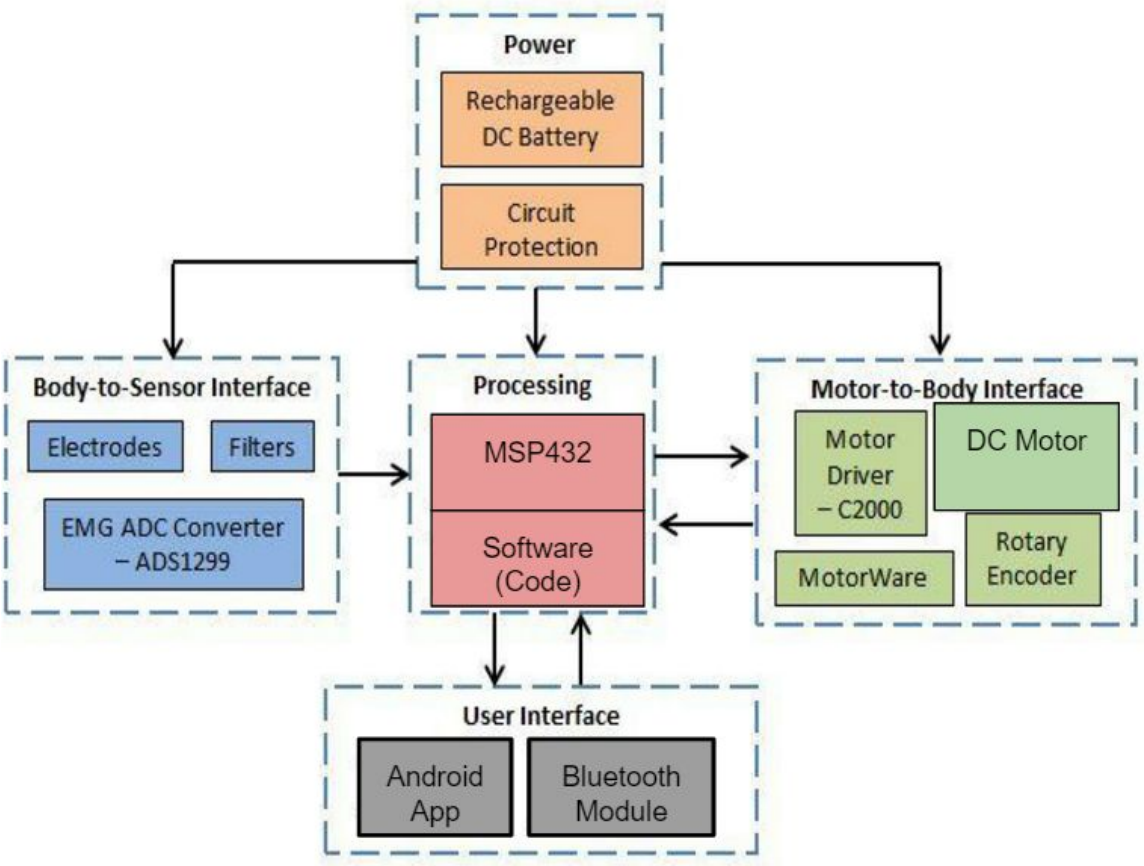



Figure 1.1: System Block Diagram

As outlined in Figure 2, the orthosis structure itself provides a mount for several of the sub-components, including sensors, a motor, a user interface, and several computational modules. Electrodes line the orthosis interior, contacting the user's biceps and triceps. Their placement on the upper arm provides a more direct measurement of the muscles responsible for elbow bending. The motor also rests on the orthosis. However, in an effort to reduce its profile, the motor's axis lies parallel to the upper arm, misaligned from that of the orthosis by 90 degrees. A series of beveled gears allow the motor to drive the orthosis with the appropriate torque and direction. To discern angular position, a rotary encoder aligns with the axis of rotation. A user interface attaches to the orthosis, complete with several buttons and a digital readout display. Finally, a printed circuit board merges each of the integrated circuit chips onto a single mountable board. The only component not firmly attached to the orthosis is the battery pack; because of weight considerations, it is instead mounted at the waist.

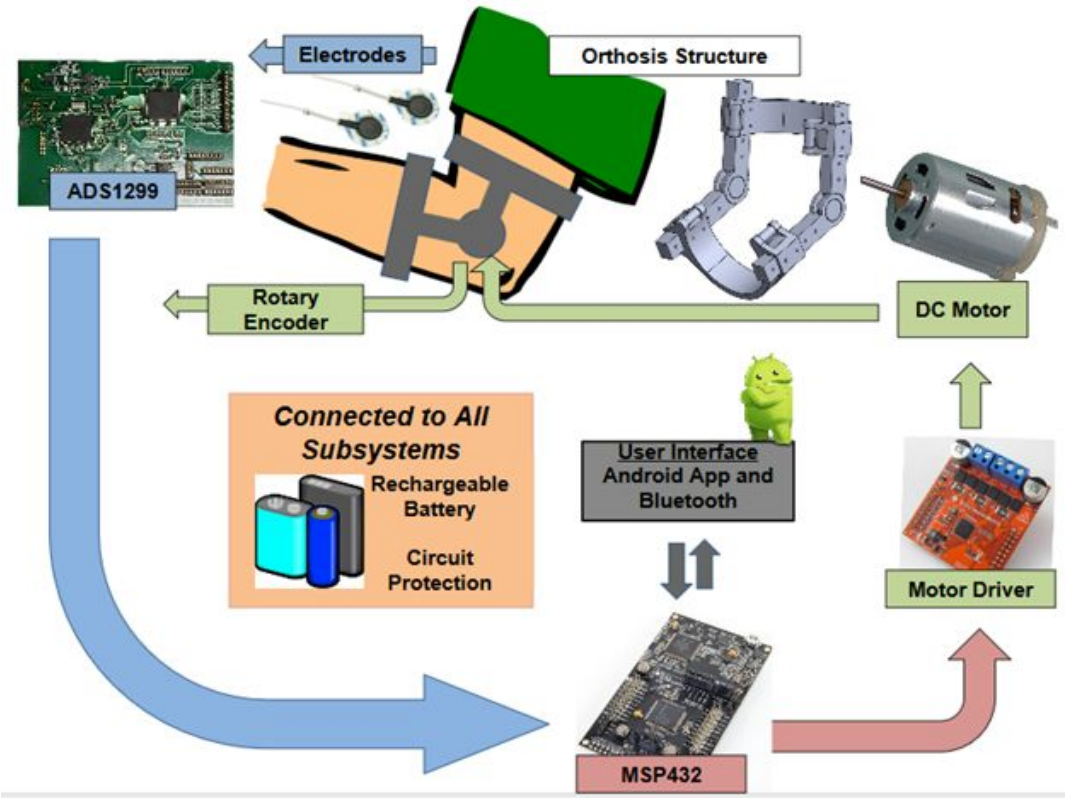



Figure 1.2: Physical Sketch of System

## 1.2 Project Deliverables

The final implementation of the Elbow Orthosis stabilizes, limits, and assists the movements of a user's elbow. In its final form, this project restores upper arm functionality to debilitated users. In broad terms, the orthosis intercepts weakened muscle signals, discerns muscular intention, and subsequently drives a motor to help articulate elbow movement. A user interface allows for dynamic manipulation of desired settings, and a battery supplies portable and long-lasting system power. Additionally, the elbow orthosis integrates with a modular wrist orthosis system. The two systems share a combined power system and similar hardware components.

The Body-to-Sensor Interface features a set of wearable sensors which transmit elbow angle and muscle activity. This interface must deliver clear, filtered information regarding whither and whether the user intends to move. The Data Processing Subsystem features a controller specially programmed to receive sensor signals, convert and manipulate data, and transmit motor signals. This subsystem must log calibration data. It must then normalize the filtered EMG signals to this data to eventually provide logic for the motor. The Motor-to-Body Interface features a motor which articulates an elbow based on the controller signal. This interface must safely provide 1 N • m of torque and must limit the elbow to an arbitrary angle within 180 degrees. It must also establish a feedback loop with the main microcontroller to

ensure smooth movement and critical damping. The Power Subsystem features a rechargeable battery which provides portable power to the subsystems. This subsystem must supply 12 V to power the motor and must subdivide this value to power the other circuit peripherals. The User Interface features user-friendly buttons and readouts for user calibration and dynamic setting. This system must communicate settings to and from the main microcontroller. The overall orthosis structure integrates and mounts all components with considerations to form and stability. All components must communicate through their respective SPI, UART, or GPIO pins.

### 1.2.1 Project Specifications and Interfaces

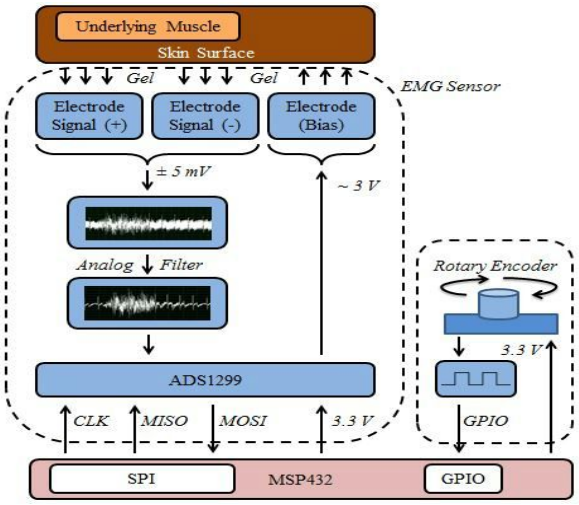



Figure 1.3: Body-to-Sensor Interface Block Diagram

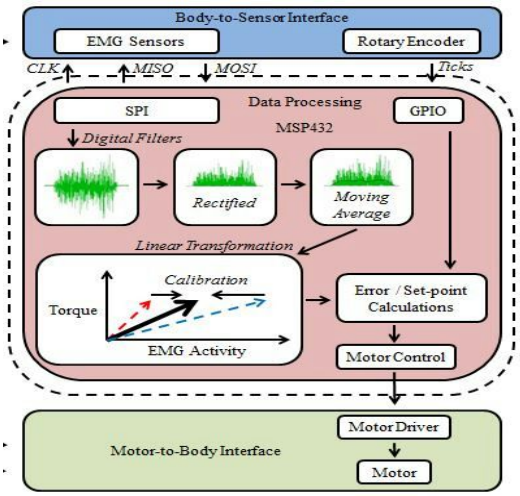



Figure 1.4: Data Processing Block Diagram

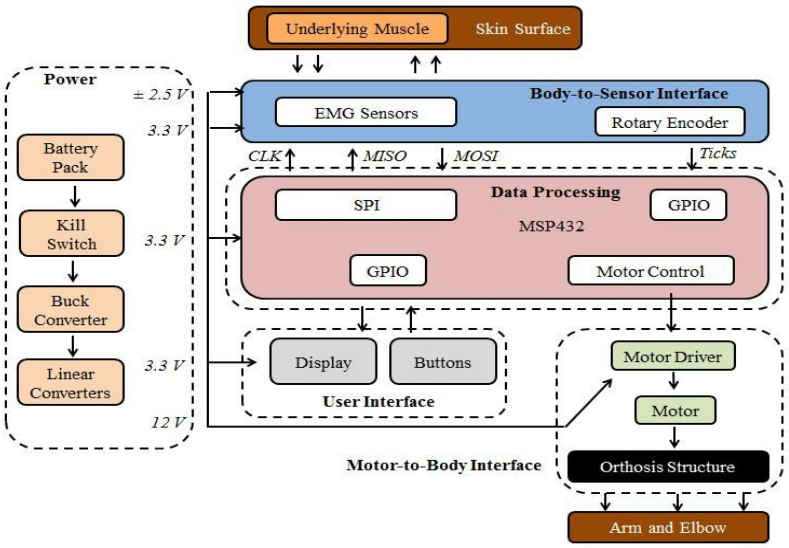


Figure 1.5: Motor-to-Body Interface, Power, and User Interface Block Diagrams

*Body-to-Sensor Interface*

The impetus for elbow movement begins at the surface electrodes of the Body-to-Sensor Interface. These gel patches connect with the ADS1299, a dedicated analog-to-digital front-end for biopotential measurements. The ADS1299 feeds to the user a bias voltage through one electrode and collects from the user 8 differential biopotential signals through pairs of electrodes. The ADS1299 communicates with the MSP432 central processor through a Serial Peripheral Interface (SPI). Through this serial connection, the CPU transmits to the ADS1299 the desired firmware bits and default settings. Additionally, the ADS1299 transmits to the CPU the converted analog biopotential readings as a series of 24-bit digital values. Beyond the EMG sensor front-end, the rotary encoder pulses angular readings to one of the central processor's General Purpose Input Output (GPIO) pins. The force sensitive resistors change the voltage across another GPIO pin, which the processor may interpret through an analog to digital conversion.

*Data Processing*

The Data Processing Subsystem includes the MSP432 microcontroller, its data processing software libraries, and its communication interface with the other modules. The Data Processing system receives input signals from three sources: the ADS1299 EMG front-end, the rotary encoder, and the force sensitive resistors. The ADS1299 signal is a bit stream that communicates with the MSP432 via a serial peripheral interface (SPI). The MSP432 then digitally filters the signal to reduce undesired noise. The rotary encoder and the force sensitive resistor signals connect to the MSP432 GPIO pins, which are configured for analog to digital conversion. All signals are used to create control variables for motor logic. The new variables are inputted into the motor driver through another set of GPIO pins. The Data Processing Subsystem provides a calibration function to match a maximum voluntary contraction with motor torque. The User Interface wirelessly communicates with the MSP432 through an Radio Frequency (RF) Booster Pack.

*Motor-to-Body Interface*

The Motor-to-Body Interface consists of three parts: the MSP432 microcontroller, the DRV8848 Booster Pack, and two .5 Amp High Torque DC Motors. The microcontroller serves as the master controller for the motor-to-body interface. By resolving external signals and user-defined parameters (such as voltage, current, and torque ratings), the microcontroller produces logic to drive the motor. The microcontroller relays this logic to the DRV8848 Motor Driver, and this peripheral converts the logic into wave pulses. These pulses induce magnetic fields in different coils within the motor, subsequently driving the rotation. The stepper motor's high voltage and current specifications prompt the need for a motor driver.

*Power Distribution*

The power system is meant to take power from a rechargeable voltage source and convert it to the appropriate voltages for the other subsystems. To clarify, the system will take in a large voltage source, around 12 V, and then regulate the voltage for the microcontroller and ADS1299, 3.3 V. The body-to-sensor interface also requires a low noise voltage regulation of ±2.5 V. Fuses will also be implemented to protect highly sensitive components, such as the ADS1299. A kill switch, when activated,

disconnects the circuit loop and stops all system function. If any subsystem experiences a dangerous fault, the user has the option to immediately halt all functions.

*User Interface*

The user interface consists of an Android app with options for calibration, limits sets, and bluetooth connectivity. The phone connects to the MSP432 microcontroller through the use of a bluetooth module. The limit set works with the rotary encoder of the motor-to-body subsystem and sets what angles the orthosis will not move past. This keeps the user's arm from moving into any harmful or dangerous positions. The calibration setting allows the system to read a user's bicep emg signal, while flexing, allowing for the system to more accurately respond to different users.

Table 1.1
Specification Compliance Matrix

| Specification | Input | Min | Nominal | Max |
|---|---|---|---|---|
| DROK Buck Converter | 12 V | | 5 V | |
| LM1117IMP-3.3/NOPB | 5 V | 3.267 V | 3.3 V | 3.333 V |
| LP5907MFX-2.5/NOPB | 3.3 V | | 2.5 V | |
| LP5907MFX-3.3/NOPB | 3.3 V | | 3.3 V | |
| LM2663M/NOPB | 3.3 V | -3.267 V | -3.3 V | -3.333 V |
| TPS72325DBVT | -3.3 V | -2.45 V | -2.5 V | 2.55 V |
| DRV8711 Motor Driver | | 0 A | 1.7 A | 4.5 A |
| 42BYGHM809 Stepper Motor | | 0 NM | .4 NM | .4 NM |
| EMG Signal Frequency | | 50 Hz | 100 Hz | 300 Hz |
| Rotary Encoder RM22I | | | 8192 res | |
| ADS1299 CMRR | | | -110 dB | |
| Data Rate | | 250 SPS | | 16 KSPS |
| ADS1299 Analog VDD | | -.3 V | | 5.5 V |
| ADS1299 Digital VDD | | -.3 V | | 3.9 V |
| ADS1299 Analog Signal Voltage | | | 5 mV | |
| ADS1299 Digital Signal Voltage | | | 3.3 V | |

## 1.2.2 Significant Changes in Direction

*Body-to-Sensor Interface*

The body-to-sensor interface underwent a few significant direction changes. The subsystem shifted from an analog filter focus to a hardware filter focus. Furthermore, the subsystem shifted from frequency-domain signal filtering to time-domain filtering. Also, the orthosis must collect two calibration points (resting EMG activity and maximum EMG activity) as opposed to one calibration point (maximum EMG activity). Finally, the method for convolving the filter coefficients with the sample window may undergo significant changes. These changes and their reasonings are described in their corresponding subsystem section.

*Data Processing*

The Data Processing Subsystem has had only one major change throughout the development process. Initially we started out using an MSP430 microcontroller but later chose to use an MSP432 microcontroller. This change was initiated by the fact that the MSP432 has an extensive digital signal processing library that could be used to further process EMG signals. There is also the added benefit of the low power capabilities of the MSP432. Implementation of the subsystem only changed through the software libraries used.

*Motor-to-Body Interface*
The motor-to-body interface had significant changes in the motor driver and motor selection. The elbow orthosis will be actuated using two high torque dc motors and the respective motor driver DRV8848. See Section below for more information.

*Power Distribution*
The power subsystem has stayed relatively the same. The only major addition is the discharge protection system added for the protection of a lithium polymer battery. This is achieved through the use of a commercially available discharge alarm and an electric latching relay.

*User Interface*
This subsystem has gone through quite a few different iterations. Originally the system was to simply use an LCD screen and some pushbutton wire directly into the microcontroller. This was then altered to utilize the Texas Instruments Chronos smartwatch which would connect with the microcontroller using an RF module. This proved to be quite difficult as little resources were available. It was then decided to look for better options to more properly utilize the little time available in the semester. Currently the system is utilizing an Android app developed with Android Studio and connecting to the MSP432 with the use of a bluetooth module.

## 1.3 Project Timeline and Task Ownership

Nathan was responsible for the Body-to-Sensor Interface. The main purpose of the body-to-sensor interface was to capture, filter, and interpret signals relating an elbow's current and intended positions. The sensors implemented in this subsystem include rotary encoders, force sensitive resistors (FSRs), and electromyography (EMG) sensors. Nathan worked on characterizing the sensors and interfacing them with the central processor. Furthermore, he investigated the methods and signal processing for extracting intended elbow movements from muscle signals.

Rafael was responsible for the conversion of the sensor data to controller values for the motor driver. This data had to be manipulated to satisfy the user specification and the wave signals that the motor driver is expecting. His code would then take into consideration the data coming from the radial encoder and correct for error.

David took charge of the Motor-to-Body. The motor driver was able to actuate the motor based on the PWM signals that are being sent by the microcontroller. The Motor was be fined tuned to be able to provide precise and fluid movements that responded in fractions of a second. Two motors were then used to share the load. Both of the motors were tuned so that they worked together without causing interference.

The Power System was handled by Joe Loredo and provides the appropriate voltages for the different components of the powered orthosis. Power is taken from a large power source, a bench supply, and is regulated through the use of a buck converter and a set of linear voltage regulators. These regulated voltages are used to power microcontrollers and the ADS1299 of the other subsystems. The system was

tested by placing the components on a breadboard and monitoring their output as voltages were applied. Another aspect of the system is safety for both the user and subsystem components. Fuses were used in making sure a spike in current does not damage our circuits. Temperature damage control is handled by the voltage regulators themselves as they all have internal ways of shutting down in times of excess temperature, as stated by their data sheets. For the user a switch was implemented allowing a user to instantly shut off the device in case of an emergency. A rechargeable lithium polymer battery will be used as the system's power supply along with a relay circuit that will shut off the battery to prevent the battery from fully discharging.

The user interface was also handled by Joe loredo and allows the user to calibrate and set safety limits on the orthosis. The system uses an Android app with menus that allow the user to calibrate emg sensors of the ADS1299 and set angle limits. The main MSP432 has an bluetooth attachment that will allow it to communicate with an android powered smartphone. For testing the Android app is connected to an MSP432 with an bluetooth module and test code. The connection was tested by wirelessly causing the led of the MSP432 to blink.

Table 1.3: Responsibility Matrix

| Subsystem | Primary Contact | Responsibilities |
|---|---|---|
| Body-to-Sensor Interface | Nathan Glaser | – Develop and test wearable sensors which will transmit elbow angle and muscle activity<br>– Filter noise from raw sensor signal<br>– Ensure setup is not invasive and is not restrictive |
| Data Processing | Rafael Salas | – Convert sensor data into controller variables<br>– Manipulate data to satisfy user specifications<br>– Reconvert result into motor signal |
| Motor-to-Body Interface | David Cuevas | – Actuate motor and articulate elbow based on microcontroller signal<br>– Ensure movement is assistive and natural |
| Power | Joe Loredo | – Provide portable and sustained power to each subsystem |
| User Interface | Joe Loredo | – Connect user-friendly interface which will dynamically calibrate EMG sensors and specify custom angle limits (without customizing microcontroller source code for every user) |

Figure 1.6 shows the Gantt chart for the Spring 2016 semester. It shows dates for how long each subsystem should take to reach certain milestones.

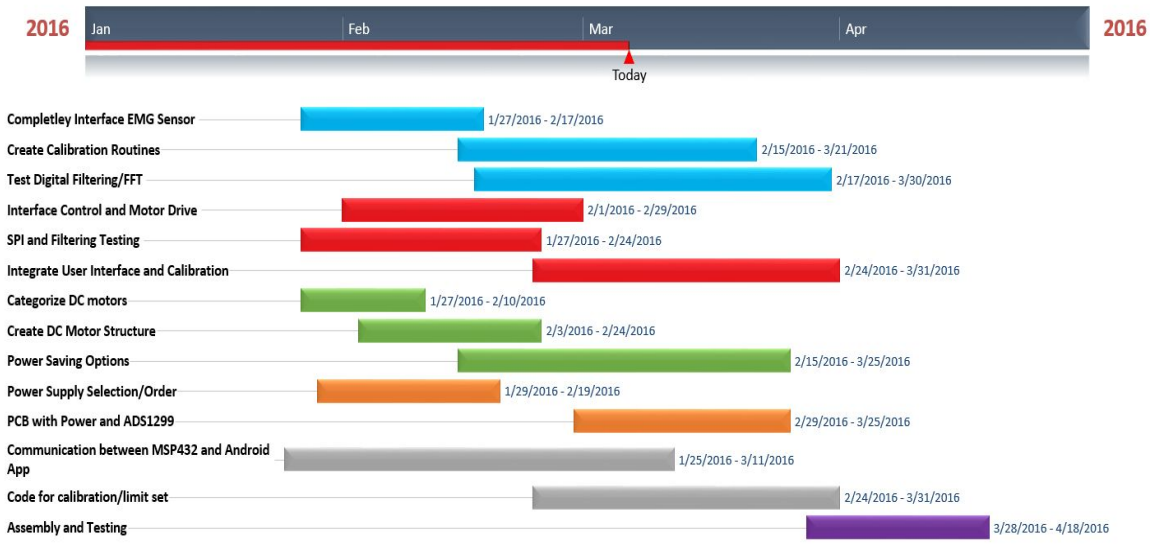



Figure 1.6: Gantt Chart

# 2 Subsystem: Body-to-Sensor Interface

The main purpose of the body-to-sensor interface is to capture, filter, and interpret signals relating an elbow's current and intended positions. The sensors implemented in this subsystem include rotary encoders, force sensitive resistors (FSRs), and electromyography (EMG) sensors.

## 2.1 Significant Changes in Direction

A unity-gain, band-pass filter helps constrain the collected signal to within the accepted region for EMG signals [1]. The cutoff frequencies listed in Table V align with these accepted parameters, and the PSpice model depicted in Figures B6 and B7 conforms to the specifications. Beyond the front end filtering, the ADS1299 encodes the analog readings and passes them to the microcontroller as digital values.

TABLE V
UNITY-GAIN, BAND-PASS FILTER DESIGN

| Parameter | Specification | Simulation results |
|---|---|---|
| Gain | 0 dB | ~ -0.1 dB |
| $\omega_1$ (Lower Cutoff) | 10 Hz | ~ 10 Hz |
| $\omega_2$ (Upper Cutoff) | 300 Hz | ~ 300 Hz |

However, in practice, connecting the analog filters to the ADS1299 EVM board was fruitless. Any additional filtering on top of the board's built-in 5,000 Hz Low Pass Filter negated the signal. Testing included both passive and active filters. Theories for this malfunction include the use of imprecise passive resistor or capacitor components, the use of noisy amplifiers, and the use of a large high pass resistance value which dissipated the low voltage signal.

These challenges encouraged the pursuit of other noise-cancelling methods. Focus shifted from exterior analog filtering the internal instrumentation amplifiers of the ADS1299. Their use involved adjusting some of the device's register bits. Additionally, emphasis shifted towards digital filtering, an already important component. The MSP432 and its associated CMSIS DSP libraries offered favorable digital filtering prospects, compared with the lesser MSP430.

## 2.2 Subsystem Specifications

Since the orthosis structure only allows rotation about a hinged joint, tracking this sole degree of freedom involves only one sensor—a rotary encoder. Successful tests for the rotary encoder include verifying its transmitted pulses, verifying that one pulse precedes the other pulse to provide an indication of direction, and verifying that the MSP432 properly tracks angular position through its GPIO pins.

To distinguish an elbow's unwanted and wanted movements, the orthosis fuses information from force sensitive resistors (FSR) and electromyography (EMG) sensors, respectively. To detect unwanted elbow movements, the orthosis detects changes in contact pressure, specifically using FSRs which change resistance when deformed. Specifically for the orthosis, the FSRs serve as a secondary safety measure. However, if the orthosis contacts the elbow or arm with an undesirable amount of pressure, the FSRs register a threshold voltage, and the microcontroller acts to reduce this presumed discomfort. Successful tests for the FSR include verifying its decrease in resistance with applied pressure and verifying that the MSP432 ADC reads the voltage change across the resistor,

Most importantly, the EMG sensors produce the logic which eventually drives the orthosis. For the case of the orthosis, more active biceps muscle signals indicate intent to close the elbow joint, and more active triceps muscle signals indicate intent to extend the elbow joint. By intercepting these myoelectric signals at the source—at the motor units themselves—the orthosis acts on less muddled and less contingent data. Successful tests for the EMG sensors include verifying that the electrode placement properly collects the EMG signals, verifying that the ADS1299 communicates with the MSP432 along an SPI channel, and verifying that the digital filters clean the raw signal.

Table 2.1
Body-to-Sensor Interface Specification Compliance Matrix

| Specification | Min | Nominal | Max |
|---|---|---|---|
| EMG Signal Frequency | 50Hz | 100Hz | 300 Hz |
| Rotary Encoder RM22I | | 8192 res | |
| ADS1299 CMRR | | -110dB | |
| Data Rate | 250 SPS | | 16 kSPS |
| ADS1299 Analog VDD | -0.3V | | 5.5V |
| ADS1299 Digital VDD | -0.3V | | 3.9V |
| ADS1299 Analog Signal Voltage | | 5 mV | |
| ADS1299 Digital Signal Voltage | | | |

Table 2.1
Body-to-Sensor Interface Connection Matrix

| (Version) | | | (Interfaces) | |
|---|---|---|---|---|
| Project-Subsystem | Description | Power | COM | Signal |
| V1-00 | Gel Electrodes | Bias = ~3V DC | — | ± 5 mVpp (Skin Surface) |
| V1-00 | ADS1299 EVM with MMB0 | USB | USB | ± 5 mVpp (Electrodes) |
| V1-01 | ADS1299 EVM | 2.7 ± 0.9V (DVDD)<br>5 ± 0.25V (AVDD) | 3.3 V (MSP432 SPI) | ± 5 mVpp (Electrodes) |
| V1-02 | ADS1299 PCB | 6 V (VDD) | 3.3 V (MSP432 SPI) | ± 5 mVpp (Electrodes) |
| V1-00 | Rotary Encoder | 3.3 V | 3.3 V (MSP432 GPIO) | 3.3 V Pulse (Axial Knob) |
| V1-00 | Force Sensitive Resistor | 3.3 V | 3.3 V (MSP432 GPIO) | 3.3 V Variable (Pressure Pad) |

## 2.3 Subsystem Status

Since the orthosis structure only allows rotation about a hinged joint, tracking this sole degree of freedom involves only one sensor—a rotary encoder. This sensor uses a rotating axle to encode angular displacement. As its axle rotates through a series of angular checkpoints, the encoder transmits a pulse. Given a calibrated starting position, the microcontroller counts these pulses to determine the hinge's angular displacement and hence, an elbow's current position. Currently, the MSP432 uses its GPIO pins to interface with the rotary encoder and to properly track angular position and direction.

To distinguish an elbow's unwanted and wanted movements, the orthosis fuses information from force sensitive resistors (FSR) and electromyography (EMG) sensors, respectively. To detect unwanted elbow movements, the orthosis detects changes in contact pressure, specifically using FSRs which change resistance when deformed. With this variable resistor and a fixed resistor in a voltage divider, the microcontroller measures the deformation as a voltage. Specifically for the orthosis, the FSRs serve as a secondary safety measure. The first safety measure is the orthosis structure itself, with mechanical stops that physically prevent the orthosis and elbow from entering certain ranges of motion. However, if the orthosis contacts the elbow or arm with an undesirable amount of pressure, the FSRs register a voltage, and the microcontroller acts to reduce this presumed discomfort. Currently, the FSR registers a decrease in

resistance with increases pressure. The MSP432 uses a GPIO pin to interface with this component, and it uses an ADC to determine whether the voltage, and hence pressure, exceeds a threshold value.

Most importantly, the EMG sensors produce the logic which eventually drives the orthosis. In general, EMG sensors capture the cumulative motor unit action potentials of muscle fiber membranes [1]-[2]-[3]. In simpler terms, these sensors measure muscle activity and to some degree, intended movement. For the case of the orthosis, more active biceps muscle signals indicate intent to close the elbow joint, and more active triceps muscle signals indicate intent to extend the elbow joint. By intercepting these myoelectric signals at the source—at the motor units themselves—the orthosis acts on less muddled and less contingent data. The EMG sensors are more "upstream" than other sensors, bypassing some of the physical sensing issues associated with the user's physical limitations. For example, physical contact sensors depend on some degree of neurological and muscular strength; however, EMG sensors can cater to users without physical strength but with sufficient myoelectric intention. Currently, experimentally-determined placements of electrodes collect a signal for analysis within a computer LabView program. A series of tests isolated desired digital filter parameters. Furthermore, the initial SPI communication link between the ADS1299 and the MSP432 is established.

## 2.4 Subsystem Technical Details

The most complicated component of the body-to-sensor interface is the EMG sensor interface, and therefore it merits the most description. It consists of filtering, encoding, and interpreting the user's myoelectric signals.

Specifically for the orthosis, surface electrodes connect the biological world with the electronic world. The system's information flow begins here with registering muscular activity through electromyography (EMG) sensors. To sustain a muscular contraction, independent local muscle fibers must pulse. These individual pulses combine into a natural, Gaussian distribution of frequencies centered around 100 Hz, and the surface electrodes capture this asynchronous combination of individual pulses. These mechanisms provide the bases for the MATLAB Simulated EMG Signal in Figure B8, which closely matches an Actual EMG Signal as in Figures B1-B5 and B9. Furthermore, these mechanisms guide the signal processing, isolating the analysis window between 10 Hz and 300 Hz.

The EMG measurements require two groupings of surface gel electrodes—one bias electrode to elevate the user's overall voltage by a small amount and several pairs of electrodes to capture the differential muscular activity at each region. Unity gain buffers isolate sections of the signal path. Analog bandpass filters admit the relevant spectrum for an EMG signal (10 Hz to 300 Hz) [1]. A high resolution (1μV), low noise analog-to-digital converter (ADC), namely the ADS1299, converts the ~5 mV analog voltage signals into digital values. The high voltage resolution of the ADS1299 and the expense of precision amplifiers undermine the need for dedicated analog amplifiers. Instrumentation amplifiers within the ADS1299 improve the common mode rejection ratio for the differential signal inputs. The central microprocessor uses digital notch filters to remove the permeating power hum (60 Hz) and its harmonics [1]-[2]-[3]. Since myoelectric signals originate from a random combination of individually firing motor units, they follow a natural statistical distribution [1]-[2]-[3]. Digital statistical smoothing algorithms, namely the moving average and root mean square methods, eliminate the sharp signal peaks and provide a simplified, quantitative measure for signal strength [3]. Furthermore, recording a maximum voluntary contraction provides an upper bound for normalizing and calibrating the signal. This normalization allows the orthosis to operate for users of different fitness levels. Figure 3 details the major components of the EMG aspect of the body-to-sensor interface.

Figures B1, B2, and B3 show a non-active EMG signal as registered by the ADS1299 and as evaluated with the ADS1299EE-FE Evaluation Module Software. The first plot displays the unfiltered signal, and the second plot decomposes its noise with a Fourier transform. The signal needs filtering to exclude the power line interference spike at 60 Hz and its harmonics. Additionally, it must exclude the EKG frequency peaks around 70 Hz and the low frequency wander contributions at the lower end of the spectrum. The third plot shows the impact of filtering. Specifically, a 10 Hz high pass filter and a 60 Hz digital notch filter drastically clean the signal. However, the resulting signal is not noiseless—the remaining peaks are the EKG signals relating to the heartbeat. Unfortunately, since these signals vary between individuals, a fixed filter design cannot single them out. Furthermore, the range of their possible frequencies discourages a band cut filter since their frequencies lie near the center frequency (~70 Hz) of EMG signals [1]-[2]-[3]. Figures B4 and B5 show a pulsed EMG signal before and after applying the same filters.

The digital modules in the orthosis communicate through either a Universal Asynchronous Receiver / Transmitter (UART) or a Serial Peripheral Interface (SPI) standard. Both standards transmit and receive numerical data as a sequence of bits. These "1" or "0" binary signals travel as "high" or "low" voltages either along conductive wires or as wireless electromagnetic radiation. The SPI communications are synchronous and require an associated clock signal while the UART communications do not.

## 2.5 Subsystem Testing

The testing for the Body-to-Sensor Interface primarily focused on collecting, transmitting, and characterizing the electromyography (EMG) signals produced by different muscles. Specifically, these tests investigated the operation of a precision biopotential analog-to-digital converter. The tests analyzed the effects of analog and digital filters, probed muscle groups, and characterized the relationship between EMG activity and intended force. Additional tests characterized the peripheral sensors, such as the rotary encoder and force sensitive resistor.

In the following tests, a bench supply powered the components, an oscilloscope displayed analog signals, and its logic analyzer attachment displayed the digital signals. The ADS1299 converted analog EMG signals to digital signals with 24-bit precision and high common mode rejection. The ADS1299 EVM served as a breakout board for the ADS1299 and allowed for easy setting of its hardware pins. The MMB0 Platform powered and connected the ADS1299 EVM Module to a computer. A LabView program then communicated firmware settings and analyzed any collected data.

### 2.5.1 ADS1299 Evaluation Module (EVM): General Operation and Filter Test

The purpose of this test was to verify the general operation of the ADS1299 and to explore the functionality of its associated software. The post-processing signal analysis of the test provided insight regarding the necessary signal filtering. The major purpose and pass condition of this test was to observe a verifiable output signal as per an input test signal.

#### *Test Setup*

The ADS1299 EVM Module, the MMB0 Motherboard Platform, and its associated software were configured per online manuals. The software allows for easy setting of the ADS1299 firmware bits. For this initial test, the ADS1299 was naively deployed in Reference Mode instead of Differential Mode; the

distinction between these firmware settings is discussed in the conclusion. Three gel electrodes were connected to the subject--a bias electrode on the elbow (an electrically inactive region), a reference electrode also on the elbow, and a signal electrode on the biceps. An EMG activity signal was then collected for approximately 10 seconds. The collected signals were analyzed with custom post-processing digital filters.

### *Data*

The ADS1299 transmits data through a Serial Peripheral Interface (SPI), and the following plots represent the numerical equivalent of this bitstream. The horizontal axis unit is sample number, but this unit is interchangeable with time since the ADS1299 transmits data sequentially at a predefined rate. The vertical axis is voltage, though its precise value is irrelevant since all signals represent a proportion of some calibrated maximum signal. The two important features of these plots are the relative amplitudes of signal peaks and their signal densities.

Figure 2.1 displays the raw EMG data for a person doing nothing at all. The expected signal for an inactive person should be a noisy, but constant signal. However, the actual signal is an unstable one with considerable periodic noise contributions. Figure 2.2 decomposes the frequency contributions of this raw EMG signal with a Fourier transform. The frequency plot for an inactive person should also be a noisy, but flat frequency distribution. However, this figure shows an elevated spectrum at lower frequencies, a sharp peak at 60 Hz, and smaller peaks at 70 Hz and 120 Hz. These elevated lower frequencies correspond to electrode drift and slight changes in skin conductivity. The 60 Hz and 120 Hz contributions correspond to significant power-line interference. The 70 Hz frequency corresponds to the electric stimulus for heart contractions.

Figure 2.3 displays the time-domain signal after digital filtering. Specifically, a digital 10 Hz High Pass Filter and a digital 60 Hz Notch Filter clean the signal. The result of the filtering is a flat and narrow signal as desired, but there are periodic sharp peaks. These peaks correspond to the 70 Hz electric stimulus for a heartbeat. However, since this noise contribution is biological, and hence dynamic by nature, it may not be easily removed by a rigid Notch Filter.

Figures 2.4 and 2.5 display two muscular contractions without and with the previously mentioned digital filters. The contractions are somewhat discernable in Figure 2.4, displayed as distortions to a periodic noise pattern. The contractions are clearly discernable in Figure 2.5, displayed as dense and amplified signals relative to the background.

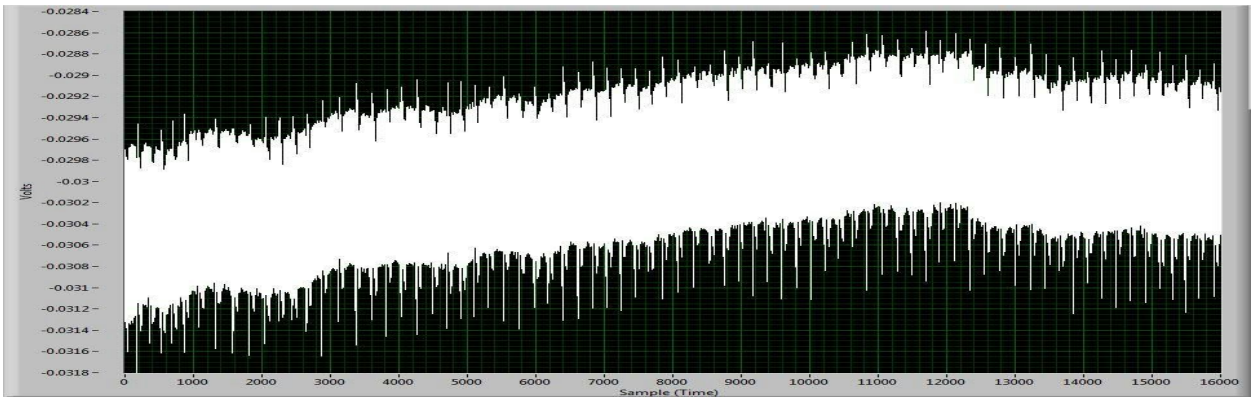



Figure 2.1: EMG Activity (Time-Dependent, Unfiltered, Resting)

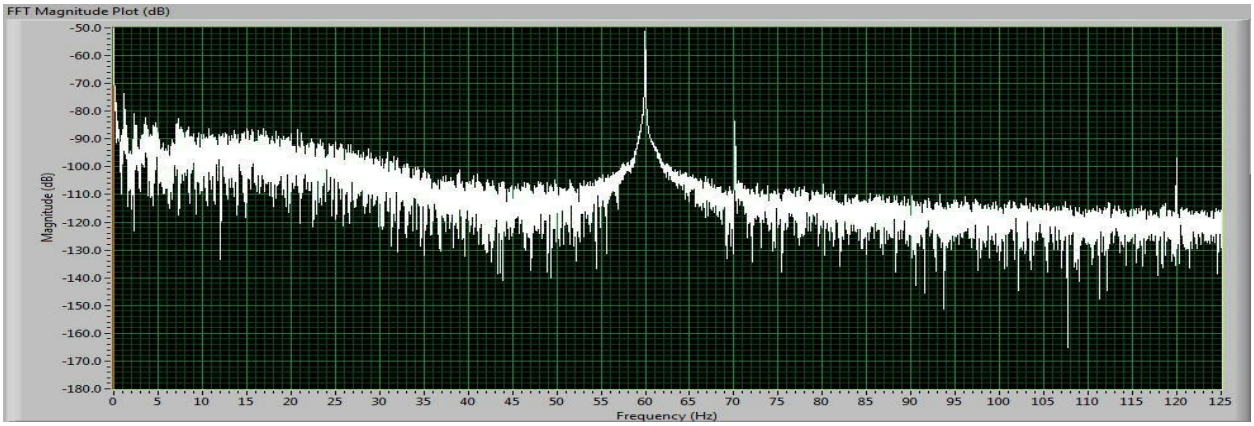



Figure 2.2: EMG Activity (Frequency-Dependent, Unfiltered, Resting)

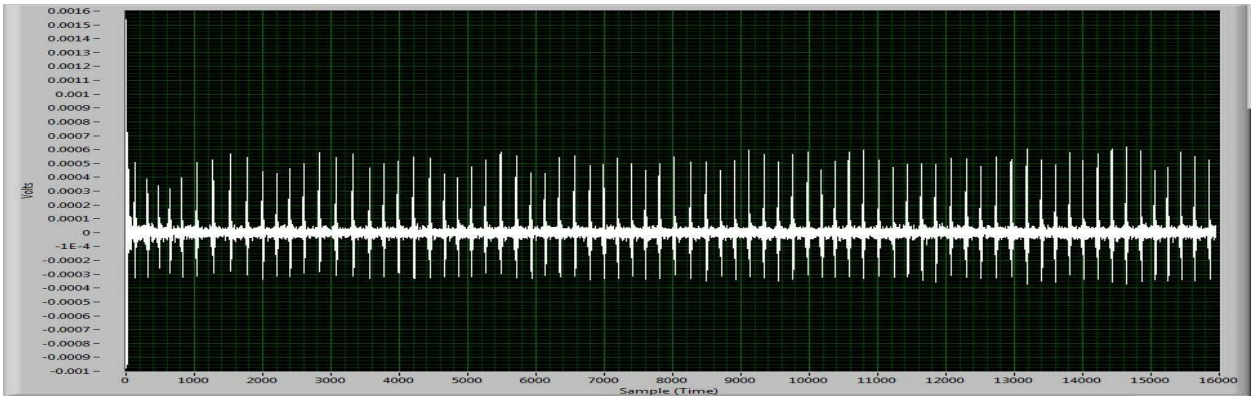



Figure 2.3: EMG Activity (Time-Dependent, Filtered (10 Hz High Pass, 60 Hz Notch), Resting)

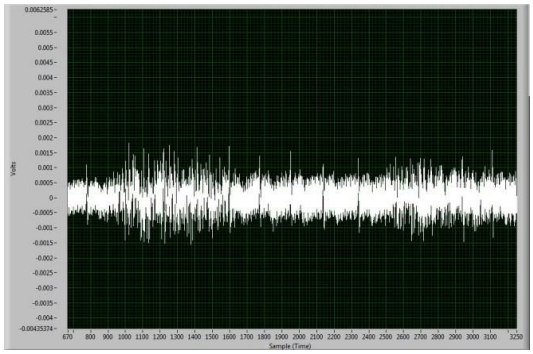


Figure 2.4: Unfiltered, Pulsed EMG Signal

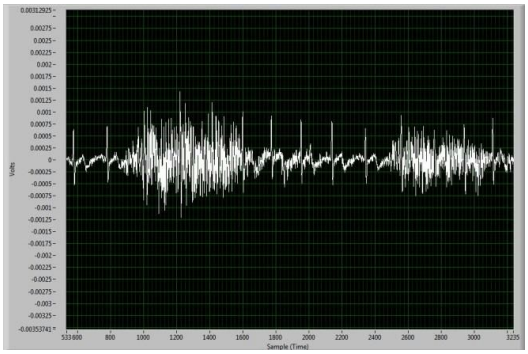


Figure 2.5: Filtered, Pulsed EMG Signal

*Conclusion*

In conclusion, the General Operation and Filtering Test passed its requirements. This test verified the general operation of the ADS1299 and explored the functionality of its associated software. This test followed the firmware configuration for Reference Mode, a signal input mode which collects differential voltages between each signal electrode and a single reference electrode. Reference Mode contrasts with Differential Mode, an input configuration which collects differential voltages between pairs of electrodes. While Reference Mode reduces the number of necessary electrodes, Differential Mode vastly improves the common mode rejection by placing the relevant pairs of electrodes near each other. As evidenced by this test (which used Reference Mode) compared to subsequent tests (which use Differential Mode), the raw signals from Differential Mode require less filtering for power-line interference and cardiological signals. Another triumph was the determination of the front-end filter parameters. A 10 Hz High Pass Filter and 60 Hz Notch Filter sufficiently sleuthed the EMG signal from noise.

### 2.5.2 ADS1299 EVM: Electrode Placement and Muscle Group Independence Test

The purpose of this test was to provide insights regarding electrode placement. Furthermore this test hoped to affirm that the EMG signals of different physical movements are independent, despite these distinct motor neuron signals traveling along intertwined axons. The major purpose of this test was to find both an optimal and independent placement of electrodes. The major pass condition of this test was the eventual determination of electrode placement and improved signal quality and independence.

*Test Setup*

The ADS1299 EVM Module, the MMB0 Motherboard Platform, and its associated software were configured per online manuals. The software allows for easy setting of the ADS1299 firmware bits. For this test, the ADS1299 was deployed in Differential Mode. Five gel electrodes were connected to the subject--a bias electrode on the elbow (an electrically inactive region) and two pairs of signal electrodes, one placed across the biceps and one placed across the inside of the forearm. Trial runs were performed with different electrode placements, and eventually, an optimal configuration was obtained. Two electrodes placed laterally across the biceps and two electrodes placed longitudinally along the interior of the forearm produced the strongest signal with the least amount of noise. With this configuration, an EMG activity signal was then collected for approximately 20 seconds. First, the wrist was flexed closed then

opened. Next, the biceps was flexed closed then opened. The collected signals were analyzed with custom post-processing digital filters.

### *Data*

Figure 2.4 displays the EMG activity signal where the wrist and elbow were independently flexed. The red signal corresponds to the EMG activity signal of the wrist, and the blue signal corresponds to the EMG activity signal of the biceps. Both motions registered a signal on both electrodes. However, the relevant muscle signal overpowered the irrelevant signal in both cases. Another important feature includes the discrepancy between the amplitudes of both independent signals.

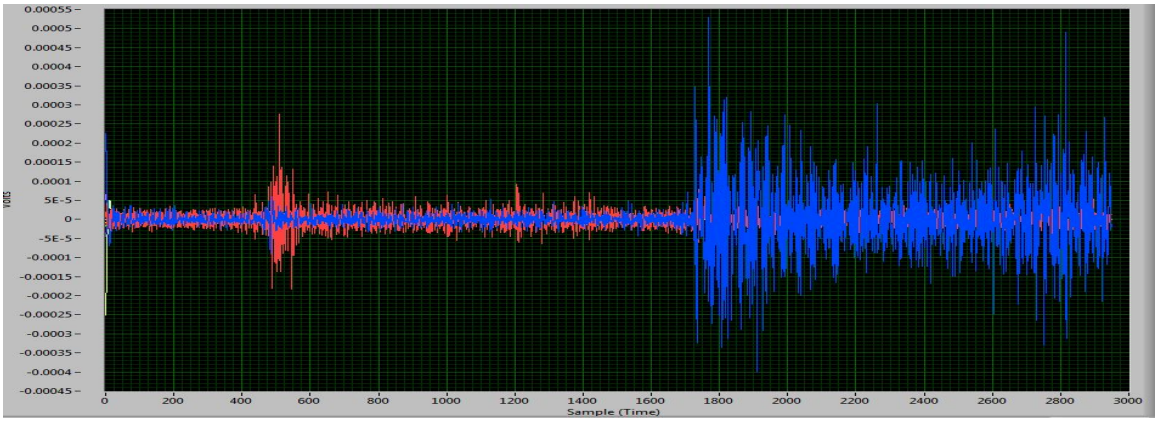



Figure 2.4: EMG Activity of Two Independent Muscle Groups

### *Conclusion*

In conclusion, the Electrode Placement and Muscle Group Independence Test passed its requirements. This test searched for and found an optimal electrode configuration, corresponding to electrodes placed laterally across the biceps and triceps muscles. Furthermore, the test showed that the elbow muscles acted independently from the wrist signals.

In the language of Linear Algebra, independence and signal orthogonality is an important condition for establishing a spanning basis set. In the scope of the orthosis, the signals for one range of motion do not obscure the signals of the other range of motion, so each muscle group directly maps to an independent a physical movement. However, these two independent muscle groups do not produce the same voltage amplitude. To establish a spanning orthonormal basis set, the corresponding signals must normalize to a calibrated value.

## 2.5.3 ADS1299 EVM: Linearity Test (Force and EMG Activity)

The purpose of this test test was to investigate the supposed linear relationship between EMG activity and produced force. Furthermore, the relationship discovered by this test hoped to complete the logic chain of the EMG Sensors for the Body-to-Sensor Interface. The major purpose of this test was to characterize the

relationship between EMG activity and produced force, and its major pass condition was the ability to use the determined relationship to serve as a transfer function between measured EMG activity and motor logic.

### *Test Setup*

The ADS1299 EVM Module, the MMB0 Motherboard Platform, and its associated software were configured per online manuals. The software allows for easy setting of the ADS1299 firmware bits. For this test, the ADS1299 was deployed in Differential Mode. Three gel electrodes were connected to the subject--a bias electrode on the elbow (an electrically inactive region) and a pair of signal electrodes placed laterally across the biceps. With this configuration, an EMG activity signal was collected for approximately 30 seconds. First, the subject flexed his or her biceps as much as possible, allowing the ADS1299 to collect the calibration signal (the subject's maximum voluntary contraction). Next, the subject lifted and lowered an unweighted arm for 10 seconds. Then, the subject lifted and lowered a weighted arm for 10 seconds. Finally, the subject sustained one final maximum voluntary contraction.

### *Data*

Figure 2.5 displays the raw, unfiltered EMG signal from the test test procedure. Figure 2.6 displays the filtered EMG signal. Figures 2.5 and 2.6 display the four movements listed in the test procedure. The calibration, maximum voluntary contraction signals border the two linearly scaled EMG signals. Figure 2.7 focuses on the two linearly scaled EMG signals.

The bordering maximum voluntary contraction signals serve as the calibration signal, whereby all other collected signals represent some proportion of it. In Figure 2.7, the weightless arm produced an EMG activity signal with an amplitude smaller than that of the weighted arm. Furthermore, these activity signals increase linearly with added weight.

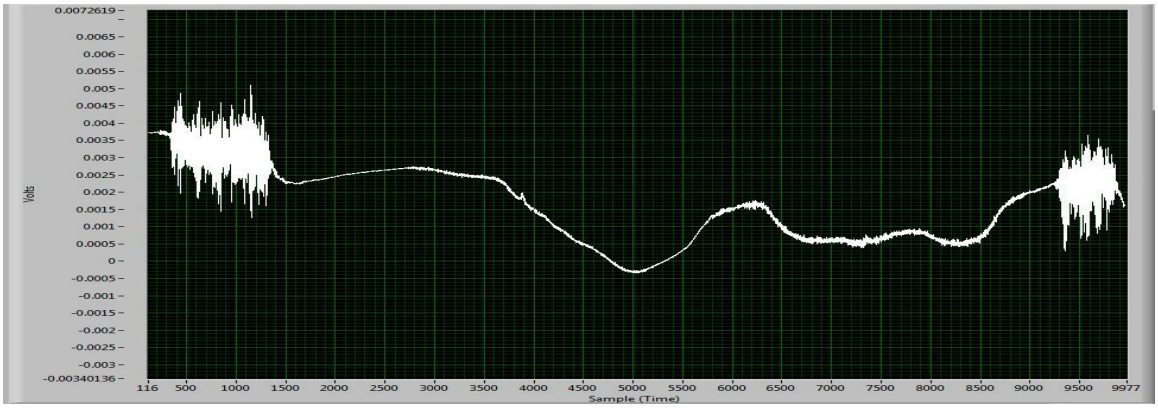



Figure 2.5: Raw, Unfiltered EMG Signals (Calibration → No Weight → Weight → Calibration Signal)

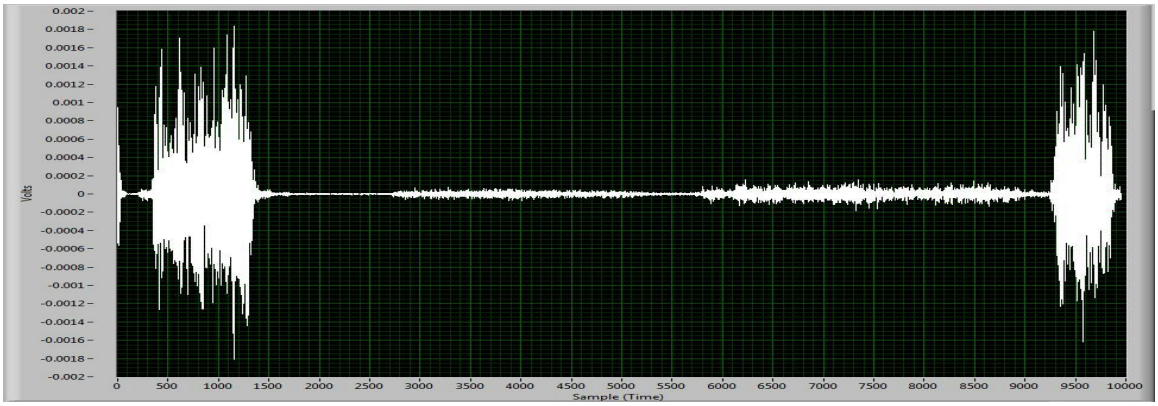



Figure 2.6: Filtered EMG Signals (Calibration → No Weight → Weight → Calibration Signal)

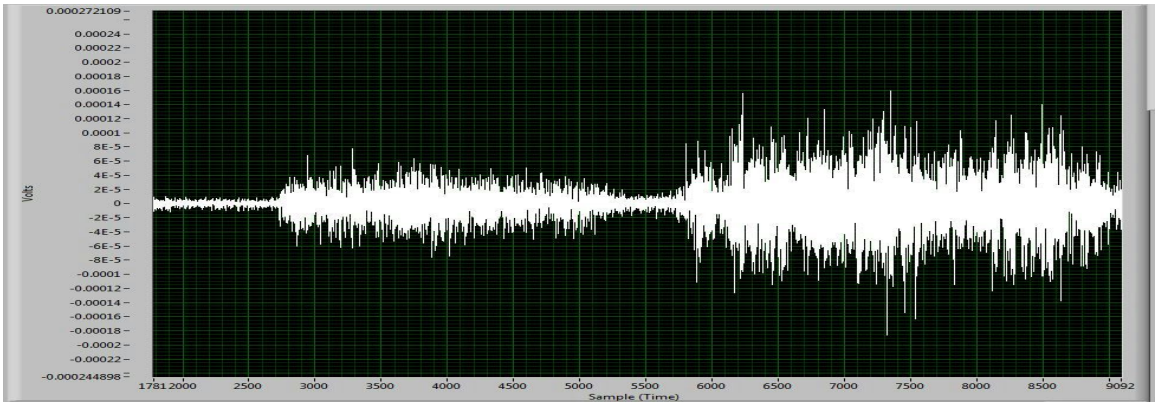



Figure 2.7: Focus on Filtered EMG Signals (No Weight → Weight Signal)

### *Conclusion*

This test confirmed the linear relationship between EMG activity and produced force. Furthermore, this relationship completes the logic chain of the EMG Sensors for the Body-to-Sensor Interface: biopotential signals travel along the electrodes to the ADS1299; the ADS1299 rejects the common mode interference and converts these analog signals into digital data; the digital data travels along an SPI communication line to the central processor; a digital high pass filter and a notch filter remove noise artifacts; a moving average smooths the signal to a stable voltage value; a calibration signal divides all voltage values, causing all signals to represent a proportion of the user's maximum voluntary contraction; and finally, because of the direct relationship between intended force and EMG activity, this proportion (signal / max signal) directly maps to a variable to drive motor motion.

A linear relationship is important because only two points are necessary to completely quantify the relation between EMG activity and intended force--the (0 activity, 0 force) point and the (max activity, max force) point:

$$y = mx + b \; ; \; m = (y_2 - y_1)/(x_2 - x_1) \tag{2.1}$$

$$activity = (max\ activity\ /\ max\ force) * force \tag{2.2}$$

While the orthosis has full control over the max force parameter (global constant), the max activity varies between different users (constant for an individual user, variable between different users). Variation among users necessitates an activity parameter which does not change between users. This parameter is the normalized EMG activity signal (signal / max signal) which is constant between users. With this parameter in the direct relationship, the orthosis directly maps the EMG activity to intended output force, regardless of a user's muscular fitness. Figure 2.8 shows the linearity of EMG signals and slopes change between persons of varying muscular fitness.

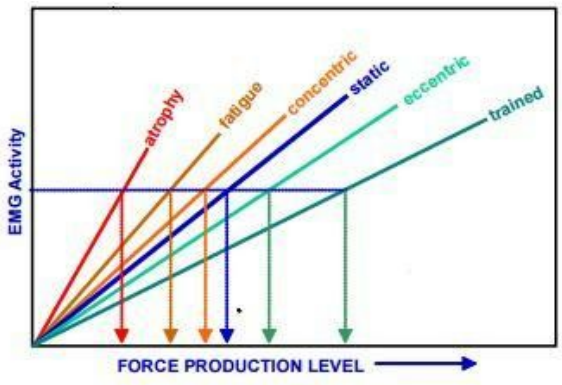



Figure 2.8:Linearity of EMG Signals, Independent of Muscular Fitness[5]

## 2.5.4 ADS1299 EVM: Linearity and Subject Diversity Test

The purpose of this test was to investigate the discrepancies in EMG signals between different individuals. Furthermore, this test investigated the similarities between each individual's characteristic transfer function between force and EMG activity. The major purpose of this test was to gain insight regarding the standardization of the sensor system to diverse users. If each user's characteristic curve (relating EMG activity and Force) differed by some offset or some small scalar, the test would be deemed successful.

### *Test Setup*

The ADS1299 EVM Module, the MMB0 Motherboard Platform, and its associated software were configured per online manuals. The software allows for easy setting of the ADS1299 firmware bits. For this test, the ADS1299 was deployed in Differential Mode. Three gel electrodes were connected to the subject--a bias electrode on the elbow (an electrically inactive region) and a pair of signal electrodes placed laterally across the biceps. With this configuration, an EMG activity signal was collected for approximately 30 seconds. First, the subject flexed his or her biceps as much as possible, allowing the ADS1299 to collect the calibration signal (the subject's maximum voluntary contraction). Next, the subject lifted and lowered an unweighted arm for 10 seconds. Then, the subject lifted and lowered progressively heavier weight for 10 seconds. Then, the subject sustained one final maximum voluntary contraction. This same procedure was repeated for multiple different users. Figures 2.9 and 2.10 depict the signals associated with this procedure for two different subjects. Each signal spike corresponds to a specific

weight, monotonically increasing from left to right. A Matlab program extracted the activity averages from these test signals.

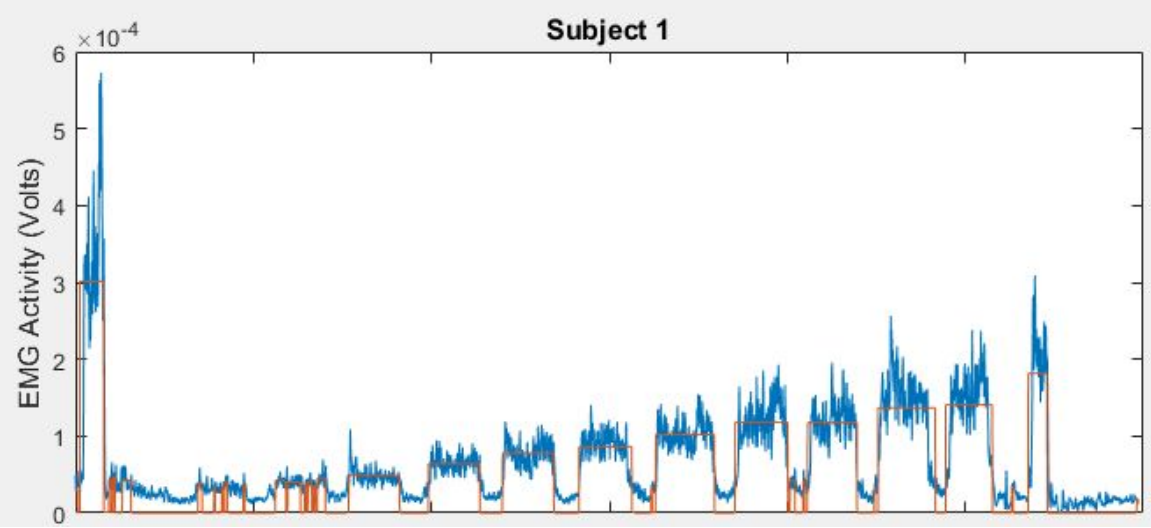



Figure 2.9: Subject 1 Test Signal (Consecutive Peaks Correspond to Increased Weight)

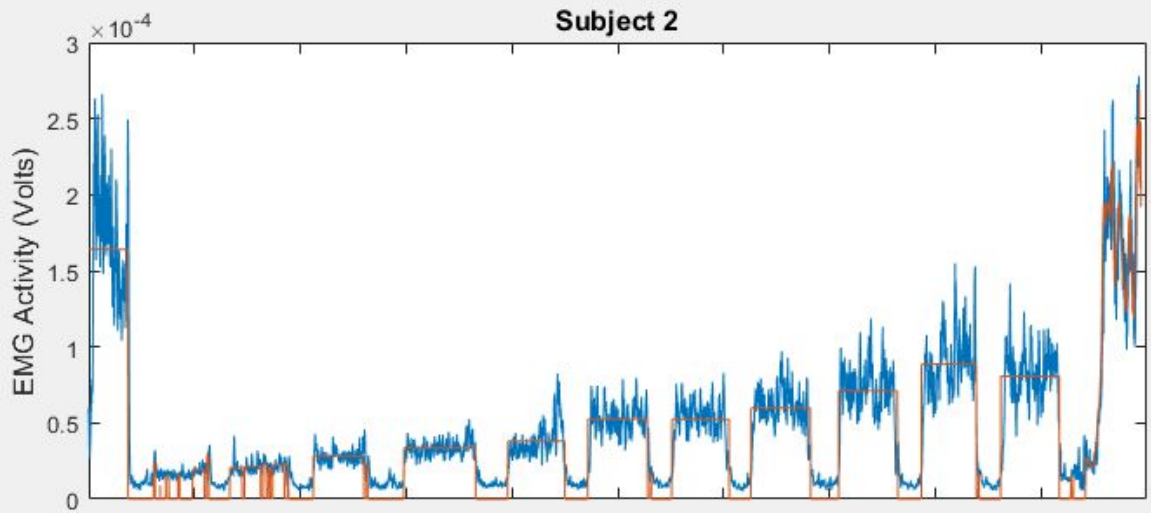



Figure 2.10: Subject 2 Test Signal (Consecutive Peaks Correspond to Increased Weight)

### *Data*

Figure 2.11 summarizes the data collected from this test. The data and general trends for this experiment resemble the previously collected data from the EMG Signal and Force Characterization Test. Even though the characteristic curves between users differed, these transfer functions differed from each other by a slight offset and proportionality constant. The figure depicts three important features. First, each of the data series is linear and linear to a strong degree. Specifically, Subject 1's data correlated with an $R^2$ value of

0.9608, Subject 2 with 0.9560, Subject 3 with 0.9495, and Subject 4 with 0.9846. Figure 2.12 depicts these trends. These results are significant because they indicate that a linear regression correctly predicts EMG Activity with 95% accuracy, given a multiple of an arbitrary weight. Second, each trendline intersects the y-axis at a non-zero value. This trend indicates that the (zero force, zero EMG activity) point is no longer valid for all users. Two calibration signals may be required to fully characterize the transfer function for an arbitrary user. Third, the trendlines are substantially different between each other. For instance, Subject 2's EMG Activity at the maximum tested weight of 10 units was larger than all other Subjects at the minimum tested weight of 1 unit. These substantial differences necessitate some sort of calibration and normalization routine.

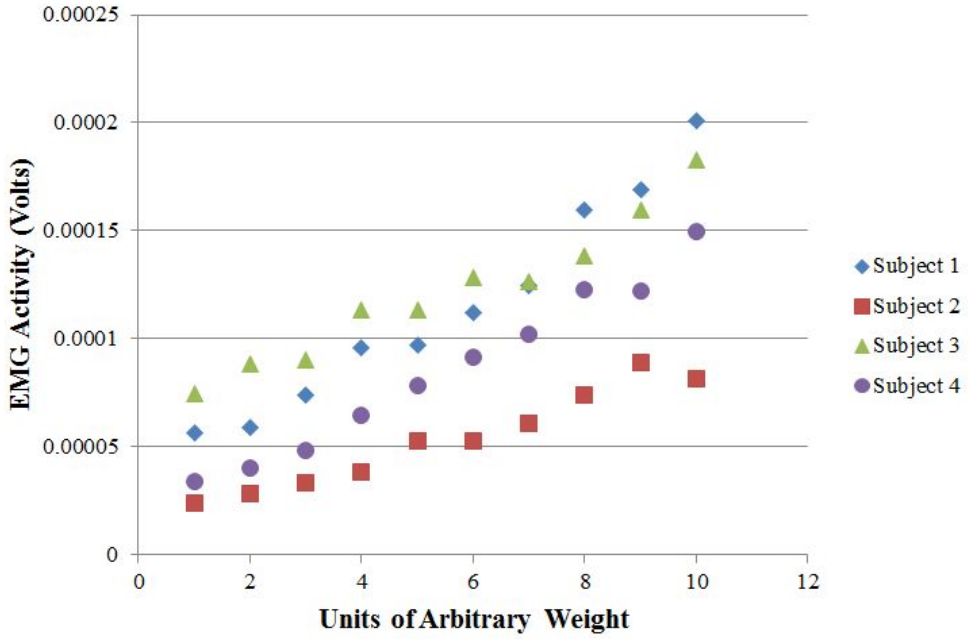



Figure 2.11: Diversity of EMG Activity versus Restraint Weight

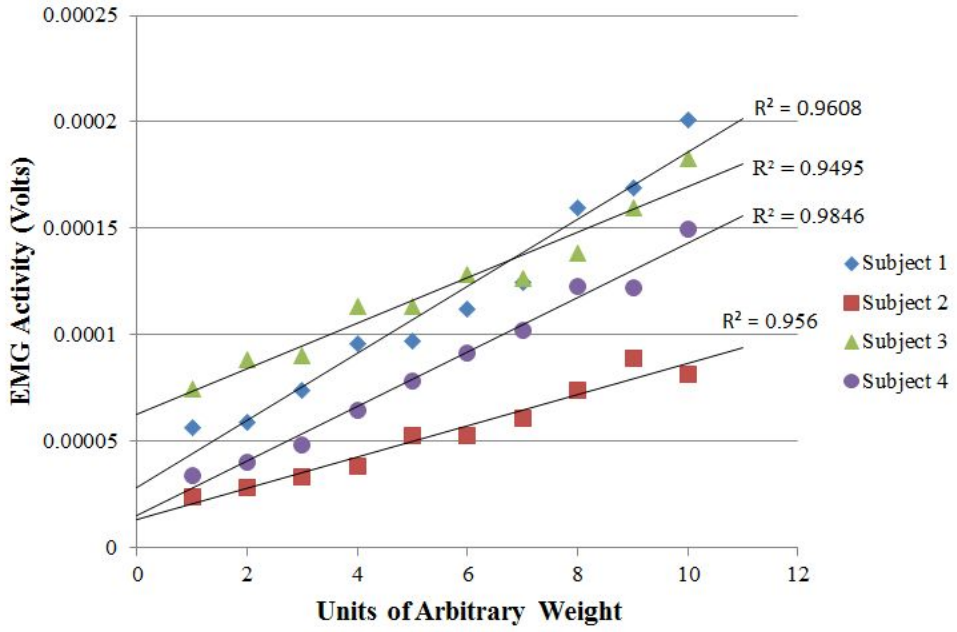



Figure 2.12: Diversity of EMG Activity versus Restraint Weight Trendlines

### *Conclusion*

This test revealed a general trend between different users of unique muscular builds. Furthermore, based on this trend, a calibration routine was developed such that it standardizes the signals of most users. The general expectation was that a calibration signal would divide all voltage values, causing all signals to represent a proportion of the user's maximum voluntary contraction. Because of the direct relationship between intended force and EMG activity, this proportion directly maps to a variable to drive motor motion. The modified, two point calibration routine involves the following equation:

$$y = (x - MIN(x))/(MAX(x) - MIN(x)) \tag{2.3}$$

$$force = (activity - MIN(activity))/(MAX(activity) - MIN(activity)) \tag{2.3}$$

Figure 2.13 depicts this two point calibration routine as applied to the collected data. Figure 2.14 shows how the data align as nearly linear, with an $R^2$ value of 0.9403. Applied to an arbitrary (but calibrated) signal, the previously defined transfer function maps EMG activity to force with nearly 95% accuracy.

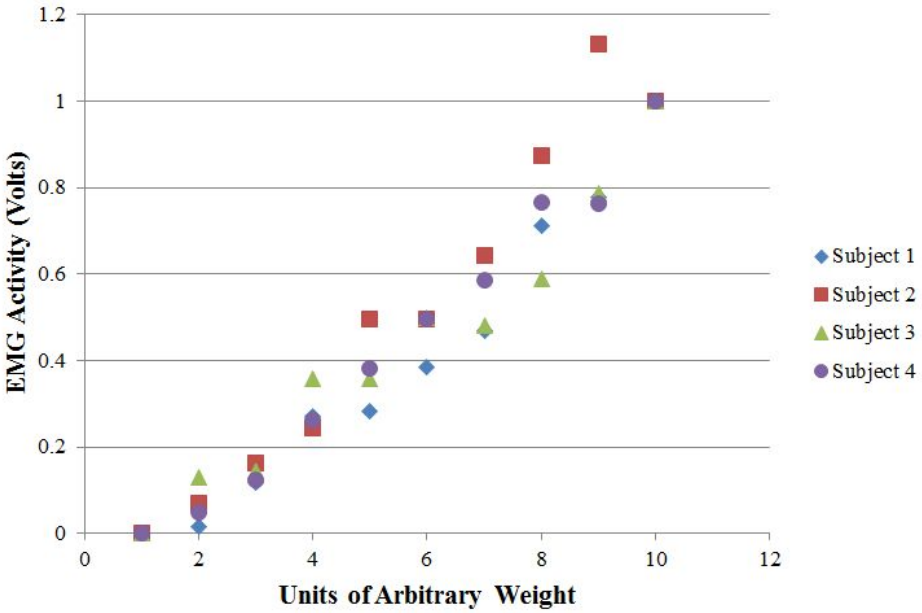



Figure 2.13: Calibrated Signals for Each Subject

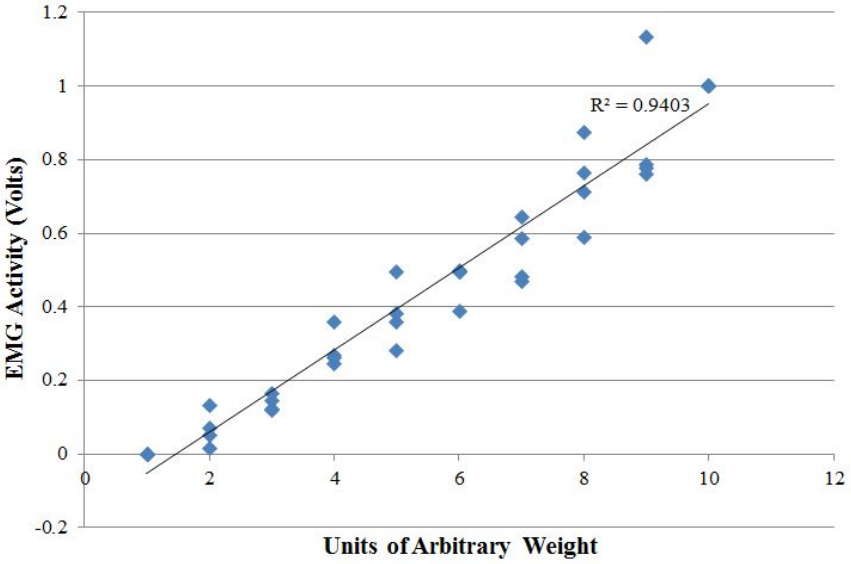



Figure 2.14: Calibrated Signals Regression Line

# 3 Subsystem: Data Processing

The objective of the data processing subsystem is to receive sensor data input and transform it into suitable control variables for motor logic. Figure 1.4 details the process. The MSPWare Library handles all hardware-software interfaces for the MSP432. This specific chip and its library suite perform digital signal processing and establish all necessary communications with the other subsystems.

This system processes three data signals: EMG signal data, radial encoder data, and force sensitive data. MSPWare configures all of the pins required to collect these inputs. The EMG data undergoes the most processing. The CMSIS software library performs a Fast Fourier Transform on the EMG data to convert the signal into the frequency domain. The CMSIS library contains the software methods that utilize the MSP432's Cortex, a digital signal processor. In the frequency domain, a digital Notch filter eliminates the 60 Hz signal generated by power line interference with the electrodes. Additional CMSIS methods are used to filter any other noise. The filtered EMG signal is then transferred back to a time domain signal. Figures 3.2 through 3.4 detail the procedure. Microcontroller C Code rectifies and averages the filtered EMG signal using primitive methods. Rectification and averaging smooth any sharp spikes in the signal and transform it into a suitable variable to be used by the Motor-to-Body Interface. Figures 3.5, 3.6, and 3.7 show an unaltered, a rectified, and an averaged signal.

The radial encoder signal is collected by a GPIO pin configured to ADC mode. The number of square wave pulses from the radial encoder determines motor movement and arm positioning. A simple software loop counts the square wave pulses. The force sensitive resistor signals also enter the microcontroller through a GPIO pin. This signal is used as simple limit switch to provide a electrical safety stop for the motor. The three inputted signals will be used to create set point, error, and safety stop variables. These variables are used to create closed loop motor control.

The MSP432 also communicated with the User Interface system to provide a calibration method. The MSP432 receives a RF signal to initiate a calibration algorithm that matches muscle exertion with motor movement. Figure B10 shows a diagram of muscle exertion to EMG signal. This allows for a proportional relation to be made between the muscle exertion and EMG data. This feature is used when calibrating for a new user.

## 3.1 Significant Changes in Direction

The Data Processing Subsystem has had only one major change throughout the development process. Initially we started out using an MSP430 microcontroller but later chose to use an MSP432 microcontroller. This change was initiated by the fact that the MSP432 has an extensive digital signal processing library that could be used to further process EMG signals. There is also the added benefit of the low power capabilities of the MSP432. Implementation of the subsystem only changed through the software libraries used.

## 3.2 Subsystem Specifications

The Data Processing subsystem's main purpose is to transform sensor data into usable control and indication variables. Currently, it is comprised of a main body of code/software which establishes two types of communications SPI and UART. The software also takes in two types sensor data not related to SPI. An

ADC sequencer can take in analog sensor data; for this project the analog sensor is the FSR. The second type of sensor reading the software supports is discrete digital reads which is used for the radial encoder.

The software is setup to perform necessary data manipulation but has yet to be tested on the MSP432 which is the microcontroller used for data processing. The code has been tested using MATLAB software and performs the data manipulation satisfactory.

The software platform used is code composer which flashes the instruction set onto the MSP432 when prompted. All physical connections made to the MSP432 are currently through breadboard prototyping and usb cable.

Table 3.1
Data Processing Specification Compliance Matrix

| Specification | Min | Nominal | Max |
|---|---|---|---|
| MSP432/ADS1299 SPI Clock | | 500KHz | 20MHz |
| MSP432 Vss | -0.03 V | | 4.7V |

Table 3.2
Data Processing Connection Matrix

| (Version) Project-Subsystem | Description | Power | (Interfaces) COM | Signal |
|---|---|---|---|---|
| V1-00 | MSP432 | 3.3V | SPI / UART | GPIO |
| V1-01 | MSP432 PCB | 3.3V | SPI / UART | GPIO |
| V1-00 | RF Booster | 3.3V | UART | RF |
| V1-00 | Code Composer | — | USB Micro-C | — |
| V1-00 | MSPware | — | Code Composer | — |
| V1-00 | CMSIS DSP | — | Code Composer | — |

## 3.3 Subsystem Status

The Data Processing subsystem's main purpose is to transform sensor data into usable control and indication variables. Currently is comprised of a main body of code/software which establishes two types of communications SPI and UART. The software also takes in two types sensor data not related to SPI. An

ADC sequencer can take in analog sensor data; for this project the analog sensor is the FSR. The second type of sensor reading the software supports is discrete digital reads which is used for the radial encoder.

The software is setup to perform necessary data manipulation but has yet to be tested on the MSP432 which is the microcontroller used for data processing. The code has been tested using MATLAB software and performs the data manipulation satisfactory. Motor control software is in place to drive the motor bidirectionally. This has been tested using the DRV 8848 motor driver in conjunction with the MSP432.

The software platform used is code composer which flashes the instruction set onto the MSP432 when prompted. All physical connections made to the MSP432 are currently through breadboard prototyping and usb cable.

## 3.4 Subsystem Technical Details

The Data Processing Subsystem has several processes SPI communication, two's compliment conversion, and data manipulation. The first process deals with the transmission of EMG data between the ADS1299 and The MSP432. The ADS1299 provides a bitstream of the EMG data and transfers it at the rate set by the MSP432 SPI clock. The MSP432 starts the processes by sending over configuration data and a start signal. Once the start signal is received the ADS1299 continuously transfers the bitstream. This process is implemented through C code and TI Code composer. Code snippet 3.1 demonstrates the software functionality for SPI read which takes place after the ADS1299 commences streaming bits.

Code 3.1 SPI Receive and Store Functions

```
for(iterator = 0;iterator < bit_stream_length;++iterator){
        for(iter = 0; iter<delay ;++iter){}// delay for 4 cycles
        container[iterator%3] = SPI_receiveData(EUSCI_B0_MODULE); //read a byte
        if(iterator%3 == 2){
                sample[spi_index ]= twos_to_signed (container[0],container[1],container[2]);
        }
}
```

The second process involves conversion of the bitstream into a usable value. The bitstream needs to be converted from two's compliment into signed integer format. Code 3.2 snippet shows the conversion method used. C's bitwise operations were leveraged to perform the conversion. This conversion was necessary to perform the signal manipulation since previous code was written with signed int format in mind.

Code 3.2 Two's Compliment

```
int32_t twos_to_signed (uint32_t msb, uint32_t mid, uint32_t lsb){
        uint32_t num = (msb<<24)|(mid<<16)|(lsb<<8);// concatenate bytes
        int32_t num2;
        if(((int)num)<0){//check to see if neg
                num2=-((~num)+1); //2's complement
        }
        else{
                num2 = num;
        }
```

```
        num2 = num2>>2; //shift to correct for 2 bit offset needed for 2's complement
        return num2;
}
```

The final process was data manipulation and digital filtering. The microcontroller software digitally filters the incoming signals and then standardizes them across different users. Specifically, a 60 Hz digital notch filter removes the significant power line interference. Filtering this power hum is critical because its contributions are often on the same order of magnitude (~5mV) as the EMG signal [1]. The microcontroller then rectifies the filtered, digital voltage values, transforming them into a statistically manageable format. To smooth the data, the microcontroller calculates a simple moving average. Equation (6) outlines the averaging procedure with |V| representing the rectified voltage and with N representing number of samples in a designated time frame (typically the time frame is 50ms):

$$Moving\ Average = \langle V(N) \rangle_i = \frac{\sum_{i=1}^{N} |V_i|}{N} \tag{6}$$

After statistical smoothing, the microcontroller normalizes the muscle activity of each user. Illustrated in Figure B10, EMG activity relates linearly with force produced, regardless of muscle type [3]. Only two data points are necessary to define this line for each user—the zero activity, zero force point and the maximum activity, maximum force point. In fact, during initial setup, the orthosis runs a calibration routine which collects this second data point. By normalizing all subsequent activity measurements to this initial maximum value, the orthosis achieves a universal "degree of intention", independent of the user's physical strength. This standardized metric controls the torque settings for the motor, passively assisting the elbows of normal users and augmenting the atrophied elbows of disabled users.

## 3.5 Subsystem Testing

Data Processing system involved testing sensor and data manipulation through internal and external test equipment. UART communications were established between the microcontroller and a terminal on a PC to view data manipulation and sensor input. Along with this a logic analyzer and Oscilloscope were used to test SPI communications.

### 3.5.1 Serial Peripheral Interface Test

This test is used to examine the proper SPI transmission. Proper SPI transmission involves observing a 500 KHz signal and seeing the correct bitstream sent and received between the MSP432 and ADS 1299.

#### *Test Setup*

SPI pins were connected to a logic analyzer to view the SPI clock, chip select, MSP432 out, and ADS1299 output signals . These signals correspond to the first,second, third, and fourth channel respectively. The clock was also connected to the analog channel to observe it further.

#### *Data*

The signals can be seen in figure 3.1. The analog signal displays the clock while channel one of the logic analyzer shows the digital delta function representation. The second line shows the chip select line which

is driven low to select the ADS1299. The third shows data transmission from the MSP432 and the last line shows data transmission to the MSP432.

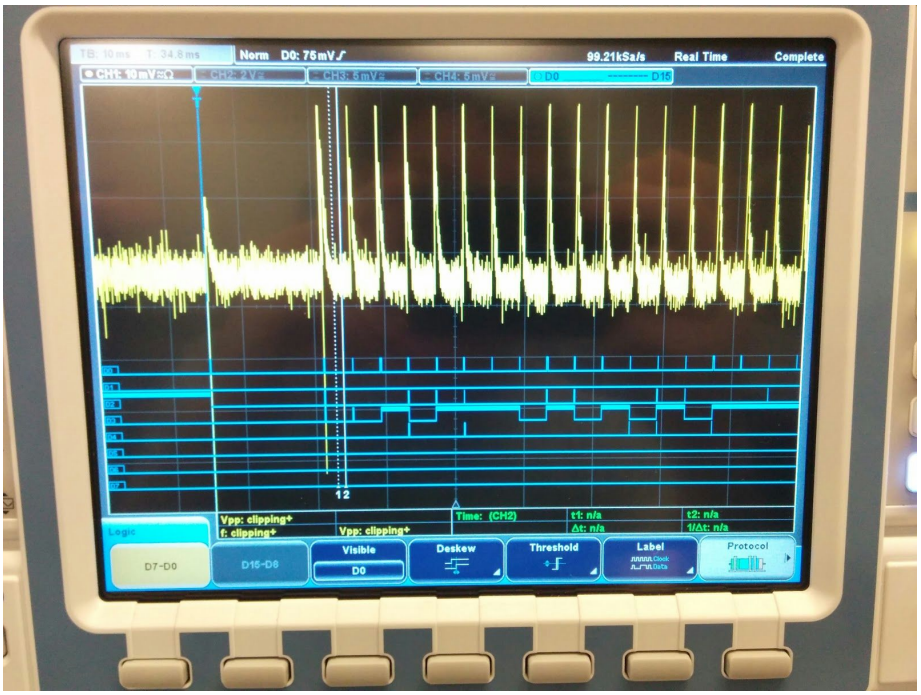



Figure 3.1 Logic Analyzer

*Conclusion*

Although most of the test was past satisfactory, transmission to the MSP432 failed to get the proper results. Modifications to the SPI software is needed to achieve proper input from the ADS1299.

## 3.5.2 Auxiliary Sensor Tests (Force Sensitive Resistor and Rotary Encoder)

This test is used to observe proper input from the radial encoder and force sensitive resistors. UART communications were used to view the proper input/data.

*Test Setup*

Pins for both sensors were attached to the MSP432 GPIO pins. The radial encoder was tested using interrupts that performed a digital read on the encoders three signal pins. The FSR was tested using interrupts and an ADC function. The FSR was configured to be in a voltage divider circuit and the ADC pins connected to measure the potential across the FSR. UART communications were used to display these numbers to a terminal screen. The interrupt handler code for the radial encoder and the ADC/FSR can be seen in code snippets 3.3 and 3.4 respectively.

Code 3.3

```
void gpio_isr4(void)
{
   uint8_t val[3];
   val[0] = GPIO_getInputPinValue(GPIO_PORT_P4,GPIO_PIN0);
   val[1] = GPIO_getInputPinValue(GPIO_PORT_P4,GPIO_PIN1);
   val[2] = GPIO_getInputPinValue(GPIO_PORT_P4,GPIO_PIN2);
        uint32_t status;

        status = MAP_GPIO_getEnabledInterruptStatus(GPIO_PORT_P4);
        MAP_GPIO_clearInterruptFlag(GPIO_PORT_P4, status);
        printf(EUSCI_A0_MODULE,"\r\n\r\n");

        if(val[0]==val[1]==val[2] ){ //one step CW
                printf(EUSCI_A0_MODULE,"in neg\r\n");
                raw_position++;
         }
        else{ //one step CCW
                printf(EUSCI_A0_MODULE,"in pos\r\n");
                raw_position--;
         }
         printf(EUSCI_A0_MODULE,"%i\r\n", raw_position);
}
```

Code 3.4

```
void adc_isr(void)
{
        uint64_t status;
        status = MAP_ADC14_getEnabledInterruptStatus();
        MAP_ADC14_clearInterruptFlag(status);
        if(status & ADC_INT1)
        {
                MAP_ADC14_getMultiSequenceResult(resultsBuffer);
                a = resultsBuffer[0];
                b = resultsBuffer[1];
                printf(EUSCI_A0_MODULE,"result1: %i result1: %i",a,b );
        }
}
```

### *Data*

The radial encoder would cause an interrupt whenever it was manipulated causing a square waves to be generated. When the clockwise and reference square waves aligned the interrupt would increment a value otherwise when a counter clockwise signal was generated it would decrement this value. This was seen while testing.

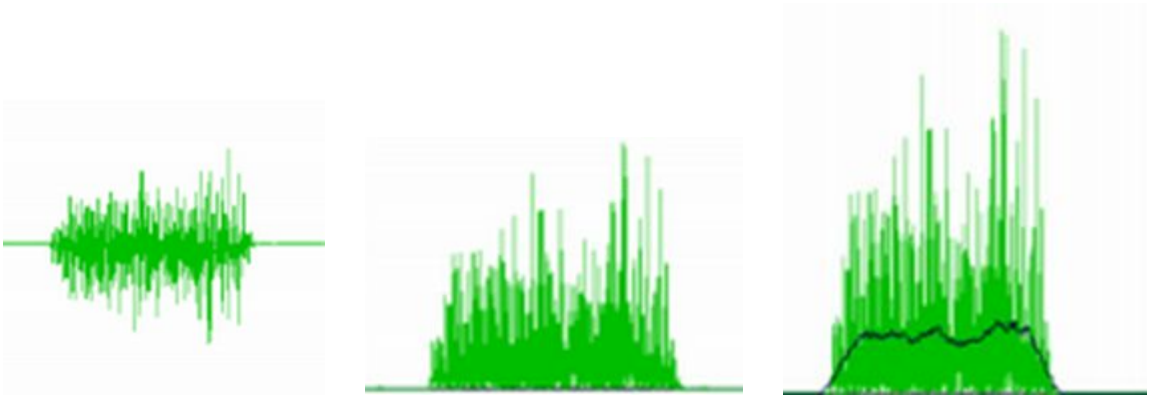
When the FSR was actuated resistance would decrease and cause less of a voltage drop across the FSR. Linearity of the FSR is not much of a factor. We are more concerned with reaching a certain setpoint with the FSR. Visible change was seen using the FSR while viewing the ADC values on the terminal screen.

*Conclusion*

The tests from the radial encoder and the FSR proved to be satisfactory. Although the FSR threshold has yet to be determined, we were able to observe the ADC values change when the FSR was actuated. The radial encoder resolution played a big factor in the tests, the higher the resolution was the more successful the test.

### 3.5.3 Data Processing Test

This test involved testing the software that rectified and performed a moving average on the inputted EMG signal from the ADS1299. The code was written in C code then converted to Matlab to test. The matlab code took in a sample of about 5 thousand points and performed the filtering. A successful test is indicated by a plotted signal that looks similar to that of plots found in EMG filtering scholarly journals. These plots can be seen through figures 3.2 to 3.4.



Figures 3.2-3.4: Filtered, Rectified, and Averaged EMG Signal [6]

*Test Setup*

First EMG data was collected and saved to an excel file. Next the C code from Code Composer was converted to Matlab code so that we could run the test. Function were added to read the EMG data and feed it into the processing portion of the code.

*Data*

The EMG Data was plotted at three points during the data processing execution. First the signal was plotted with the initial filtered data to be inputted. Next the another plot was made after rectification. Finally the moving average was plotted. These data plots can be seen in figure 3.5.

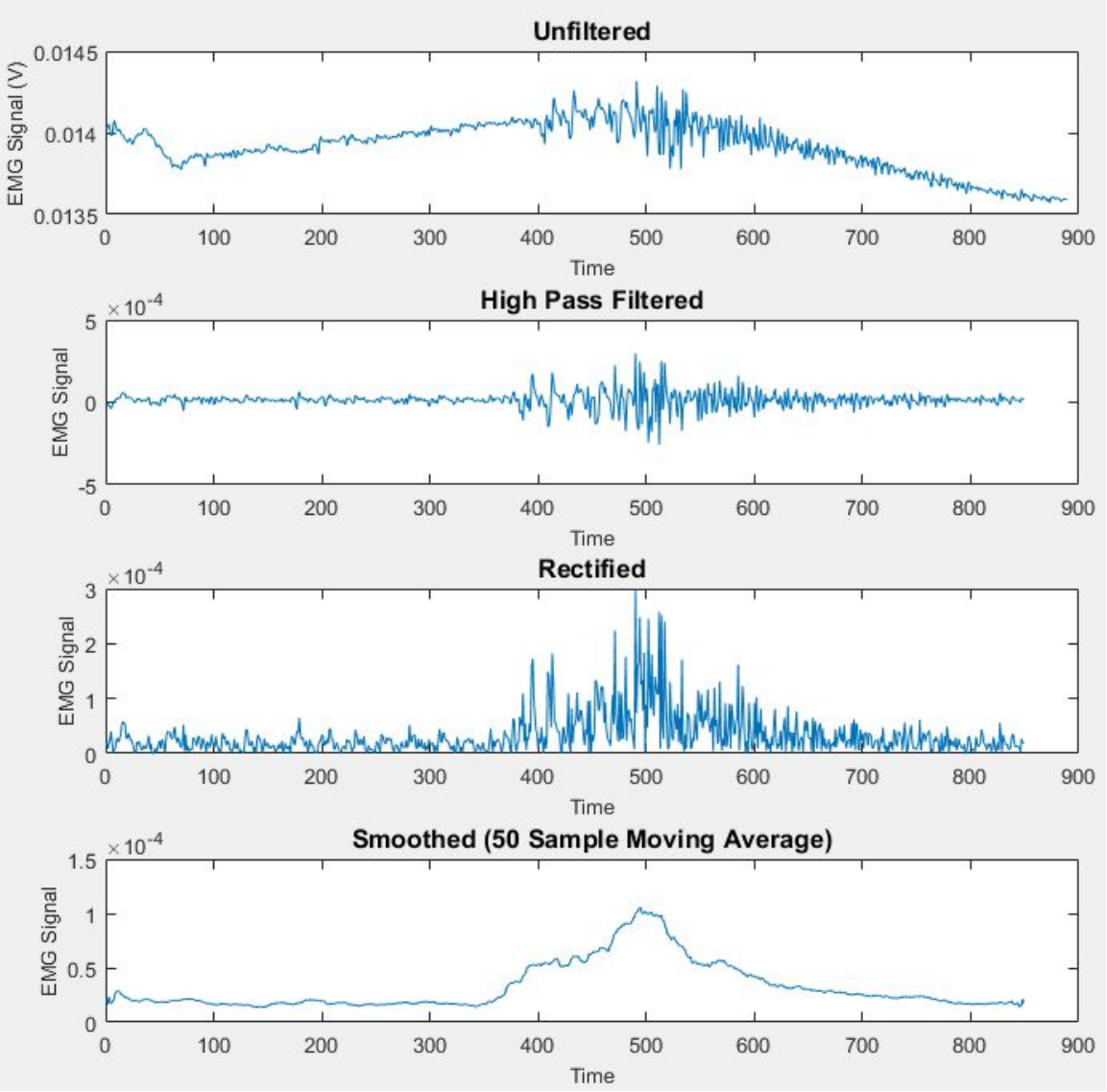



Figure 3.5: EMG Signal Processing (Matlab)

### *Conclusion*

The data processing software was tested using Matlab to get a visual representation of the filtered input signals. This method is a very accurate way of testing the data processing software because it allows for comparison of different data sets through visual representations. The test was successful because the signals are similar to other previously recorded scholarly data.

### 3.5.4 Motor Driver Test

This test is used to ensure control signals from the data processing system can actuate motor movement.The test involves the ability to move the motor with software commands, actual emg signals, and filtered variables.

*Test Setup*

The motor driver is hooked up to to the MSP432 pins. Figure 3.8 shows the signals that the motor driver will be hooked up to.

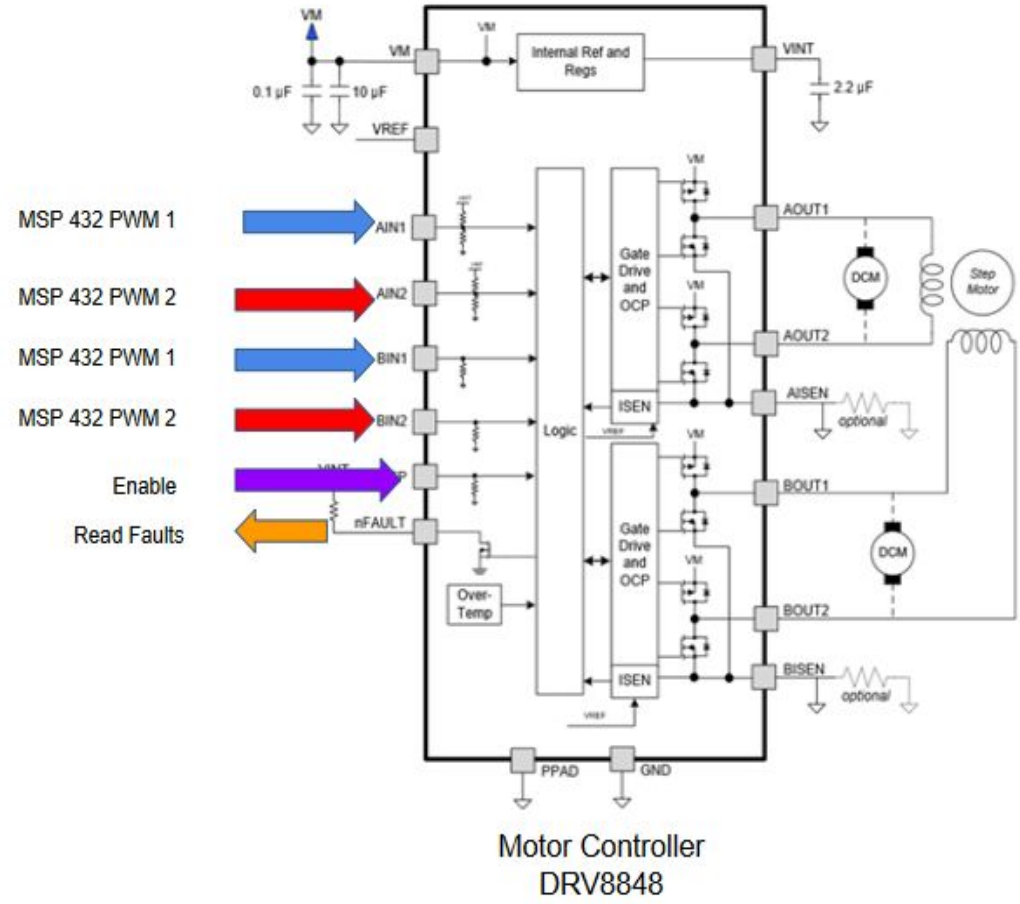



Figure 3.8 Motor Set Up

*Data*

Motor rpm will be determined based on signal sent. The first test involving software commands resulted in max rpm at 100 percent duty cycle and half of max at 50 percent. When testing is performed with emg

signal and filtered signal the duty cycle will vary. Duty cycle and rpm will be measured to verify proper signal magnitudes.

*Conclusion*

The software command method has been tested for 100 percent and 50 percent duty cycles resulting in proper motor speeds. Motor control has yet to be tested for emg signals and filtered signals but based on the software command tests, normal motor operation is expected.

# 4 Subsystem: Motor-to-Body Interface

Many specifications must be considered when choosing the best motor for an elbow orthosis. The motor must produce the largest amount of torque for its size while also not requiring a large amount of power. Power consumption is very important for this project's mobility and feasibility. Every movement must be efficient yet powerful.

Brushed DC motors with a high torque gear box are the best approach to this problem.They apply the highest degree of torque at all times, allowing for smooth elbow movements with small degrees of rotation. Due to the worm gear within the Gearbox. The motors will only move when power is being applied. Power consumption can be reduced by distinguishing intent to move versus normal, relaxed arm swinging.

With the size of the elbow orthosis being a constraint, every component must be judiciously chosen. Other subsystems use the MSP432, so an optimal and ubiquitous implementation would involve the same microcontroller. Even though the DVR8848 motor driver pairs specifically with the MSP430, the MSP432 can also recreate these reverse engineered motor signals.

## 4.1 Significant Changes in Direction

The motor-to-body interface had significant changes in the motor driver and motor selection. The elbow orthosis will be actuated using two high torque dc motors and the respective motor driver DRV8848. After calculating the amount of torque and realizing that the motor would have to move at slow speeds, we chose to use a DC motors instead. Our original designs was designed to be driven by a single motor and applying the torque from only one side of the brace, this was changed after concluding that spreading the load onto two motors would reduce the load and allow for smaller motors. Our last major change was adding a high torque gear box that will allow us to achieve a higher torque.. Gearing was already involved from the beginning but the high torque load required us to find new ways of getting the most out of our small motors.

## 4.2 Subsystem Specifications

The main purpose of the Motor-to-Body is to actuate the elbow. The subsystem is to provide more than 1N*M of torque at any given moment and hold the torque. For our demonstration we used two DC motors

and a DRV8848 motor drivers. Each motor requires .5A for maximum torque. When maxed out each one of these motors can produce 0.85 N*M. The DRV8711 requires any voltage from 8V to 25V and can produce up to 2A. With such high Amp rating we can run both motors using the same motor driver if we choose to in the future. For the DRV8848 we chose to use 12V because of the amount of variety in voltage regulators that are in the market for 12V.

Table 4.1
Motor-to-Body Specification Compliance Matrix

| Specification | Min | Nominal | Max |
|---|---|---|---|
| DRV8848 Motor Driver | 0 A | .5 A | 2 A |
| Brushless DC Motors | 0 NM | .5 NM | .83 NM |

Table 4.2
Motor-to-Body Interface Connection Matrix

| (Version) Project-Subsystem | Description | Power | (Interfaces) COM | Signal |
|---|---|---|---|---|
| V1-00 | C2000 | 3.3V | SPI / UART | GPIO |
| V1-01 | DRV8312 | 12V | UART | GPIO |
| V2-00 | MSP430 | 3.3V | SPI / UART | GPIO |
| V2-01 | DRV8711 | 12V | UART | GPIO |
| V3-00 | MSP432 | 3.3V | SPI / UART | GPIO |
| V1-00 | MSPware | — | USB C-Micro | — |
| V3-01 | DRV8848 | 12V | UART | GPIO |

## 4.3 Subsystem Status

The main purpose of the Motor-to-Body is to actuate the elbow. The subsystem is to provide more than 1N*M of torque at any given moment and hold the torque. For our demonstration we used two Brushless DC motors and a DRV8848 motor drivers. Each motor requires .5A for maximum torque. When maxed out each one of these motors can produce .83 N*M. The DRV8848 requires any voltage from 8V to 25V and can produce up to 2A. With such high Amp rating we can run both stepper motors using the same motor driver if we choose to in the future. For the DRV8848 we chose to use 12V because of the amount of variety in voltage regulators that are in the market for 12V.

## 4.4 Subsystem Technical Details

After signal acquisition and processing, the orthosis focuses on moving the elbow. Specifically, the orthosis uses a stepper motor since it can vary the speed of the shaft with a fixed voltage and current. This property allows the orthosis to use a portable power source and allows it to accurately assess battery life. Another benefit is its ability to be controlled using pulse signals, where each pulse signal moves the motor one step. This property allows precise motor control and easy determination of the angular position of the rotor [9].

The motor driver requires voltages between 8 V and 52 V to operate. However, this project's quest for a portable, yet powerful, battery constrains the voltage to a lower value. Specifically, 12V meets these requirements and is also a common voltage, providing more readily available portable power options [10].

Smooth motor transitions play a big role in the effectiveness of the Elbow Orthosis. For instance, drinking from a glass of water is difficult with discrete, "lurching" arm movements. The step angle of the motor determines the lurch of the elbow orthosis. Motors have a certain angular resolution based on the number of teeth of their stator and rotor; this orthosis motor has a resolution of 0.9 degrees. The motor driver also has the ability to reduce the step angle down to 0.1125 degrees.

While the stepper motor's "steps" establish smoothness of operation, a Proportional-Integral-Derivative (PID) loop and rotary encoder establish stable movement in attaining a position. A PID loop uses the information of the rotary encoder to stabilize itself at the desired position by minimizing set-point error. Simulations were created using the equation below, where $K_p$, $K_i$, and $K_d$ denote the coefficients for the proportional, integral, and derivative terms, respectively. The proportion term accounts for present values of the error, the integral term accounts for past values of the error, and the derivative term accounts for possible future values of the error, based on its current rate of change [11]. Figure 8 displays a standard PID loop.

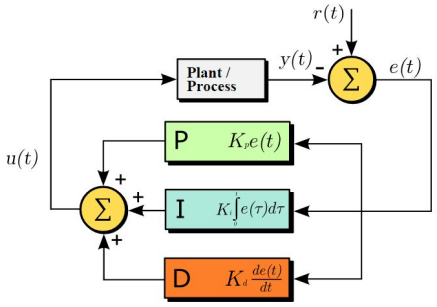



Figure 4.1: Motor Control Loop

*$K_p$ = Proportional constant ; $K_i$ = Integration constant ; $K_d$ = Differential constant ; t = time*

Figures B17 through B19 show PID responses as simulated in Matlab. An overshoot in a graph represents the elbow going past its wanted position before settling. Figure B17 shows tuning for a quick response with a small overshoot which settles slowly over time. The small overshoot is desirable, but the ripple at the end means that it takes a while for the elbow to stop moving. Figure B18 shows a bigger overshoot but a quick settling time, and Figure B19 shows a slower response time with slow settling time. The PID needs to be finely tuned to give the smallest overshoot and settling time.

The last hurdle includes safely controlling the motor movements of the orthosis--the orthosis may force the elbow into a harmful position. The orthosis applies both mechanical and electrical restrictions to prevent harm from happening. The mechanical restrictions do not allow the elbow to extend beyond 180 degrees. The electrical restrictions use information from the rotary encoder and PID loop to prevent the motor traveling beyond its limits.

## 4.5 Subsystem Testing

### 4.5.1 Proportional-Integral-Derivative (PID) Loop Test

An accurate PID loop was necessary to have a smooth moving elbow. The PID loop had to be finely tuned, for this to happen we needed to test different parameters and understand how they would affect the movements of the elbow. Correcting for error had to be done fluidly and unnoticeably. To test this we needed to send signals into the Microcontroller and manually moved the elbow. We kept an eye out for response time and any jerking motion while the elbow corrected itself. The Parameters to be tested where Voltage, RPM, and Setpoint.

*Test Setup*

The test was set up by connecting two motors and a Rotary encoder to a elbow Orthosis Simulator. The Orthosis Simulator consists of two boxes that hold both motors together in a way that their rotors can be connected together with a two headed female coupler. The Coupler is 3d Printed with a shaft that simulates the elbow Orthosis. This allows for us to reproduce the torque that will be applied to the motors. Code was created with the setpoint set to hold the stator perpendicular to the floor. By placing the Weight perpendicular to the floor we create the maximum torque possible by the weight. Once the Elbow orthosis is stable we manually move the stator position and observe how the PID loop return to the stable position. Repeat test for different positions and weight and check for fluidly, responsiveness and jerking motions.

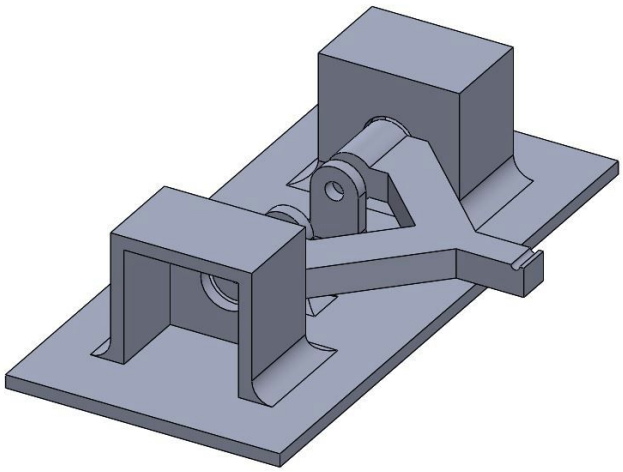


Figure 4.2: Orthosis Simulator

### *Data*

The elbow passed on responsiveness but not on fluidness. When applying more than 1 kg the motors are subjected to too much torque. The motors have a jerking motion whenever coming to a stop. This shows that the PID loop is working at a fast response rate but it is too fast for our application. Another abnormality is the speed of the Elbow when it is subjected to more than a 90 degree manual move. The motor continues to accelerate throughout the motion instead of reaching an fluid RPM that can be useful for the user.

### *Conclusion*

The Top Speed that the Elbow can reach while accelerating through the correcting motion must be set to lower than 9RPM. This will allow for a slow and fluid motion. This will also help with the jerking motion but the only solution to that will be slowing down the rpm when reaching the desired position. This will make our position graph below longer in the time axis. Changing the Step mode will also help by giving it more accurate readings and allow for us to be more precise around the setpoint of the PID loop. The highest resolution that the motors can actuate is 1/400 degrees, meaning that the user will not be able to sense the small movements.

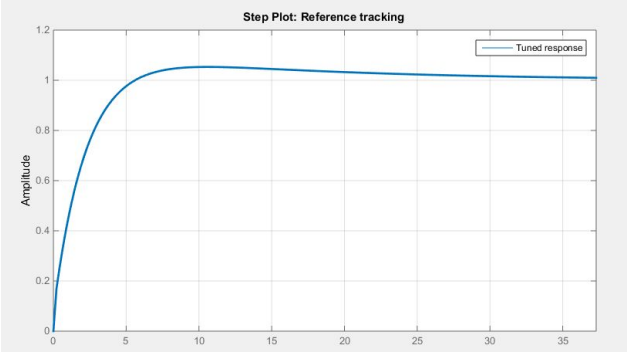



Figure 4.3: Response with: Kp= 4.907, Ki= 1.074 Kd= 0.2205

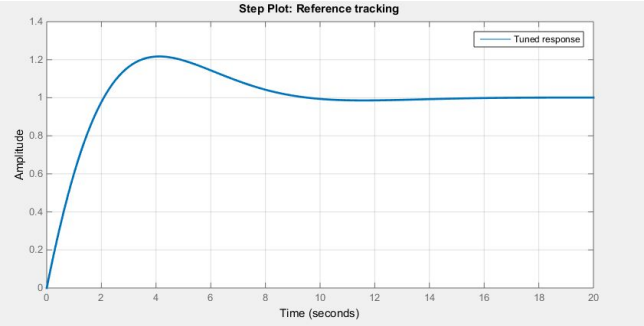



Figure 4.4: Response with: Kp= 1.645, Ki= 0.7145 Kd= 0

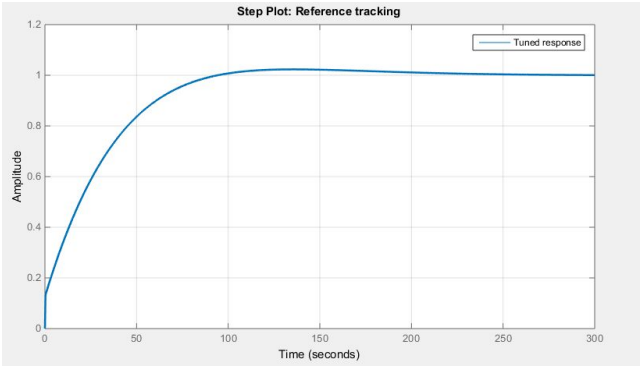


Figure 4.5: Response with: Kp= 0.11877, Ki= 0.0032163 Kd= 0.117755

### 4.5.2 Torque Test

The motors have the option of being driven by two separate motor drivers or be connected in parallel and ran of the same motor driver. Being able to use the same motor driver gives us the benefit of less connections and easier control. There were few things that had to be tested before we assume that one option was better than the other. First they have to consume the same amount of current that they would have if connected separately and they must also provide the same amount of torque. We put both of the motors through different variations and recorded the Current, Voltage and Power Consumption.

#### *Test Setup*

This test will us the Orthosis Simulator described in the previous test. The motors will both be placed within the holding positions of the Simulator and connected depending on the appropriate test. First connect both motors to individual motor drivers and max out the current by applying the highest possible holding torque to the shaft. After this, connect both motors to the same drive, insert in elbow simulator and max out the current by applying the highest possible holding torque to the shaft.

#### *Data*

Each Motor had a holding torque current of about .5A when connected separately. When both motors were driven separately the torque current would increase proportionally but when using the same motor driver they would only produce the torque that one motor would alone.

| A Motor Connected, 1 Motor Driver | |
|---|---|
| Current : | 0.49 A |
| Voltage | 12 V |
| Power | 70.56 Watts |
| Max Current Set: | 1.5 A |

| 2 Motors Connected , 1 Motor Driver | |
|---|---|
| Current | 0.5 A |
| Voltage | 12 V |
| Power | 72 Watts |
| Max Current Set: | 2.26 A |

| B Motor Connected, 1 Motor Driver | |
|---|---|
| Current : | 0.48 A |
| Voltage | 12 V |
| Power | 69.12 Watts |
| Max Current Set: | 1.5 A |

| 2 Motors Connected, 2 Motor Driver | |
|---|---|
| Current | 1.02 A |
| Voltage | 12 V |
| Power | 146.88 Watts |
| Max Current Set: | 2.26 A |

*Conclusion*

Connecting the motors together from the same motor driver allows for currents within the motors to feed back into each other and create a sort of parallel connection. This loop reduces the maximum current that the motors draw and also reduces the maximum torque that can be applied. Using only one motor driver is destructive to the max torque and also does not allow for correction if one of the motors stalls or slips. The best solution would be to use individual motor drivers for each motor. This allows for max torque and control.

# 5 Subsystem: Power Distribution

The power system is meant to take power from a rechargeable voltage source and convert it to the appropriate voltage levels for the different subsystems. To clarify, the system will take in a large voltage source, around 12 V, and then regulate the voltage for the MSP432 microcontroller, 3.3 V, used by the motor drive, data processing, and body-to-sensor interface. The body-to-sensor interface also requires a low noise voltage regulation of ±2.5 V. The system also takes into consideration the user's safety by implementing a physical kill switch, separate from any software, allowing the user to shut the system off in case of an emergency. Fuses will also be implemented to protect highly sensitive components, such as the ADS1299. Since a lithium polymer rechargeable battery will be utilized another layer of safety must also be implemented. A relay switch in conjunction with a LiPo discharge alarm, premade, will shut off the battery supply to protect the battery from overdischarge.

## 5.1 Significant Changes in Direction

Originally the system used a bench power supply to provide the power for the motors and other subsystems. Now that more components are finalized a decision has been made to use a lithium polymer hobby battery. This type of battery comes with a balance connector allowing for the testing of each individual cell's, within the battery, voltage. Using this connector a circuit, utilizing a relay and a commercially available discharge alarm, is designed to stop the battery from fully discharging during usage.

## 5.2 Subsystem Specifications

The main purpose of the subsystem is to take the large voltage of the main power source and regulate so that it is usable by the MSP432, ADS1299, and motor drive. The devices in Table 5.1 are used to power the circuit. The DROK buck converter takes in 12 V, current source voltage for motors, and regulates it down to 5 V at .01 A. The LM1117, a linear voltage regulator, takes this 5 volts and outputs 3.3 V which is used by the microcontrollers and the rest of the regulators, excluding the TPS723. The rest of the regulators, LP59 through TPS723, output voltages that are required by the ADS1299 and specified in its data sheet. These last 4 regulators are also rated for low noise which is required to improve signal clarity with the ADS1299.

Table 5.1
Power System Specification Compliance Matrix

| Specification | Input (V) | Min (V) | Output (V) | Max (V) |
|---|---|---|---|---|
| DROK Buck Converter | 12 | | 5 | |
| LM1117IMP-3.3/NOPB | 5 | 3.267 | 3.3 | 3.333 |
| LP5907MFX-2.5/NOPB | 3.3 | | 2.5 | |
| LP5907MFX-3.3/NOPB | 3.3 | | 3.3 | |
| LM2663M/NOPB | 3.3 | -3.267 | -3.3 | -3.333 |
| TPS72325DBVT | -3.3 | -2.45 | -2.5 | 2.55 |

## 5.3 Subsystem Status

The power subsystem, which converts the power from the battery for use by the MSP432 and ADS1299, has been tested and approved by the rest of the team and our Texas Instruments mentor. Once all components are received, from their retailers, the components will be soldered onto a PCB board designed by David Cuevas, motor-to-body interface. Once soldered they will be tested once again to make sure all the voltages and currents are still suitable for the ADS1299 and MSP432.

## 5.4 Subsystem Technical Details

The large battery voltage accommodates the most physically demanding component of the orthosis, namely the high torque (1 N • m) motor. It also reduces the need for a series of different voltage converters, namely boost converters in addition to buck converters. For initial development and testing, a bench power supply helps isolate problems within each individual subsystem, without the looming concern of a battery or power distribution failure. Once all systems work properly with this regulated and reliable bench power, the power distribution system conforms to a new design with only a single voltage supply. The mobile design includes a series of regulators, filters, and converters.

The DC to DC buck converter is used over the linear voltage regulator, another device that reduces large voltages, due to the fact that the buck converter is much more efficient. A linear voltage regulator's output voltage is dictated by both the load resistance and the load current (1). This load current is controlled by a transistor with a modifiable base current. However, problems come from the difference between the output voltage and source voltage. Any voltage not used in the load by the output is absorbed by transistor, across its collector and emitter, which results in power loss according to (2). This equation means that as the difference between the source and output voltage increases, the power loss also increases. This constrains the linear converter to low voltage systems. The linear voltage regulator dissipates this excess power as heat which leads to a lower efficiency and lower battery life [7]. Also, this increased heat output can be bothersome or problematic for the user.

$$V_o = R_L I_L \quad (1), \qquad P_{CE} = V_{CE} I_L \qquad (2)$$

A standard buck converter model is shown in Figure 5.1a. Standard switching converters include a resistor, voltage source, and transistor switch. The switch periodically opens and closes according to a designer's specified duty ratio, D, which depends on the designer specified switching frequency (3). These equations result in a voltage waveform, as seen in Figure 5.1b, which is much too noisy for a DC circuit [12]. The right half of the circuit (inductor and capacitor, excluding the resistor) act like a LC low pass filter. When the switch is closed a current in the inductor rises linearly but the inductor resists the change in current, producing a voltage proportional to the rate of change, which then results in a voltage drop. When the switch is opened, the charge stored in the inductor then powers the circuit. The diode completes the circuit without requiring a voltage source, and the switch closes before the inductor discharges to maintain a steady voltage [13]. Ideally, this process produces output voltages as dictated by (4).

Buck converters are more efficient than linear converters since the power absorbed by the diode, according to (5), determines how much power is lost by the converter. However, the buck converter produces more noise than the linear converter, so a bank of capacitors at the subsystems' positive ends reduce this effect. The device data sheets aid in deciding the noise-cancelling capacitors.

$$D = t_{on} f_{switching} \quad (3), \quad V_o = V_S D \quad (4), \quad P_D = V_D (1 - D) I_o \quad (5)$$

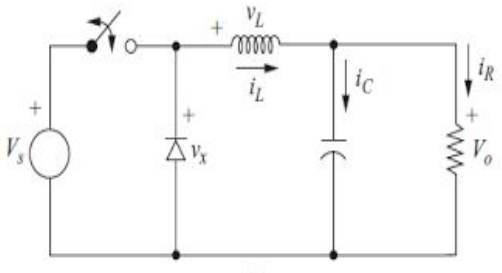



Figure 5.1a

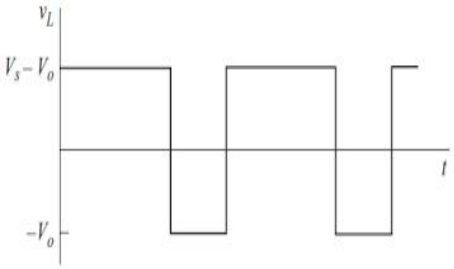



Figure 5.1b

Figures 5.2 and 5.3 depict a PSpice simulation of a standard buck converter. This simulation represents a simple theoretical converter, and the purchased model includes additional circuitry which provides protection against heat and surge damage. The converter reduces the initial 12 V input to a 5 V output which will be further lowered, through the use of linear voltage regulators, to match the needed voltage of the different circuit components. The load resistor in the actual circuit substitutes a potentiometer to more finely tune the output voltage. Figure 5.3 shows the voltage before the inductor where an internal switch induces a ripple, from around -720 mV up 12 V, which closely matches the waveform of Figure 5.1b. The

inductor then stabilizes the voltage around 5 V which both charges the capacitor and serves as the voltage across the resistor.

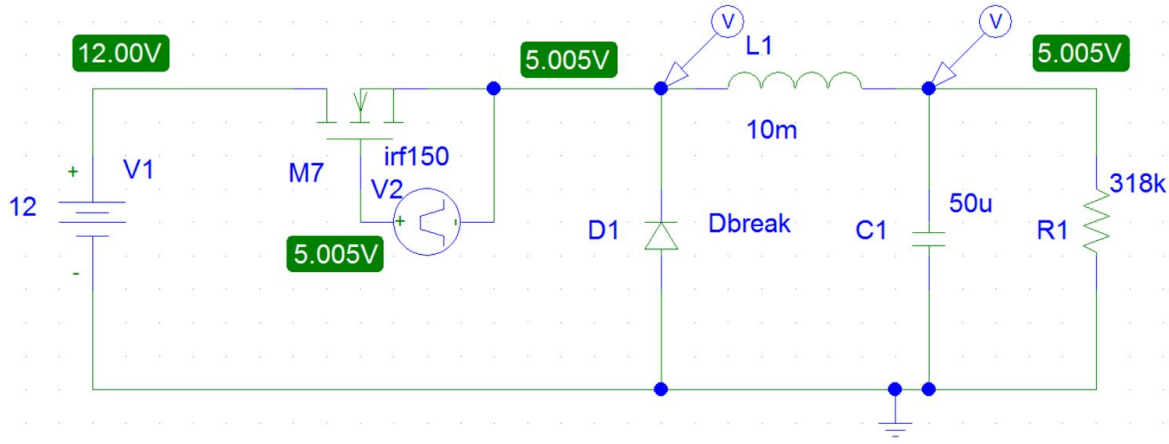



Figure 5.2: PSPICE Buck Converter Simulation

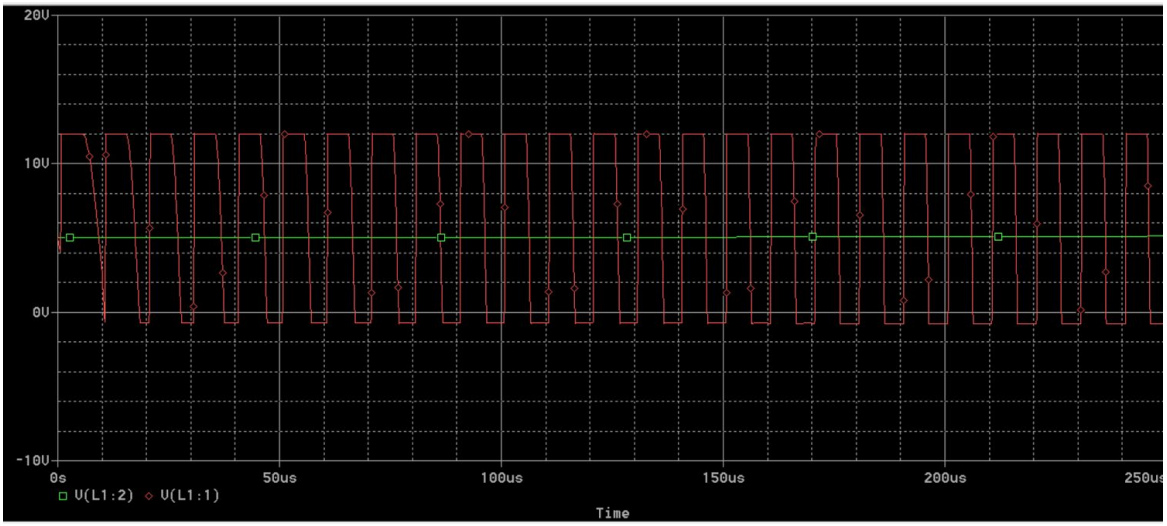



Figure 5.3: PSpice Buck Converter Voltage Waveform

## 5.5 Subsystem Testing

### 5.5.1 Power System Voltage Test

The purpose of this test was to make sure all the components listed in Table 5.2 are outputting the appropriate voltage. The circuit was breadboarded with the buck converter taking bench power and sending it to the ADS1299 power circuit. The components had the voltages at the output capacitors tested with a digital multimeter to make sure they were outputting at or near their ideal values.

Table 5.2: Power Distribution Components

| Symbol | Value | Description |
|---|---|---|
| LM1117 | | Voltage Regulator 5 V to 3.3 V |
| LM2663 | | Voltage Regulator 3.3 V to -3.3 V |
| LP590725 | | Voltage Regulator 3.3 V to 2.5 V |
| LP590733 | | Voltage Regulator 3.3 V to 3.3 V |
| TPS72325 | | Voltage Regulator -3.3 V to -2.5 V |
| PDC1 | 10 uF | Capacitor |
| PDC2 | 100 uF | Capacitor |
| BYP2,PDC7 | 47 uF | Capacitor |
| PDC3,PDC4,PDC5,PDC6 | 1 uF | Capacitor |
| PDC8 | 2.2 uF | Capacitor |
| BYP1 | .01 uF | Capacitor |
| Power Plug | | Takes in power from buck converter |

### *Test Setup*

The DROK converter was set on a small breadboard with 2 inputs connected to a bench power supply providing 12 Volts at 500 mA. The outputs of the converter went to a larger breadboard which had the linear regulators arranged as shown in Figure 5.4. The LM1117 takes in the voltage from the buck converter and redistributes it to the other regulators. The components were first all tested individually to ensure that they were functioning correctly and that none of the soldered jumpers were causing shorts. This was done by disconnecting a component from the circuit and then powering the input,with the appropriate voltage and current, with the power supply and then testing the output voltage with a multimeter. Once all components were tested the whole system was then connected, see Figure 5.4, and then each of the outputs were tested to see if they matched the ideal outputs. To test that the voltage and current coming from the LM1117 was enough to power a our microcontroller the output from this component was connected to a MSP430 with a generic sample code. This program allowed the LED of the microcontroller to be turned off when it's button is pressed.

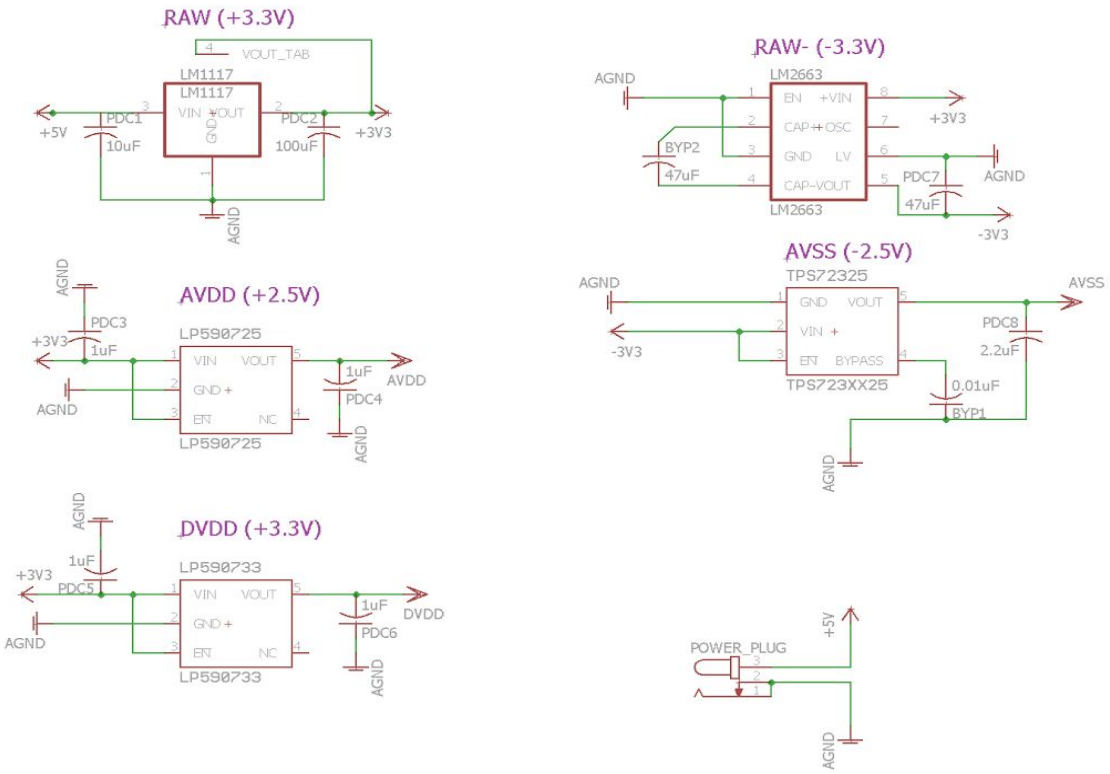



Figure 5.4: Power Distribution Schematic

### *Data*

The actual voltages and currents received are shown in Table 5.3.The real outputs were slightly lower than the voltages noted in the voltage regulators datasheets. Also the LP5907MFX-2.5/NOPB was outputting 2.8 V instead of 2.5 V. The program on the microcontroller executed correctly allowing us to switch the LED on and off at will.

Table 5.3
Power System Test Output

| Specification | Input (V) | Input (mA) | Actual output (V) | Actual output (mA) | Ideal output (V) |
|---|---|---|---|---|---|
| DROK Buck Converter | 12 | 500 | 5.01 | 31 | 5 |
| LM1117IMP-3.3/NOPB | 5.01 | 31 | 3.26 | 31 | 3.3 |
| LP5907MFX-2.5/NOPB | 3.26 | 31 | 2.85 | 31 | 2.5 |
| LP5907MFX-3.3/NOPB | 3.26 | 31 | 3.27 | 31 | 3.3 |
| LM2663M/NOPB | 3.26 | 31 | -3.27 | 31 | -3.3 |

| | | | | | | |
|---|---|---|---|---|---|---|
| TPS72325DBVT | -3.27 | 31 | -2.51 | 31 | -2.5 | |

*Conclusion*

The circuit as a whole was outputting, with exception of the first LP59, near the ideal voltages necessary for the ADS1299. The reason for the incorrect output of the first LP59 was due to an acquisitional error where the LP5907MFX-2.8/NOPB was received instead of the required LP5907MFX-2.5/NOPB. An appropriate regulator will be ordered to replace the incorrect one but the overall system still worked as planned. The actual output voltages were much lower than the ideal voltages labeled in the regulators datasheets. The lower output voltage can be attributed to internal resistances and voltage drops of MOSFETs and OpAmp circuits amongst others. The currents also stayed relatively low, fluctuating between 31 and 32 mA, which is well under the 100 mA max input current for the ADS1299. The microcontroller also functioned properly indicating the current and voltage was sufficient. This could however change when the full data processing program is executed.

### 5.5.2 Battery Discharge Cutoff Test

The purpose of this test is to make sure the discharge relay switch shutoffs the battery when the voltage of any cell within the battery gets below 3.6 volts. The normal fully charged capacity of a battery cell is around 3.7 volts. This small difference between fully charged and shut off allows us to test the circuit multiple times without having to fully charging and discharging the battery every time a test is performed. Once testing is complete this shutoff value of 3.6 volts can be changed for full time use.

*Test Setup*

For testing the circuit in Figure 5.5 is constructed on a breadboard and placed in will be powered by a LiPo battery with a balance connector. A LiPo discharge alarm, off the shelf, is used with a latching relay, DSP1a-L2-DC5V, along with a capacitor and with to create the circuit. Once the circuit is constructed it is attached to a fully charged LiPo battery. The switch is pressed putting the relay into a closed position allowing voltage to flow through. A multimeter is used to verify that voltage is being outputted from the battery. The alarm is set to go off at 3.6 volts which is only .1 volts lower than the cells fully charged capacity. Once the voltage reaches that level the relay should switch resulting in no more voltage output.

*Conclusion*

This test is still awaiting to be completed as some parts were ordered but have yet to be received.

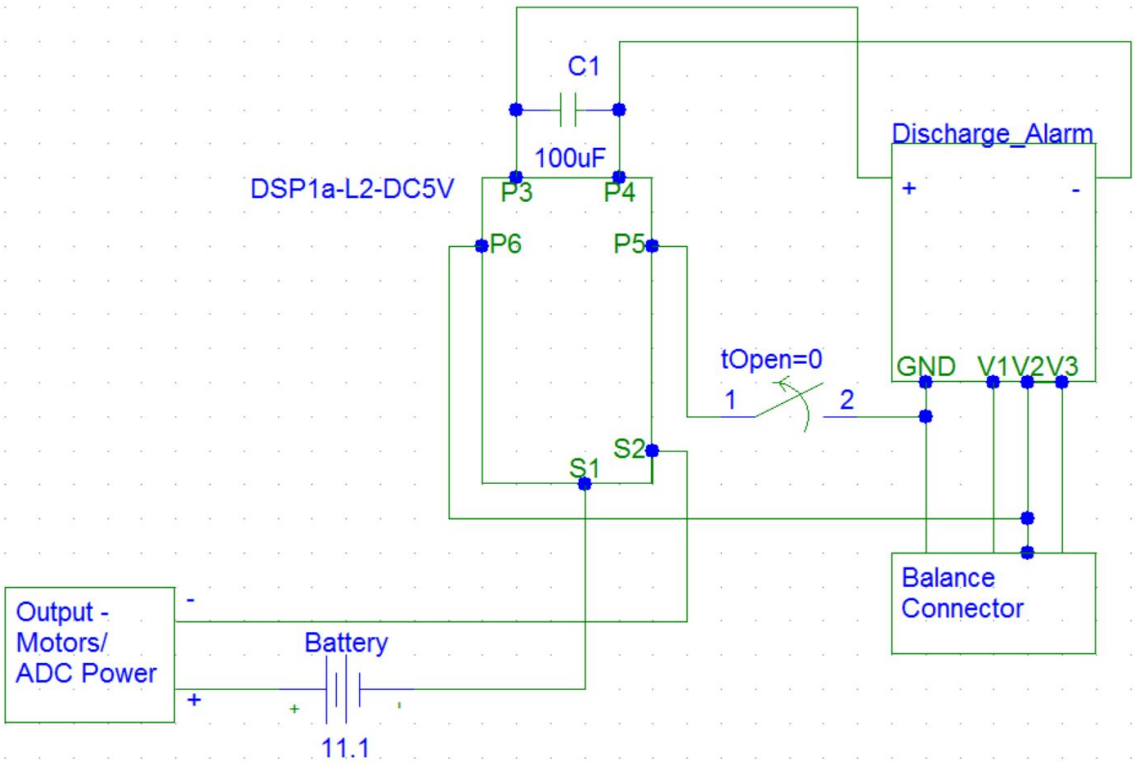



Figure 5.5 Discharge Disconnect Circuit

# 6 Subsystem: User Interface

The purpose of the user interface is to allow the user of the orthosis to both calibrate and set limits for the unit. The current system uses an Android app, created in Android Studio, along with a bluetooth module to communicate with the MSP432. The app allows the user to visually move through orthosis options and then use buttons on the screen to set limits and calibration. These limits are angles being read from the motor drive subsystem's radial encoder and the user sets a max and minimum angle so the orthosis never forces the forearm into uncomfortable positions. The calibration setting has the user flex which allows the data processing and body-to-sensor interface subsystems to read an average of the user's bicep and triceps electrical activity. This calibration makes sure that the system reacts appropriately to different users.

## 6.1 Significant Changes in Direction

At first the system had a 16 by 2 LCD screen and multiple pushbuttons, wired directly to the MSP432, as the user interface. The only problem is the system would be attached to the orthosis at the elbow which could be hard to reach for some users. For this reason we explored different options one being the Texas Instruments Chronos which utilizes a version of the MSP430 and can be programed with Code Composer, which is being used in programing data processing and the motor drive. The Chronos was also originally intended to communicate wirelessly with the MSP432 of the Data Processing subsystem through the use of

an RF booster pack, the CC110L. This proved challenging as it seems that the RF booster and the Chronos do not want to communicate. Information from online sources and students from other senior design projects informed us that while getting the two to communicate was possible our time would be better spent investing in a different communication option.

For this reason we have shifted towards the development and use of an Android app in conjunction with a bluetooth module, the FC114, attached to the MSP432's UART connection. Unlike the TI Chronos, Android app development has plenty of resources online to aid in achieving the final design of communication between a user interface system and the MSP432.

## 6.2 Subsystem Specifications

The main component of the subsystem is the Android app being run on any android phone running KitKat 4.4 or better. The app is being developed through Android Studio a free to use app development tool from Google. Currently the system has three functions with the first being establishing a bluetooth connection with an MSP432. The other two option are the limit set and calibration options. The module being used for bluetooth is an FC114 bluetooth module.

## 6.3 Subsystem Status

The system is currently still in development with bluetooth connectivity with the MSP432 being the first priority. The menu options and different instructional prompts have been created. Once connection has been established work will begin, in conjunction with the Data Processing subsystem, on integrating the app with the emg sensors and radial encoder.

## 6.4 Subsystem Technical Details

The main purpose of this interface is to provide a calibration routine that standardizes the muscle signals of each user. Upon pressing a button on the interface, the user begins his or her maximum voluntary contraction. The microcontroller initializes a data collection routine with this button push, collects muscle signals over a short period of time, and subsequently normalizes all future muscle signals to this maximum muscle activity value.

Additionally, the user may want to define maximum angle limits for the orthosis. Unlike a spring-loaded mechanical goniometer, the orthosis uses electromechanical motor resistance to discourage harmful movements. Through interaction with the User Interface, users may dynamically customize these limits.

Initially, the main microcontroller establishes the serial connection with the peripheral watch device at a 9600 BAUD rate. Depressing the orthosis calibration button on the app initiates a signal collection routine for the user's maximum voluntary contraction. Specifically, the button push prompts the watch microcontroller to transmit logic bits to the main microcontroller through an UART connection. These bits then drive logic within the microcontroller--when the user depresses the button, the main microcontroller stores a period of high activity EMG signal data. All future signals are normalized to this sample data. The electromechanical limit setting follows a similar procedure. The watch screen displays the current settings for the upper and lower elbow angle limits. Buttons on the app allow the user to manipulate these settings.

## 6.5 Subsystem Testing

### 6.5.1 Communication Test

The purpose of this test is to show that the App is properly communicating with an MSP430 that had a bluetooth module attached. The test had a user go to the bluetooth menu option and hit an in the option that should allow the user to connect to the MSP432. Once connected another button in the bluetooth menu is a blink option that when pressed should make an LED on the MSP432 blink. The blinking of an LED indicates a successful connection.

#### *Test Setup*

First the code is compiled in Android studio and is then uploaded to a smartphone, in this case an Xperia Z3v. The MSP432 is then turned on to ensure that it's bluetooth module is active and the code to test the LED blink is compiled onto the board. The user will then open the app and the bluetooth menu option is selected. Once the MSP432 appears on the device the option to connect will be activated. Then another button in the bluetooth menu, labelled blink, is pressed which should cause an LED on the microcontroller to blink

#### *Conclusion*

This test has yet to be complete. The development is ongoing to establish bluetooth connectivity between the MSP432 and the Android app.

# 7 Conclusions

In conclusion, the powered, programmable elbow orthosis stabilizes, limits, and assists the movements of a user's elbow. Upon introducing the proper electrical muscle signal to the electrodes of the elbow orthosis, the orthosis subsequently filters, converts, communicates, and analyzes the collected data to discern intended movement. After processing intention, the orthosis pulses a signal to drive the motor. Throughout the process, rotary encoders and force sensitive resistors ensure this stability and safety. Furthermore, a battery efficiently supplies portable and long-lasting system power. A user interface allows for dynamic display and manipulation of desired settings.

The Sensor-to-Body Interface collects a variety of signals relating to current state and future intention. Specifically, this interface intercepts muscular biopotential signals with EMG sensors, transmits angular position with the rotary encoder, and registers harmful forces with the force sensitive resistors. While the rotary encoder produces digital voltages "ticks", the force sensitive resistors and EMG sensors produce continuous analog voltages. Harmful forces information passes to the microcontroller through a simple analog-to-digital voltage conversion. However, the EMG sensors require significant filtering. For this portion of the interface, unity gain buffers isolate the signal paths, analog bandpass filters ignore the noise outside of the relevant frequency band, the instrumentation amplifiers of the ADS1299 reject common mode noises, and the ADS1299 converts the analog voltages to high resolution digital values. Finally, this information enters the MSP432 main processor through a standard SPI bit stream. The orthosis' Data Processing subsystem accomplishes the task of converting EMG sensor data to useful motor control variables. Along with this vital task this subsystem keeps track of motor positioning and enforces electrical stop limits.

For the Motor-to-Body Interface, the microcontroller relays the logic received to the motor driver and begins the PID loop. The motor driver amplifies the signals using an external power source to drive the stepper motor. The microcontroller monitors feedback from the motor using the rotary encoder to calculate set-point error and accordingly adjusts the motor signals. For the Power Subsystem, through the use of a buck converter, the orthosis system uses a high voltage battery without the need of a boost converter or loss of battery life. The voltage after the buck converter is at a low enough voltage and current where linear regulators may power the microcontrollers without excessive power loss. The User Interface allows an individual to adjust the orthosis to his or her specific needs while also providing easily activated safety precautions. The TI chronos, interfaced with the MSP432, allows a user to easily reset and adjust tolerance values so that the device never becomes uncomfortable for the user. A toggle switch directly intercepts the voltage supply at the source and allows the user to quickly deactivate the device.

The power subsystem utilizes an 11.1 volt lithium polymer rechargeable battery to properly power the motors, MSP432 and ADS1299 of the other subsystems. Through the use of an off the shelf discharge alarm and a latching relay the battery is able to run without the need of physically checking if the system is discharging past a limit that would damage the battery and in turn possibly damage the user. Multiple linear regulators and a buck converter provide the appropriate low noise voltages required by the ADS1299 and MSP432. The buck converter also improves the efficiency of the system by preventing any power loss through heat that usually occurs when a linear regulator is used at high power settings.Finally the physical power switch allows the user to quickly shut off the entire system incase of any failure regardless if the source is digital or analog.

Utilizing an Android app, developed in Android Studio, for the User Interface the user is able to calibrate and set physical limits for the orthosis. Through the use of the systems's emg sensors the user is able to calibrate the system to their needs based on the strength of their bicep while flexing. This allows the system to adjust in sensitivity depending on the strength or weakness of the user. The limit setting allows the user to move their arm to positions that would cause great discomfort and then having the system record that value with the aid of a radial encoder. This ensures that the user will feel no unnecessary pain or discomfort from an instance where the orthosis would try push their forearm to angles their body can't handle. The main purpose is to allow customizability between different users and enhance their experience with the device.

# References


[1] M. B. I. Reaz, M. S. Hussain, and F. Mohd-Yasin (2006). Techniques of EMG signal analysis: detection, processing, classification and applications. Retrieved October 19, 2015, from http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1455479

[2] M. Al-Faiz and A. Miry (2015). Artificial Human Arm Driven by EMG Signal. Retrieved October 19, 2015, from http://www.cdn.intechopen.com/pdfs-wm/39325.pdf

[3] Konrad, P. (2006). The ABC of EMG: A Practical Introduction to Kinesiological Electromyography. Retrieved October 19, 2015, from http://www.noraxon.com/wp-content/uploads/2014/12/ABC-EMG-ISBN.pdf

[4] F. Ogce and H. Ozyalcin (2000, December 24). A Myoelectrically Controlled Shoulder-Elbow orthosis for Unrecovered Brachial Plexus Injury. Retrieved October 19, 2015, from http://www.ncbi.nlm.nih.gov/pubmed/11195363

[5] N. Vitiello, T. Lenzi, S. Roccella, S. M. M. De Rossi, E. Cattin, F. Giovacchini, F. Vecchi, and M. C. Carrozza (2013, February). NEUROExos: A Powered Elbow Exoskeleton for Physical Rehabilitation. Retrieved November 8, 2015, from http://www.ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6304936

[6] Motor Neuron Diseases Fact Sheet. (2015, July 15). Retrieved October, 19 2015, from http://www.ninds.nih.gov/disorders/motor_neuron_diseases/detail_motor_neuron_diseases.htm

[7] Keeping, Steven. "Understanding the Advantages and Disadvantages of Linear Regulators." Understanding the Advantages and Disadvantages of Linear Regulators. Digi Key, 5 Aug. 2012. Web. 29 Oct. 2015. <http://www.digikey.com/en/articles/techzone/2012/may/understanding-the-advantages-and-disadvantages-of-linear-regulators>.

[8] NSPE (2007). Code of Ethics. Retrieved November 10, 2015, from http://www.nspe.org/resources/ethics/code-ethics#sthash.FcMtfqQt.dpuf

[9] Honeywell, D. (n.d.). PID Control. Retrieved November 23, 2015, from http://www.cds.caltech.edu/~murray/books/AM08/pdf/am06-pid_16Sep06.pdf

[10] How to choose a Portable Battery/PowerPack. (n.d.). Retrieved November 23, 2015, from http://www.ecogeekliving.com/how-to-choose-a-portable-battery-powerpack.html

[11] Stepper Motors. (n.d.). Retrieved November 23, 2015, from http://www.circuitspecialists.com/stepper-motor

[12] Hart, Daniel W. "Ch. 6 DC-DC Converters." Power Electronics. New York: McGraw-Hill, 2011. N. pag. Print.

[13] "Buck Converters." Learnabout Electronics, n.d. Web. 22 Nov. 2015. <http://www.learnabout-electronics.org/PSU/psu31.php>.

# Appendix A Budget Table

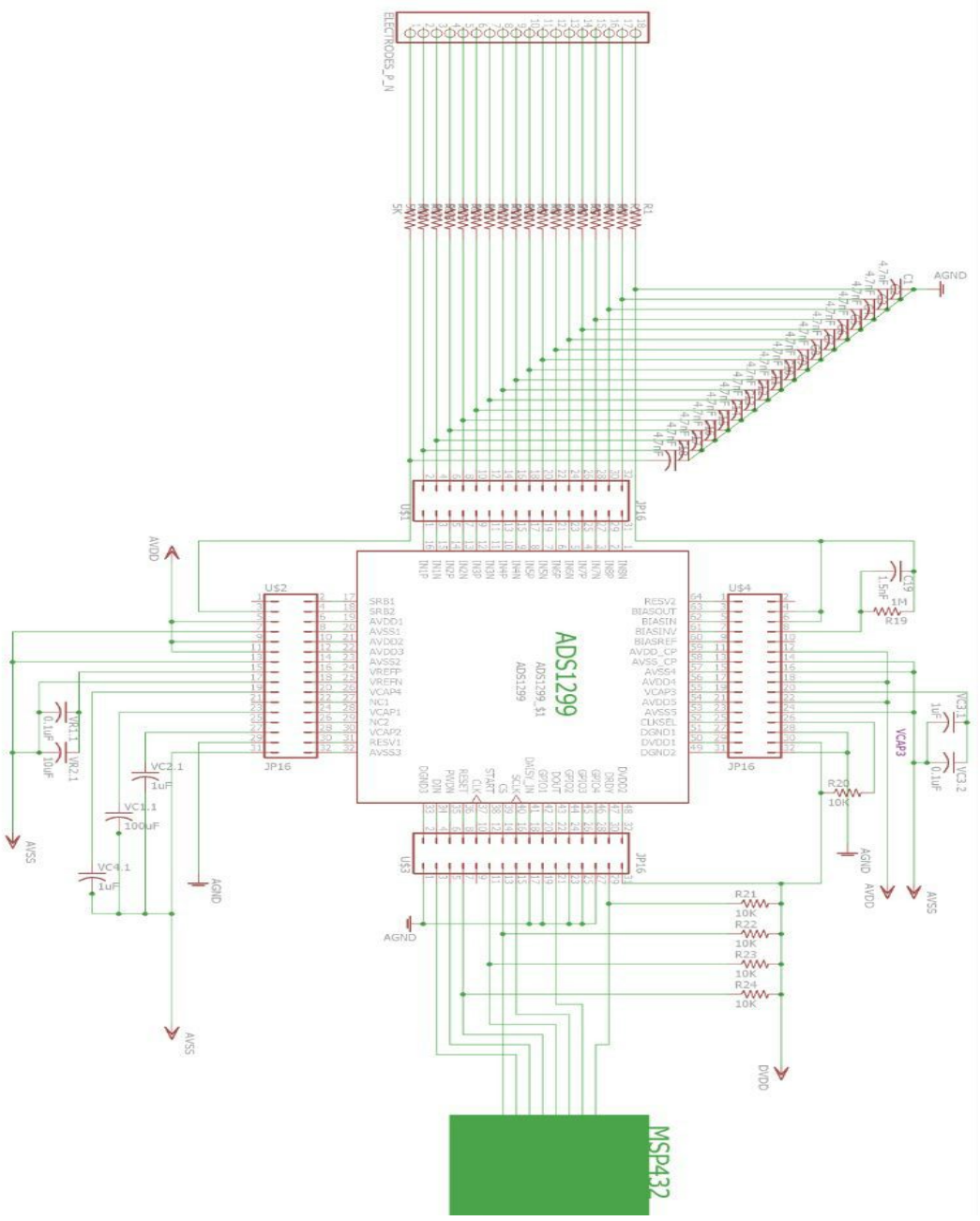
Table A-1: Project Budget

| Part - Description (Manufacturer) | Quantity | Item Cost | Subtotal |
|---|---|---|---|
| Lead Wires for TENS/EMS Machine (KONMED) | 8 | 7.99 | 7.99 |
| FS-TB-30 - EKG ECG Foam Electrodes (Skintact) | 30 | 8.85 | 8.85 |
| LM2596 - DC-DC Buck Converter (DROK) | 1 | 4.99 | 4.99 |
| LMP7721MA - High Precision Amplifier (TI) | 2 | 11.84 | - |
| LM1117IMP-3.3/NOPB - 800mA Linear Regulator (TI) | 1 | 1.65 | - |
| LM2663M/NOPB - Capacitor Voltage Converter (TI) | 1 | 2.52 | - |
| TPS72325DBVT - High Reliability Regulator (TI) | 1 | 3.13 | - |
| LP5907MFX-3.3/NOPB - 250mA Linear Regulator (TI) | 1 | .56 | - |
| LP907MFX-2.85/NOPB - 250mA Linear Regulator (TI) | 1 | .56 | - |
| LAUNCHXL-F28027 - C2000 LaunchPad (TI) | 3 | 17.70 | - |
| BOOSTXL-DRV8301 - Motor Driver (TI) | 3 | 49.00 | - |
| ADS1299 EVM - Precision Biopotential ADC (TI) | 1 | 199.00 | - |
| EZ430-CHRONOS-915 - RF Smartwatch (TI) | 2 | 58.00 | - |
| MSP432P401R - MSP432 “Falcon” (TI) | 2 | 12.99 | - |
| eZ430-RF2500 - Wireless Development Tool (TI) | 2 | 120.00 | - |
| 430BOOST-CC110L - RF Boosterpack (TI) | 2 | 19.00 | 19.00 |
| | | | 40.33 |

( - ) Indicates No Charge

# Appendix B Body-to-Sensor Interface Subsystem

## B-1 **ADS1299 Breakout Schematic**



## B-2 ADS1299 Bill of Materials

| Symbol | Value | Description |
|---|---|---|
| R1-R18 | 5.1 kΩ | RESISTOR, 1/10W 1% 0603 |
| C1-C18 | 4.7 nF | CAPACITOR, CER 4700pF 50V 10% X7R 0603 |
| JP1-JP4 | 32-pin | 100mil, 2 row JUMPER |
| P1 | 16-pin | 100mil, 1 row PIN HEADER |
| VC3.1 | 1 uF | CAPACITOR, CER 1uF 50V 10% X5R 0603 |
| VC3.2 | 0.1 uF | CAPACITOR, CER 0.1uF 50V 10% X7R 0603 |
| VC2.1 | 1 uF | CAPACITOR, CER 1uF 50V 10% X5R 0603 |
| VC1.1 | 100 uF | CAPACITOR, CER 100uF 6.3V 20% X5R 1206 |
| VC4.1 | 1 uF | CAPACITOR, CER 1uF 50V 10% X5R 0603 |
| VR1.1 | 0.1 uF | CAPACITOR, CER 0.1uF 50V 10% X7R 0603 |
| VR2.1 | 10 uF | CAPACITOR, CER 10uF 10V 20% X5R 0603 |
| C19 | 1.5 nF | CAPACITOR, CER 1500PF 50V 5% NP0 0603 |
| R19 | 1 MΩ | RESISTOR, 1/10W 5% 0603 |
| R20-R24 | 10 Ω | RESISTOR, 1/10W 5% 0603 |

## B-3 Supporting Figures

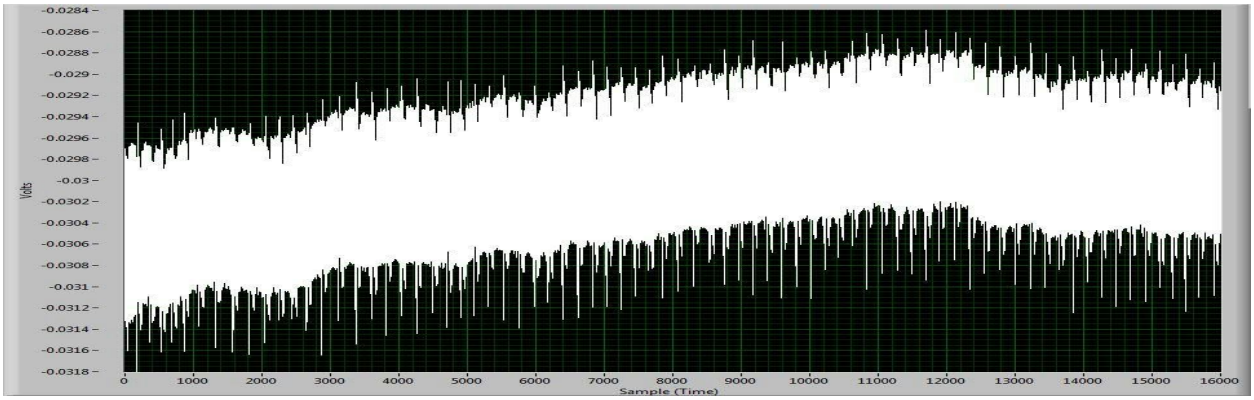



Figure B1: Time-Dependent, Unfiltered, Resting EMG

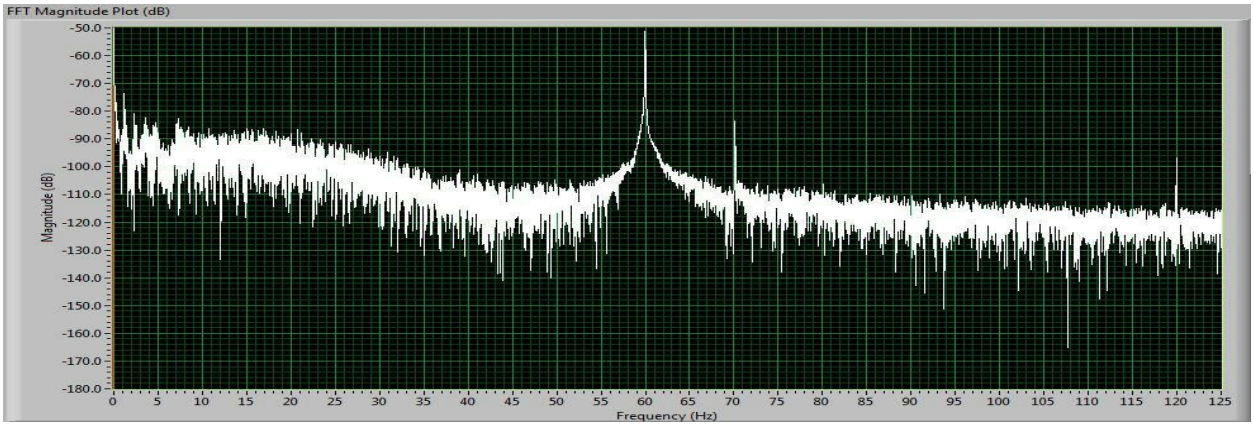



Figure B2: Frequency-Dependent, Unfiltered, Resting EMG

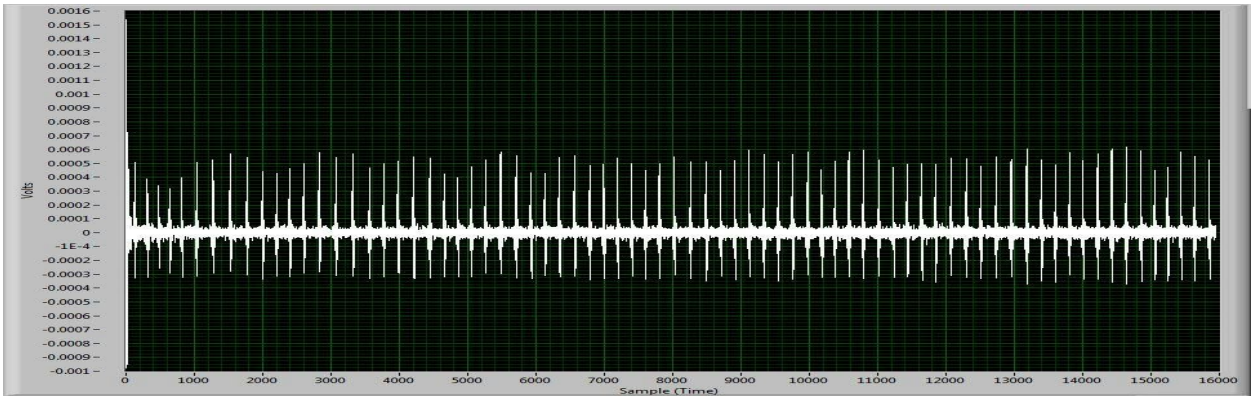



Figure B3: Time-Dependent, Filtered (10 Hz High Pass, 60 Hz Notch), Resting EMG

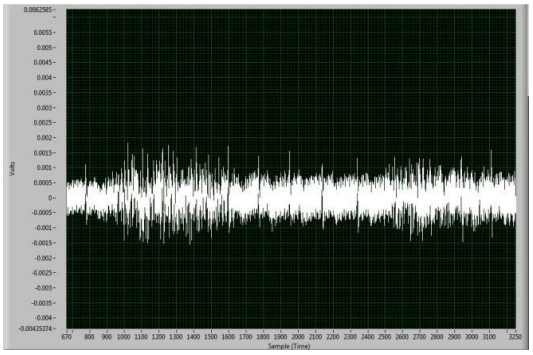


Figure B4: Unfiltered, Pulsed EMG Signal

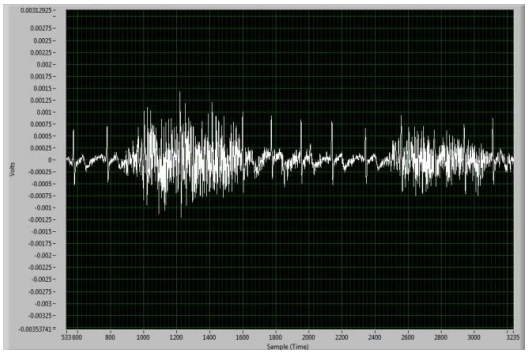


Figure B5: Filtered, Pulsed EMG Signal

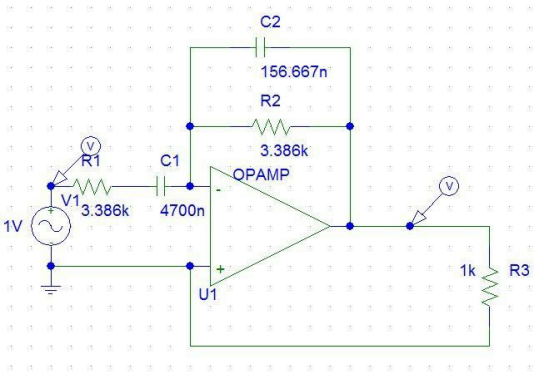



Figure B6: Unity-Gain, Band-Pass Filter

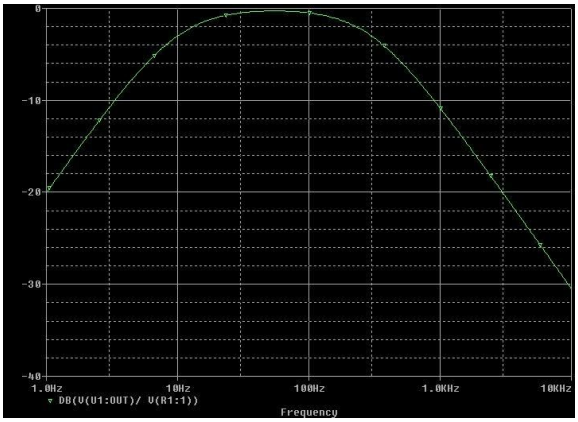



Figure B7: Frequency Response of A6

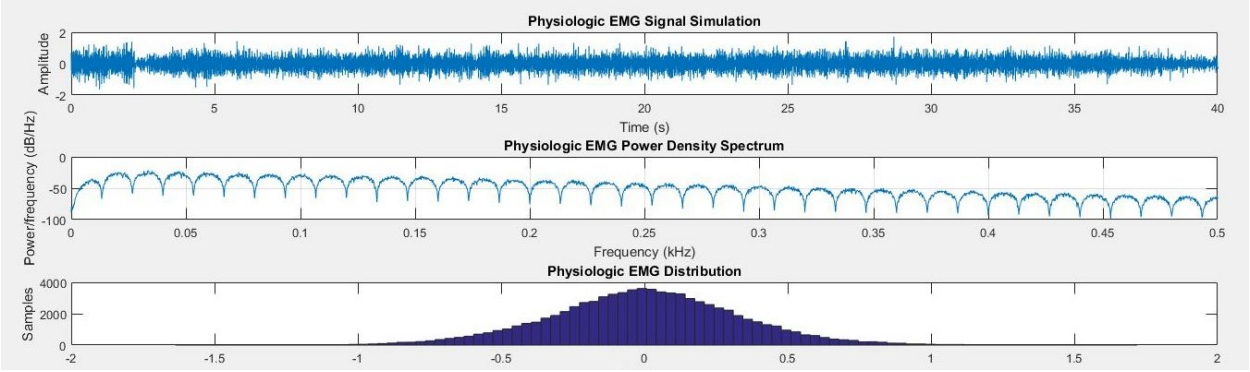



Figure B8: Simulated EMG Signal (Time, Power Spectrum, and Distribution Plots) [4]

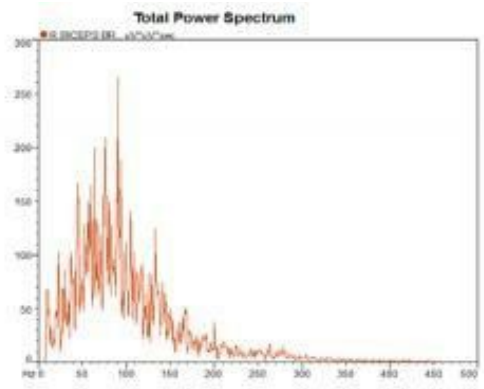



Figure B9: Actual EMG Signal (Power Spectrum) [5]

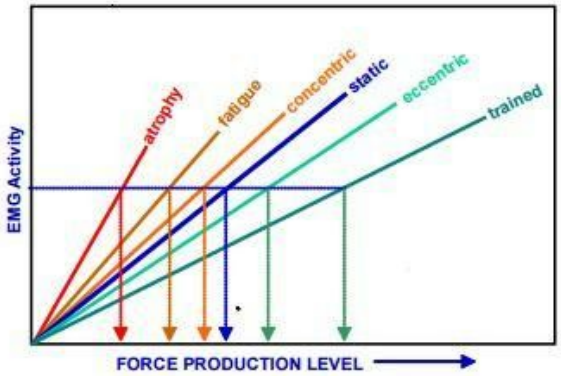



Figure B10: Linear Relationship between EMG Activity and Produced Force [5]

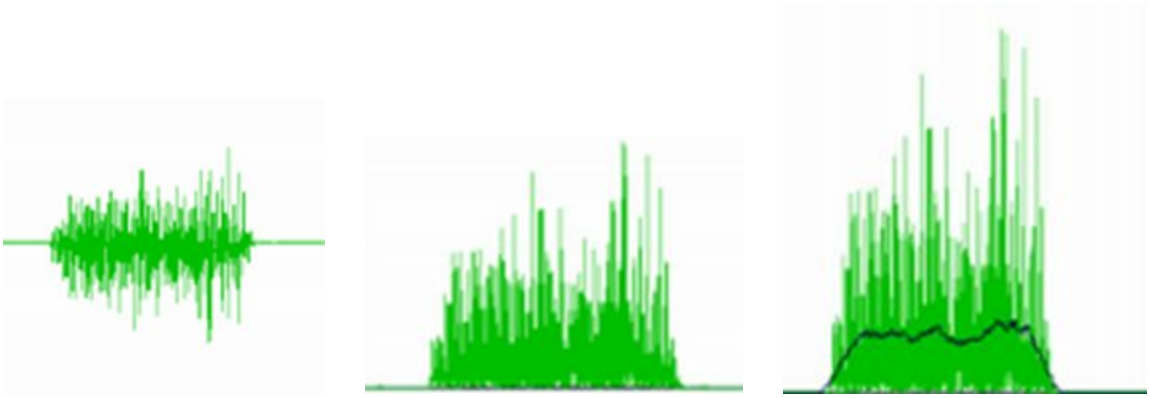


Figure B11-B13: Filtered, Rectified, and Averaged EMG Signal [6]

# Appendix C Data Processing Subsystem Code

## C-1 MSP432 Main Code

```
/*******************************************************************************
 * MAIN
 *
 *  Created on: Dec, 2015
 *         Author: Rafael Salas
*******************************************************************************/

#include <driverlib.h>
#include <stdint.h>
#include <stdbool.h>
#include <QMathLib.h>
#include "printf.h"
#include "RadialEncoder.h"
#include <string.h>

//-----------------------------------------------Variables
const int bit_stream_length = 24; // length in bytes, data + status byte
const int msp_clk_rate = 48000;//48Mhz
const int ads_clk_rate = 2048; //2.048 MHz
uint8_t Drdy = 0x00; //Flag for SPI
uint8_t sample_rdy=0x00; //Flag when window amount of samples read.
uint8_t spi_data[50][24];
int32_t sample[50];
uint8_t spi_index = 0; // used to keep track of the first dimension of the spi_dta matrix

const uint8_t window = 50; //moving average window
volatile int16_t samples [50]; //raw signal
volatile int16_t filtered [50]; //filtered signal
static uint32_t sum = 0; //sum for moving average

volatile int32_t raw_position = 0;
volatile int32_t pos_pulse_count = 0;
volatile int32_t neg_pulse_count = 0;

static uint16_t resultsBuffer[2];// used for ADC
volatile uint16_t a,b =0; //used for adc testing
//-----------------------------------------------Filtering
```

```
void conditionSamples(){
   uint8_t i = 0;
   for(i = 0; i < window;++i){
          samples[i] = _Q1abs(samples[i]);//rectify
          //moving average
          sum = sum + samples[i];
          if(i>window){
                sum = sum - samples[i-window];
                filtered[i] = sum/window;
          }
          else if(i = window){
                filtered[i] = sum/window;
          }
          else{
                filtered[i] = 0;
          }
          MAP_UART_transmitData(EUSCI_A0_MODULE, filtered[i]);
   }
}
//-----------------------------------------------ADC

void adc(){

          MAP_PCM_setPowerState(PCM_AM_LDO_VCORE1);
          MAP_CS_setDCOCenteredFrequency(CS_DCO_FREQUENCY_48);

          /* Zero-filling buffer */
          memset(resultsBuffer, 0x00, 2);

          /* Setting reference voltage to 2.5  and enabling reference */
          MAP_REF_A_setReferenceVoltage(REF_A_VREF2_5V);
          MAP_REF_A_enableReferenceVoltage();

          /* Initializing ADC (MCLK/1/1) */
          MAP_ADC14_enableModule();
          //MAP_ADC14_initModule(ADC_CLOCKSOURCE_MCLK, ADC_PREDIVIDER_1,
ADC_DIVIDER_1, 0);
          MAP_ADC14_initModule(ADC_CLOCKSOURCE_MCLK, ADC_PREDIVIDER_1,
ADC_DIVIDER_4,
          0);
          /* Configuring GPIOs for Analog In */
```

```
        MAP_GPIO_setAsPeripheralModuleFunctionInputPin(GPIO_PORT_P5,
        GPIO_PIN5 | GPIO_PIN4, GPIO_TERTIARY_MODULE_FUNCTION);

        /* Configuring ADC Memory (ADC_MEM0 - ADC_MEM7 (A0 - A1)  with no repeat)
        * with internal 2.5v reference */

        MAP_ADC14_configureMultiSequenceMode(ADC_MEM0, ADC_MEM1, true);
        MAP_ADC14_configureConversionMemory(ADC_MEM0,
ADC_VREFPOS_INTBUF_VREFNEG_VSS, ADC_INPUT_A0, false);
        MAP_ADC14_configureConversionMemory(ADC_MEM1,
ADC_VREFPOS_INTBUF_VREFNEG_VSS, ADC_INPUT_A1, false);

        /* Enabling the interrupt when a conversion on channel 7 (end of sequence)
        *  is complete and enabling conversions */
        MAP_ADC14_enableInterrupt(ADC_INT1);

        /* Enabling Interrupts */
        MAP_Interrupt_enableInterrupt(INT_ADC14);
        MAP_Interrupt_enableMaster();

        /* Setting up the sample timer to automatically step through the sequence convert.*/
        MAP_ADC14_enableSampleTimer(ADC_AUTOMATIC_ITERATION);

        /* Triggering the start of the sample */
        MAP_ADC14_enableConversion();
        MAP_ADC14_toggleConversionTrigger();

}
//-----------------------------------------------SPI

int32_t twos_to_signed (uint32_t msb, uint32_t mid, uint32_t lsb){


   uint32_t num = (msb<<24)|(mid<<16)|(lsb<<8);// concatenate bytes
        int32_t num2;
        if(((int)num)<0){
        num2=-((~num)+1); //2's complement
        }
        else{
        num2 = num;
        }
        num2 = num2>>2; //shift to correct for 2 bit offset needed for 2's complement
```

```
        return num2;
}


void spi_setup(){
/* SPI Master Configuration Parameter */
   const eUSCI_SPI_MasterConfig spiMasterConfig =
   {
        EUSCI_B_SPI_CLOCKSOURCE_ACLK,               // ACLK Clock Source
        32768,                              // ACLK = LFXT = 32.768khz
        500000,                             // SPICLK = 500khz (110k)
        EUSCI_B_SPI_MSB_FIRST,                      // MSB First
        EUSCI_B_SPI_PHASE_DATA_CHANGED_ONFIRST_CAPTURED_ON_NEXT,    // Phaseb
        EUSCI_B_SPI_CLOCKPOLARITY_INACTIVITY_LOW, // low polarity
        EUSCI_B_SPI_3PIN                            // 3Wire SPI Mode
   };

        /* Starting and enabling LFXT (32kHz) */
   MAP_GPIO_setAsPeripheralModuleFunctionOutputPin(GPIO_PORT_PJ, GPIO_PIN0 |
GPIO_PIN1, GPIO_PRIMARY_MODULE_FUNCTION);
   MAP_CS_setExternalClockSourceFrequency(32768, 0);//32.768 KHz
   MAP_CS_initClockSignal(CS_ACLK, CS_LFXTCLK_SELECT, CS_CLOCK_DIVIDER_1);// clock
source CS_ACLK, Use external clock, no clock divisions
   MAP_CS_startLFXT(CS_LFXT_DRIVE0); //LFXTDRIVE_0 give the lowest current consumption
according to the msp432 header file for this board

   //selecting pins for spi mode
   //pin 1 = clk  pin 6 = mosi pin 7 =miso
   MAP_GPIO_setAsPeripheralModuleFunctionOutputPin(GPIO_PORT_P1 , GPIO_PIN5 |
GPIO_PIN6 | GPIO_PIN7,
                GPIO_PRIMARY_MODULE_FUNCTION );

   //configuring SPI for 3wire master mode
   MAP_SPI_initMaster(EUSCI_B0_MODULE,&spiMasterConfig);//EUSCI_B0_MODULE = Base
address of module registers
   MAP_SPI_enableModule(EUSCI_B0_MODULE);
        Interrupt_enableInterrupt(INT_EUSCIB0);
        Interrupt_enableSleepOnIsrExit();

  /* Delaying waiting for the module to initialize */
        __delay_cycles(500);
```

```
}

void read_message(){
   unsigned int cycle = (msp_clk_rate/ads_clk_rate)+1; // cycle ratio needed for delays
between reading bytes
   unsigned int delay = 4*cycle;
   unsigned int iterator = 0;
   unsigned int iter = 0;
   uint32_t container[3];

   if(spi_index == window){//condition sample and reset  spi_index
         conditionSamples();
         spi_index = 0;
   }
   uint8_t it = 0;
   for(it=0;it<3;++it){
         SPI_receiveData(EUSCI_B0_MODULE); //read status byte
   }
   for(iterator = 0;iterator < bit_stream_length;++iterator){

                  for(iter = 0; iter<delay ;++iter){}// delay for 4 cycles
                  container[iterator%3] = SPI_receiveData(EUSCI_B0_MODULE); //read a byte
                  if(iterator%3 == 2){
                        sample[spi_index]= twos_to_signed
(container[0],container[1],container[2]);
                        printf(EUSCI_A0_MODULE, "value: %i \r\n", sample[spi_index]);
                  }
   }
   /*if(spi_index==49){
         sample_rdy = 0x01;
   }*/
   Drdy = 0x01;
   spi_index++; //increment first dimension of spi_data index
}

void spi_start(){
   __delay_cycles(10000); //Wait 150ms
   SPI_transmitData(EUSCI_B0_MODULE, 0x06); //RESET
   __delay_cycles(10000); //Wait 150ms
   SPI_transmitData(EUSCI_B0_MODULE, 0x06); //RESET
   __delay_cycles(100000); //Wait 150ms
```

```
    // Power Up sequencing - Single ended Input
    SPI_transmitData(EUSCI_B0_MODULE, 0x11); //SDATAC
//    __delay_cycles(960);
    __delay_cycles(10000); //Wait 150ms
    SPI_transmitData(EUSCI_B0_MODULE, 0x43); //CONFIG3 address
//    __delay_cycles(960);
    __delay_cycles(10000); //Wait 150ms
    SPI_transmitData(EUSCI_B0_MODULE, 0x00); //Write to 1 register
//    __delay_cycles(960);
    __delay_cycles(10000); //Wait 150ms
    SPI_transmitData(EUSCI_B0_MODULE, 0x6C); //Register data
    __delay_cycles(960);
    __delay_cycles(100000);

    SPI_transmitData(EUSCI_B0_MODULE, 0x41); //CONFIG1 address
//    __delay_cycles(960);
    __delay_cycles(10000); //Wait 150ms
    SPI_transmitData(EUSCI_B0_MODULE, 0x00);// Write to 1 register
//    __delay_cycles(960);
    __delay_cycles(10000); //Wait 150ms
    SPI_transmitData(EUSCI_B0_MODULE, 0x96); //Register data - 500Hz data rate
//    __delay_cycles(960);
    __delay_cycles(10000); //Wait 150ms

    SPI_transmitData(EUSCI_B0_MODULE, 0x42); //CONFIG2 address
//     __delay_cycles(960);
     __delay_cycles(10000); //Wait 150ms
    SPI_transmitData(EUSCI_B0_MODULE, 0x00); //Write to 1 register
//    __delay_cycles(960);
    __delay_cycles(10000); //Wait 150ms
    SPI_transmitData(EUSCI_B0_MODULE, 0xC0); //Register data
//    __delay_cycles(960);
    __delay_cycles(10000); //Wait 150ms

    SPI_transmitData(EUSCI_B0_MODULE, 0x45); //CH1SET address
//    __delay_cycles(960);
    __delay_cycles(10000); //Wait 150ms
    SPI_transmitData(EUSCI_B0_MODULE, 0x07); //Write to 8 registers
//    __delay_cycles(960);
    __delay_cycles(10000); //Wait 150ms
```

```
   SPI_transmitData(EUSCI_B0_MODULE, 0x00); //Register data - normal electrode
//    __delay_cycles(960);
   __delay_cycles(10000); //Wait 150ms
   SPI_transmitData(EUSCI_B0_MODULE, 0x00); //Register data
//    __delay_cycles(960);
   __delay_cycles(10000); //Wait 150ms
   SPI_transmitData(EUSCI_B0_MODULE, 0x00); //Register data
//    __delay_cycles(960);
   __delay_cycles(10000); //Wait 150ms
   SPI_transmitData(EUSCI_B0_MODULE, 0x00); //Register data
//    __delay_cycles(960);
   __delay_cycles(10000); //Wait 150ms
   SPI_transmitData(EUSCI_B0_MODULE, 0x00); //Register data
//    __delay_cycles(960);
   __delay_cycles(10000); /Wait 150ms
   SPI_transmitData(EUSCI_B0_MODULE, 0x00); //Register data
//    __delay_cycles(960);
   __delay_cycles(10000); //Wait 150ms
   SPI_transmitData(EUSCI_B0_MODULE, 0x00); //Register data
//    __delay_cycles(960);
   __delay_cycles(10000); //Wait 150ms
   SPI_transmitData(EUSCI_B0_MODULE, 0x00); //Register data
//    __delay_cycles(960);
   __delay_cycles(10000); //Wait 150ms

   SPI_transmitData(EUSCI_B0_MODULE, 0x55); //MISC1 address is 15h
//    __delay_cycles(960);
   __delay_cycles(10000); //Wait 150ms
   SPI_transmitData(EUSCI_B0_MODULE, 0x00); //Write to 1 register
//    __delay_cycles(960);
   __delay_cycles(10000); //Wait 150ms
   SPI_transmitData(EUSCI_B0_MODULE, 0x20); //Register data - Set SRB1
//    __delay_cycles(960);
   __delay_cycles(10000); //Wait 150ms
   SPI_transmitData(EUSCI_B0_MODULE,0x44); //LOFF address
//    __delay_cycles(960);
   __delay_cycles(10000); //Wait 150ms
   SPI_transmitData(EUSCI_B0_MODULE,0x00); //Write to 1 register
//    __delay_cycles(960);
   __delay_cycles(10000); //Wait 150ms
```

```
   SPI_transmitData(EUSCI_B0_MODULE,0x06); //Register data - 24nA, 31.2Hz
//    __delay_cycles(960);
   __delay_cycles(10000); //Wait 150ms
   __delay_cycles(1920);
   SPI_transmitData(EUSCI_B0_MODULE, 0x08); //START
//    __delay_cycles(960);
   __delay_cycles(10000);
   SPI_transmitData(EUSCI_B0_MODULE, 0x10); // read data cont
   __delay_cycles(10000);
}

//-----------------------------------------------Drdy
void drdy_setup(){
        /* Configuring P3.5 as an input and enabling interrupts */
        MAP_GPIO_setAsInputPinWithPullUpResistor(GPIO_PORT_P3, GPIO_PIN5);
        MAP_GPIO_clearInterruptFlag(GPIO_PORT_P3, GPIO_PIN5);
        MAP_GPIO_enableInterrupt(GPIO_PORT_P3, GPIO_PIN5);
        MAP_Interrupt_enableInterrupt(INT_PORT3);

        /* Enabling MASTER interrupts */
        MAP_Interrupt_enableMaster();
}

//-----------------------------------------------Uart
void uart_setup(){
   const eUSCI_UART_Config uartConfig =
   {
        EUSCI_A_UART_CLOCKSOURCE_SMCLK,          // SMCLK Clock Source
        26,                                 // BRDIV = 26
        0,                                  // UCxBRF = 0
        111,                                // UCxBRS = 111
        EUSCI_A_UART_NO_PARITY,                 // No Parity
        EUSCI_A_UART_LSB_FIRST,                 // LSB First
        EUSCI_A_UART_ONE_STOP_BIT,              // One stop bit
        EUSCI_A_UART_MODE,                      // UART mode
        EUSCI_A_UART_OVERSAMPLING_BAUDRATE_GENERATION  // Oversampling
   };

        /* Selecting P1.2 and P1.3 in UART mode and P1.0 as output (LED) */
        MAP_GPIO_setAsPeripheralModuleFunctionInputPin(GPIO_PORT_P1,GPIO_PIN1 |
GPIO_PIN2 | GPIO_PIN3, GPIO_PRIMARY_MODULE_FUNCTION);
```

```
        //MAP_GPIO_setAsOutputPin(GPIO_PORT_P1, GPIO_PIN0);
        //MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN0);

        /* Setting DCO to 48MHz (upping Vcore) */
        MAP_PCM_setCoreVoltageLevel(PCM_VCORE1);
        CS_setDCOCenteredFrequency(CS_DCO_FREQUENCY_48);
        //CS_setExternalClockSourceFrequency
        /* Configuring UART Module */
        MAP_UART_initModule(EUSCI_A0_MODULE, &uartConfig);

        /* Enable UART module */
        MAP_UART_enableModule(EUSCI_A0_MODULE);
        MAP_Interrupt_enableMaster();
}


volatile int css = 0;
void main(void)
{
   MAP_WDT_A_holdTimer();
   drdy_setup();
   adc();
   uart_setup();
   encoderInit();
   spi_setup();
   spi_start();
        while(1){}
}

//-----------------------------------------------Interrupts

/* This interrupt is fired whenever a conversion is completed and placed in
 * ADC_MEM1. This signals the end of conversion and the results array is
 * grabbed and placed in resultsBuffer */

void adc_isr(void)
{
        uint64_t status;
        status = MAP_ADC14_getEnabledInterruptStatus();
        MAP_ADC14_clearInterruptFlag(status);
```

```
        /*  if(status & ADC_INT0)
        {
        a = ADC14_getResult(ADC_MEM0);
        }*/
        if(status & ADC_INT1)
        {
        MAP_ADC14_getMultiSequenceResult(resultsBuffer);
        a = resultsBuffer[0];
        b = resultsBuffer[1];
        }

}


void gpio_isr4(void)
{
   uint8_t val[3];
   val[0] = GPIO_getInputPinValue(GPIO_PORT_P4,GPIO_PIN0);
   val[1] = GPIO_getInputPinValue(GPIO_PORT_P4,GPIO_PIN1);
   val[2] = GPIO_getInputPinValue(GPIO_PORT_P4,GPIO_PIN2);
        uint32_t status;

        status = MAP_GPIO_getEnabledInterruptStatus(GPIO_PORT_P4);
        MAP_GPIO_clearInterruptFlag(GPIO_PORT_P4, status);

  //  printf(EUSCI_A0_MODULE,"gpio1 %i", GPIO_getInputPinValue(GPIO_PORT_P4,GPIO_PIN0));

        printf(EUSCI_A0_MODULE,"\r\n\r\n");
        if(val[0]==val[1]==val[2] ){ //one step CW
         printf(EUSCI_A0_MODULE,"in neg\r\n");
         raw_position++;
   }else{ //one step CCW
         printf(EUSCI_A0_MODULE,"in pos\r\n");
         //temp_pos++;
         raw_position--;
   }

   /*if(temp_pos > temp_neg){
         raw_position++;
         printf(EUSCI_A0_MODULE,"temp_pos: %i",temp_pos);
   }
```

```
    else if(temp_pos < temp_neg){
            raw_position--;
            printf(EUSCI_A0_MODULE,"temp_neg: %i",temp_neg);
    }*/
}

void gpio_isr3(void)
{
        uint32_t status;

        status = MAP_GPIO_getEnabledInterruptStatus(GPIO_PORT_P3);
        MAP_GPIO_clearInterruptFlag(GPIO_PORT_P3, status);

        /* set Drdy Flag*/
        if(status & GPIO_PIN5)
        {
        //    Drdy = 0x01;
        //   MAP_GPIO_toggleOutputOnPin(GPIO_PORT_P1, GPIO_PIN0);
        //        read_message();
        }

}
```

## C-2 Supporting Figures

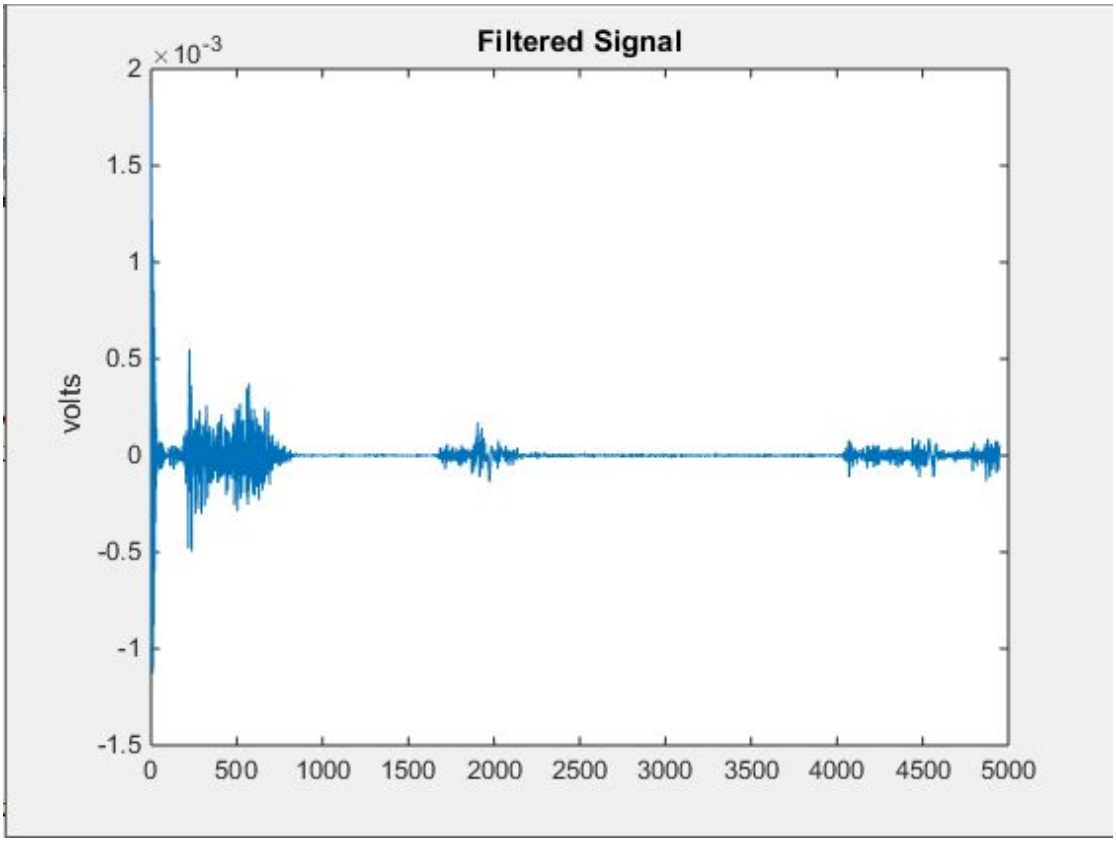



Figure C1: EMG Signal (Matlab)

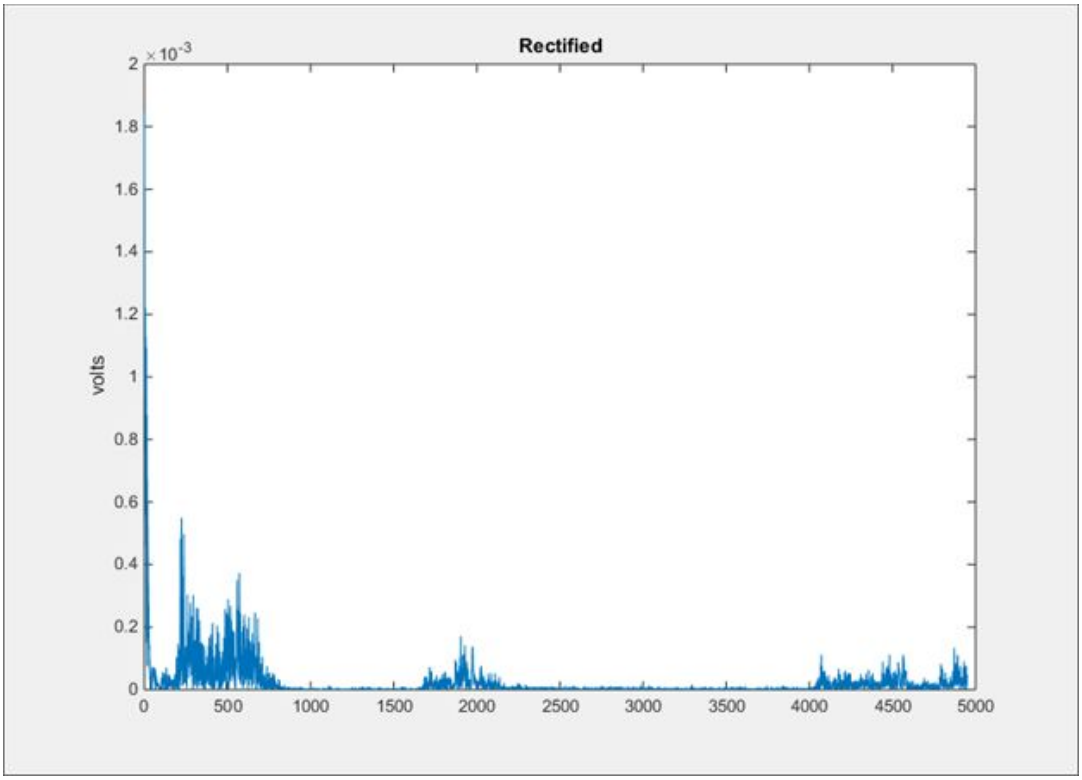



Figure C2: Rectified EMG Signal (Matlab)

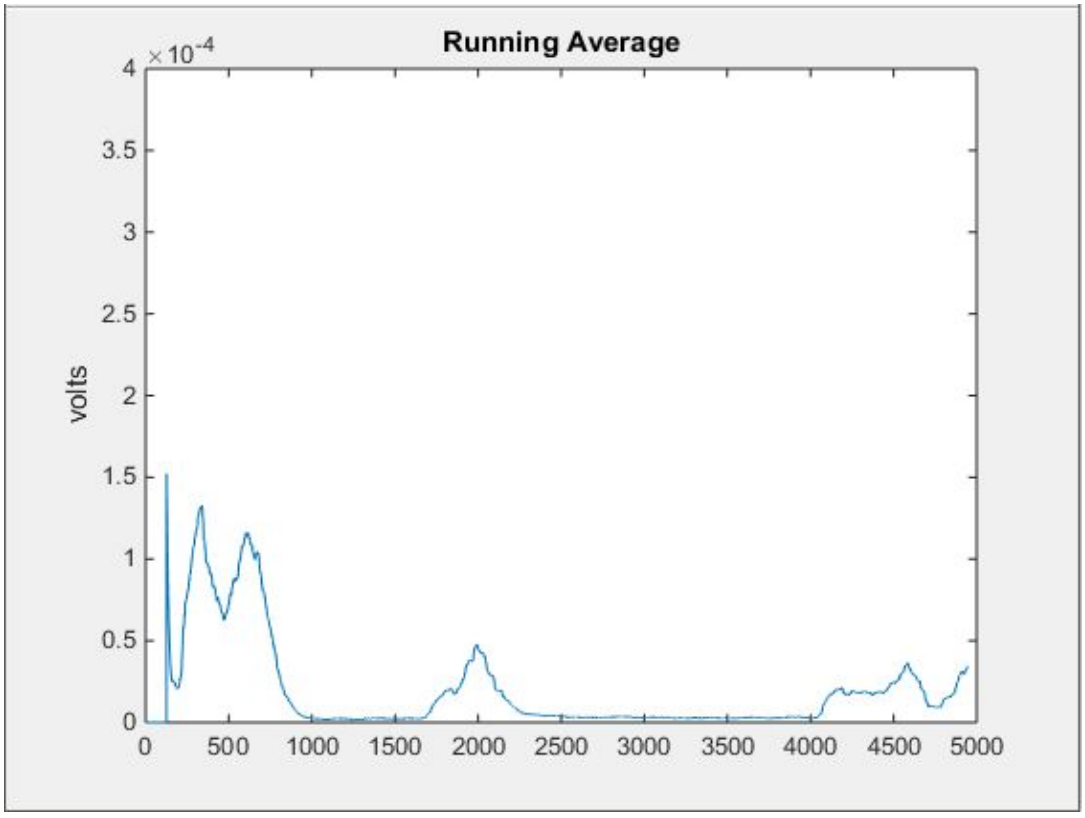



Figure C3: Running Average of Rectified Signal(Matlab)

# Appendix D Motor-to-Body Interface Subsystem

## D-1 **Supporting Figures**

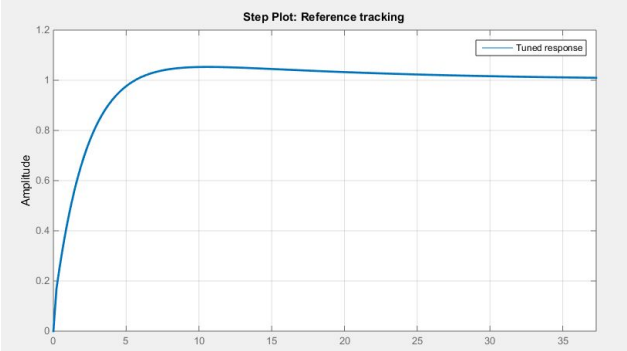



Figure D1: Response with: Kp= 4.907, Ki= 1.074 Kd= 0.2205

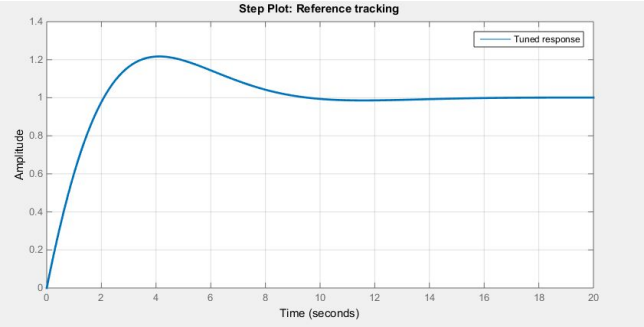



Figure D2: Response with: Kp= 1.645, Ki= 0.7145 Kd= 0

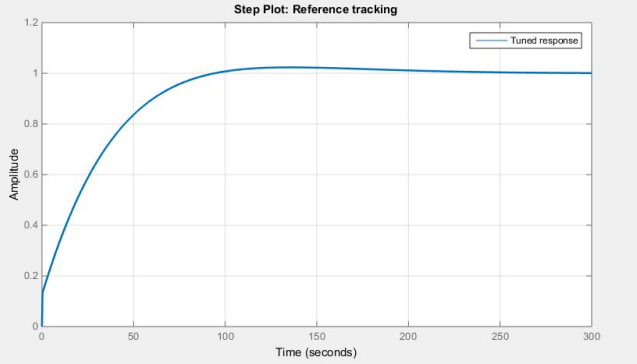
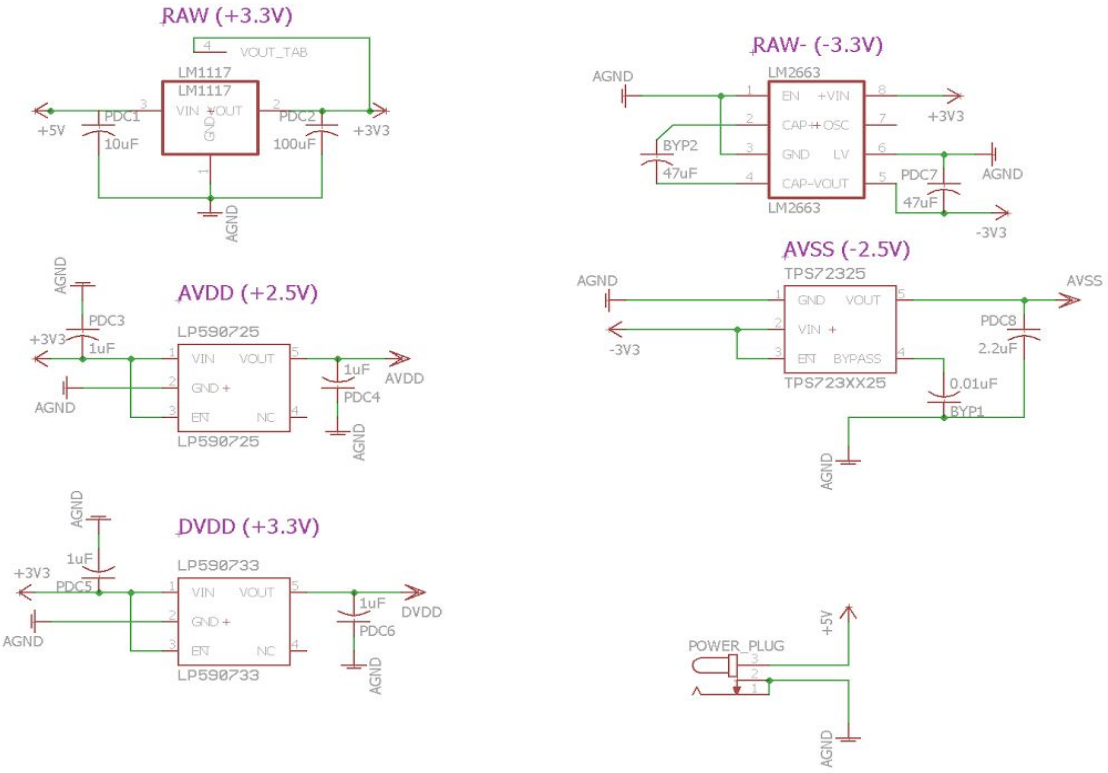



Figure D3: Response with: Kp= 0.11877, Ki= 0.0032163 Kd= 0.117755

# Appendix E Power Subsystem

## E-1 ADS1299 Power Distribution Schematic




## E-2 ADS1299 Power Distribution Bill of Materials

| Symbol | Value | Description |
|---|---|---|
| LM1117 | | Voltage Regulator 5 V to 3.3 V |
| LM2663 | | Voltage Regulator 3.3 V to -3.3 V |
| LP590725 | | Voltage Regulator 3.3 V to 2.5 V |
| LP590733 | | Voltage Regulator 3.3 V to 3.3 V |
| TPS72325 | | Voltage Regulator -3.3 V to -2.5 V |
| PDC1 | 10 uF | Capacitor |
| PDC2 | 100 uF | Capacitor |
| BYP2,PDC7 | 47 uF | Capacitor |
| PDC3,PDC4,PDC5,PDC6 | 1 uF | Capacitor |
| PDC8 | 2.2 uF | Capacitor |
| BYP1 | .01 uF | Capacitor |
| Power Plug | | Takes in power from buck converter |

# Appendix F User Interface Subsystem

## F-1 modified Chronos menu - menu.c

```
// system
#include "project.h"

// driver
#include "display.h"

// logic
#include "menu.h"
#include "user.h"
#include "clock.h"
#include "date.h"
#include "alarm.h"
#include "stopwatch.h"
#include "temperature.h"
#include "altitude.h"
#include "battery.h"
#include "bluerobin.h"
#include "rfsimpliciti.h"
#include "acceleration.h"
#include "Calibration.h"
#include "rfbsl.h"

// *************************************************************************************************
// Defines section
#define FUNCTION(function)  function

// *************************************************************************************************
// Global Variable section
const struct menu *ptrMenu_L1 = NULL;
const struct menu *ptrMenu_L2 = NULL;

// *************************************************************************************************
// Global Variable section

void display_nothing(u8 line, u8 update)
{
}
```

```
u8 update_time(void)
{
   return (display.flag.update_time);
}

u8 update_stopwatch(void)
{
   return (display.flag.update_stopwatch);
}

u8 update_date(void)
{
   return (display.flag.update_date);
}

u8 update_alarm(void)
{
   return (display.flag.update_alarm);
}

u8 update_temperature(void)
{
   return (display.flag.update_temperature);
}

u8 update_battery_voltage(void)
{
   return (display.flag.update_battery_voltage);
}

u8 update_acceleration(void)
{
   return (display.flag.update_acceleration);
}

u8 update_calibration(void)
{
   return (display.flag.update_calibration);
}
```

```
// ********************************************************************************************
// User navigation ( [____] = default menu item after reset )
//
//      LINE1:  [Time] -> Alarm -> Temperature -> Altitude -> Heart rate -> Speed ->
Acceleration
//
//      LINE2:  [Date] -> Stopwatch -> Battery  -> ACC -> PPT -> SYNC -> Calories/Distance -->
RFBSL
// ********************************************************************************************

// Line1 - Time
const struct menu menu_L1_Time = {
    FUNCTION(sx_time),              // direct function
    FUNCTION(mx_time),               // sub menu function
    FUNCTION(display_time),          // display function
    FUNCTION(update_time),           // new display data
    &menu_L1_Alarm,
};

// Line1 - Alarm
const struct menu menu_L1_Alarm = {
    FUNCTION(sx_alarm),             // direct function
    FUNCTION(mx_alarm),              // sub menu function
    FUNCTION(display_alarm),         // display function
    FUNCTION(update_alarm),          // new display data
    &menu_L1_Temperature,
};

// Line1 - Temperature
const struct menu menu_L1_Temperature = {
    FUNCTION(dummy),                // direct function
    FUNCTION(mx_temperature),        // sub menu function
    FUNCTION(display_temperature),    // display function
    FUNCTION(update_temperature),     // new display data
    &menu_L1_Altitude,
};

// Line1 - Altitude
const struct menu menu_L1_Altitude = {
    FUNCTION(sx_altitude),          // direct function
    FUNCTION(mx_altitude),           // sub menu function
    FUNCTION(display_altitude),      // display function
```

```
    FUNCTION(update_time),          // new display data
    &menu_L1_Heartrate,
};

// Line1 - Heart Rate
const struct menu menu_L1_Heartrate = {
    FUNCTION(sx_bluerobin),         // direct function
    FUNCTION(mx_bluerobin),         // sub menu function
    FUNCTION(display_heartrate),    // display function
    FUNCTION(update_time),          // new display data
    &menu_L1_Speed,
};

// Line1 - Speed
const struct menu menu_L1_Speed = {
    FUNCTION(dummy),                // direct function
    FUNCTION(dummy),                // sub menu function
    FUNCTION(display_speed),        // display function
    FUNCTION(update_time),          // new display data
    &menu_L1_Acceleration,
};

// Line1 - Acceleration
const struct menu menu_L1_Acceleration = {
    FUNCTION(sx_acceleration),      // direct function
    FUNCTION(dummy),                // sub menu function
    FUNCTION(display_acceleration), // display function
    FUNCTION(update_acceleration),  // new display data
    &menu_L1_Calibration,
};

// Line1 - Calibration
const struct menu menu_L1_Calibration = {
    FUNCTION(sx_calibration),       // direct function
    FUNCTION(mx_calibration),       // sub menu function
    FUNCTION(display_calibration),  // display function
    FUNCTION(update_calibration),   // new display data
    &menu_L1_Time,
};
// Line2 - Date
const struct menu menu_L2_Date = {
```

```
    FUNCTION(sx_date),              // direct function
    FUNCTION(mx_date),               // sub menu function
    FUNCTION(display_date),          // display function
    FUNCTION(update_date),           // new display data
    &menu_L2_Stopwatch,
};

// Line2 - Stopwatch
const struct menu menu_L2_Stopwatch = {
    FUNCTION(sx_stopwatch),          // direct function
    FUNCTION(mx_stopwatch),          // sub menu function
    FUNCTION(display_stopwatch),     // display function
    FUNCTION(update_stopwatch),      // new display data
    &menu_L2_Battery,
};

// Line2 - Battery
const struct menu menu_L2_Battery = {
    FUNCTION(dummy),                 // direct function
    FUNCTION(dummy),                 // sub menu function
    FUNCTION(display_battery_V),     // display function
    FUNCTION(update_battery_voltage), // new display data
    &menu_L2_Rf,
};

// Line2 - ACC (acceleration data + button events via SimpliciTI)
const struct menu menu_L2_Rf = {
    FUNCTION(sx_rf),                // direct function
    FUNCTION(dummy),                 // sub menu function
    FUNCTION(display_rf),           // display function
    FUNCTION(update_time),           // new display data
    &menu_L2_Ppt,
};

// Line2 - PPT (button events via SimpliciTI)
const struct menu menu_L2_Ppt = {
    FUNCTION(sx_ppt),               // direct function
    FUNCTION(dummy),                 // sub menu function
    FUNCTION(display_ppt),           // display function
    FUNCTION(update_time),           // new display data
    &menu_L2_Sync,
```

```
};

// Line2 - SXNC (synchronization/data download via SimpliciTI)
const struct menu menu_L2_Sync = {
   FUNCTION(sx_sync),              // direct function
   FUNCTION(dummy),                // sub menu function
   FUNCTION(display_sync),         // display function
   FUNCTION(update_time),          // new display data
   &menu_L2_CalDist,
};

// Line2 - Calories/Distance
const struct menu menu_L2_CalDist = {
   FUNCTION(sx_caldist),           // direct function
   FUNCTION(mx_caldist),           // sub menu function
   FUNCTION(display_caldist),      // display function
   FUNCTION(update_time),          // new display data
   &menu_L2_RFBSL,
};

// Line2 - RFBSL
const struct menu menu_L2_RFBSL = {
   FUNCTION(sx_rfbsl),             // direct function
   FUNCTION(dummy),                // sub menu function
   FUNCTION(display_rfbsl),        // display function
   FUNCTION(update_time),          // new display data
   &menu_L2_Date,
};
```

# Appendix G Structure Subsystem

## G-1 Three-Dimensional CAD Images

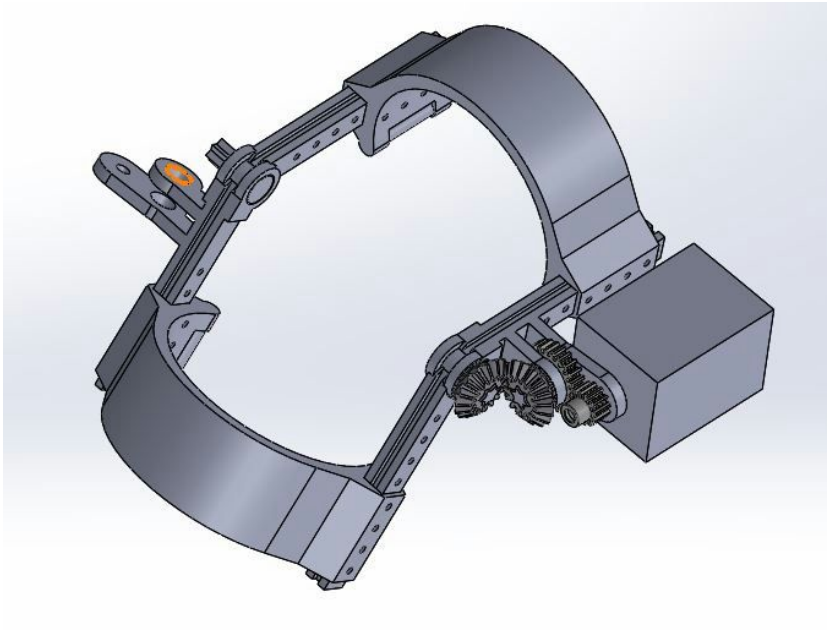


Figure G1: Orthosis Structure (Geared Design)

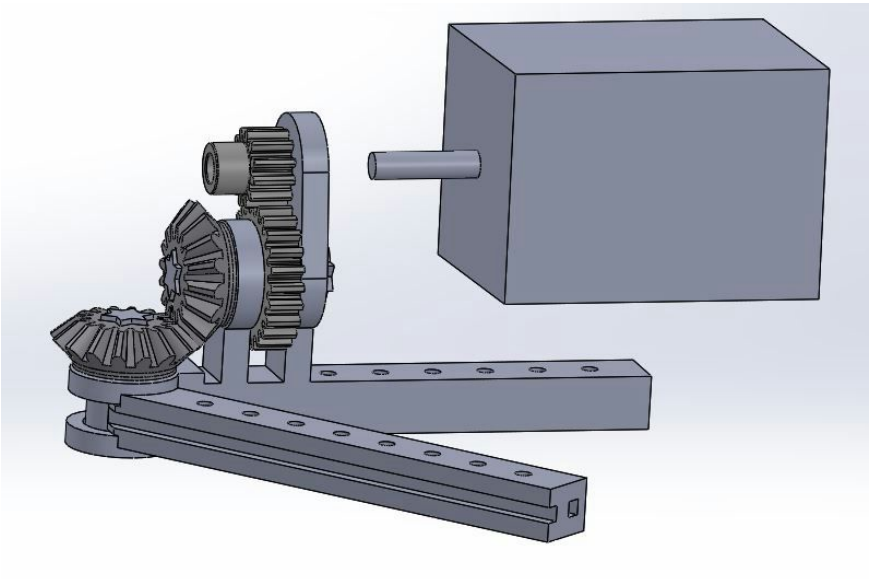


Figure G2: Orthosis Joint (Geared Design)

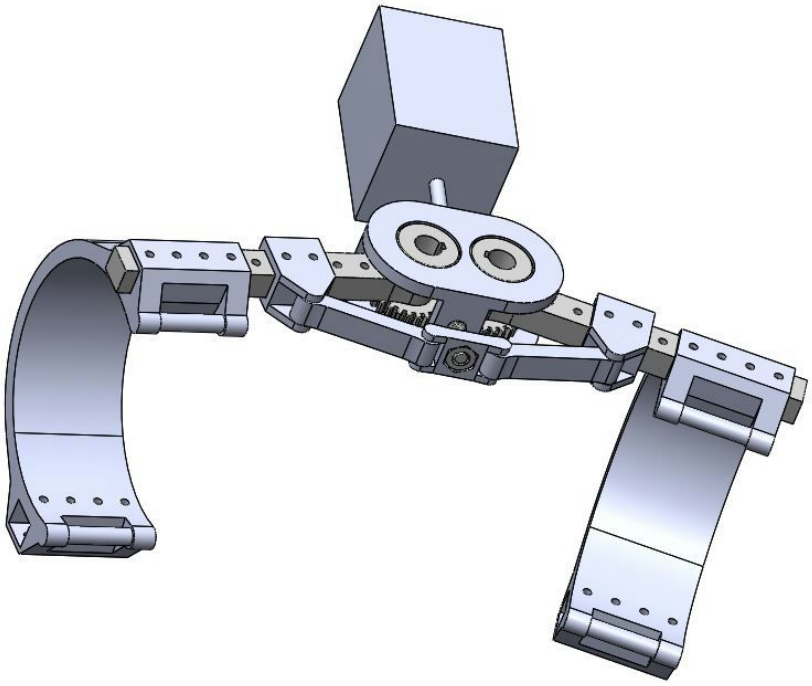


Figure G3: Orthosis Opened (Winch Design)

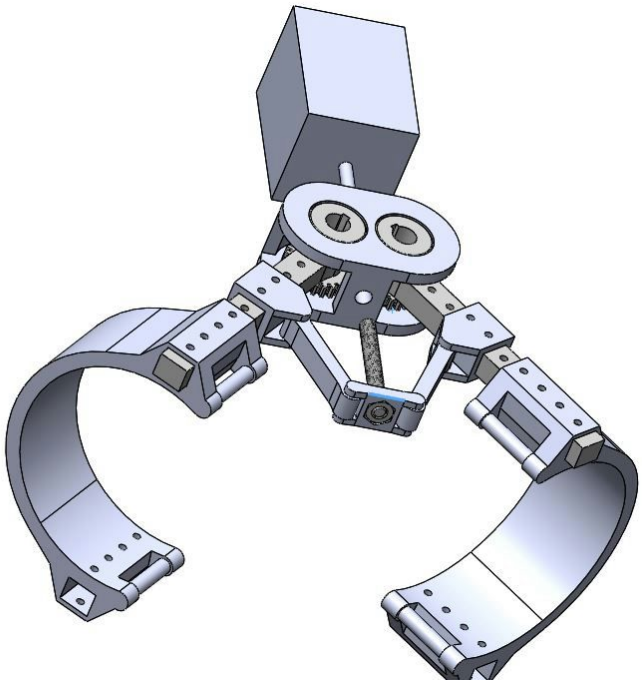


Figure G4: Orthosis Closed (Winch Design)