“Extended Reach” Haptic Array

Final Report

December 12, 2012

Nathan Murray
Todd Pixton

ECE 445 Fall 2012
TA: Lydia Majure

Table of Contents

Abstract:

We designed, built, and verified a robust, modular haptic biofeedback system. It consists of 32 shaftless vibration motors housed within a Velcro armband that can be strapped onto a user's arm. The motor array is controlled by a driver program executed on an Arduino and it is powered by a circuit primarily composed of two TI TLC5940 16-channel constant-current sink LED drivers, fitted with circuitry that allows the ICs to operate small motors instead of LEDs.

The design was successfully completed and verified. We demonstrated that a user can strap the device onto their arm, and then control the device through serial communication. Rich and interesting haptic feedback is delivered to the user, and a standard of robustness and modularity was maintained throughout the project.

While we were successful, there are areas of improvement for this project. Mechanical isolation could be increased between the motors and armband to prevent mechanical coupling. More dimensions of haptic feedback could be implemented, such as timbre. Finally, a user-friendly GUI interface could be introduced to facilitate manual motor control.

1. Introduction

1.1 Description

The motivation for this project stems from a desire to extend and expand human sensory perception. There are many signals that humans simply don't have the capacity to sense; ultrasound, infrasound, the vast majority of the electromagnetic spectrum, and the temperature of distant objects are all invisible to humans. Our project aims to allow humans to experience these signals by translating them into a form that humans can recognize.

Specifically, this project will provide the tools to map any quantified characteristics of a signal detected by a sensor into a haptic response on an arm-worn array of motors. This haptic feedback system will use an array of micro-transducers to stimulate the user's arm, utilizing the body's haptic perception of different intensities, locations (two dimensions), and frequencies. Every tenth of a second, the haptic device will accept a packet of information from the sensor. This packet will give values for up to three signal characteristics, which will then be intelligently mapped to haptic intensity, location, and frequency. The mapping will be modular and easy to use; the user will have the ability to decide what signal characteristic gets mapped to each haptic quality. The end result will be a "natural" psychophysical stimulus to a signal imperceivable to humans.

1.2 Goals

The primary goal of this project is to create a robust haptic biofeedback device that delivers a continuous stimulus to the user's arm, allowing the user to become aware of sensory data normally outside the range of human perception. This device will include a simple Arduino program that will facilitate the mapping of sensory data to haptic stimulus. Also, the device will include circuitry to amplify and control the motors such that they deliver natural and repeatable feedback to the user.

1.3 Benefits and Features

- Augments human sensory perception
- Delivers "natural" psychophysical haptic feedback
- Arm-worn haptic feedback array
    - Sturdy, compact design
    - Four dimensions of haptic feedback (intensity, x and y location, frequency)
- Haptic feedback driver with simple programmable interfacing with signal characteristics
    - Modular, simple user interface
    - Supports any quantifiable signal qualities
- Unobtrusive haptic feedback works in noisy and bright environments.
- Modularity allows expansion and integration into other human augmentation designs

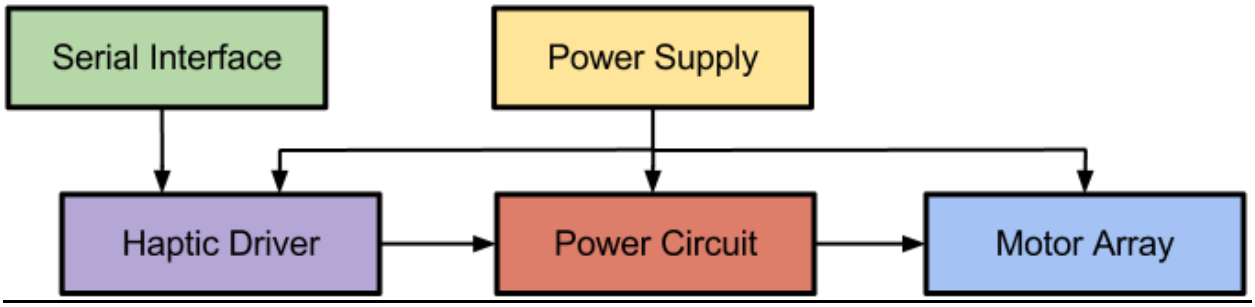

Figure 1.1: Haptic Array Block Diagram

1.4 Block Descriptions

Power Supply: Power will be supplied by DC battery sources.

Inputs: None

Outputs: 4.8V and 9V DC

Alternative: DC battery sources were chosen to allow the device to be portable. However, if DC battery sources do not work, DC power supplies can also be used.

Serial Interface: Data will be input serially through USB or RS-232 .

Inputs: Quantified signal data through USB or RS-232

Outputs: Quantified signal data though Arduino's Type B USB connection or TTL pins

Alternative: Two different Serial Interfaces have been described in order to allow the user flexibility with how he inputs data. Furthermore, the USB serial interface with the Arduino is a failsafe; it is required for the Arduino to function in general.

Haptic Driver: An Arduino Duemilanove microcontroller will take in quantified signal characteristics and map these characteristics to haptic features of the array. PWM values for every motor will be calculated to achieve the haptic stimulus. Commands for these values will be output through the digital I/O pins of the Arduino board to the Power Circuit.

Inputs: 9V DC (9V battery)
4.8V DC (Battery pack or bench-top supply)
UART data from sensor

Outputs: PWM commands for every motor through digital I/O pins

Alternative: The Haptic Driver must function in some capacity; without it, there cannot be processing of sensor data. However, the requirements for this component have been loosely specified to allow freedom in determining how the Haptic Driver will be implemented.

Power Circuit: The Power Circuit will maintain, amplify, and control the PWM commands from the Haptic Driver such that they can be used to drive the motor array. The TI TLC5940, a 16 Channel LED Driver, will be used to amplify and control the PWM response of the motors.

Inputs: 4.8V DC (4 NiMH AA batteries or bench-top supply)

PWM commands from Haptic Driver

Outputs: Amplified PWM signals for every motor

Alternative: The four PWM digital output pins on the Arduino can also be used to drive four motors. However, we have chosen to use the current design because it allows us to control up to thirty-two motors.

Motor Array: The motor array will be two 5x3 arrays of shaftless vibration motors (one array for the top of the forearm, and one for the bottom). The motors will be secured in a robust armband.

Inputs: Amplified PWM signals from Power Circuit

Outputs: Haptic stimulus on user's arm

Alternative: If the large arrays of motors fails, the number of motors can be reduced to a more reasonable amount, such as a signal 2x4 array, or even a line of only four motors. Even with these smaller number of motors, interesting haptic stimulus can be achieved, although in a much narrower scope. The chosen design provides richer feedback than the alternatives.

1.5 Performance Requirements

- Achieve a robust device by employing durable build techniques
- Achieve continuous perception (motors indistinguishable from one another)
- Achieve repeatable results for dependable feedback
- Achieve large range of frequency and intensity across entire array

2. Design

2.1 Schematic

The full Hardware Schematic can be found in Appendix A: Hardware and Software Figures.

2.1.1 Hardware

Sub-circuit Descriptions

The circuit for the haptic feedback array will include an Arduino Duemilanove, two TLC5940 LED driver IC chips, thirty shaftless vibration motors, thirty diodes, thirty 17.4 Ohm resistors, two 650 Ohm resistors to set the channel current to 60 mA, a 9V power source for the Arduino, a 4.8V power source for the chips and motor array, and a PCB to house the Power Circuit.

TLC5940:

The main purpose of the TLC5940 is to maintain the PWM signals to the motors. This way, the Arduino is free to process future PWM values instead of working to maintain current PWM signals. The TLC5940 has 28 pins, which can be seen in the schematic. The notable ones are as follows [1]:

- XERR: this output indicates that the TLC5940 has overheated or has a burnt out LED. We are not using this output.
- SOUT: this is the serial data output pin. We are using it for the daisy-chaining functionality to chain the two chips together. It is connected to the SIN of the second chip.
- DCPRG: this input selects the source of the current limiter register. We will be keeping it high.
- IREF: This sets the current output of every channel. The equation for determining the value of this current is as follows:

$$I_{max} = \frac{V_{(IREF)}}{R_{(IREF)}} \times 31.5 \quad (2.1)$$

  Using this formula, we determined that a 650 Ohm resistor is necessary to achieve a channel current of 60 mA.
- XLAT: this input is used to latch data after shifting. The TLC5940 library functions control what is input to XLAT.
- SCLK: this input is used to clock data shifting. The TLC5940 library functions control what is input to SCLK.
- SIN: this is the serial input pin for the TLC5940. For the first chip, this pin is connected to the Arduino. For the second chip, it is connected to the first chip's SOUT.
- VPRG: this selects the chip's mode of operation. We will tie VPRG to ground to select GS (grayscale) mode.
- GSCLK: this is the grayscale clock for the PWM. It will be driven by the Arduino's digital I/O pin 3.

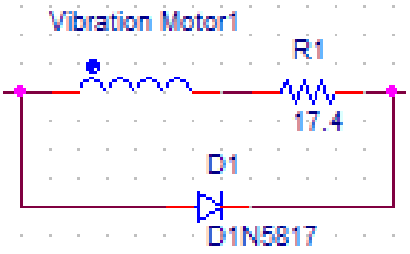

Figure 2.1: Flyback Diode Circuit.

- BLANK: this indicates that a grayscale PWM cycle has completed, and also blanks the output. We will tie this to the Arduino's digital I/O pin 10.
- OUT: there are 16 current-sink output pins. These are the channels that will maintain the PWM on our motor array.

Flyback Diode Circuit:
A best practice for connecting small motors to a current sink is to use a Flyback Diode Circuit design. Because motors have not only real impedance, but an imaginary component from the wire coil as well, the circuit must be designed to deal with the inductive properties.

The voltage across the motor is defined (modeled as an inductor) :

$$V_L = -L\frac{dI}{dt} \tag{2.2}$$

The voltage the TLC5940 maintains is controlled by a resistance R_c:

$$V_{IC} = I \cdot R_{IC} \tag{2.3}$$

When the TLC5940 stops the current, the voltages are equal:

$$I \cdot R_{IC} = -L\frac{dI}{dt} = V_{IC} \tag{2.4}$$

Above it can be inferred that as the TLC5940 attempts to stop the current, R_{IC} jumps very high as it is designed to mitigate I. However, $\frac{dI}{dt}$ will jump as the inductive part of the motor attempts to maintain the current. This will cause a large negative spike in V_{IC} and place the TLC5940 in danger.

We can solve this problem by putting a diode in parallel in the motor. Normally, as the motor is forward biased, the diode is reverse biased and does not allow current to run alongside the motor. Without the diode, when the motor loses its bias, an excess of charge spikes the voltage down at the TLC5940 end of the motor. However, with the diode, the diode becomes forward

biased and allows the current to flow back into the motor. The excess energy gets dissipated as work by the motor and heat by resistance in series with the motor.

Note that the Flyback Diode Circuit is only shown four times in the Hardware Schematic. Furthermore, note that OUT15 for both TLC5940 chips is not used.

Arduino Duemilanove
The Arduino Duemilanove is a small Arduino microcontroller that has five analog in/out pins (that we won't use), 14 digital in/out pins (including two serial TTL pins and four PWM pins). It is programmable using a C-like language. It also can connect to computers via USB, and can by powered via USB or a 7-12V power source.

We will be using the Arduino to process all of the motor PWM information. It will be receiving serial data, interpreting the data, computing new PWM values for the motors, and then outputting the new values to the TLC5940 chips.

Motor Array
The motor array will consist of 2 arrays of 15 motors each, arranged in a 5x3 pattern with each motor having a distance of 3 cm from center to center. We arrived at this distance by using the arm's Two-Point Discrimination, which is at best 3.07 cm [2]. The motors will be driven by the PWM outputs of the two TLC5940 chips. Each motor has a typical operating current of 60mA and a typical operating voltage range of 2.5V – 3.5V [3].

9V Power Supply
The 9V Power Supply is a power supply capable of delivering 9V to the Arduino. We plan to use a 9V battery for this element, but if that fails we will use a bench-top power supply to supply the Arduino with 9V.

4.8V Power Supply
The 4.8V Power Supply is a power supply capable of delivering 4.8V and 2A to the Power Circuit and the Motor Array. We plan to use 4 AA NiMH batteries for this element. These AA NiMH batteries deliver 1.2V each, are rechargeable, maintain a stable voltage during discharge, and can discharge 5A [4].

Decoupling Capacitor
A decoupling capacitor is placed between the VCC and GND pins of both TLC5940 chips. Inevitably, an IC's power requirements will fluctuate as it operates. Usually, a power supply is not capable of meeting the demands for more or less current quickly enough. To resolve this issue, a decoupling capacitor is placed between VCC and GND. This capacitor stores a small amount of energy and is able to quickly supply a small amount of energy to the IC when it is needed. That is, the decoupling capacitor acts as a short-term, fast-response power supply for the IC. The value of 100nF was chosen because it is the recommended value in Texas Instrument's Application Example [1].

Serial Interface
The Arduino has USB and serial TX and RX pins. However, we would like to use the RS232 standard instead of the TTL standard. Thus, we will use a TTL-RS232 converter to convert RS232 serial input to TTL serial input that can be used by the Arduino. Furthermore, for testing purposes, we can use a USB-RS232 converter to send serial signals from a computer to the Arduino.

The PCB
The haptic array was designed to be robust and easy to use. A popular method of designing circuits to interface with arduino boards is to build an arduino shield. By designing and assembling a custom circuit board with two 8-pin headers and two 6-pin headers on opposite sides of a board, the PCB can sit on top of an arduino board like a "shield".

Our haptic driver interface uses 32 parallel flyback-diode circuits in junction with our TLC5940 ICs. To simplifiy construction, increase robustness, and generally clean up our circuit, we designed the power circuit entirely on a custom PCB by using the Eagle CAD program. The Jones lab graciously awarded us an indispensible set of funds to have our PCB fabricated by Advanced Circuits.

The PCB is roughly 3" x 4" in size. It interfaces the TLC5940 chips thorugh IC sockets. The use of IC sockets allowed tolerance for the design to fail and destroy a PCB (which happened on one occasion, although likely due to a reverse insertion of the IC). The key discrete devices to interface the IC to the circuit: the capacitors, resistors, and diodes were soldered on the board as surface-mount components.

An important feature of the PCB is the use of color labeled power wires and the labeling and separating of motor current sink wires. Of the 34 wire connections on the PCB, two account for Vin and Gnd connections to the power supply. The Vin connection also connects to all of the motors on the armband. This way, instead of 64 wires (Vin * 32 + Signal * 32) connecting to the PCB, only 32 come from the motors to be connected to the PCB. This made wire management easier for our design.

2.1.2 Software

The full Software Flowchart can be found in Appendix A: Hardware and Software Figures.

User Interface
The code will be written as an Arduino sketch. We will make use of a library provided at the Arduino playground code wiki [5]. This library provides functions for setting the grayscale PWM values for the channels of the TLC5940. There are four main functions. `Tlc.init()` initializes the PWM values to zero. `Tlc.clear()` sets every grayscale PWM value to zero. `Tlc.set(channel, value)` will set the grayscale PWM value for the specified channel. For our application, the channels are 0 to 31, although we will only be using thirty. The grayscale values can range from 0 to 4095, corresponding with fractions of duty-cycle, where 4095 is the

maximum duty-cycle. However, `Tlc.set` commands do not take effect until `Tlc.update()` is called. `Tlc.update()` sends the updated channel values to the TLC5940 chips.

Function Descriptions

Following is an explanation and justification for each portion of the software in the order of information flow; that is, beginning with the global constants, moving to the setup loop, and continuing to address each function as the code flows.

Global Constants and Variables

The program uses many global constants so that the user has an easy location to control the parameters of the program. There are global constants for the number of data points displayed, the size of the cosine lookup table (200 samples), the size of a data point (four, for intensity, x location, y location, and frequency), and the number of TLC OUT pins (16 for one chip, 32 for two chips). There are also 16 constants used for mapping data points: each data point quality has an input min and max as well as an output min and max.

The program also makes use of global variables. There are five variables used when a data point is being processed (four for the qualities and one for the “tag”, or the number of the data point in a set, which can be from one to three). There are two Booleans for functional_mapping (as opposed to direct mapping) and a flag when new data points have been read. There is the cosine lookup table as well as an array of pointers that contain each data point’s location in the lookup table and an array of skip values that determine how far a data point skips in the lookup table each iteration. There is an array containing the data points, and finally an array containing the maximum values of each motor, as calculated by the program.

void setup()

The setup loop is run before loop() and initializes the program. It calls Tlc.init() which initializes the Tlc functions, calls Serial.begin(9600) which initializes serial communication at a baud rate of 9600 (which is standard), calculates the cosine values for the lookup table, and cleans all the arrays to remove any garbage.

The rest of this section deals with functions occurring within **void loop()**

getInput()

This function reads serial data when a full data point is available. A data point is seven bytes: one for the tag, two for intensity, one for x location, one for y location, and two for frequency. After the bytes are read, they are converted to ints and put into the points array. Then the new_points flag is set to true, and the function exits.

Data Point Mapping

After completing two checks (for functional mapping and new points), the program loops through every data point in the points array. It then maps each data point value to a new output value, as dictated by the input/output global constants and the mapping functions; mapIntensity(intensity), mapX(x), mapY(y), and mapFreq(freq). These are all functionally

equivalent, so I will only discuss mapIntensity(intensity). It begins with a check that the intensity value in the data point is within the input bounds set by the global constants. Then, it returns one (and only one) of three mapping functions: linearMapping, exponentialMapping, and logarithmicMapping. It must provide the intensity value, as well as the input min and max values and the output min and max values as parameters to this function.

Mapping Functions

The mapping functions are exactly as they sound: they provide functional mapping for the data point values. They take a value in the range set by the input global constants, and map it to the range set by the output global constants. The mapping is either linear, exponential, or logarithmic. For linear, the mapping reads:

```
value = (value-iMin)/(iMax-iMin)*(oMax-oMin)+oMin;
```

For exponential: `value = (exp(value)-exp(iMin))/(exp(iMax)-exp(iMin))*(oMax-oMin)+oMin;`

And logarithmic: `value = (log(value)-log(iMin))/(log(iMax)-log(iMin))*(oMax-oMin)+oMin;`

Furthermore, each of the functions has a check to make sure that there is no divide by zero error, and returns the output minimum if a divide by zero would occur. Plots for these functions can be found in Appendix C: Software Plots and Source Code.

calcMaxMotorIntensities(data_point)

After the data points have been mapped to their new output values, the maximum intensities for every motor must be determined. I say maximum intensity because the intensity can be modulated by the cosine values if a frequency is set. This function iterates through every motor, calculating its x and y position in the motor array. It then calculates the distance between the motor and the location given in the data point (the index is the parameter for the function). If the distance is less than 1 (so the data point falls between two motors) then the new intensity is simply given by: `new_intensity = intensity/(1-dist);`

Otherwise, the new intensity is zero. This intensity is then input into the max_motor_int array, after passing through **intensityBounds()**.

intensityBounds(value)

This is a simple function that bounds the value to the range 0 to 4095.

calcCurrMotorIntensities()

This function is where the intensities from frequency and superposition of data points are calculated. A new intensity is calculated by applying the cosine lookup table values to the max intensities, and then summing the result for all data points per motor. Then, the function Tlc.set is called, which sets a new intensity for a given motor. Finally, the cos_ptrs global variable is updated such that it advances through the cosine lookup table.

After **calcCurrMotorIntensities()**, the only thing left to do in **loop()** is send the new intensities to the TLC5940 chips. This is accomplished by calling the Tlc.update() function.

2.2 Simulations and Calculations

2.2.1 Flyback Diode Circuit

As discussed previously, when an inductive motor loses its bias, a strong negative bias is seen across the element. This flyback EMF can be simulated in PSPICE. Note that this simulation was designed to mimic how the TLC5940 controls the bias at the negative terminal of the motor. The negative voltage seen when the inductive motor loses its bias peaks near 200V, which is far too great for the TLC5940.

By adding a diode in parallel with the motor, a pathway is opened up for the current to flow after the motor has lost its bias, allowing the motor to safely run down its stored energy. Furthermore, by adding a resistor in series with the motor, we are able to accomplish two goals: we further reduce the effect of the flyback EMF, and we introduce another voltage drop between the 4.8V source and the TLC5940, reducing power dissipation of the chip. The 17.4 Ohm resistor choice is discussed in the next section.

The circuits and simulations for the Flyback Diode Circuit can be found in Appendix B: Simulation and Verification Figures.

2.5.2 TLC5940 Power Dissipation

Power dissipation in the TLC5940 chips is a concern. They are designed to work with LEDs, which operate at lower currents than the coin vibration motors. Therefore, it is important to confirm that the TLC5940 chips will be able to safely operate even in the worst case scenario. The power dissipation equation is given as [1]:

$$P_D = (V_{CC} \cdot I_{CC}) + (V_{OUT} \cdot I_{MAX} \cdot \frac{DC_n}{63} \cdot d_{PWM} \cdot N) \tag{2.5}$$

where, for the worst case scenario,

$V_{CC} = 4.8V$,
$I_{CC} = 60mA$,
$I_{MAX} = 60mA$,
$\frac{DC_n}{63} = 1$,
$d_{PWM} = 1$,

and

$N = 15$

One area of concern is V_{OUT}. If we can reduce V_{OUT}, then we can drastically reduce power dissipated in the chip. We reduce V_{OUT} by adding a resistor in series with each motor. We want to drop about 1V across this resistor when the motor is operating at 60mA, so the resistor

needs to be about 16.7 Ohm. We were able to find a cheap 17.4 Ohm resistor that works with PCB. Therefore, the voltage drop across our series resistor is 1.044V. Furthermore, the operating voltage of the motors is given to be between 2.5V and 3.5V. We will assume the worst case, 2.5V. Therefore,

$$V_{OUT} = 4.8V - 1.044V - 2.5V = 1.256V \tag{2.6}$$

and the worst case power dissipation can be found:

$$P_D = (4.8V \cdot 60mA) + (1.256V \cdot 60mA \cdot 1 \cdot 1 \cdot 15) = 1.4184W \tag{2.7}$$

Given that the maximum power dissipation is 2.026W [1], we can find what percentage of maximum power we will dissipate in the worst case scenario:

$$\begin{aligned} \%P_{D,MAX} &= \frac{P_D}{P_{D,MAX}} \cdot 100\% \\ &= \frac{1.4184W}{2.026W} \cdot 100\% \\ &= 70.01\% \end{aligned} \tag{2.8}$$

Therefore, we are well within the specifications and should be able to operate the motor array without power dissipation concerns.

3. Verification

3.1 Testing Procedure

Every block in the above diagram was tested individually for completeness and operability. Each block was also tested with the other blocks to make sure that the entire system works as a whole. In order to test the Haptic Driver, we wrote a simple test program that allowed user to quickly test each motor's PWM capabilities, which was tested within the array to ensure that each motor works. For the Power Circuit block, we supplied input commands from the Arduino and measured the current drain using a power supply, ensuring that the motors were being running correctly. For the Motor Array block, we used a function generator to test the boundaries of motor behavior, and tested the functionality of the entire array after the previous blocks have been constructed and tested, specifically using the Haptic Driver.

3.2 Tolerance Analysis

The Power Circuit block for this project must work to a tight tolerance. All 32 motors cannot run when powered by the Arduino, because the motors require more current than the Arduino can supply without burning out. For this reason, we ensured that the Power Circuit block is capable of amplifying the signals from the Arduino into the range of operation for the motors.

In order to verify that the Power Circuit block works as intended, we subjected it to a variety of PWM signal scenarios, such as single to numerous motors running at once, motors running close and far from one another, and motors running with various frequencies. We tested the output vibrations and measured the output current using the power supply. We were able to confirm that the Power Circuit is able to conform to the acceptable range of inputs for the motors.

3.3 Verification Table

The Requirements and Verification table can be found in Appendix B: Simulation and Verification Figures.

In the event that we were not able to complete certain blocks of our design, following is our contingency plan. Fortunately, we did not need to follow this plan for any design blocks.

Motor Array: If the array does not work with motors, LEDs can be used to test the Power Circuit.
Power Circuit: If the Power Circuit does not work, then 4 motors can be driven directly by the Arduino.
Haptic Driver: There is no contingency plan for the haptic driver. This block must work.
Power Supply: If portable power supplies can not be achieved, then we will use a benchtop source.

3.4 Verification Tests and Results

Each component of the design was tested and verified individually and together.

The Motor Circuit was tested first by testing a single motor. A power supply was connected to a motor through the breadboard and provided 3 V across a motor. A 60 mA drain was observed from the power supply, verifying that the motor operated as specified. Then a verification circuit was built on the breadboard which included 2 TLC5940 chips, the Arduino, and all necessary connections. A simple test program was written to test a single motor operating on this circuit. We found that we did not initially specify a correct connection for ground between the Arduino and the TLC5940, and one of the chips was burned out. Once the grounds were correctly connected together, and the chip replaced, we were able to run the motor using the TLC5940. In order to test the entire array, a test program was written to allow quick testing of any individual motor by serially inputting that motor's number in the array. We then were able to quickly check that every motor was operational. We found that the chips would occasionally lose connection to the PCB, and they would need to be pushed back in place. We also found that the motor leads would occasionally break at the soldering point and need to be resoldered. Care was taken to ensure that no further breakages occurred.

The Flyback Diode Circuit was tested by connecting an oscilloscope to the output terminal of a motor, and the motor was then supplied enough voltage to begin running. The voltage characteristic was observed on the oscilloscope, and no significant voltage spike was found. The power supply was then shut off, and again the voltage characteristic was observed on the oscilloscope, and no significant voltage spike was found.

The Haptic Driver was tested first by building a verification circuit on the breadboard. A simple program was written to test the library functions of initializing and setting motor intensities. Then, the Haptic Driver code was tested piece by piece, incrementally including more functions and complexity to ensure that any bugs were easily found and fixed. Once the code appeared to be functional, serial output messages were added to numerically check computed values against expected values to ensure that all math was being calculated correctly. Finally, the code was tested using the array on skin, to ensure that differences in intensity, frequency, and location could be relayed to the user.

The Printed Circuit Board required careful checking and verification to make sure that all the power and flyback-diode circuits were implemented correctly. After soldering each component, we used an ohmmeter to test the connection across the soldered joint. We verified shorts across every pin from the TLC5940 chips to their designated connections. Every connection, including the header pins, was checked. There was also a 'negative check' implemented. Instead of testing for a 'positive' connection, every pin between the ICs to have an open or 'negative connection'. Having these verifications supported the troubleshooting process while testing the device.

4. Ethical Considerations

We will abide by the IEEE Code of Ethics. Specifically:

1. To accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment

   We will consider safety of the user. Considering we are using electronics that will be worn on the user, we will take every precaution to ensure that the user is not subjected to electric shock or burnt. Furthermore, we will not compromise the user's comfort.

3. To be honest and realistic in stating claims or estimates based on available data

   We will accurately and promptly report all results of our calculations and tests, not hiding results or misrepresenting any calculation results.

5. To improve the understanding of technology; its appropriate application, and potential consequences

   We will document the entirety of our project such that it can be understood and continued after completion.

7. To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others

   We will work to resolve and criticism or issues brought to us. Furthermore, we will properly cite and credit all sources that we use for our design or any other aspect of the project.

5. Cost And Schedule

5.1 Cost Analysis

5.1.1 Labor Cost

Billable Hours Per Week Per Engineer	Weeks Of Contract Per Engineer	Total Billable Hours Per Engineer	Multiplicity Factor	Per Hour Billing Rate Per Person	Total Contract Cost
9	13	117	2.5	$34.00	$9,945.00
No. Of Engineers in Team					2
Total Labor Cost of Project					$19,890

5.1.2 Parts Cost

Part	Quantity	Unit Cost	Total Cost + Shipping
Shaftless Vibration Motor 8x3.4mm	50	$2.79	$146.45
Arduino Duemilanove	1	$29.95	$33.59
TI TLC5941 16-Channel LED Driver	2	$2.32	$12.64
22 Gauge Wire	10 ft	$0.30/ft	$3.00
Resistors	50	$0.10	$5.00
9V Battery	12	$1.40	$16.80
AA NiMH Battery	4	$16.52	$16.52
1n5817 Schottky Diode	50	$0.11	$5.50
Total Parts Cost			**$239.50**

Total Cost of Project: $20,129.50

5.2 Schedule

Week	Date	Task	Member Responsible
1	9/10/2012	Write Proposal	Todd Pixton

		Haptic Array Design	Nathan Murray
		Haptic Driver Design	Todd Pixton
2	9/17/2012	Decide Array Parts	Nathan Murray
		Design Sensor Interface	Todd Pixton
3	9/24/2012	Order Power/Motor Parts	Nathan Murray
		Order Arduino, Interface Parts	Todd Pixton
		Design Review	Nathan, Todd
4	10/01/2012	Build Power Circuit	Nathan Murray
		Build Arduino Interface, Testing Code	Todd Pixton
5	10/08/2012	Build Motor Array	Nathan Murray
		Test Motors with Arduino	Todd Pixton
6	10/15/2012	Signal Group Communication Collab.	Nathan Murray
		Code Motor Interface Protocols	Todd Pixton
7	10/22/2012	Build Arduino Enclosure	Todd Pixton
		Build Motor Array Enclosure	Nathan Murray
8	10/29/2012	Test Connections and Functionality	Todd Pixton
9	11/05/2012	Mock Up Demo	Nathan Murray
10	11/12/2012	Mock Presentations	Nathan Murray
11	11/19/2012	Thanksgiving break	
12	11/26/2012	Resolve Problems from Mock Ups	Todd Pixton
		Prepare Presentation	Nathan Murray
		Prepare Final Paper	Todd Pixton
13	12/03/2012	Demo	Nathan Murray
		Presentation	Nathan Murray
14	12/10/2012	Final Paper	Todd Pixton

6. Conclusion

We were quite pleased with the results of our project. We were able to successfully complete our objectives, and delivered a device that met and exceeded our expectations. All sub-circuits in the design were verified to be fully functional, and they interfaced with one another as specified. The Arduino was able to receive serial input from the user, then send serial commands to the TLC 5940 chips, which were in turn able to output PWM signals for each individual motor both reliably and quickly. Each individual motor was able to respond to the TLC5940 signals, and output a strong, consistent vibrational response. The arm-band was robustly sewn and afforded the user with great modularity and configurability for the motor array. Overall, the device performed beyond our anticipation.

There are some uncertainties remaining in our design, as well. While we performed calculations that showed the circuit could run at full power without burning, that calculation is under ideal conditions. In reality, heat transfer could be worse than expected, or the motors could draw more current than expected, and the TLC5940 chips may overheat and break. Furthermore, we do not know what sort of lifespan to expect from these chips. Under heavy use, the power dissipation would still be higher than expected if they were running LEDs instead of motors. The circuit may or may not have long-lasting durability.

We are also unsure of how quickly the circuit can respond to serial input. That is, we do not know at what speed the circuit can read input, process the data, and output commands to the TLC5940 chips. The circuit works with a 20 ms delay in the Arduino processing loop, but if a user or sensor requires constant processing at speeds too great, one part of the Arduino code may fall short.

That being said, there is some room for improvement and future work. Though the device performed precisely as specified, the specifications could be made tighter to afford the user with even more precise and full haptic stimulus. To begin, we noticed that there was some degree of mechanical coupling between the vibrational motors and the arm-band. This would result in some haptic noise for the user as the arm-band would vibrate as a whole. In the worst case, when the entire topside array was vibrating, the user could clearly feel strong vibrations from the underside of the arm-band, which contained no vibrating motors. By increasing the mechanical isolation between motors and the arm-band past simple Velcro connections, this problem could be greatly reduced and a more accurate depiction of haptic data could be perceived by the user.

Furthermore, the number of dimensions of haptic feedback could be increased to allow users to experience even richer forms of stimulus. One obvious dimension that could be included in the future is timbre. Timbre could be used to describe a surface's texture or other patterns in sensory signals.

Finally, a graphical user interface would suit the Haptic Driver well. We found that controlling the motor array using serial inputs was tedious and error-prone, so a visual representation of the array that can be used to control individual motors or groups of motors would serve the program

well. It would also be beneficial to allow the user to learn to perceive the haptic stimulus by augmenting the experience with visual cues, as we found interpreting the haptic stimulus to be difficult and confusing at times.

Regarding the cost of the device, we felt that we were able to deliver a phenomenal project on a small budget. The cost listed for parts is little compared to the unique haptic experience delivered by the project.

References

[1] Texas Instruments. 2007, Oct. 16 CHANNEL LED DRIVER WITH DOT CORRECTION AND GRAYSCALE PWM CONTROL. [Online]. Availabe: http://www.ti.com/lit/ds/slvs515c/slvs515c.pdf.

[2] M. F. Nolan. 1982. Two-Point Discrimination Assessment in the Upper Limb in Young Adult Men and Women. *PHYS THER.* [Online]. p967. Available: http://www.physther.net/content/62/7/965.full.pdf

[3] Pololu. Shaftless Vibration Motor 8x3.4mm. [Online]. Available: http://www.pololu.com/catalog/product/1637

[4] PowerStream. Discharge tests of AA Batteries, Alkaline and NiMH. [Online]. Available: http://www.powerstream.com/AA-tests.htm

[5] Arduino Playground. Texas Instruments TLC5940. [Online]. Available: http://www.arduino.cc/playground/learning/TLC5940