

# Remote Vibrotactile Feedback

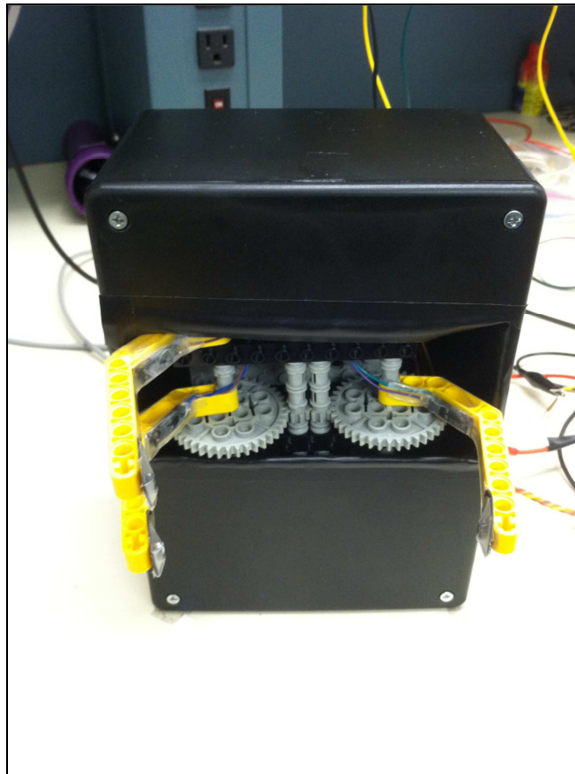
---

## A Microprocessor Based Design

Cody Hogan, Zachary Pritchett & Camilo Tejeiro

**12/10/2011**

**EE 478**

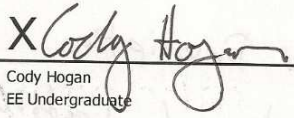


The following report details the development of a system designed to control a manipulator using electromyography and provide vibrotactile feedback to upper limb amputees based on applied force.

**Individual Contributions:**

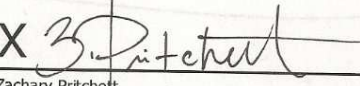
**Cody Hogan**

- |  |                   |
|--|-------------------|
| ✓ <i>Design Phase (UML; Pseudo Code; etc.)</i> | * <i>30 Hours</i> |
| ✓ <i>Hardware</i>                              | * <i>40 Hours</i> |
| ✓ <i>Coding</i>                                | * <i>15Hours</i>  |
| ✓ <i>Report</i>                                | * <i>5 Hours</i>  |

X   
Cody Hogan  
EE Undergraduate

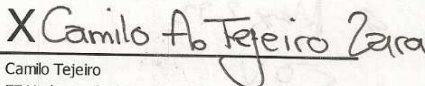
**Zachary Pritchett**

- |                         |                   |
|-------------------------|-------------------|
| ✓ <i>Hardware</i>       | * <i>40 Hours</i> |
| ✓ <i>Coding</i>         | * <i>40 Hours</i> |
| ✓ <i>Debugging(CCS)</i> | * <i>20 Hours</i> |
| ✓ <i>Report</i>         | * <i>4 Hours</i>  |

X   
Zachary Pritchett  
EE Undergraduate

**Camilo Tejeiro**

- |                         |                   |
|-------------------------|-------------------|
| ✓ <i>Hardware</i>       | * <i>40 Hours</i> |
| ✓ <i>Coding</i>         | * <i>40 Hours</i> |
| ✓ <i>Debugging(CCS)</i> | * <i>25 Hours</i> |
| ✓ <i>Report</i>         | * <i>3 Hours</i>  |

X   
Camilo Tejeiro  
EE Undergraduate

## Table of Contents

ABSTRACT.....	5
INTRODUCTION.....	5
DISCUSSION OF THE LAB.....	6
Requirements Specification .....	6
System Description (Summary).....	6
Specification of External Environment .....	6
System Input and Output Specification .....	6
Inputs .....	6
Outputs .....	6
User Interface .....	7
Use Case Diagrams.....	8
Timing Constraint Requirements .....	12
Operating Specifications .....	12
Reliability and Safety Specification .....	12
Design Procedure .....	12
Summary .....	12
High Level View.....	12
Pseudo Description of Functions .....	13
Timing Constraints .....	14
Error Handling .....	14
Software Implementation.....	14
High Level Architecture.....	14
Functional Blocks .....	14
Discussion of Specific Tasks .....	15
EMG Acquisiton.....	15
Wireless Protocol .....	15
Wireless Transmission.....	16
Wireless Reception .....	16
Vibrotactile Transducer Control.....	18

Robotic Manipulator .....	18
Force Feedback .....	19
Flow of Control.....	19
Data and Control Flow .....	20
Sequence Diagram .....	22
Design Choices .....	23
EMG.....	23
Wireless Protocol .....	24
Wireless Transmission.....	24
Wireless Reception .....	24
Vibrotactile Transducer Control.....	24
Robotic Manipulator .....	25
Force Feedback .....	25
Hardware Implementation .....	25
Block Diagram .....	25
Discussion of Modules .....	25
EMG.....	25
Wireless Communication .....	26
Robotic Manipulator .....	26
Force Feedback .....	26
Vibrotactile Transducer Control.....	27
Design Choices .....	27
EMG.....	27
Robotic Manipulator .....	28
Force Feedback .....	28
Vibrotactile Transducer Control.....	28
Testing.....	28
Test Plan.....	28
Test Specification .....	29
Test Cases.....	30
Results.....	31
Error Analysis .....	35

Summary .....	35
Conclusion.....	35
Appendices.....	36
Appendix A: Bill of Materials.....	36
Appendix B: Schematics.....	37
Appendix C: Gant Chart.....	38
Appendix D: Sources .....	<b>Error! Bookmark not defined.</b>
Appendix E: Code .....	39
Tactor and EMG Control Module Code.....	39
Manipulator and Force Control Module Code.....	44
References .....	49

## ABSTRACT

This report details the design of a remote vibrotactile feedback system for upper limb amputees. The project went from conception to working prototype in only five weeks including the formulation of specifications, a design, and implementation and testing. The project was centered around an MSP430 microcontroller, and utilized various discrete hardware elements. It was rigorously tested against various test cases to verify functionality. These tests verified that the design met overall system specifications for performance and functionality, including timing. Unfortunately, the system drew much more power than originally desired, a side effect of the vibrotactile transducer and servo that had to be used to guarantee accurate outputs. There was also a scheduling issue between the network and the control of the robotic manipulator. This sometimes resulted in the jittering of the robotic manipulator.

## INTRODUCTION

There are over 387,500 upper limb amputees in the United States of America, with hundreds of thousands more around the world. The causes range from traumatic injury to circulatory disorders to birth defects. While prosthetics have made vast improvements in both dexterity and cosmetics in the past decade, upper limb amputees have expressed the need for sensory feedback to aid daily motor tasks and object manipulation, yet tactile feedback in upper limb prosthetics is still a relatively infant technology. The purpose of this project was to develop a prototype to provide remote vibrotactile feedback to upper limb amputees. Further goals of this project are to provide the afore mentioned functionality in a low power efficient battery operated system that is robust enough for use on the human body, while maintaining economic affordability (typical prosthetics range from \$10,000 to over \$57,000).

When a person moves an appendage, electrical activity is produced by skeletal muscles. In order to reproduce natural hand movements, this project uses electromyography (EMG) to tap into this potential and provide control to the prosthetic fingers (referred to as a robotic manipulator). The project was designed around the MSP430 RF2500, a low power microcontroller from Texas Instruments (TI). Because users do not want to hassle with wires, wireless transmission of data between the robotic manipulator to the device on the body was completed using RF communications. Force sensitive resistors (FSR's) on the fingers provide feedback to the user about the amount of pressure applied. This feedback is reported in the form of a vibrotactile located on the upper arm.

This project was taken from conception to functioning prototype in the course of 5 weeks. This included formulating a system requirements specification, a design specification and implementation and prototyping. All software for the system was coded in C using Code Composer Studio 4, a free IDE provided by TI and modeled around Eclipse. Multiple pieces of hardware including digital to analog and analog to digital converters, op-amps and diff-amps were utilized to provide the required functionality. Various pieces of lab equipment were also utilized in the development of this project including a Tektronix logic analyzer and oscilloscope, along with a digital multi-meter, and DC power supply.

## DISCUSSION OF THE LAB

### Requirements Specification

#### System Description (Summary)

This specification describes and defines the basic requirements for a remote vibrotactile feedback system for upper limb amputees. The system is to be able to measure the force with which a prosthetic finger grips or presses on an object and transmit this force wirelessly to a controller located on a remote location on the body. Surface EMG electrodes are attached to the user to enable the remote manipulation of a servo controller end effector. The controller must then relay force exerted by the end effector to a vibro-tactile transducer to provide feedback the user. The system is to be low cost, low power and robust enough for use on the human body.

#### Specification of External Environment

The system is to operate in a consumer environment, both indoors and outdoors, under residential temperature and lighting environment. The unit will scavenge solar power as well as battery operation.

#### System Input and Output Specification

##### Inputs

The system shall be able to measure the following signals. An actuating force from a robotic limb, and EMG from a user forearm.

Actuating Force:

- Minimum: .1 N
- Maximum: 10 N
  - Tolerance:  $\pm .05$  N

Electromyography (EMG):

- Minimum : 100  $\mu$ V
- Maximum : 2 mV
  - Tolerance:  $\pm 50$   $\mu$ V

##### Outputs

The system shall measure and output the following signals. The vibrations go to a vibrotactile to provide force feedback while the PCM relays the actuated force input to a servo for control of a simulated robotic hand.

Vibrations (as sine waves at 250 Hz):

- Minimum: 0 V
- Maximum: 5 V
  - Tolerance:  $\pm 100$  mV

#### Pulse Coded Modulation (for controlling the gripping servo)

- 1 to 2 ms pulses @ 3.6 V
  - Pulse width Tolerance:  $\pm 10\mu\text{sec}$

## User Interface

For this system prototype the user shall be able to send measurements from an EMG sensor to a prosthetic limb and receive pressure feedback via a tractor. The EMG sensor will consist of 3 discrete sensors, one or around the elbow to provide ground, and two above the bicep to provide a voltage difference. A tactor will also be located on the body, most likely on the chest or upper arm. A diagram of the setup for this user interface is provided below

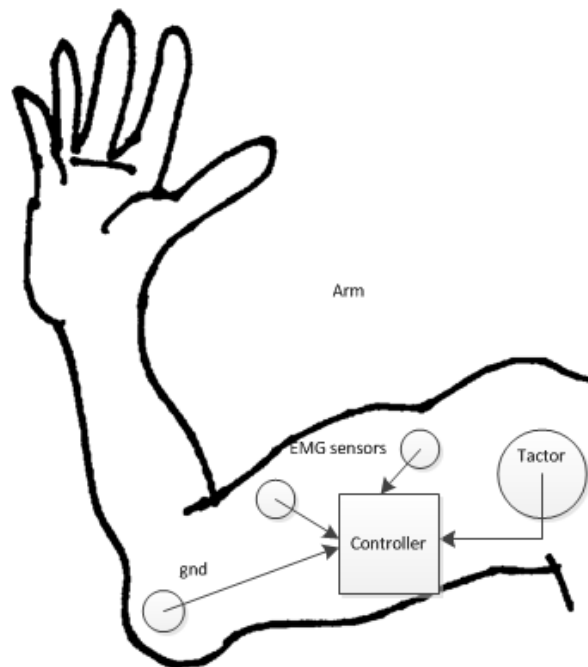
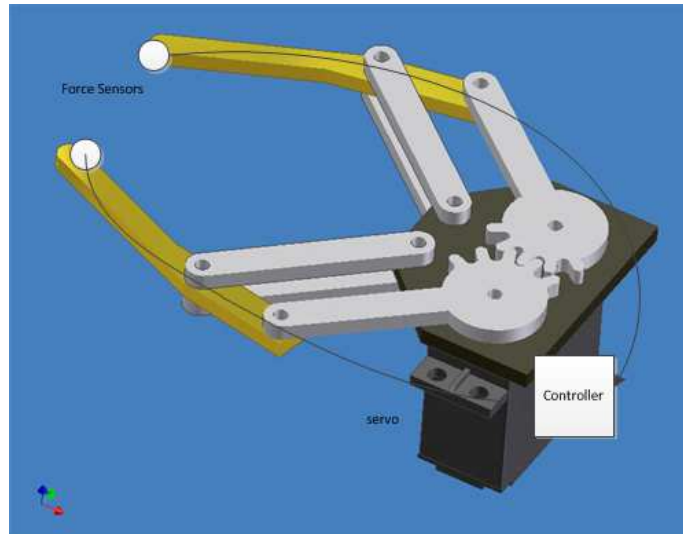


Figure 1: Local Interface

The data collected from the EMG sensors will then be used by a remote robotic manipulator pictured below. The EMG sensor will control the position of the arms, and the user can manipulate the location and orientation of the manipulator in order to utilize the arms.





**Figure 2: remote interface**

For this initial prototype the user will have power of the following switches located on the local controller:

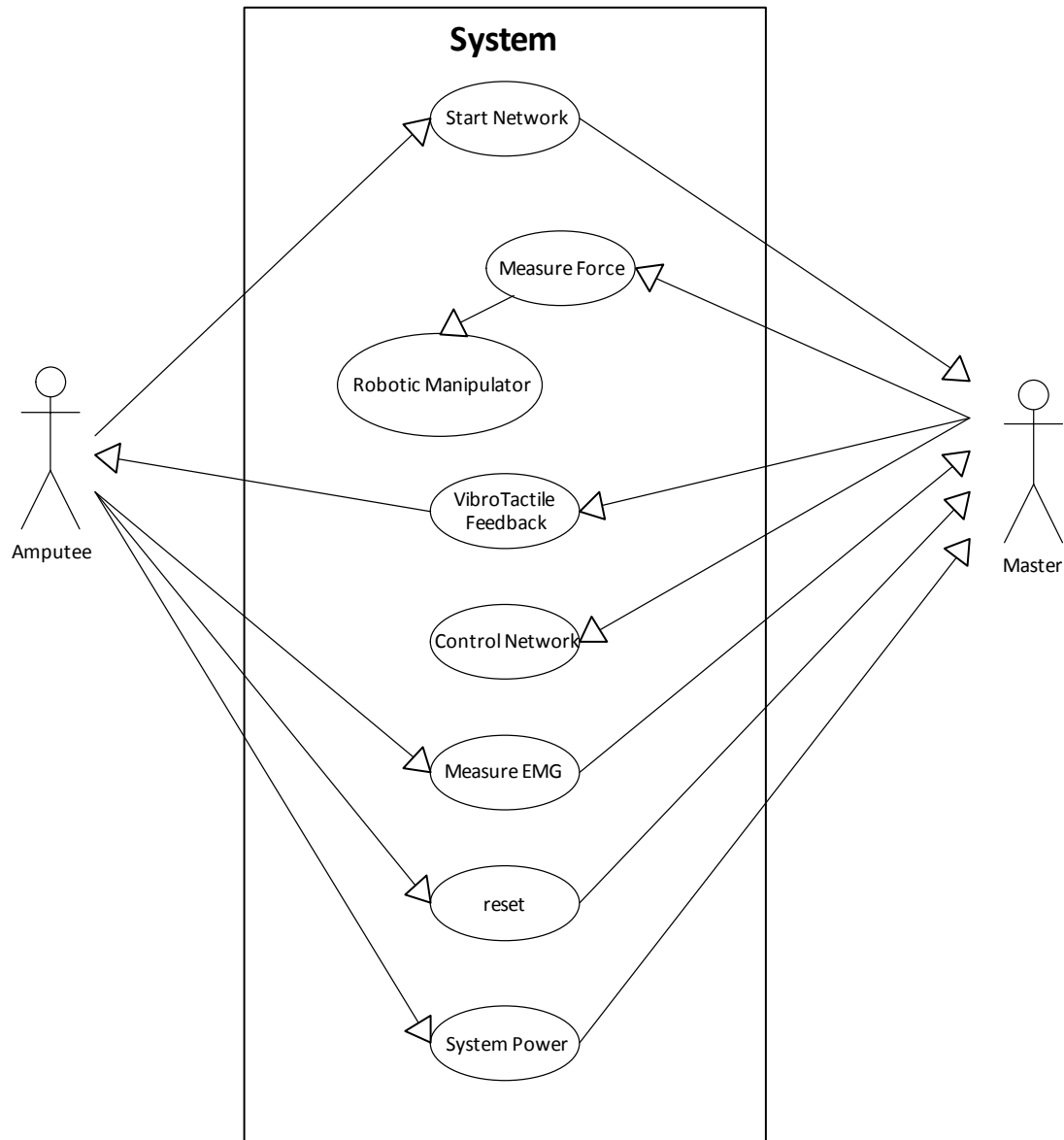
- Power: The user will be able to conserve power by turning the system on or off for extended periods of time such as at night.
- Feedback: The user will be able to enable or disable the tactor using a switch. This will block it from vibrating at all regardless of the amount of pressure on the system.

The user will also interact with the system via four LEDs, two on the remote manipulator and 2 on the local controller. These buttons will display the following status:

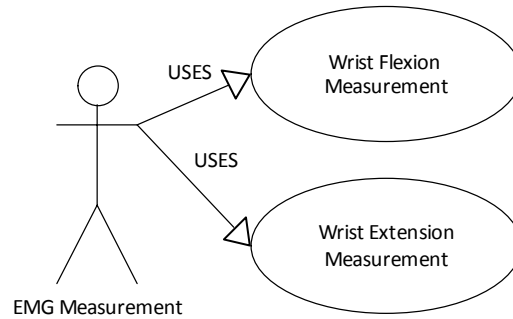
- LED1: Network Connectivity
- LED2: Robotic error
- LED3: Feedback Error
- LED4: ON/OFF

## Use Case Diagrams

The Use Cases for the system are given in the following diagram. The user can make use of either the system power push button or the EMG measurement, to power up/down the system or control the robotic manipulator respectively, upon operation the robotic manipulator makes use of the user's muscular activity to open or close a gripper around a plastic cup, the force measurement module then sends the gripper force to a feedback module which uses this data to deliver proportional vibration to the user.

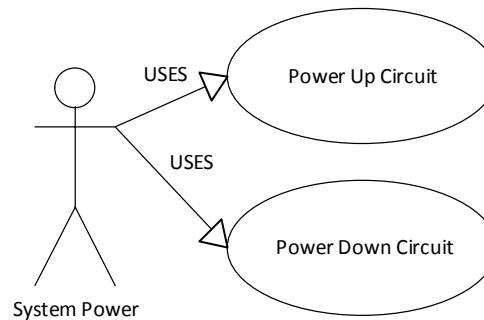


**EMG Measurement** the EMG measurement system will be comprised of two EMG surface electrodes, referenced to the user's elbow as the ground node. The user will have the ability to flex and extend his wrist to control the gripping force in a robotic manipulator. These electrodes produce a voltage difference, which is then amplified and cleaned of noise.



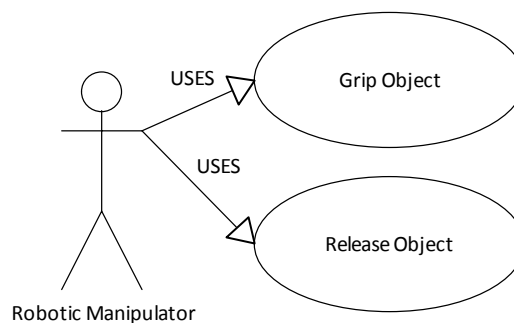
Exception: If the system fails the user will need to reconnect the electrodes at further distances to create a large potential. If the system still fails to register this, reset the master module.

**System Power** the system power module involves the use of a push button to power down the system and put the hardware into the lowest power consumption mode, and when pushed again the user will be able to reactivate the system to its normal operation.



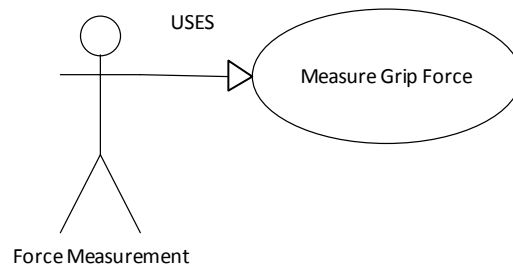
Exception: If the system continues to operate at full power after it has been powered down, or operate at power down even though it has been repowered, use the reset. This will take it back to initial stages and reboot the system.

**Robotic Manipulator** the robotic manipulator will be controlled directly by the user's flexion and extension and will provide a gripping force against a fragile plastic cup, upon successive training trials the user will be able to control the amount of force and grab the cup without breaking it.



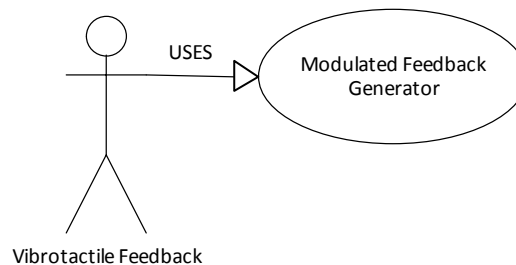
Exception: if the manipulator fails to properly grip or release, the user will need to try again. If it continues to operate an LED will flash at a one second rate indicating to reset the system.

**Force Measurement** the force measurement system will record forces at the tip of the robotic manipulator and increase relative to the pressure applied to the plastic cup.



Exception: if the force is not properly relayed an LED will turn on. In this case the master will reset the slave that is driving the force sensor and try again.

**Vibrotactile Feedback** The system will continuously use the data provided by the force sensors in real time to deliver vibration as a frequency of differing amplitudes to a tactor located somewhere on the human body.



Exception: if the feedback is not properly received it will flash a red LED on the master indicating to restart the system.

**Start Network** The start network is triggered on power on or reset. It tells the master to initialize the RF network with the slave.

Exception: If the network fails to initialize, the master will continue to try to find the remote node indefinitely. Make sure the remote has power, and reset the system.

**Control Network:** The local node will control the RF network by sending and receiving data from the remote node. It will have the power to put the slave in low power mode.

Exception: If control network fails reset the system.

**Reset** the reset command will take the system back to its initial state.

Exception: if reset fails, unplug the batteries, wait 10 seconds reinstall them and try again.

## Timing Constraint Requirements

The system must provide feedback with no noticeable lag. For this design it has been determined that a 150 ms delay is the limit for having the user feel an instantaneous reaction.[1] Therefore the user must be able to activate the emg and receive back feedback within 150 ms.

## Operating Specifications

The system shall operate in standard residential conditions.

- Operating temperature: 32° to 95° F (0° to 35° C)
- Nonoperating temperature: -4° to 113° F (-20° to 45° C)
- Relative humidity: 5% to 95% noncondensing
- Maximum operating altitude: 10,000 feet (3000 m)
- Power: 3.3 VDC, with a minimum battery life of 400 transmissions on one charge(doesn't apply).

## Reliability and Safety Specification

The counter shall comply with the appropriate standards

Safety: UL 60950-1, IEC 60950-1, CSA 60950-01

EMC: CISPR-11, EN55 011

MTBF: Minimum of 2 years

## Design Procedure

### Summary

The design of this project was influence by the specific inputs and outputs that the system was to envelop. This included the acquisition of force exerted as well as the electromyography signal from the user. The system needed to include hardware to process these inputs and components to generate the corresponding outputs. The highest level constraint that needed to be addressed was human perception of both the end effector control and vibrotactile feedback. The input to output propagation delays needed to be less than what the user would be able to detect. This value is found to fall within 150 ms to 200 ms [1].

### High Level View

Below is a high level block diagram for our system. It is a symmetric design that involves the acquisition of two inputs, their respective processing and then the appropriate output signal generation. Note that the dashed lines represent wireless communication between the main control modules. TECM stands for Tactor and EMG Control Module and MFCM stands for Manipulator and Force Control Module.

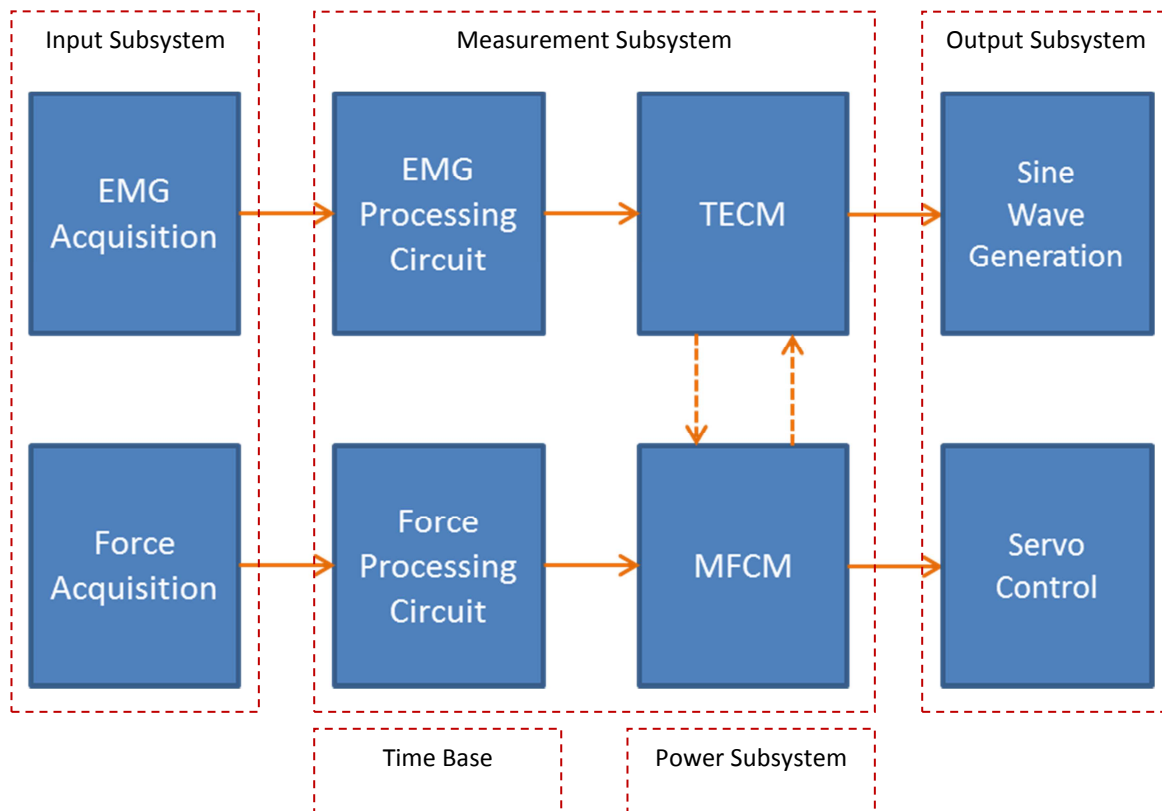


Figure 3 High Level Block Diagram of System

## Pseudo Description of Functions

**Input Subsystem** – The input subsystem enables the acquisition of 5 total signals. They include 3 electromyography signals garnered from the actuation of the user’s wrist, and 3 force signals taken at the prosthetic end effector. These signals are then routed to the measurement subsystem.

**Measurement Subsystem** – This subsystem is comprised of two 10-bit analog to digital converters that are responsible for transforming the voltages levels obtained from the input system into transferable digital signals. It also consists of two power amplifiers used to regulate the EMG signals. The measurement system is responsible for the respective transmission of data to the output system.

**Output Subsystem** – This subsystem is responsible for the generation of output signals to the two user feedback locations. This includes the vibrotactile transducer (tactor) and the prosthetic end effector. The two outputs will be a variable sine wave to the tactor and a PCM signal to the servo control system.

**Time Base** – The Local and Remote microcontrollers envelop an internal timing system that allows for various clocks and timers. This particular system utilizes two timers clocked at 8 MHz.

**Power Subsystem** – The TECM and MFCM are powered using 2 1.5 V AAA batteries each. The EMG processing circuit uses 4 1.5V AA batteries. All other modules use a 5 V supply for power.

## Timing Constraints

As mentioned, the main timing constraint adhered to was one regarding the time in which a human being would be able to notice a delay in either feedback or manipulator transmission. With this constraint in mind, it was necessary to run the microprocessors at a frequency fast enough such that the wireless transmission would enable seamless data transfer of force feedback and end effector control. Other supplementary timing constraints included those imposed by the digital to analog and analog to digital converter modules. Each component had unique timing parameters necessary to perform the appropriate calculations. These specific values are discussed further in the following sections. Links to the corresponding data sheets are also provided in the appendices.

## Error Handling

The error handling for this project was an imperative inclusion as it involves the integration of electronic signals and the human body. At any point the user should be aware if the system has malfunctioned. This is addressed by using the two main modules' on board LEDs as a mechanism for alerting the user to potential faults in the system. At any time should transmission between the two modules cease, the LEDs on the TECM node will sequentially flash every second until communication is restored. As the main functionality of the system is accomplished via the wireless transmission protocol, other errors such as the malfunction of the servo control the end manipulator will trigger this error handling event.

## Software Implementation

### High Level Architecture

The figure below shows the high level architecture of the system as broken into modules, classes and specific tasks.

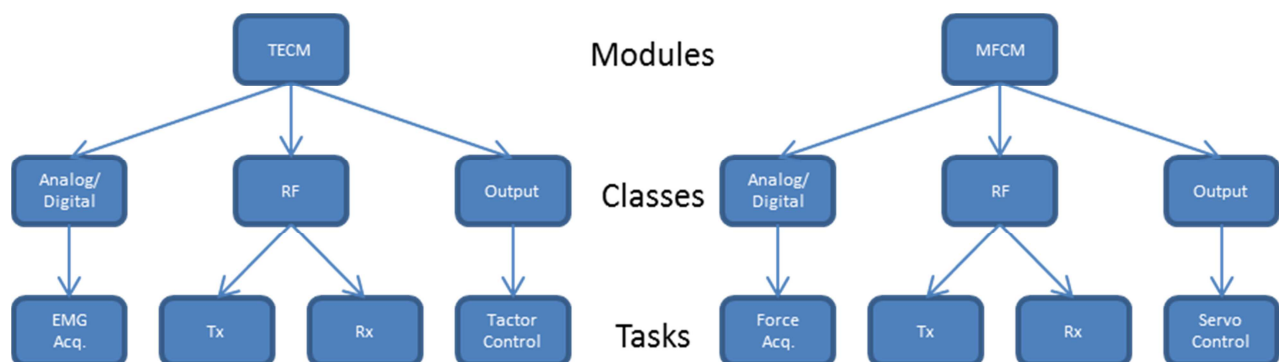


Figure 4 Software Architecture Diagram

### Functional Blocks

The table below provides insight into each of the functional blocks shown above.

**Table 1** Software Functional Blocks with Descriptions

<i>Component</i>	<i>Module</i>	<i>Class</i>	<i>Description</i>
<b>Tactor and EMG Control Module (TECM)</b>	-----	-----	Module that garners EMG signal and controls sine wave output
<b>Analog to Digital Class</b>	TECM	-----	Responsible for EMG acquisition
<b>Radio Frequency Class</b>	TECM	-----	Responsible for transmit and receive classes
<b>Output Class</b>	TECM	-----	Responsible for vibrotactile transducer output
<b>EMG Acquisition Task</b>	TECM	A/D	Garners EMG voltage and converts to digital value
<b>Transmission Task</b>	TECM	RF	Transmits EMG digital value to MFCM
<b>Receive Task</b>	TECM	RF	Receives Force value from MFCM
<b>Tactor Control Task</b>	TECM	Output	Generates unfiltered 9-point sine wave for Tactor
<b>Manipulator and Force Control Module (MFCM)</b>	-----	-----	Module that garners Force signal and controls servo output
<b>Analog to Digital Class</b>	MFCM	-----	Responsible for Force acquisition
<b>Radio Frequency Class</b>	MFCM	-----	Responsible for transmit and receive classes
<b>Output Class</b>	MFCM	-----	Responsible for servo control output
<b>Force Acquisition Task</b>	MFCM	A/D	Garners Force voltage and converts to digital value
<b>Transmission Task</b>	MFCM	RF	Transmits Force digital value to MFCM
<b>Receive Task</b>	MFCM	RF	Receives EMG value from MFCM
<b>Servo Control Task</b>	MFCM	Output	Generates PWM output for servo control

## Discussion of Specific Tasks

### EMG Acquisition

The EMG signal acquisition of the system was mostly implemented using analog circuitry. However the digitization of the analog signal was done using the 10 bit digital to analog converter integrated in the MSP430F2274. This module was configured to execute conversion using the least power demanding characteristics, the Analog to digital conversion was configured on pin 2.0 of the board, the reference was taken from Vcc and the module was configured to be able to put the system in low power mode while the conversion was running and wake up based on an interrupt to resume operations, the digital results from the conversion were then calibrated to be as close as possible to the expected values. Following this the ADC module was turned off and the value returned by the EMG task was the digital EMG value.

### Wireless Protocol

The transmission protocol for this system was designed to allow for bidirectional communication. Both nodes were required to send and receive data therefore acting as transceivers, one node the Manipulator and Force Control Module (MFCM) was required to receive force and transmit EMG data, and the other node, the Tactor and EMG Control Module (TECM) was required to receive EMG data and transmit force. The peer to peer scheme was used for communication with the advantages of direct linkage between nodes, personalized addressing, interrupt driven reception of packages with receive function subscription and an increased speed of transmission without the continuous drops of packets



experienced using different schemes. The usage of this scheme required us to synchronize the linking of both nodes upon initialization of the network. The radio was also turned on to allow for reception after a successful link was established. After this set up was completed each node was ready for bidirectional transmission and reception of data.

### Wireless Transmission

Wireless transmission was done similarly in both nodes with some minor differences. In the node MFCM transmission was performed continuously only restrained by the use of a network delay to determine the rate of transmission, following this, data was broken up into two bytes which constituted the messages to be send and then a predefined function command was used to send data. In the TECM node the main difference was the implementation of a semaphore, which was needed since the MFCM node started transmission and so it will always transmit first, so in the case that a packet was dropped the TECM node could be pre emptied in the middle of its transmission state thus producing unstable results, therefore a semaphore was used to prevent transmitting and receiving at the same time.

### Wireless Reception

Wireless reception was done through subscribed functions that will get called upon reception of a packet from the other node, in the MFCM node a quick check was performed to certify the port from which the frame was received, after checking that the port was correct the predefined function receive was used and its return value was checked to make sure that the received data was correct, after doing so the two bytes in the message were concatenated and the data was interpreted as a force measurement. At the other end at the TECM the only difference was the implementation of the semaphore to protect the receiving and transmission operations to be performed at the same time.

The pseudo code for both the transmit and receive tasks are provided below.

```
void main()
{
    if (MCU clock values are not initialized)
    {
        // dont run and start alert routine
        // flash LED's
        // then enter low power mode to save energy
    }

    // blink LED 1 & 2 to indicate start up

    // configure self RF address
    if (self address is missing)
    {
        // create new random address
    }

    // use pre defined BSP init function to
    // initialize eZ MSP430 hardware.

    // start the very low power clock using
    // control register 3

    // set timer B control to capture compare mode
    // enable interrupts on match

    // set the value to compare to using CCR0 register

    // initialize the simplicity protocol
```

```

    // try to join the wireless network.

    // upon failure go to low power mode and wait
    // for next time up interrupt

    // wake up the radio module and make sure that the
    // access point node is listening for messages

    // call the link function which takes
    // care of the periodic wireless transmissions.
}
void link()
{
    // perform a link attempt operation
    while (transmitter cannot link to receiver)
    {
        // keep trying to link and use timer B
        // value in CCR0 to wake up periodically and attempt
        // again
    }

    // execute forever
    while (1)
    {
        // send radio to sleep while other operations
        // are executed
        // perform analog to digital operations

        // now wake up the radio,
        // keep a running count of the number of packets to be
        // sent

        // use predefined function to send packets

        if (sending was successful)
            // delay and flash LED's in
            // good status patterns to indicate the user
            // packets are sent

        else if (sending was unsuccessful)
            // flash LED's in bad status pattern

            // now, send the radio to sleep using pre defined function

            // put the MCU in low power mode, to sleep until next transfer
            // time is up, leave interrupt for timer B on to know when to
            // wake up

            while (the push button at the receiver has not been toggled)
                // turn of the system completely until the button is toggled
                // again.
    }
}

For receiver
Initialize radio address location
Stop WDT

//interrupt driven
if adx match
    if new data
        take semaphore
    if semaphore
        get data
        release semaphore

```

## Vibrotactile Transducer Control

The tactor control was realized by generating an initial low amplitude sine wave. This waveform was stored in a 9 element array called a look up table. As it was necessary to generate a unique frequency, specifically 250 Hz, certain calculations were required. These included calculating the period of the sine wave and the time sampling time in which the next point in the array would be processed.

$$\tau_{clk} = \frac{1}{f_{clk}} = \frac{1}{250 \text{ Hz}} = 4 \text{ ms}$$

$$\tau_{sample} = \frac{\tau_{clk}}{\# \text{ of points}} = \frac{4 \text{ ms}}{9} = 444 \mu\text{s}$$

This means a new point from the look up table needs to be processed every 444  $\mu\text{s}$ . The MSP430 microcontroller's extensive interrupt system enabled this functionality. Using a timer driven interrupt, each point was processed and sent to the output pins of the MSP430 and then fed serially into a high speed digital to analog converter. The pseudo code below provides insight into this process.

```
sineWaveGenerator()
{
    //Create sine wave look up table
    //Initialize interrupts
    //Set appropriate digital output direction

    If(interrupted){For (current index of sine wave)
        {
            //Set output to value at current index using Dac function()
        }
    }
}
DAC function()
{
    //Create control word and data
    //Set ~CS low
    //Shift MSB into register
    //Set SCLK high
    //Set ~CS high
    //Set SCLK low
}
```

## Robotic Manipulator

The robotic manipulator control was realized by creating a PWM in the remote microcontroller. This was done by setting up two timers, one running every 20 ms, and one at a variable length of 1-2 ms. Every 20 ms the micro controller would generate a high pulse of 1ms-2ms depending on the second timer, with 1ms corresponding to a servo position of 0 degrees and 2ms 180 degrees. It is best described by the following pseudo code:

**Manipulator pseudo Code:**

```
Initialize 20 ms timer
Initialize variable timer
Get digital EMG signal from RF coms
//Convert to servo control
servoVal = 15*EMG + 5000
set variable timer = servoVal

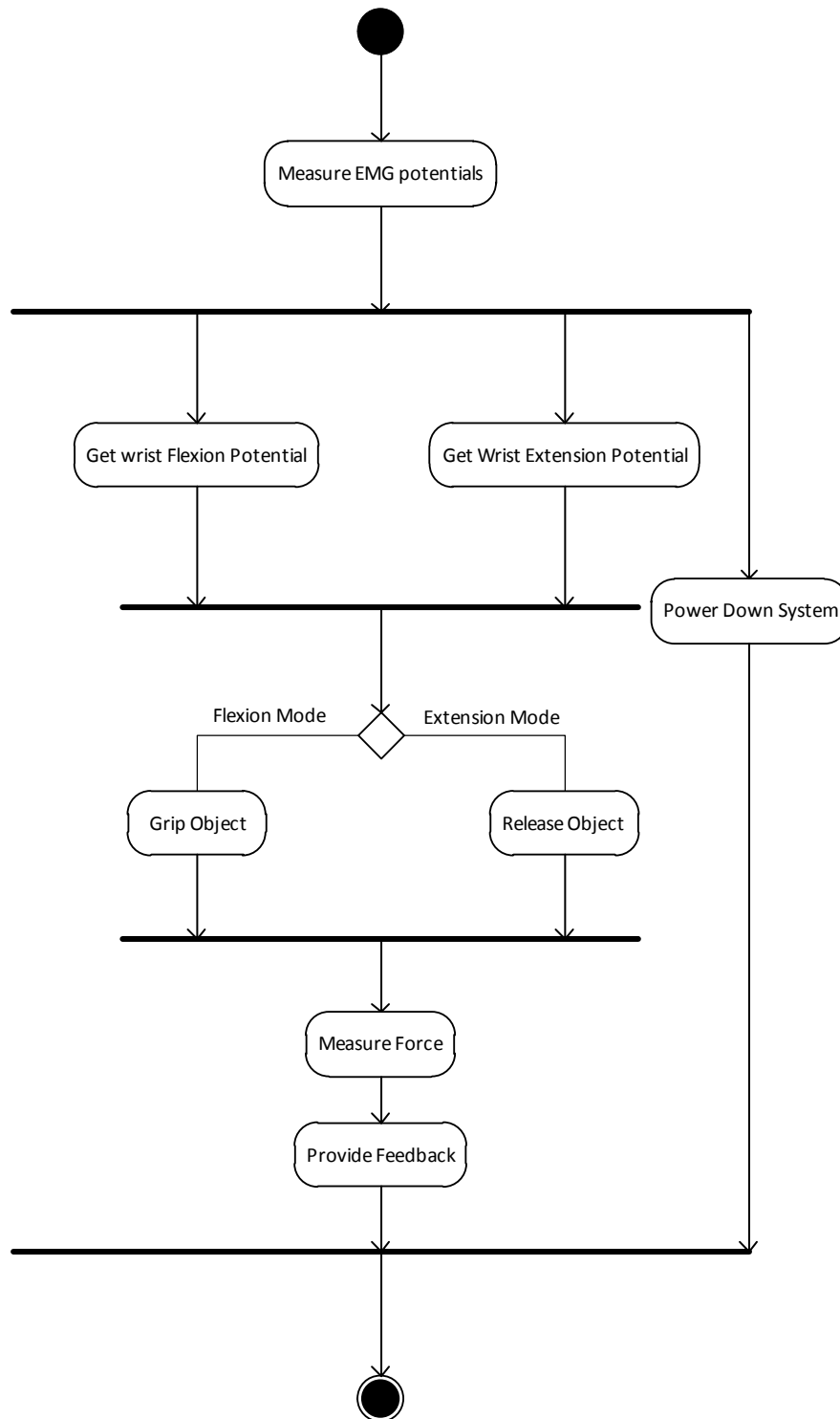
every 20 ms set digital out high
pull digital out low after variable
timer time
```

**Force Feedback**

At the end of each arm of the robotic manipulator is a FSR which provides back the amount of pressure exerted. For the software segment of this, the analog value is sampled every 100ms and converted from analog to digital, and then transmitted through the RF network.

**Flow of Control**

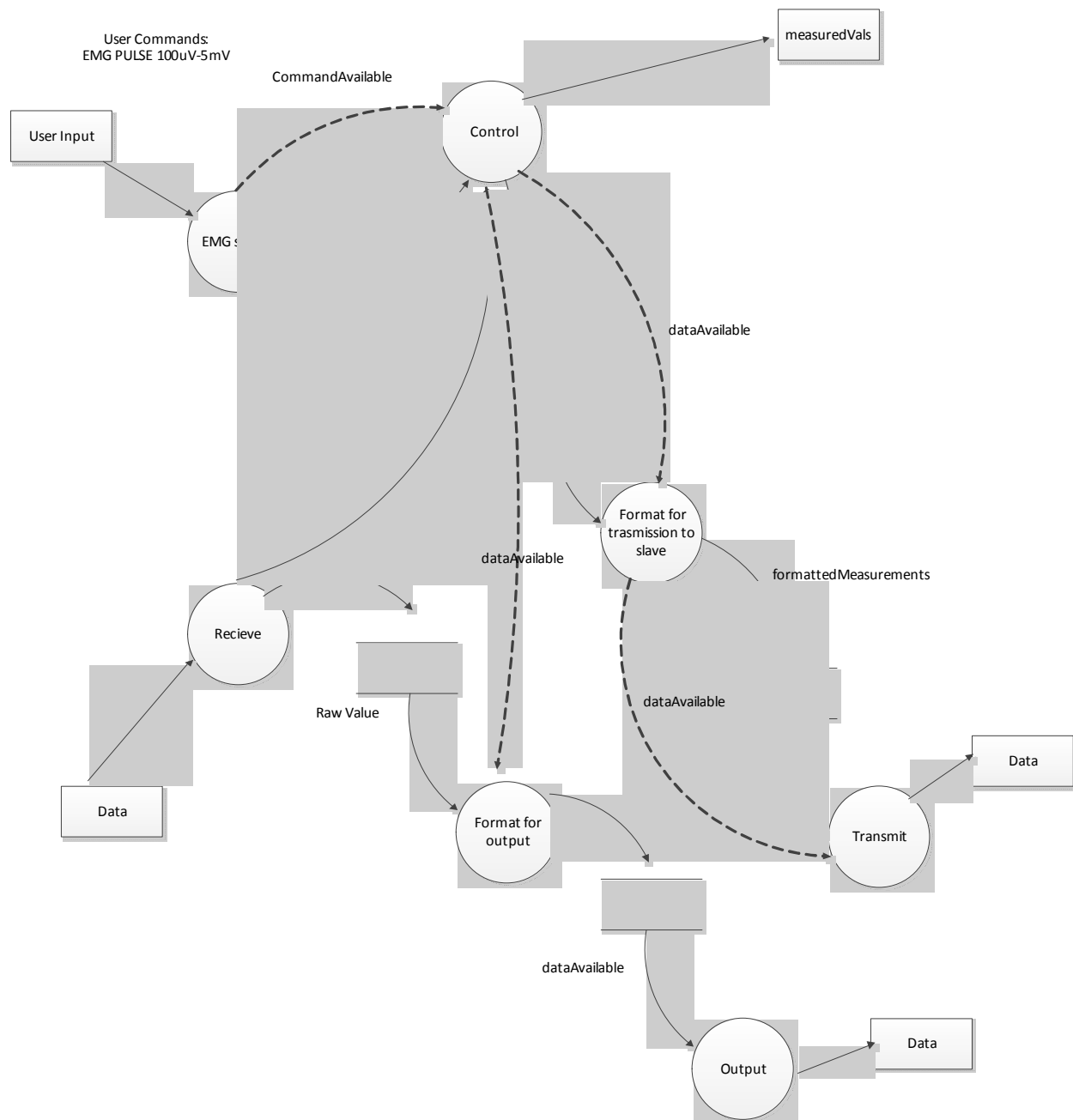
The system is to take analog force and electromyography signals and then route them to the appropriate location in the output subsystem. The activities associated with the system are shown in the following diagram.



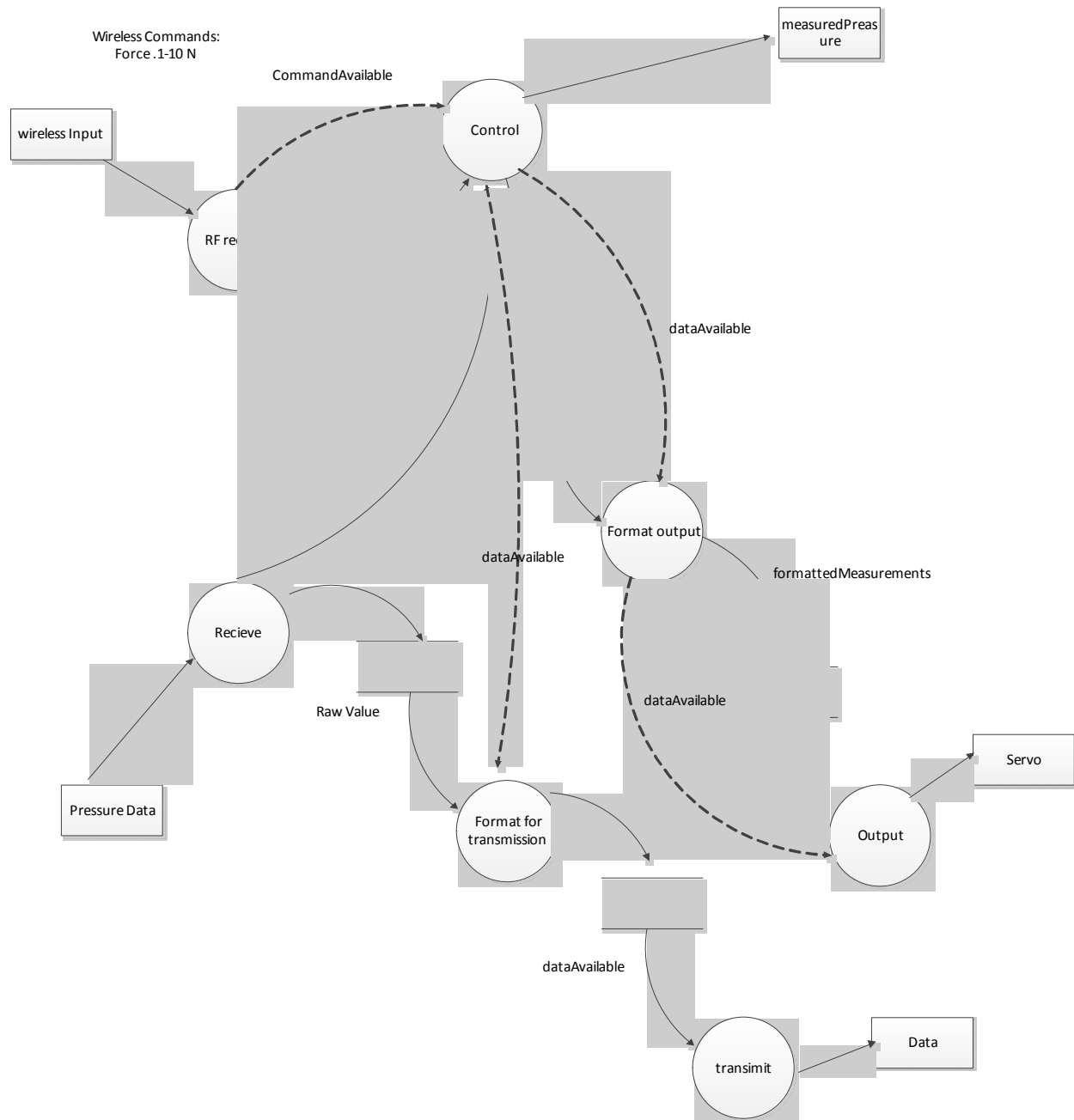
## Data and Control Flow

Below is the data and control flow for the master. It takes in EMG pulse from the user via an EMG sensor. This raw value is stored then converted for wireless transmission. Once it is formatted it is

transmitted to the slave node. Once the slave sends back data it goes to the receive process. The control then formats this raw value and outputs it to a factor on the human body.

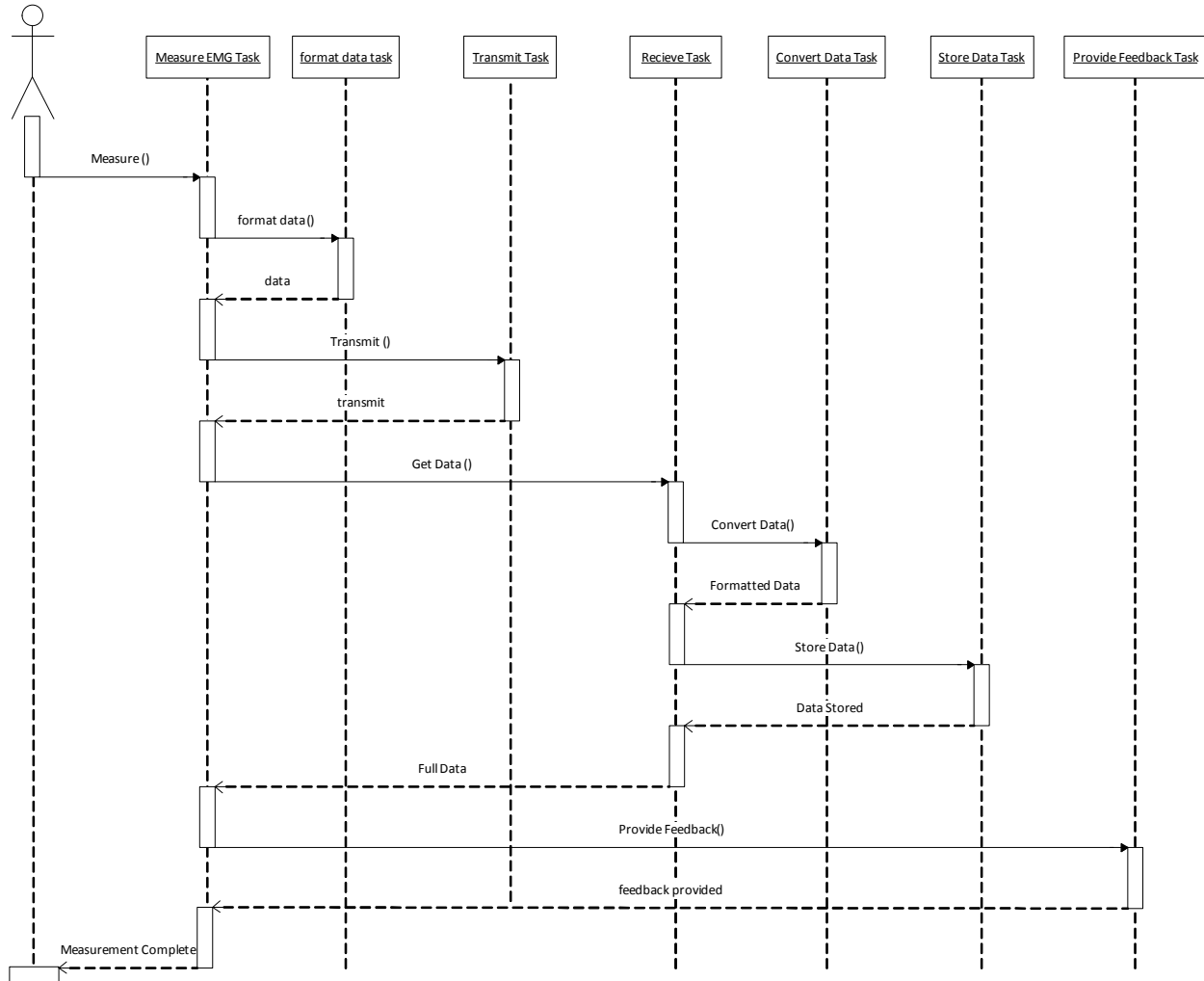


Below is the flow of control for the slave. Its behavior is very similar to that of the master. Instead of user input, it takes in the transmitted data from the master, formats that data to control a servo. Then it also takes inputs of pressure from sensors located on the servo arm, and formats those and send them back to the master.



## Sequence Diagram

Below is a sequence diagram for the parent system. It shows how a measure input from the user is propagated through the system. It shows how each task gains control and preforms its operation and then control moves on to the next task once it is done.



## Design Choices

### EMG

A few considerations were taken when converting the analog EMG data to its digital counterpart, a low power implementation of this conversion was used to reduce the burden of power consumption on batteries since the MSP430's are run on AAA batteries. This tradeoff between power consumption and speed was realizable due to the fact that sacrificing some of the conversion speed in this case was not as critical since the upper bound on the overall timing delay of the system was human perception which ranges from 150 – 200 mS [1] much greater than the conversion time . Another important design choice in this section was the usage of an interrupt to wake up the system after conversion was completed. This interrupt was used with the same purpose as the reduction in speed; it was implemented to reduce power consumption by allowing the system to go into sleep mode until the A/D conversion was completed.



## Wireless Protocol

As explained in the software implementation section, we needed bidirectional transmission with direct linkage between nodes, fast reception of packages and minimization of the number of packets drops. First the decision for the type of microcontroller and transceivers was faced where it was decided that an integrated microcontroller with an integrated antenna was the best approach, at first an attempt was made to find this device from Microchip due to our experience with their tools, however their wireless development board was more expensive than other similar products and not many low power capabilities were found. Therefore other low power alternatives were explored, with the MSP430F2274 integrated with the CC2500 transceiver into the EZ430F2500 which was chosen as the finalist. Further research pointed to three different possible wireless configurations: The access point end device configuration, the end device end device configuration and the peer to peer configuration all with their advantages and drawbacks. The access point end device configuration proved to be very fast but lacked bi directional communication and so was discarded from the options. The end device end device configuration did not establish direct linkage between nodes but instead worked with a series of fast transmissions which were issued repetitively and no acknowledgement was required, this resulted in a considerable amount of lost packets when decreasing speeds below a certain threshold. Therefore the Peer to peer configuration was used with bi-directional communication, improved speed and a minimum amount of dropped packets.

## Wireless Transmission

Two design decisions were made for wireless transmission; the transmission speed was finally set to be 25ms. This decision was based on reducing the transmission delay to a minimum but still maintaining a constant transmission of packets and a constant operation of the other operations than needed to be run concurrently when transmitting. At the other end (TECM) the implementation of a semaphore was done to prevent pre-emption of the transmission code when the reception interrupt was triggered which would cause problems in synchronization. This decision proved to be important when the rate of transmission was increased to 25 ms.

## Wireless Reception

Reception of data was done interrupt driven; this allowed for a reduction in power consumption since now the reception code was only executed when a frame was ready to be received. The subscription of the receive function upon the reception of new packages gave us the flexibility to turn the whole system on or off by simply setting the radio to idle mode and then turn in it back on. This ability allowed us incorporate a push button where the user could turn the system off (e.g. when going to sleep) and put it in lowest power mode and then turn it back on when needed.

## Vibrotactile Transducer Control

Initially, the use of PWM generation was being used to create the desired sine wave to drive the vibrotactile transducer. This soon proved to be inefficient and slow. Instead, a look up table was used to create the initial output that would then be modulated based on the force seen at the end manipulator. This method was more elegant and effective than the initial PWM scheme as it did not take as much post processing circuitry.

## Robotic Manipulator

The robotic manipulator was built using a MPI MAXX servo. This specific servo was selected because it utilizes metal gears which provide the torque necessary to power the arms without slipping. While the servo has 180 degrees of motion, human fingers only move roughly 90 degrees. In order to give more control over the fingers, rationed gears were used so that the servo would move through its full range of motion, but the fingers only half that.

## Force Feedback

In order to provide accurate feedback a total of three FSRs were utilized (one at the tip of each finger). Due to limited A/D channels on the msp430, these forces were averaged together in hardware and sampled at one A/D port. This increased the overall accuracy of the pressure measurements, based upon or test case of picking up a plastic cup, but for cases where only one finger experienced force this setup would not provide accurate feedback.

## Hardware Implementation

### Block Diagram

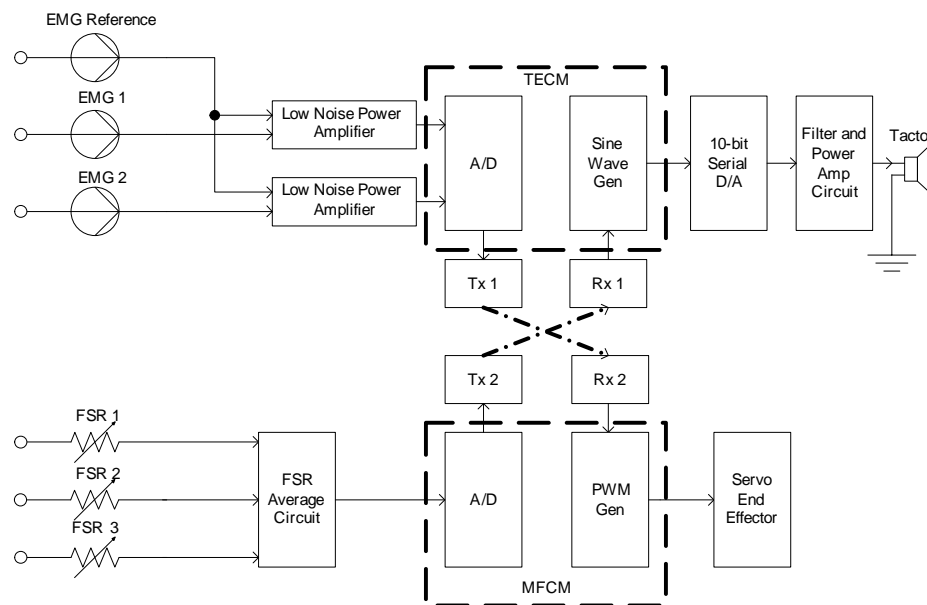


Figure 5 Hardware Block Diagram

## Discussion of Modules

### EMG

The hardware part of the EMG circuit involved the design of an EMG acquisition circuit which was built using discrete analog components. The Motor Unit Action Potential (MUAP) was recorded in two places: Biceps and Flexor Carpi Ulnaris. The difference in potential was measured across two points along the targeted muscle referenced to a ground point which was chosen to be the bony part close to your elbow. The signal gathered from the two muscle electrodes was sent into two buffers to isolate the

input signal and provide driving capacity. Then these pair of signals was introduced into a differential amplifier and the common mode rejection ratio was chosen to be as high as possible (CMMR ~ 90) to reject common mode noise (mostly 60 Hz from electrical wiring). Then the signal was amplified using an inverting amplifier scheme to put it as close to rail as possible under maximum input signal amplitude. This amplified signal was passed through a bypass capacitor to center the signal at 0 volts. Then a further inverting amplifier was added to invert the signal back to its original state and take the amplitude as close to negative and positive rail. Then an active full wave rectifier was used to invert the negative peaks of the signal, after that an active inverting low pass filter in an inverting configuration with a cutoff frequency of ~3 Hz was used to take the running average of the signal. Finally another inverting amplifier was used to bring back the signal to its original state and a potentiometer was put in the feedback loop to provide variable gain for targeting different muscles. A calibration circuit was also added to provide input signals at very low amplitudes for testing of the circuit without the need for a human subject.

### **Wireless Communication**

The Ez430-RF2500 has a complete integrated interface between the MSP430 microcontroller and the CC2500 transceiver (ISM band multi-channel low power, low range transceiver 2.4 MHz – 2.4835 MHz). The CC2500 transceiver has an integrated ceramic antenna and communicates with the MSP430 microcontroller through an SPI protocol, The CC2500 also has integrated analog circuitry to allow for frequency hopping and multi-channel operation as well as frequency compensation and an integrated analog temperature sensor. The Ez430-RF2500 also has the ability to operate from 1.8 to 3.6 supplies and can put both the transceiver and itself in sleep mode and low power mode respectively.

### **Robotic Manipulator**

For the manipulator built in timers A0 and B0 were utilized to provide the timing of the digital out to the servo. A0 counted to 20ms, while B0 These had various control registers which were set in software to provide the correct period.

### **Force Feedback**

In order to provide accurate feedback a total of .5 inch three FSRs were utilized (one at the tip of each finger). Due to limited A/D channels on the msp430, these forces were averaged together in hardware using the circuit provided below. It is a simple passive average where the voltages are provided by the FSRs. The resistance values were chosen to cancel each other out and limit current into the A/D converter, in this case 10 kilo ohms each.

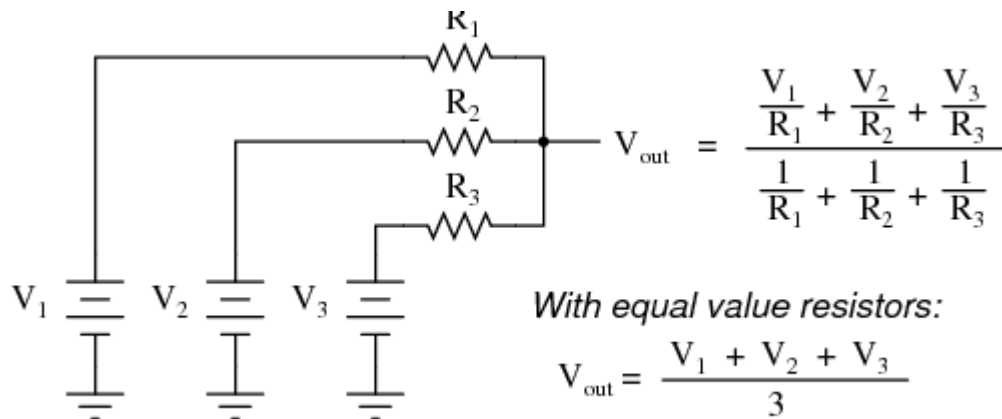


Figure 6: averaging circuit source: all about circuits.com

### Vibrotactile Transducer Control

The hardware involved in the tactor control consists of various filtering and amplification elements. The first component encountered by the output of the TECM is a 10-bit high speed digital to analog converter. This is used to create a variable amplitude sine wave based on force. The output of the DAC needs to be filtered to get the desired 250 Hz smooth sine wave. This is done using a passive 2<sup>nd</sup> order low pass filter consisting of appropriate cascaded RC networks. As this signal does not have any driving capabilities it is necessary to first provide voltage amplification and then current amplification. This is done by cascading a simple non-inverting amplifier with a gain of 5 V/V and then sending that to a Darlington coupled power transistor. The final signal generated is a 0 V<sub>pp</sub> to 3.6 V<sub>pp</sub> sine wave that is capable of driving the vibrotactile transducer.

### Design Choices

#### EMG

Several design choices were made when building the EMG acquisition circuit, the first was whether to use a pre-built instrumentation amplifier with a very high set common mode rejection ratio (110) and laser trimmed resistors. This instrumentation amplifier was purchased but the group preferred to build the instrumentation amplifier from scratch using discrete op amps and resistors to learn the basic intuition behind instrumentation amplifiers, the use of differential amplifiers and gathering signals from noisy environments such as the human body. Note that a notch filter at 60 Hz was not used because a lot of useful information is stored at this frequency. The next design choice was whether to use a series of inverting amplifiers or to use non inverting amplifiers, in this case inverting amplifier proved to be a better choice since it was easier to amplify the signal coming out from the instrumentation amplifier and then centered at zero and then further amplify it to the supply rails, the same was true for the last stage where we drove the rectified signal to its maximum level and then use the last inverting op amp with a potentiometer in its feedback loop to attenuate it if needed.

## Robotic Manipulator

For the manipulator it was decided to utilize hardware interrupts to generate the PWM over using a software timer or polling. This was decided upon because the hardware provided much more accurate results, which are required for servo manipulation. If the output pulse had differed by any value, the servo would have moved at the wrong time. This would lead to the manipulator shifting around when it should be steady. The hardware timers provide the required accuracy to keep the manipulator steady.

## Force Feedback

For the force feedback it was decided to use a simple averaging circuit as opposed to a more complex one due to a combination of cost and comparative accuracy. While a complex circuit gives marginally superior results, the added costs and complexity of design did not warrant this design approach. In order to provide accurate feedback a total of .5 inch three FSRs were utilized (one at the tip of each finger). Due to limited A/D channels on the msp430, these forces were averaged together in hardware using the circuit provided below. It is a simple passive average where the voltages are provided by the FSRs. The resistance values were chosen to cancel each other out and limit current into the A/D converter, in this case 10 kilo ohms each.

## Vibrotactile Transducer Control

As mentioned previously, Pulse Width Modulation was originally used to generate the sine wave though this soon proved to be a lesser technique when compared to high speed DAC generation. The main advantage of using the later method is the elegance of code needed to accomplish the task. Also, higher order active filtering was necessary to obtain a sine wave that was still lesser quality than that garnered from simple 2<sup>nd</sup> order passive filtering. The specific DAC module was chose based on its speed in conversion settling time as well as serial input capabilities. The filter used was created based on the desired frequency and the amplification was created based on the power requisites of the transducer which stated optimal driving occurred with 250 Hz sine wave bursts at 500 mW. The final power generated by the elements constituent to the tactor control are capable of provided values around 430 mW.

# Testing

## Test Plan

In order to determine if this system is functioning correctly an in depth series of test cases were developed to verify that each subsystem and the overall system preform as expected. This includes speed, power consumption (static and dynamic) and accuracy. Each individual subsystem was isolated and fully verified using a combination of the Tektronix logic analyzer for functionality accuracy, a DMM for power consumption and operating voltages, and an oscilloscope for verifying operating speed. In order to verify that the measurements taken are accurate, replace various dynamic inputs with fixed known values and compared to the resulting values displayed on the logic analyzer.

In addition to testing for normal operating conditions, the operating limits were also tested. This includes operating voltages and input voltages.

## Test Specification

Due to the complexity of the overall system, testing was broken up into each of the subsystems. Therefore each subsystem will have individual test specifications and will build upon each other until the final overall system can be tested. These test specifications will determine whether the system is functioning as required by the system specification. The inputs and outputs of each subsystem are detailed below in individual charts. For information on how to actually test these specifications, see the test cases section.

### EMG Acquisition

INPUT	ANALOG OUTPUT	DIGITAL VALUE
Low Activity	0V	0V
Medium Activity	1V	324
High Activity	2V	621

### Servo Control

INPUT	OUTPUT(DEGREES)
1ms	0
1.5ms	90
2.0ms	180

### Force Sensing

FORCE INPUT	ANALOG OUTPUT
No Pressure	100 mV
Low Pressure	1.36 V
Medium Pressure	3.41 V
High Pressure	4.73 V

### Feedback

DIGITAL VALUE	DIGITAL OUTPUT	DAC OUTPUT	SCALED OUTPUT
---------------	----------------	------------	---------------

<b>0</b>	<b>0</b>	<b>0</b>	<b>0v</b>
<b>465</b>	465	1.5v	2.27v
<b>775</b>	775	2.5v	3.8v
<b>1024</b>	1024	3.3v	5v

## System integration

<b>EMG INPUT (peak to peak)</b>	<b>SCALED OUTPUT (peak to peak)</b>
<b>0v</b>	<b>0v</b>
<b>3v</b>	1.5v
<b>5v</b>	2.5v
<b>7.2v</b>	3.6v

## Test Cases

Due to the complexity of the overall system, testing was broken up into each of the subsystems. Therefore each subsystem will have individual test cases and will build upon each other until the final overall system can be tested. These test cases will determine whether the system is functioning as required by the system specification.

### EMG Acquisition

The EMG acquisition module is composed of three parts: the sensors, amplifier, and analog to digital converter. First check the A/D converter by directly inputting know values from a power supply, and read them into the MSP430 and output them on digital output pins at a frequency of 1MHz and display them on the logic analyzer. Verify that the corresponding digital value is equivalent to the analog values. Once this has been verified add the amplifier output to the A/D input. Input a sine wave to the amplifier and again read out the values on the logic analyzer and make sure that the gain is correct. Finally integrate the actual sensor and run the test again.

### RF Communication

To test the RF communications load predefined data into the local terminal and transmit it to the remote. At the remote terminal, output the data received onto digital outs and display them on a logic analyzer. Repeat for the remote to local. Then integrate with the human body sensor, by transmitting the digital value from the body to the remote and back to the local for display.

### Servo Control

To test the servo, output PWMs from the micro controller and verify that the gear spins the correct amount of degrees in each direction.

**Force Sensor**

To test the force sensor apply a known force and read the resulting analog voltage on a DMM. Verify the analog voltage corresponds to the applied force.

**Feedback**

The feedback consists of three stages, a DAC, sine wave generator and a voltage scalar. First output a digital value from the microcontroller to the DAC and read out the analog value on a DMM. Verify its functionality. Next output PWMs from the microcontroller into a low pass filter. Verify that it is at 250 Hz using an oscilloscope. Finally use the output of the DAC to scale the sine wave. This is the output to the tactor, which is providing vibrational feedback to the user.

**System integration**

Finally integrate the whole system together. Use the leads on a function generator(to eliminate the differences in different peoples strength) to provide electrical pulses to the system. Visually verify that the servo is gripping correctly (displayed on an oscilloscope) and releasing based upon the input voltage (displayed on a DMM). Also verify the amplitude of the feedback. All of these values should be proportional to each other for all values inputted by the electrodes. Finally hook the electrodes up to to a human arm and verify that the system is working using the same approach.

**Power Consumption**

Use a DMM in current mode to verify current computation by passing the ground through its terminals and recording the corresponding values. Use these to verify power consumption.

**Results**

The results for overall system are detailed below. Each subsystem was tested thoroughly following the test cases mentioned above, but to avoid repetition in relaying results, only the overall systems results are detailed below.

As described in the above report, there is a lot of noise affecting the acquisition of EMG signals. The first step in the system was therefore to isolate those signals using the EMG circuit. 7 displays the EMG circuit output that is not averaged, and is followed by figure 9, the averaged values. As can be seen there is limited noise. The averaged values are the ones that went into the system.



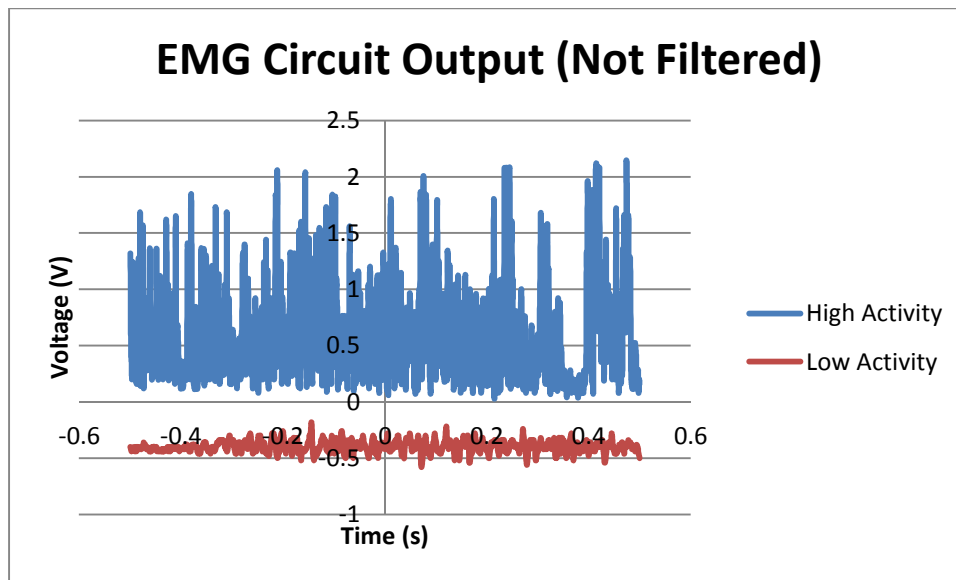


Figure 7 Raw EMG signal from user

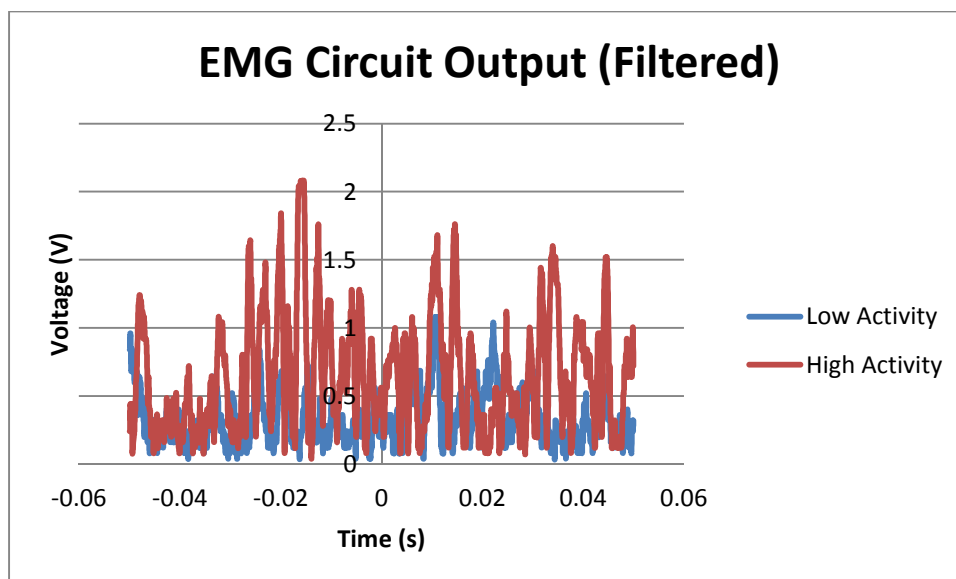


Figure 8 Filtered EMG signal from user

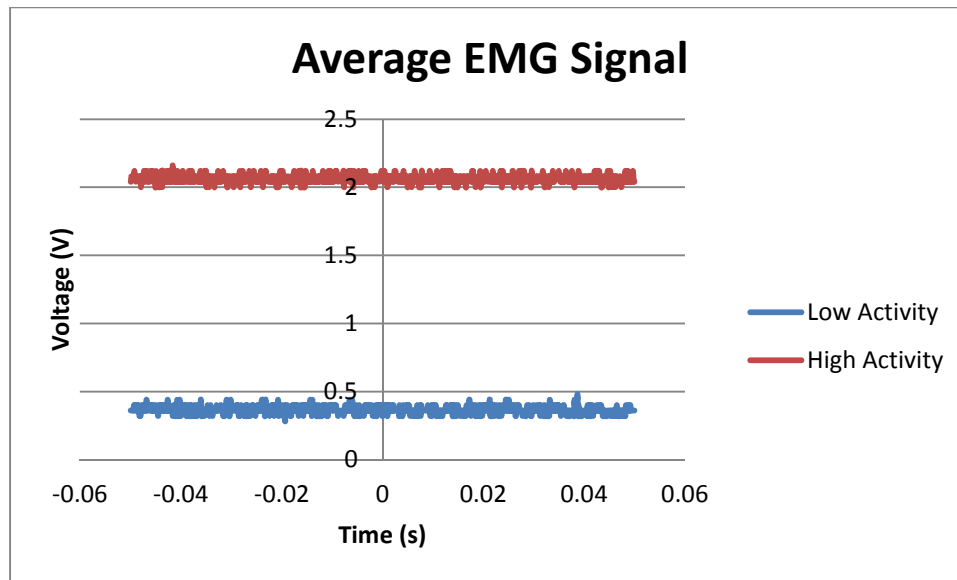


Figure 9 Averaged EMG signal

After the EMG signal was captured, it was transmitted to the robotic manipulator. The results of high activity and low activity are displayed below in figure 10. As can be seen, relaxed (the blue pulse is at 1 ms (completely open fingers), while a maximum flex resulted in a 2 ms pulse (completely closed). This was exactly as expected.

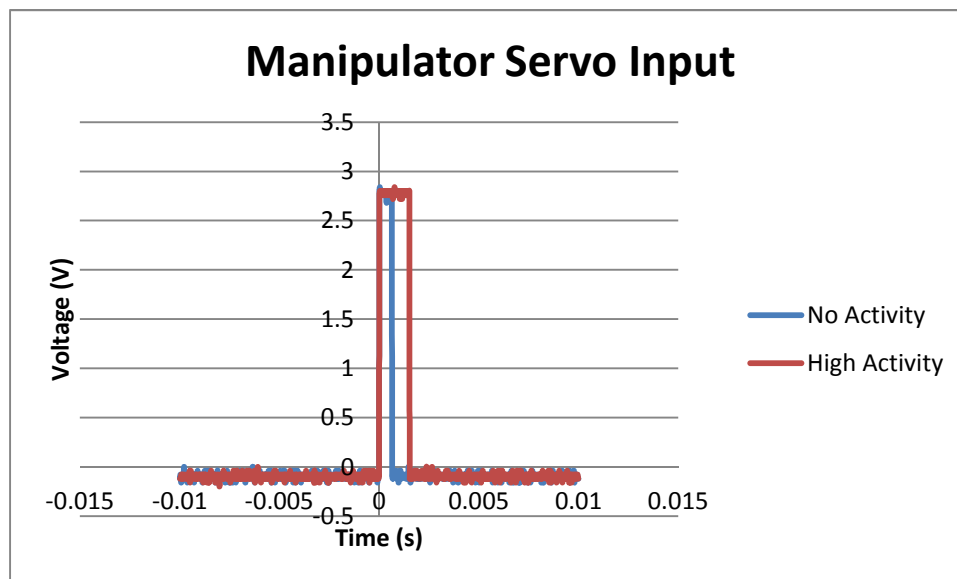


Figure 10 PWM input to servo of end effector

On each of the fingers was a FSR. Figure 11 below show the resulting voltage from both low pressure and high pressure. Again these results were as expected and verified correct functionality of both the resistors and the averaging circuit.

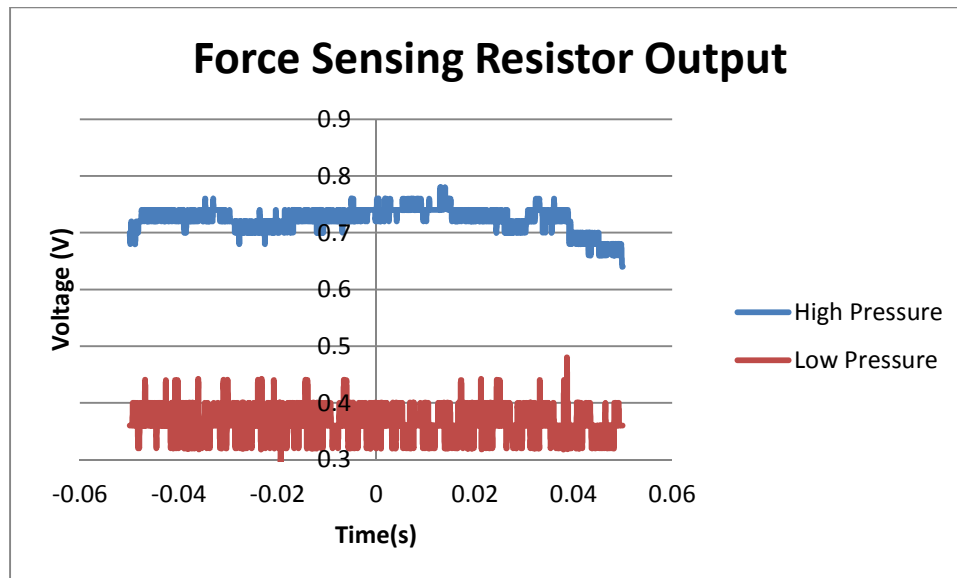


Figure 11 Force sensing array output from manipulator

Finally the voltage from the FSRs was transmitted to the local controller and outputted as a scaled sine wave. Figure 12 shows the resulting outputs of the scaled sine waves at low, medium and high force.

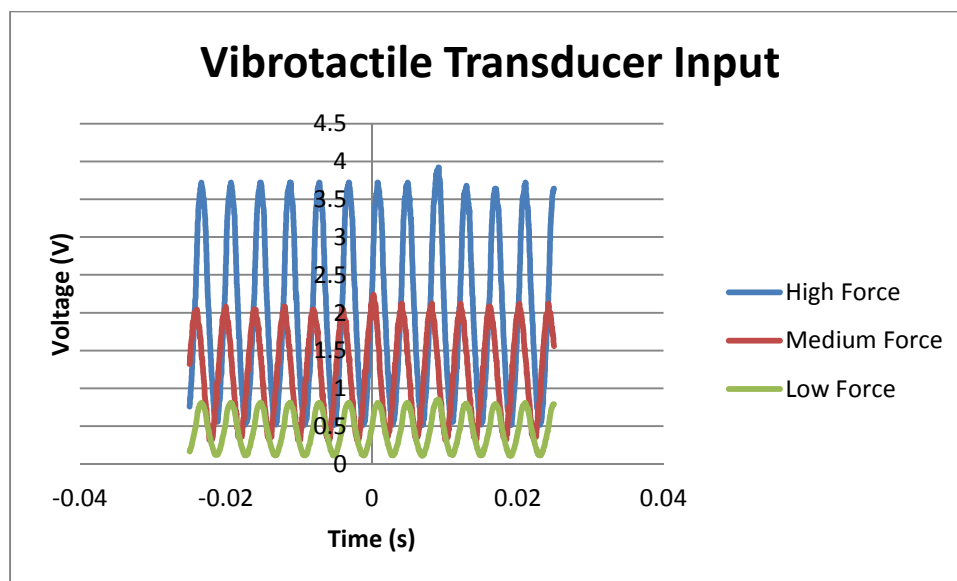


Figure 12 Final Feedback Output

The sequence detailed above functioned almost exactly as expected, with only one minor glitch (as reported below in error analysis). The timing was excellent, with no noticeable lag between applying an EMG force and receiving the haptic feedback. Power consumption was okay, ranging from 30mA to 50 mA during operation, but when the servo was engaged large current spikes of up to 300 mA were observed. Depending how heavy the usage of this system is, this could severely limit battery life, and needs to be addressed in future designs.

## Error Analysis

The biggest error encountered during this project was one involving the scheduling of tasks. As there is a hard requirement to generate the transmission of both the force and EMG signals such that human perception is not compromised, a sacrifice in the output of the servo control was made. Specifically, to ensure that transmission is conducted seamlessly, it was necessary to allow its preemption of the output generation. This sometimes results in the jittering of the PWM signal to the servo. Numerous techniques were incorporated to attempt to remedy this. This included the disabling of the receive interrupt, this however compromised the fidelity of transmission and was not sustainable. In the end a work around was implemented by decreasing to the transmission speed such that it would not preempt the servo control as often.

## Summary

A vibrotactile feedback system was generated in this project using a combination of embedded microcontroller and analog devices. This provides an inexpensive method for upper limb amputees to garner significant system response that can be integrated in their existing prostheses. Specifically, Texas Instrument's MSP430 microcontroller was used to create a system network responsible for acquiring end manipulator force and then transmitting it to a vibrotactile transducer. This provides important information regarding the force being exerted by the user. Additionally, a second microcontroller was responsible for acquiring an electromyography signal from the user. It would then process this signal and use it to generate control signals for the end effector.

## Conclusion

Several observations were made during the design and implementation of this project. Particularly, the significance of human perception was a reoccurring theme. It was discovered that it is imperative to keep the transmission of the feedback signals to less than 150 ms else the user would begin to notice ensuing delays. Unfortunately, even though transmission for this system was kept well below this rate, the interrupt driven nature of the wireless network enabled the preemption of output signals, causing a small glitch. Further research would be wise to eliminate this issue and make the system as robust as possible. Research into the interface between the feedback end manipulator and the physical prosthetic would be necessary to make this system absolutely viable. It is also necessary to investigate the learning curve of using such a system and the mechanisms implemented by a human body to control such a device. This includes determining how specific users isolate their personal muscle control in order to manipulate the end effector.

## Appendices

### Appendix A: Bill of Materials

**Project Name:** Remote Vibrotactile Feedback for Upper Limb Amputees  
**Project Number:** 1  
**Revision:** A  
**Date:** November 13, 2011

University of Washington  
 Dept. of Electrical Engineering  
 P.O. Box 352500  
 Seattle, Washington 98195-2500

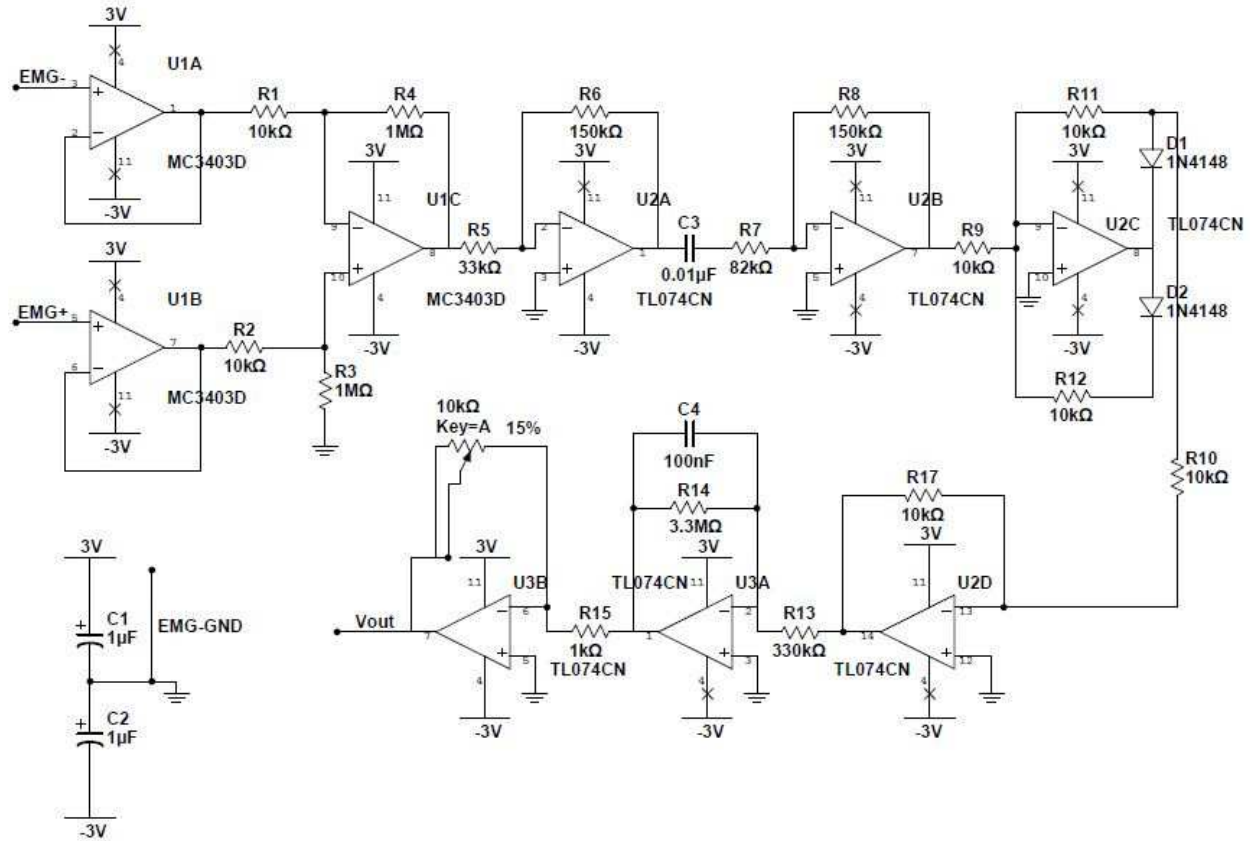
#### Bill Of Materials

Item	Quantity	Part Value	Rating	Description	Package	Vendor	Part No.	Price Each
1	1		3.6V	Wireless development Kit		Texas Instruments	MSP430 RF2500	\$49.00
2	2	1uF	25V	talium Capacitor	radial	EE Stockroom		\$0.40
3	3	10uF	50V	electrolytic capacitor	radial	EE Stockroom		\$0.30
4	2	330uF	50V	electrolytic capacitor	radial	EE Stockroom		\$0.30
5	4	0.1uF	50V	ceramic capacitor	radial	EE Stockroom		\$0.20
6	1	1000uF	50V	electrolytic capacitor	radial	EE Stockroom		\$0.30
7	2	LMTL074CN		low noise JFET amp	DIP-40	EE Stockroom	LMTL074CN	\$1.20
8	1	MC3403AD		op amp	DIP-40	EE Stockroom	MC3403AD	\$0.50
9	1	LM 386		audio amplifier, low noise	DIP-40	EE Stockroom	LM 386	\$1.00
10	5	10k	1/2W	10k potentiometer, 3/4 turn		EE Stockroom		\$0.90
11	1	33k	1/4W	5% resistor	axial	EE Stockroom		\$0.10
12	2	150k	1/4W	5% resistor	axial	EE Stockroom		\$0.10
13	2	82k	1/4W	5% resistor	axial	EE Stockroom		\$0.10
14	1	3.3k	1/4W	5% resistor	axial	EE Stockroom		\$0.10
15	1	1k	1/4W	5% resistor	axial	EE Stockroom		\$0.10
16	1	2.2k	1/4W	5% resistor	axial	EE Stockroom		\$0.10
17	9	1k	1/4W	1% resistor	axial	EE Stockroom		\$0.10
18	1	MAXX 400	5V	servo		amazon.com		\$9.50
19	1		2W	tactor		amazon.com		\$8.00
20	1			roll wire		EE Stockroom		\$1.20
21	4	1.5V		battery		radio shack		\$1.20
22	2	9v		battery		radio shack		\$3.00
23	3	DIP-2	2W	force sensing resistor	DIP-2	Trossen Robotics		\$5.00
24	50			electrodes		biomed.com		\$0.32

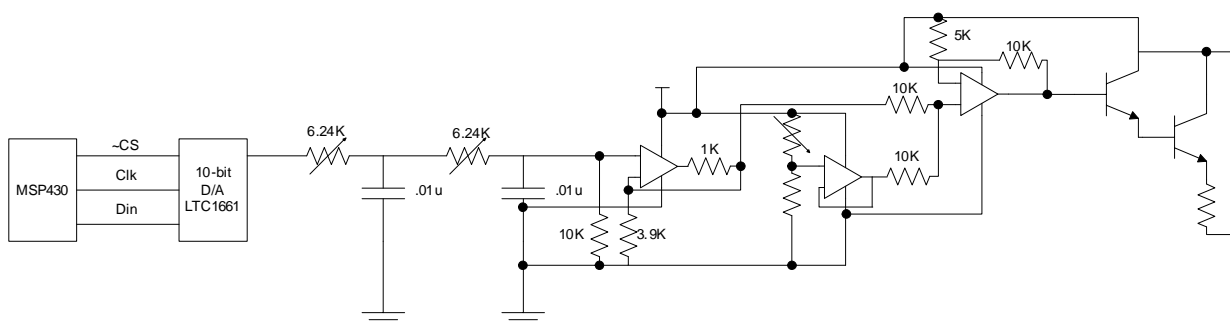
TOTAL PARTS COST:

\$ 123.00

## Appendix B: Schematics

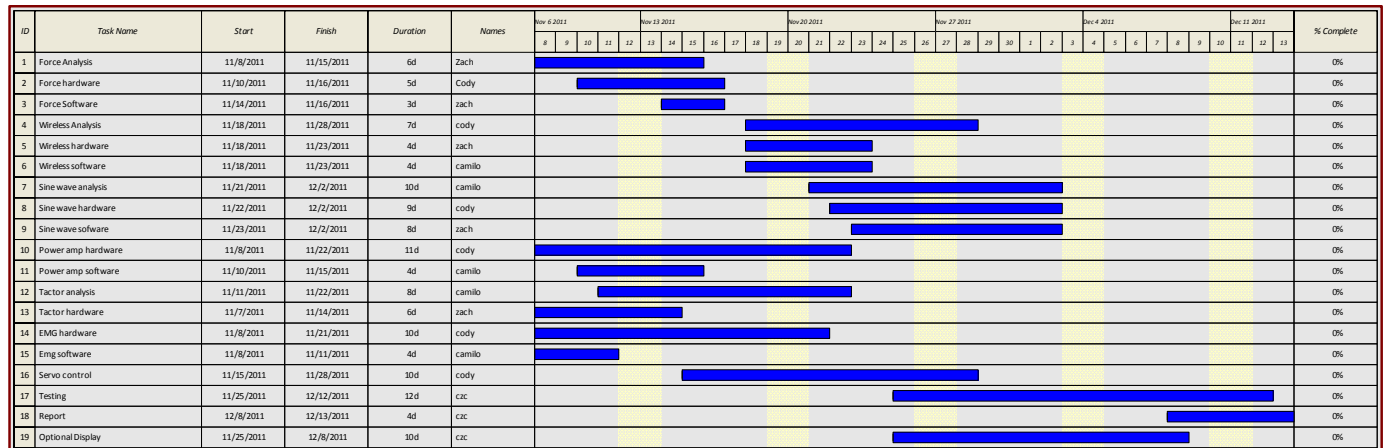


### Figure 13 EMG Acquisition Circuit Schematic



**Figure 14** Sine Wave Generation Circuit Schematic

## Appendix C: Gant Chart



## Appendix E: Code

### Tactor and EMG Control Module Code

```

/*****:)*
 * End Device 1 - TECM (Link To 1) (address 78)
 *
 * - Receives force and creates sine to
 *   drive tactor
 * - End Device one links to the network and
 *   starts communications with the End Device 2
 *
 * Authors: Zach Pritchett
 *          Cody Hogan
 *          Camilo Tejeiro
 *
 * Note: the wireless communication was implemented
 *       using the peer to peer protocol from
 *       TI Simpliciti and example code from their
 *       website.
 *****/

/*****:)*
 * include statements
 *****/
#include "bsp.h"
#include "mrfi.h"
#include "nwk_types.h"
#include "nwk_api.h"
#include "bsp_leds.h"
#include "bsp_buttons.h"
#include "app_remap_led.h"
#include "msp430x22x4.h"

/*****:)*
 * Function Declarations
 *****/
static void transmit(void);
void toggleLED(uint8_t);
static uint8_t receive(linkID_t);
void delay(unsigned int BlinkCount);
unsigned int getEMG(void);
void DACsetLines(unsigned int dig_value);

/*****:)*
 * Globals
 *****/
static linkID_t sLinkID1 = 0;
unsigned int force;
unsigned int sine_wave[9] = {6,9,11,10,7,4,1,0,2};
unsigned int index = 0;

/*****:)*
 * defines
 *****/
#define SPIN_ABOUT_A_SECOND NWK_DELAY(1000)

// Main Executable
void main (void)
{
    WDCTL = WDTW + WDTOLD;           // Stop WDT
    BSP_Init();                      // board support package
                                     // initialization
    P2DIR &= 0xFE;                   // set direction for
                                     // port 2
    P2DIR |= 0x0E;
    P4DIR |= 0x08;                   // P4.0 output

```



```

/* This call will fail because the join will fail since there is no Access Point
 * in this scenario. But we don't care -- just use the default link token later.
 * We supply a callback pointer to handle the message returned by the peer.
 */
SMPL_Init(receive);

/* turn on LEDs. */
if (!BSP_LED2_IS_ON())
{
    toggleLED(2);
}
if (!BSP_LED1_IS_ON())
{
    toggleLED(1);
}

/* wait for a button press... */
do
{
    if (BSP_BUTTON1() || BSP_BUTTON2())
    {
        break;
    }
}
while (1);

// start transmit, should not come out
transmit();

// to be safe
while (1) ;
}

/*****:*)
 * function name: Transmit
 * function inputs: void
 * function outputs: void
 * function description:establishes the conection
 * with the END device 2 and sends the EMG signal to
 * the END device 2, also sets up the sine interrupt
 *
 * author: Zach Pritchett
 *         Cody Hogan
 *         Camilo Tejeiro
 *****/
static void transmit()
{
    unsigned int EMG;
    uint8_t msg[3];

    while (SMPL_SUCCESS != SMPL_Link(&sLinkID1))
    {
        /* blink LEDs until we link successfully */
        toggleLED(1);
        toggleLED(2);
        SPIN_ABOUT_A_SECOND;
    }

    /* we're linked. turn off red LED. received messages will toggle the green LED. */
    if (BSP_LED2_IS_ON())
    {
        toggleLED(2);
    }

    /* turn on RX. default is RX off. */
    SMPL_Ioctl( IOCTL_OBJ_RADIO, IOCTL_ACT_RADIO_RXON, 0);

    /* put LED to toggle in the message */

```

```

msg[0] = 2; /* toggle red */

TACCTL0 = CCIE; // TACCR0 interrupt enabled
TACCR0 = 3552; // output every value for a complete sine
// wave 4mS/9
TACTL = TASSEL_2 + MC_1; // SMCLK, contmode
_BIS_SR(GIE); // Enter w/ interrupt

while (1)
{
    NWK_DELAY(25); // transmit every 25 mS
    EMG = getEMG(); // read the EMG from circuit
    msg[1] = EMG&0xFF; // break the EMG data in two
    // bytes, put into two messages
    msg[2] = (EMG>>8)&0xFF;
    SMPL_Send(sLinkID1, msg, sizeof(msg)); // transmit messages
}

/*****:)*
* function name: get EMG
* function inputs: void
* function outputs: digital EMG value
* function description:Read the analog values
* from the EMG circuit convert to digital
* and return this value to the calling statement
*
* note: done based on ADC interrrupt
* upon conversion
* author: Zach Pritchett
* Cody Hogan
* Camilo Tejeiro
*****:)*
unsigned int getEMG()
{
    unsigned int EMG;
    // done under less power demanding characteristics
    ADC10CTL1 = INCH_0; // channel A0 selsected for
conversion // sample and hold time of
    ADC10CTL0 = SREF_0 + ADC10SHT_3 + ADC10ON + ADC10IE + ADC10SR; // reference VCC= 3.6V
64*ADC clock, // no need to be faster, more
accurate // delay to allow reference to
    __delay_cycles(250); // settle
    ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion
start // LPM0 with interrupts
    __bis_SR_register(GIE); // enabled
    EMG = (unsigned int)(ADC10MEM); // put the result as an
unsigned int into // results array
    EMG = EMG-(unsigned int)(EMG*0.1526);

    ADC10CTL0 &= ~ENC; // turn ADC module off
    ADC10CTL0 &= ~(REFON + ADC10ON); // turn off A/D to save power

    return EMG;
}

/*****:)*
* function name: toggleLED
* function inputs: unsigned int which
* function outputs: void
* function description: toggle a certain LED
* for debugging purposes
*
* author: taken from TI sample code examples

```

```

*****:)/
void toggleLED(uint8_t which)
{
    if (1 == which)
    {
        BSP_TOGGLE_LED1();
    }
    else if (2 == which)
    {
        BSP_TOGGLE_LED2();
    }
    return;
}

/*****:)*
* function name: DAC Set Lines
* function inputs: digital value for conversion to
*                 analog
* function outputs: void
* function description: digital to analog conversion
* done serially, used to output values for the sine
* wave generation
*
* author: Zach Pritchett
*         Cody Hogan
*         Camilo Tejeiro
*****:)/
void DACsetLines(unsigned int dig_value)
{
    unsigned int ctl_code = 10240;
    unsigned int input_word = (ctl_code + dig_value) << 2;
    int shift = 15;
    unsigned int temp = 0;

    P2OUT &= ~BIT1;    // Input load low
    while(shift != -1)
    {
        temp = input_word >> shift;
        if(temp & 0x0001)
        {
            P2OUT |= BIT3; // Data high
            P2OUT |= BIT2; // SCLK high
        }else
        {
            P2OUT &= ~BIT3; // Data low
            P2OUT |= BIT2; // SCLK high
        }
        P2OUT &= ~BIT2; // SCLK low
        shift--;
    }
    P2OUT |= BIT1; // Input load high
}

/*****:)*
* function name: Receive
* function inputs: port to link to
* function outputs: unsigned int status of reception
* function description: handle received messages from
* End device 2
*
* author: Zach Pritchett
*         Cody Hogan
*         Camilo Tejeiro
*         TI sample code
*****:)/
static uint8_t receive(linkID_t port)
{

```

```
uint8_t msg[3], len;

/* is the callback for the link ID we want to handle? */
if (port == sLinkID1)
{
    /* yes. go get the frame. we know this call will succeed. */
    if ((SMPL_SUCCESS == SMPL_Receive(sLinkID1, msg, &len)) && len)
    {
        /* Check the application sequence number to detect
         * late or missing frames...
         */

        /* we're good. toggle LED in the message */
        toggleLED(*msg);
        force = msg[1] + (msg[2]<<8);

        return 1;
    }
}

/* keep frame for later handling. */
return 0;
}

/*****
 * function name: Delay
 * function inputs: unsigned int BlinkCount
 * function outputs: void
 * function description: interrupt driven delay
 *
 * author: TI sample code
 *****/
void delay(unsigned int BlinkCount)
{
    int TimerTemp;
    TimerTemp = TBCCR0;                // Save current content of TBCCR0
    TBCCR0 = BlinkCount;              // Set new TBCCR0 delay
    TBCTL |= TBCLR;                   // Clear TBR counter
    TBCTL0 &= ~CCIFG;                 // Clear CCIFG Flag
    TBCTL |= MC_1;                    // Start Timer B
    __bis_SR_register(LPM3_bits + GIE); // Enter LPM3
    TBCTL &= ~(MC_1);                 // Stop Timer B
    TBCCR0 = TimerTemp;
}

/*****
 * function name: ADC10_ISR
 * function inputs: Void
 * function outputs: void
 * function description: ISR gets executed upon
 * termination of the A/D conversion
 *
 * author: Zach Pritchett
 *         Cody Hogan
 *         Camilo Tejeiro
 *****/
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
    __bic_SR_register_on_exit(LPM0_bits); // Clear CPUOFF bit from 0(SR)
}

/*****
 * function name: Timer_A ISR
 * function inputs: Void
 * function outputs: void
 * function description: ISR gets executed upon
 * reach of the CCR0 value for timer A0.
 *
 * here is where we output the lines for the
 * sine wave generation every 4mS/9
 *****/
```

```
*
* author:  Zach Pritchett
*          Cody Hogan
*          Camilo Tejeiro
*****:)/
#pragma vector=TIMER_A0_VECTOR
__interrupt void Timer_A(void)
{
    DACsetLines(0.1*force*sine_wave[index]);
    index++;
    if(index >= 9) index = 0;
    P4OUT ^= 0x08;           // Toggle P1.0
    TACCR0 = 3552;           // Add Offset to TACCR0
}
```

## Manipulator and Force Control Module Code

```
/* *****:)*
* End Device 2 - MFCM (Link Listen 2) (address 87)
*
* - Receives Force and drives servo manipulator to
* - End Device 2 lsitens for linking and
*   establishes communications with the End Device 1
*
* Authors: Zach Pritchett
*          Cody Hogan
*          Camilo Tejeiro
*
* Note: the wireless communication was implemented
*       using the peer to peer protocol from
*       TI Simpliciti and example code from their
*       website.
*****:)*

/* *****:)*
* include statements
*****:)*

#include "bsp.h"
#include "mrfi.h"
#include "nwk_types.h"
#include "nwk_api.h"
#include "bsp_leds.h"
#include "bsp_buttons.h"
#include "app_remap_led.h"
#include "msp430x22x4.h"

/* *****:)*
* Function Declarations
*****:)*
static void transmit(void);
void toggleLED(uint8_t);
unsigned int getForce();
void delay(unsigned int BlinkCount);
static uint8_t receive(linkID_t);

/* *****:)*
* Globals
*****:)*
static linkID_t sLinkID2 = 0;
static volatile uint8_t sSemaphore = 0;
unsigned int EMG;
unsigned int servoVal = 5000;
unsigned int off = 0x0000;
unsigned char counter = 0;           // Current location in wave array
unsigned char CCRO_INDEX = 0;

// main executable
void main (void)
```

```
{
    WDTCTL = WDTPW + WDTHOLD;           // Stop WDT
    BSP_Init();

    P2DIR |= 0x0D;                       // port 2.1 set as input

    /* This call will fail because the join will fail since there is no Access Point
    * in this scenario. But we don't care -- just use the default link token later.
    * We supply a callback pointer to handle the message returned by the peer.
    */
    SMPL_Init(receive);

    /* turn on LEDs. */
    if (!BSP_LED2_IS_ON())
    {
        toggleLED(2);
    }
    if (!BSP_LED1_IS_ON())
    {
        toggleLED(1);
    }

    /* wait for a button press... */
    do
    {
        if (BSP_BUTTON1() || BSP_BUTTON2())
        {
            break;
        }
    } while (1);

    CCTL0 = CCIE;    // CCR0 interrupt enabled
    CCTL1 = CCIE;    // CCR1 interrupt enabled
    CCR0 = 27340;    // Set PWM period to 27340 clock ticks empirically measured to be 20.7 mSec
    CCR1 = 5000;    // Set first duty cycle value to 0.600mS
    TACTL = TASSEL_2 + MC_1 + TAIE + TACL; // SMCLK, upmode, enable interrupt, clear TA1R

    // start transmission
    transmit();

    //just for safety
    while (1) ;
}

/*****:*****/
* function name: Transmit
* function inputs: void
* function outputs: void
* function description:establishes the connection
* with the END device 1 and sends the force signal to
* the END device 1, also sets up the PWM interrupt
* to establish period and pulse width
*
* author: Zach Pritchett
*        Cody Hogan
*        Camilo Tejeiro
*****/
static void transmit()
{
    uint8_t    msg[3];
    unsigned int force;
    /* Turn off one LED so we can tell the device is now listening.
    * Received messages will toggle the other LED.
    */
    toggleLED(1);

    /* listen for link forever... */
    while (1)
    {
        if (SMPL_SUCCESS == SMPL_LinkListen(&LinkID2))
    }
}
```

```

    {
        break;
    }
    /* Implement fail-to-link policy here. otherwise, listen again. */
}

/* turn on LED1 on the peer in response to receiving a frame. */
msg[0] = 1;

/* turn on RX. default is RX off. */
SMPL_Ioctl( IOCTL_OBJ_RADIO, IOCTL_ACT_RADIO_RXON, 0);

while (1)
{
    /* Wait for a frame to be received. The Rx handler, which is running in
     * ISR thread, will post to this semaphore allowing the application to
     * send the reply message in the user thread.
     */

    if (sSemaphore)
    {
        force = getForce();
        msg[1] = force&0xFF;
        msg[2] = (force>>8)&0xFF;
        SMPL_Send(sLinkID2, msg, sizeof(msg));

        /* Reset semaphore. This is not properly protected and there is a race
         * here. In theory we could miss a message. Good enough for a demo, though.
         */
        sSemaphore = 0;
    }
}

/*****:)*
 * function name: toggleLED
 * function inputs: unsigned int which
 * function outputs: void
 * function description: toggle a certain LED
 * for debugging purposes
 *
 * author: taken from TI sample code examples
 *****/
void toggleLED(uint8_t which)
{
    if (1 == which)
    {
        BSP_TOGGLE_LED1();
    }
    else if (2 == which)
    {
        BSP_TOGGLE_LED2();
    }
    return;
}

/*****:)*
 * function name: getForce
 * function inputs: void
 * function outputs: digital Force value
 * function description: Read the analog values
 * from the force averager circuit, convert
 * to digital and return this value to
 * the calling statement
 *
 * note: done based on ADC interrupt
 * upon conversion
 * author: Zach Pritchett
 * Cody Hogan

```

```

*          Camilo Tejeiro
*****:)/
unsigned int getForce()
{
    unsigned int force;
    // done under less power demanding characteristics
    ADC10CTL1 = INCH_1; // channel A0 selcted for
conversion
    ADC10CTL0 = SREF_0 + ADC10SHT_3 + ADC10ON + ADC10IE + ADC10SR; // reference VCC= 3.6V
// sample and hold time of
64*ADC clock,
// no need to be faster, more
accurate

    __delay_cycles(250); // delay to allow reference to
settle
    ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion
start
    __bis_SR_register(LPM0_bits + GIE); // LPM0 with interrupts
enabled
    force = (unsigned int)(ADC10MEM); // put the result as an
unsigned int into
    force = force-(unsigned int)(force*0.1526); // results array

    ADC10CTL0 &= ~ENC; // turn ADC module off
    ADC10CTL0 &= ~(REFON + ADC10ON); // turn off A/D to save power

    return force;
}

/*****:)*
* function name: Receive
* function inputs: port to link to
* function outputs: unsigned int status of reception
* function description:handle received messages from
* End device 1
*
* author: Zach Pritchett
*         Cody Hogan
*         Camilo Tejeiro
*         TI sample code
*****:)/
static uint8_t receive(linkID_t port)
{
    uint8_t msg[3], len;

    /* is the callback for the link ID we want to handle? */
    if (port == sLinkID2)
    {
        /* yes. go get the frame. we know this call will succeed. */
        if ((SMPL_SUCCESS == SMPL_Receive(sLinkID2, msg, &len)) && len)
        {
            /* Check the application sequence number to detect
            * late or missing frames...
            */

            toggleLED(*msg);
            EMG = msg[1] + (msg[2]<<8);
            // EMG max 750.85324
            // maximun servo value = 16000
            servoVal = 15*EMG + 5000;
            if (EMG > 512)
            {
                BSP_TOGGLE_LED1();
            }
            else
            {

```



```

    }
    /* Post to the semaphore to let application know so it sends
    * the reply
    */
    sSemaphore = 1;
    /* drop frame. we're done with it. */
    return 1;
}
}
/* keep frame for later handling */
return 0;
}

/*****:)*
* function name: Delay
* function inputs: unsigned int BlinkCount
* function outputs: void
* function description: interrupt driven delay
*
* author: TI sample code
*****/
void delay(unsigned int BlinkCount)
{
    int TimerTemp;
    TimerTemp = TBCCR0;           // Save current content of TBCCR0
    TBCCR0 = BlinkCount;         // Set new TBCCR0 delay
    TBCTL |= TBCLR;              // Clear TBR counter
    TBCTL0 &= ~CCIFG;            // Clear CCIFG Flag
    TBCTL |= MC_1;               // Start Timer B
    __bis_SR_register(LPM3_bits + GIE); // Enter LPM3
    TBCTL &= ~(MC_1);            // Stop Timer B
    TBCCR0 = TimerTemp;
}

/*****:)*
* function name: ADC10_ISR
* function inputs: Void
* function outputs: void
* function description: ISR gets executed upon
* termination of the A/D conversion
*
* author: Zach Pritchett
*         Cody Hogan
*         Camilo Tejeiro
*****/
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
    __bic_SR_register_on_exit(LPM0_bits); // Clear CPUOFF bit from 0(SR)
}

/*****:)*
* function name: Timer_A ISR
* function inputs: Void
* function outputs: void
* function description: ISR gets executed upon
* reach of the CCR0 value for timer A0.
*
* here is where we determine the period of the
* PWM signal to drive the servo 20 mS
* pulls the line up.
*
* author: Zach Pritchett
*         Cody Hogan
*         Camilo Tejeiro
*****/
#pragma vector = TIMERA0_VECTOR
__interrupt void TIMERA0_ISR(void)
{

```

```

    if(CCR0_INDEX == 5){
        CCTL1 = CCIE;    // CCR1 interrupt enabled
        P2OUT |= BIT0;
        CCR0_INDEX = 0;
    }else {
        CCR0_INDEX++;
    }
}

/*****:)*
* function name: Timer_A1 ISR
* function inputs: Void
* function outputs: void
* function description: ISR gets executed upon
* reach of the CCR1 value for timer A1.
*
* Here is where pull the line low to determine
* the angle of rotation of the servo
* range: 0.600mS to 2mS
*
* author:  Zach Pritchett
*         Cody Hogan
*         Camilo Tejeiro
*****/
#pragma vector = TIMERA1_VECTOR
__interrupt void TIMERA1_ISR(void)
{
    switch(TAIV)
    {
        case 2:          // CCR1 interrupt
            CCTL1 = off;    // CCR1 interrupt enabled
            P2OUT &= ~BIT0; // Clear P1.0 to determine duty cycle.
            CCR1 = servoVal;
            break;
        default:
            break;
    }
}

```

## References

- [1] K. Amano, *et al.*, "Estimation of the timing of human visual perception from magnetoencephalography," *J Neurosci*, vol. 26, pp. 3981-91, Apr 12 2006.