University of Waterloo

Faculty of Engineering

SYDE 362 Final Report – Group 11

Mayank Gulati, Adam Wootton, Victor Vucicevich,

Lesia Nalepa, Imaad Umar

Prepared for Prof. Kofman

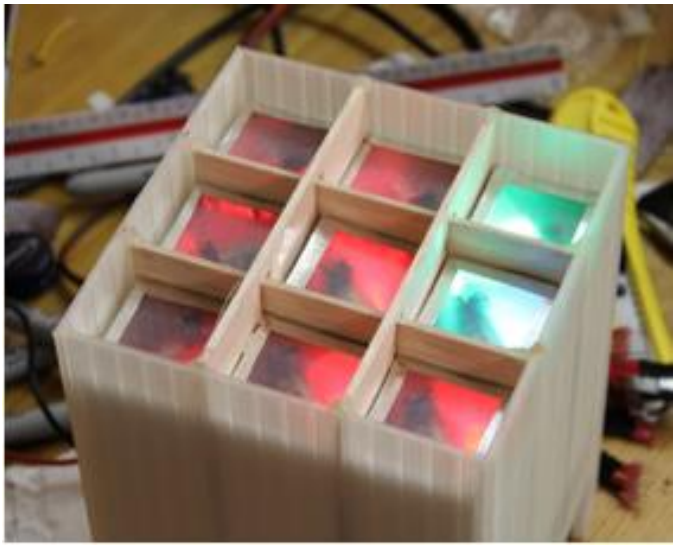
April 6 2015

Executive Summary

Create a programmable surface for makers to design, develop, and test haptic interfaces.

This is the design statement set out to be achieved. It is clear through the evaluation of current technologies that the touchscreen has taken over a significant portion of human computer interaction and taken away much of the tactile and haptic feelings of hardware controls. Even though emerging technologies are trying to tackle the challenge of programmable tacticity, there is no acceptable solution for programmable haptics.

The team achieved the goal of making a programmable haptic surface. It is a nine pixel prototype display where each pixel is capable of traditional color display and touch, as well as encoding and actuating both position and force in the vertical direction. Additionally, the group created a programmatic API and debugger for makers to test an entirely new paradigm of interfaces. A few interface demos have also been created to inspire makers and excite users for the future.



The story of the three prototypes and ample engineering analysis undertaken to fulfill the design statement has been revealed over the next 67 pages to the best of the group's abilities. The group has also indulged in the exercise of using hindsight to detail all errors and successes in the design process. If there was ever an opportunity to take further steps towards the achieving the objective, the group would work on minimizing the physical footprint of the haptic surface.

Table of Contents

Table of Figures

Table of Tables

Part A - Technical Report

1 Background Information

1.1 Design Statement

Create a programmable surface for makers to design, develop, and test **haptic** interfaces.

1.2 Project Motivation and Relevant Background

Over the last decade, especially since the introduction of the original iPhone, touchscreens have proliferated to countless devices in our lives - from smartphones, to cars, and beyond. As a result, an increasing amount of human computer interaction is now defined by finger taps, swipes, and gestures on flat glass panels. Unfortunately, the convenience of such an adaptable interaction technology comes at the cost of texture, tactile, and haptic feedback from controls.

1.2.1 Specific Situation of Concern

Touchscreens are rapidly replacing traditional interaction experiences in countless verticals, from email applications to gaming to home appliance control to even aircraft flight[1]. It is important to recognize that in many of these applications, touchscreens replace the rich tactile and haptic feedback that used to exist - joysticks and triggers on gamepads, knobs on appliances, and entire cockpits - with little more than smooth, flat glass.

Although the finger is the primary point of interaction for touchscreens, Microsoft Research is quick to point out that the primary sense engaged is vision[2]. Users need to use their sight to orient their spatial understanding of the interface with almost every tap, and developing muscle memory is much more difficult. This reorientation can drastically slow down user interaction; specifically, studies by Tactus show that "the feel of touch is about five times faster than vision," to discriminate (and use) controls [3].

The picture gets even bleaker when considering situations with limited cognitive availability of vision. In situations such as driving, where a user's vision is occupied by the road, it can be dangerous to use touchscreens. Even in everyday usage such as typing, software auto-correction is implemented to account for touchscreen inaccuracies. For visually impaired users, touchscreens can only currently be used alongside aids such as screen readers.

1.2.2 Impact

It is clear that programmable interfaces will continue to define human computer interaction in the near future. Creating the first iteration of haptic surfaces allows the primary users, makers, to design and test interfaces that can have profound HCI implications on various secondary users, visually impaired persons, drivers, and even consumers. In cases such as drivers, the impact on secondary users extends as far as societal change allowing safer roads.

As programmable tacticity and haptics continue to evolve, its conceivable that it would replace touchscreens as the de-facto standard of human computer interaction. Replacing the touchscreen will revitalize interface design across industries, providing economic injections in many markets. Furthermore, replacing single use haptic hardware with programmable surfaces will reduce physical technological waste, a large environmental stressor today.

1.2.3 State of the Art

Four current technologies which attempt to solve aspects of the situation of concern were found.

Figure 1- State of the art; from left to right: tactus, senseg, palette, and stick-on-controls.

Tactus technology uses a layer of pressurized fluid and valves to create 4 millimetre bulges in a layer of 2 millimetre elastomer film, using energy only in the transition. The layer adds programmable tacticity to touchscreens, and can distinguish touches from presses. The technology is close to market. With this technology, 3D physical haptic interfaces are achievable, however the protrusions can currently only extend to a predefined height and in a preset pattern, which reduces the number of dynamic interfaces they can be applied to.

Senseg uses an electrically variable friction panel that, when combined with accurate touch position data, can be modified to simulate textures and even ridges on an otherwise glass panel.

This allows for tactile feedback to be given to the user, but it does not enable physical haptic controls in 3D space.

Palette modularizes traditional hardware controls such as knobs, sliders, buttons, and touch panels, allowing users to set hardware controls to various desired layout and interface with them. While this does satisfy the need of having relatively dynamic hardware controls for a variety of interfaces, it does not allow on-the-fly programmatic changes to be enacted depending on the specific interface being used.

Stick-on-Controls that can be placed on touchscreens to provide a haptic and tactile interface between the touchscreen and the user have been demoed by various companies such as Lenovo. Again, these provide a physical haptic interface, but lack programmatic control.

While these solutions address different aspects of tactile and haptic controls individually, non offer haptic feedback to the user in a truly complete and programmable fashion.

1.2.4 Relevant Technologies

Two recent technological projects from MIT and Apple have also inspired the solution design.

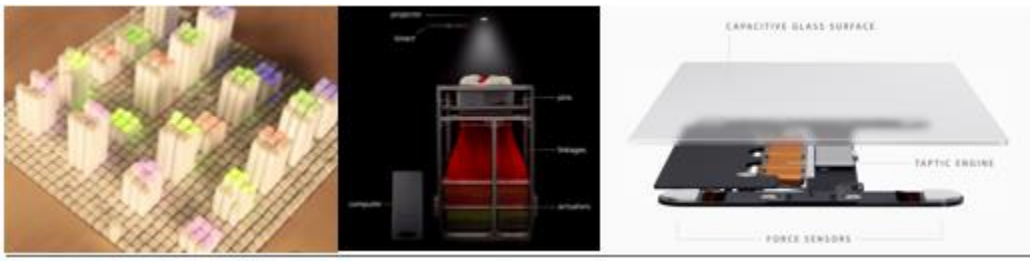


Figure 2- Relevant Technologies: MIT InForm surface and body, Macbook Force Trackpad

MIT InForm uses pneumatic pistons as pixels to actuate a 3D surface and uses a projector for display. The unit is longer than 5 feet in all three dimensions. Although multitudes of various 3D projections were demoed on the display, direct interaction with the pixels was not demonstrated.

Macbook 2015 Trackpad comes with a precise force sensor as well as a haptic vibration engine that can programmatically provide feedback. The two combine to create various click ‘feelings’. This is related to the project goal of providing rich haptic feedback in response to user interaction.

1.3 Project Goals and Objectives

The team mission is to design a display surface that will programmatically change shape and apply force to provide haptic feedback for controls. Interfaces incorporating such controls can be implemented in industrial use, aerospace, vehicle dashboards, collaborative surfaces and consumer electronics. The 5 mission goals, their champions, and objectives are identified below.

1.3.1 Goal: Shift Into Shape

Champion: Vic

Objective: Move various parts of a display surface up and down independently with enough amplitude such that it can be felt by a user. The surface must shift into the configuration as required by its current interface's control placement. The shape of the surface should allow the user to orient themselves in the interface via touch.

1.3.2 Goal: Display an Interface

Champion: Adam

Objective: Display interfaces including controls on a 3D surface. The display must be salient enough in brightness and resolution such that interface elements can be easily visually discerned by the user. The visual display must work across the screen in various 3D shape configurations.

1.3.3 Goal: Respond to User Interaction

Champion: Imaad

Objective: Receive user input from hand actions that is detailed and useful for controlling complex systems. Hand user input may include - but is not limited to - gestures, touches, presses, slides, and twists. User input data must be processed and provided in a digitally usable manner.

1.3.4 Goal: Provide Dynamic Feedback

Champion: Mayank

Objective: Provide haptic force feedback to the user's hands while they make contact with the screen. Haptic feedback may include the resistance to being deformed/pushed, or force applied. The feedback must be responsive and dynamic enough to provide haptic cues towards the functionality of controls, such as the state of a button or the limits of a range slider.

1.3.5 Goal: Design Haptic Interactions

Champion: Lesia

Objective: Improve on the quality of digital interaction in real-life situations by designing visual interfaces that sit on 3D shapes, as well as incorporate and respond to haptic controls. Designing test interfaces such as vehicle dashboards and consumer applications will allow comparative testing against existing touchscreen counterparts.

1.4 Summary of Proposed Design Solution

Any proposed solution must meet all the criteria above. The goals have been extracted into system requirements, and a proposed solution system architecture and prototype are presented.

1.4.1 High Level Solution capabilities

Touchscreen capabilities must be preserved in the solution: per-pixel capacitive touch and color display. Ideally the device would emulate touchscreens when haptics are not enabled.

Vertical Position controls for both actuating to desired positions and encoding the current position must be precise and accurate for each pixel in order to achieve tacticity.

Vertical Force controls for both actuating an active or resistive feedback force as well as encoding the force input by the user must be precise and accurate for each pixel to achieve haptics.

Form Factor of a final solution should be self contained and reasonably sized to ship to makers and allow for them to develop and test haptic interfaces out of the box.

1.4.2 Prototyped Solution

To satisfy the need for pixel-level display, touch, and vertical capabilities, the proposed solution implements a grid of identical 9 identical ‘pixels’; each one individually meeting requirements.

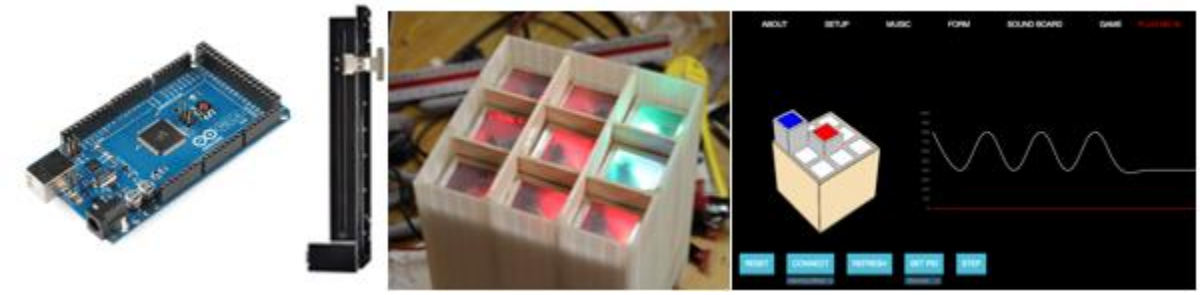


Figure 3- Prototyped Solution. Arduino Mega, Motorized Potentiometer, Form Factor, UI.

The prototyped solution uses an Arduino Mega 2560 microcontroller to interface with the 9 pixels in a 3x3 grid, emulating a screen. Each pixel is an encased square prism, incorporating a 24 bit pulse width modulation (PWM) RGB LED for display, a capacitive touch electrode surface for touch inputs, and a motorized slide potentiometer to provide vertical position and force actuation and encoding capabilities. The microcontroller is heavily optimized for smooth operation of the 9 pixel display and PID control, as well as serial communication for interfacing

with software. A Javascript API for the hardware and an API to display a live 3D render of the device in HTML have been created to allow makers to test and develop interfaces. A chrome extension containing development pages as well as interface demos is included to help makers get started. A system diagram of all the components is seen in Figure 5.

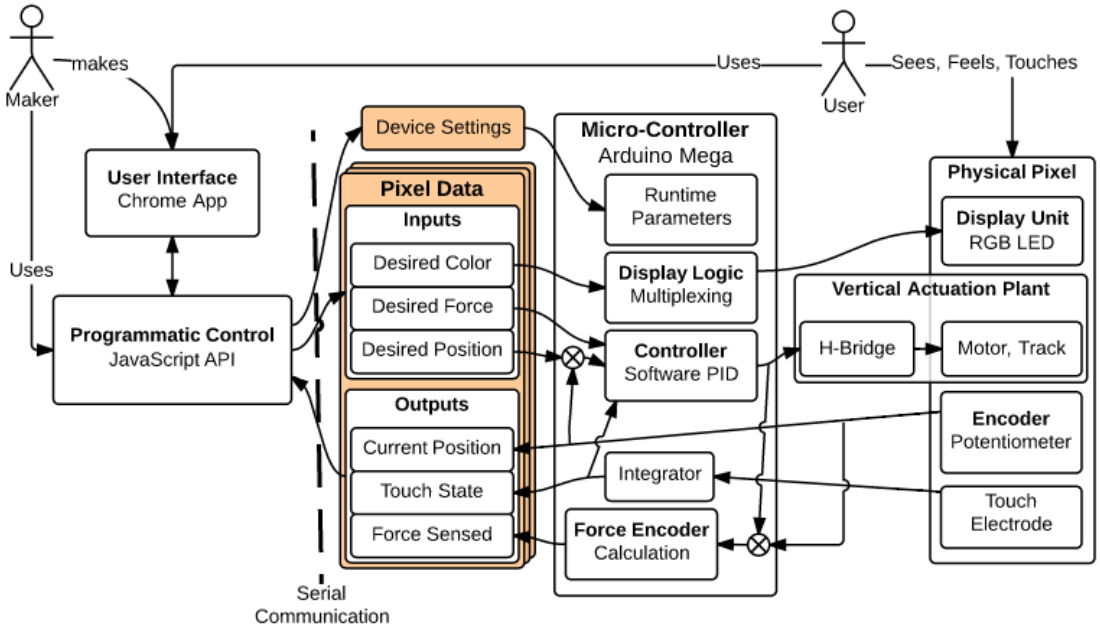


Figure 4 - System Diagram of prototyped solution. Orange indicates data.

The remainder of this report details successes and failures of the prototyped solution to meet the set requirements, justifications of the engineering decisions made during the design process of the prototype, as well as next steps towards producing a final product.

1.5 Summary of Design Requirements

In order to evaluate the success or failure of the proposed solution, the goals of the system must be converted into quantifiable engineering requirements: functional, user, data and component.

The functional requirements are those necessary to achieve the design goals of the project. In order to create an interactive haptic surface, the device needs to be able to adjust its height in a fast and accurate manner, display information, receive user input, and provide haptic feedback in response to input. All of these components are equally important to the success of the project, and all must be fulfilled in order to satisfy the situation of concern.

Table 1 - Functional Requirements

	Requirement	**Benchmark**	**Testing Procedure**
A1	**Create physical surface with meaningful tacticity**	Distance from lowest to highest achievable point should be at least 2 cm	Measure distance from lowest to highest achievable point
A2	**Display information and interface to user**	The visual and physical interface can dynamically display salient information and recognizable interfaces	Perform user testing of navigation through a multi-screen interface with distinct controls layouts
A3	**Receive user touch/press control input**	Accurate digital input is received within 20ms of initial touch	Determine average system response delay in repetitive and trivial interaction testing
A4	**Haptic response to touches**	PID loop can provide up to 1N force feedback within 40ms of interaction start	Measure time series of force feedback using a force sensor and compare to expectations
A5	**Speed of actuation**	A full stroke of actuation should take less than 500ms	Measure average time taken for a bottom to top actuation
A6	**Accuracy of actuation**	Actuation should reach target position with less than 2mm overshoot and 1mm steady-state error	Measure average overshoot and steady-state error moving to various target positions.

The purpose of the user requirements is to ensure that the product improves on existing interaction techniques and enhances a user's ability to accurately and swiftly complete tasks. The suggested testing procedures simulate interaction with a regular flat touch screen and compare the accuracy and speed of these interactions with the same ones conducted on the haptic display device. In regards to requirement B1, the haptic device must also increase the level of intuitive understanding an interface can provide. This requirement ensures that, compared to a touch screen, users will more quickly comprehend a new interface and identify all its key components.

Table 2 - User Requirements

Label	**Requirement**	**Benchmark**	**Testing Procedure**
B1	**Physical and visual interface and controls must be intuitive, salient to user**	Average time for user to correctly detect a key interface control component on a foreign interface is 100% faster than the same interface presented on a touch screen.	Design an interface containing several distinct components. On haptic display, raise interface element user must find. Conduct blind user testing and measure average time before element is identified.
B2	**Physical interface must increase interaction speed compared to touchscreens**	Identical task sequences are performed on average 15% faster with physical tacticity and haptics as opposed to a touch screen	Conduct and time user tests requiring a sequence of button presses on surface in its flat state and tactile/haptic state and compare results
B3	**Physical interface must increase accuracy compared**	Identical task sequences are performed with 20% more accuracy with tacticity vs	Conduct and score user tests requiring a sequence of button touches on

	to touchscreens	touchscreens, measured by distance from touch targets	surface in its flat state and tactile/haptic state and compare results

The data requirements listed in Table 3 ensure that the device can input and output all the information required for proper programmatic control of the interface. This includes receiving user touch input and force application and sending it to the Arduino to allow the microcontroller/computer to decide what actions to initiate in response. Current slider position must be sent to the Arduino in order to recalculate necessary motor actions and update the user interface, and these new actions must be received by the device and responded to promptly.

Table 3 - Data Requirements

Label	**Requirement**	**Benchmark**	**Testing Procedure**
C1	**Output data on multiple simultaneous user touch/press inputs**	At least 3 at a time, pixels can continuously report user interaction data, with >90% accurate touch versus press detection	Follow set pattern of multiple touches and presses on the device, compare received data with performed actions and determine accuracy
C2	**Input target position / Output current position of each 3D pixel**	Achieve target position given within 500 ms to average tolerance of 2mm per pixel, Report current position with same tolerance within 50ms	Provide matrix inputs of target positions, compare reported matrix positions and observed physical position over time, check for accuracy
C3	**Output user applied force / Input actuation force for each 3D**	Report force which user is currently applying on each slider, and receive and respond to actuation force	Measure time required to receive applied user force and respond to changes in output actuation force.

	pixel	output adjustments on each slider, both in less than 50ms	

Beyond setting functional, user and data requirements for the system as a whole, it is also important to set performance metrics and tolerances for specific system components. Display brightness and color capabilities, performance of the microcontroller and linear actuator used, physical tolerances of the cap and frame, as well as electrical component performance metrics have all been established in Table 4. The relationship between these components will be further expanded in the following sections.

Table 4 - Components

Label	**Component**	**Off the Shelf**	**Performance/Tolerance**
D1	**Arduino Microcontroller**	Yes	Must process fast enough to meet set requirements
D2	**Linear Actuators**	Yes	Stroke, speed, and force requirements from above
D3	**Display Components**	Yes	Minimum brightness 500 lumens for visibility[17] Must be able to produce 24-bit-true-color per pixel
D4	**Frame**	No	Frame is robust enough to handle 8 continuous hours of users interaction without structural failure
D5	**Caps / Screen**	No	Each pixel must have a cap that is light enough to meet actuation power requirements. Alternatively, the device must have a flexible screen capable of showing at least 2 cm deflection.
D6	**Integrated Circuits**	Yes	As required for running motors (H-bridges), PWM or Analog/Digital IO Multiplexing. Must handle current and voltage requirements of the device.

2 Engineering Specifications and Analyses

Analysis performed on various subcomponents of the system shows how the chosen solution components meet the set benchmarks and requirements. Analysis of various subcomponent alternatives that are not part of the chosen solution has also been included to justify engineering choices. Specifications derived from the solution space and requirements are reported below.

2.1 Design Specifications

The requirements in the tables above are quantified, categorized and justified as specifications below in order to allow accurate engineering analysis of subcomponents.

2.1.1 Linear Encoding and Actuation Specifications

Linear position must be reported to the maker with less than 50 millisecond lag and correct within 2 millimetres in order to allow smooth and accurate interactions for the user.

Any desired positions must be actuated within 500 milliseconds with steady state error of less than 2 millimetres in order to allow the maker to smoothly and accurately display various 3D interfaces.

Linear force up to 1 Newtown with a tolerance of 0.1 Newton must be provided within 40 milliseconds of touch interaction in the resistive case, and within 100 milliseconds in the active feedback case.

Linear force must be read with less than 50 millisecond lag and correct within 0.2 Newtons in order to allow smooth and accurate interactions for the user.

All pixels must update position readings and accept new actuation values simultaneously, defined as at least once for each pixel every 20 milliseconds.

2.1.2 Display Specifications

Each pixel must be able to reproduce all 2^{24} colors of the standard RGB model, such that makers can create interfaces with their existing color palettes.

Each pixel must be able to produce a minimum of 500 lumens for reliable visibility of display.

Each pixel must be outputting display simultaneously or equivalent thereof. Specifically, if the screen is set to fully on, no pixel should be off for more than 20 simultaneous milliseconds. This is to ensure a consistent, flicker-free display.

Every pixel screen must update at 30 times a second, in order to achieve a minimum screen refresh rate of 30 frames per second. This is important to meet standard display expectations of makers.

2.1.3 Touch Specifications

Accurate touch states must be reported within 50 millisecond lag to preserve smooth user interaction.

At least three pixels must be able to report a touch simultaneously, in order to preserve existing multi touch capabilities of touchscreens as expected by developers.

Touch must be identified if and only if the user is making direct contact with the surface of the pixel casing, both on its sides if the pixel is vertically raised and the top.

2.1.4 Electrical Specifications

The unit will run on 120 - 240VAC and must never draw more than 5A of current at once in order to safely plug in to standard 15A household outlets likely alongside computers and more.

No more than 0.5A of total current draw should occur when linear position of each pixel is stabilized at a constant value, in order to keep total power usage down in the default state.

The unit must never draw more than 1A of power from the USB port of a connected computer in order to to allow operation with most computer USB ports.

The unit must never provide more than 0.1A of power into the USB port in order to avoid damage to the motherboard of the connected computer.

2.1.5 Physical Specifications

The weight of the casings on an individual pixel must be light enough in order to satisfy the 0.5A of current draw at stable linear position requirement. Heavier casings will push down on the pixels even in passive state.

The pixels must not touch each other when placed in a base meant to allow a ¼” distance between casing walls. ¼” is a reasonable tolerance for the casing, and not too much space to make interactions awkward. If the pixels were to touch each other, the added friction would cause a failure in linear actuation and the electrical contact would fail touch readings.

The casings must be a square of width 1¼” on the top, determined based on the dimensions of the motorized slide potentiometer used as well as recommendations for designing touch targets.

The casing must be at least 4” in height in order cover the pixel’s entire linear range of motion.

The frame must be made with ⅛” walls in order to provide an attractive form factor.

The frame must allow the same ¼” between pixels and the walls as there is between pixels itself.

The frame must be tall enough that the lowest button state occurs at least ½” underneath the top of the frame, such that a maker can mimic the interactions of a depressed button.

2.1.6 Interface and Communication Specifications

A javascript API must be provided to interact with the device in order to allow makers to work with familiar languages as well as make supplementary web interfaces.

Serial communication must be fast and robust enough to update encoded pixel values as well as desired actuation values at least once for each pixel every 40ms, to meet above specifications.

A user interface that allows makers to easily test each pixel’s vertical position encoding, vertical actuation, touch encoding and display color must be provided for quick start and debugging.

2.2 Engineering Analyses of Proposed Design

Many competing design options were considered for a multitude of different aspects of the project. Each choice was made using careful engineering analysis and informed decision making. What follows is a description of the analysis completed in evaluating various options.

2.2.1 Method of Vertical Actuation

The major technological component that had to be engineered is the ‘shape shifting’ surface itself, whose purpose is to provide tactile and haptic feedback. Many techniques exist to enable such actuation on a surface, and were considered through the evolution of the design solution.

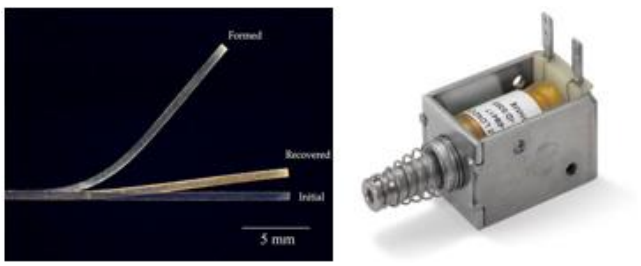


Figure 5 - Methods of Linear Actuation: SMAs, Solenoids, Motorized Linear Potentiometer.

Shape Memory Alloys are Nickel Titanium wire that are malleable at room temperature, but themselves towards a certain remembered shape when heated. It was hypothesized that SMAs would allow for creating intricate geometric shapes given a combination of remembered elementary shapes. The approach consisted of a grid of isolated, electrically actuated, shape memory alloy wires that could create rises and drops in a cloth surface. Unfortunately, testing showed that shape memory alloys provide a small (<0.5N) force towards a remembered shape rather than reliably obtain the shape itself. As such, the SMAs are suitable for small scale actuation, but not suitable for the solution. In addition, the shape memory alloys heated up significantly when conducting current, and as such would be a danger to the user if employed.

Linear Solenoids use a coil around a metal shaft to magnetically pull down the shaft when electrically activated. Using a spring underneath the shaft and careful electronic control, a range of positions can be obtained. Unfortunately in testing the solenoids used 1.5A of current and generated significant heat, which is unacceptable as this must be the device's default state.

Motorized Slide Potentiometer available from Sparkfun is a simple 10cm linear potentiometer with the slider connected to motor via a track. After initial testing, they were selected for both actuation and sensing. The unit also has a capacitive electrode situated at the slider contact, and meets all of the quantitative position actuation and encoding requirements. The largest drawbacks of using the slide potentiometer was its large physical footprint and cost.

2.2.2 Method of Encoding Position

Continuous measurements of the position of each pixel were imperative for the system to function. Various methods of encoding vertical position were compared based on following:

Accuracy(5): at minimum 2mm from actual
Precision(4): at minimum 1mm
Speed of Acquisition(5**)**: How quickly the sensor can determine the position (<50ms)
Computation Requirements(2): Computing requirements for the entire encoding system
Compactness(3): Overall size of the position encoding system

A **camera** can be positioned at an angle that allows field of view of all the pixels. The position is acquired by locating the top edge of each pixel and then determine the vertical distance of pixel

from the datum. This method requires heavy image processing and possibly two camera for proper depth perception. Furthermore, user interaction that blocks the cameras view will temporarily disable position acquisition.

A **linear potentiometer** can very accurately and quickly report the linear position of a knob on the potentiometer track. Linear potentiometers could be attached to each pixel using a variety of mechanisms, but already come built in to the chosen motorized slide potentiometers.

Magnetic Encoders such as those used to read angular position in motorized control systems could be used to encode linear position using mechanisms that translate linear movement into rotation. In addition to such mechanisms the encoders require

Table 5 - Decision matrix of possible encoders

Sensor	Accuracy (5)	Precision (4)	Speed of Acquisition (5)	Computing Requirements (2)	Compactness (3)	**Total**
Camera	2	3	2	2	1	**39**
Potentiometer	5	5	5	5	3	**89**
Magnetic	4	4	4	3	4	**74**

2.2.3 Method of Encoding Touch Input

Methods of detecting touching of pixels were evaluated on the following criteria:

Compactness (3): Overall size of the touch system.
Computation power (2): Computing requirements for entire touch system.
Accuracy (5): Correctly identify touch state of a given pixel.
Precision (5): Able to identify which singular pixel is being touched.
Granularity (2): Able to identify precise location of touch on the pixel.

A Webcam could be used to captures video of the surface which is then analyzed by a computer to determine where the users fingers and hands are placed. This faces the same computational expenses and drawbacks as a camera position encoding system.

LEAP Motion uses infrared sensors and cameras to locate fingertips in 3D space. The device is placed orthonormal at the edge of the surface, and is capable of detecting hand gestures such as pinching twisting. Testing unfortunately showed that the device does not meet desired accuracy.

Kinect, similar to the Leap Motion, uses infrared sensors to locate hands in 3D space. The computational expenses, form factor trade offs and precision were all found to be inadequate.

Photogates placed on top of each pixel would be optically tripped when a finger touches the pixel. Its limited ability to only detect touches to the top of a pixel is less than ideal.

Capacitive Touch is a popular method to detect touches on an electrode based on the change of the RC time constant due to the capacitance of the human finger. This method was selected to due ease of implementation and precision. Aluminum tape covers each pixel as the electrode.

Table 6 - Decision matrix for touch input. Brackets in headings indicate criteria weight

Sensor	Compactness (3)	Computation Requirements(2)	Precision (5)	Accuracy (5)	Granular (2)	Total
Webcam	2	2	3	3	3	**46**
Leap	3	1	4	3	5	**56**
Kinect	1	1	2	4	5	**45**
Photogate	4	3	5	4	1	**65**
Capacitive	5	4	5	5	5	**83**

2.2.4 Method of Display

Two main methods of displaying information to the user were considered.

Projectors mounted above the surface would project the interface directly onto the surface itself. This would allow for rich, high resolution information and interactions to be displayed. Unfortunately, projectors are quite expensive, require external mounting, and require a

microcontroller capable of HDMI output. Furthermore, the user's hands would obstruct a projection coming from above the device whenever the user interacts with the device.

RGB LEDs in individual pixels can be driven by 8-bit PWM to achieve 24-bit color. Although this solution requires more wiring on the microcontroller and is not nearly as high resolution as the projector, at least for the 9 pixel-prototype, LEDs allow a much smaller physical footprint.

For the reasons stated above, the per pixel RGB-LED solution was chosen for the final prototype. This allows for a common resolution of display, touch and vertical actuation; which in turn makes it easier for makers to understand and interface with the surface.

2.2.5 Number of Pixels in Grid

Deciding the number of pixels was an exercise in balancing the trade offs between cost in terms of time, work, and money and functionality of the device.

A 2x2 grid for a total of four pixels was set as the minimum goal in order to fully demonstrate different interface states and have multiple controls. However, the minimal number of controls also meant that many UI concepts would not be achievable. While four would be sufficient to demonstrate the satisfaction of the design goals, increasing the number would improve the usability and greatly increase the possible applications for the prototype device.

A 3x3 for a total of 9 pixels allowed many more possible UI designs to be realized. For example, the "Game" UI demo described in section 2.3 would not be possible without nine individual pixel elements. Increasing the scale of the project by introducing more pixels also increased the complexity of the circuit and the difficulty in constructing the prototype, however it provided a more complete solution which had a greater variety of use cases.

It was decided that financially, purchasing more than 9 sliders was unfeasible for the solution.

2.2.6 Material Selection Pixel Casings

As stated above, while resting on the slide potentiometer, the final design of the caps cannot push the slider downwards while the potentiometer is unpowered. The coefficient of static friction of the slider is unknown. As such, multiple materials were tested and the results recorded.

Figure 6 - Tested casing materials: ABS, Aluminum, Aluminum-Taped-Balsa, and 3D-Printed

ABS tubing with a 1¼" inner diameter of 3rd grade purchased from Noble Plumbing in Waterloo was tested. ABS tubing has a density of about 0.0376 lb/in³ [4]. The cost of the tube was $3/ft. The weight of a 4" tube was 76.93g. Unfortunately, when epoxied to the slider the tube pushed down in the default state, making it unsuitable for use.

Aluminum sheets of 1/16" thickness, purchased from KW Surplus for $15, as well as a roll of 1/128" aluminum were tested. A 4"x5" section of 1/128" aluminum was cut and bent into a 4" square tube weighing only 12g. When hot glued, this tube did not push down on the slider. Unfortunately the thin aluminum tubes were too fragile for reliable use. Square tubes of 1/16" thick aluminum weighed in at 123g, and moved the slider quickly down when unpowered. Three 1" diameter holes were drilled to reduce the weight by 53g, which still did not allow the slider to move up quickly enough to fulfill requirements.

Balsa wood, which is extremely light, was tested. Two sheets were bought for $10. The wood was cut and wood glue was used to create a 4" long square tube. Once dried, the balsa wood tube was measured to only 2.5g. Glued to the slider with hot glue, the slider did not move at all when stationary. When powered, the sliders moved up and down at the same speed as if it were not even on the slider. The strength, weight and cost of the material made it the ideal choice.

3D Printed PVC tubing was tested based on the model found in section 2.4.1 One 3D printed tube at high density came out to be 10g. When added to the slider, the slider stayed stationary when unpowered. However compared to the balsa, actuated upwards motion was slowed.

2.2.7 Material Selection for Cap Top

The pixel caps had to be topped off by a material that could allow light to shine through it, possibly allow for touch sensing, and not be heavy.

Plexiglass was readily and freely available, but quite hard to cut to the required dimensions with the available tools. as such it was quickly dismissed as an option.

Screen Protector material from a phone was thought to be a good choice, but found to be too expensive for use in the design.

Figure 7 - Plastic covers diffusing light.

Diffusive Plastic from the covers for collated reports costs a few cents per pixel and were unmeasurably light on a scale. They also diffuse the light very well, as seen below. For these reasons, the diffusive plastic is the selected material for the top of each pixel casing.

2.2.8 Pixel Placement in Enclosure

The placement of the pixels within an enclosure is a question of usability and modularity. With the pixels being placed in a 3x3 grid, this creates a rigidity in the design. With this rigidity of design, the possible applications of the design are limited. However, the original reasoning behind this project was to simulate a portion of a screen in which each ‘pixel’ could move. Placing the pixels in a 3x3 grid aligns closely with the original project idea and scope. It also allows for easier wiring schemes, for the wires will all be stationary. Having one cohesive unit will also allow for easier descriptions of use cases that others will understand.

However, the idea of individual pixel placement outside of an enclosure is an interesting one. If each pixel were to have its own enclosure, this would allow for far more use cases. Even the use cases of a single pixel instead of nine becomes quite large when one considers the possibility of not simply putting the pixels on a horizontal surface. Were pixels to be able to

extrude from a wall, there could be many uses such as thermostats that protrude from the wall only when needed. Individually placed pixels also allow for more ergonomic placements. A set of pixels grouped together could be less ergonomic than having them further spaced apart, depending on the use case.

With all of this considered, the design to go with will be the grid of pixels within an enclosure. It represents the idea at hand better than if there were individually placed. The issue of complex wiring is also one that is a large concern. Also, individual encasings would need to be built for each individual pixel were they to be separate. In a single encasing, only one enclosure would need to be built.

2.2.9 Enclosure Sizing

With one of each of all possible caps for the pixels created, the final pixel sizes are now known. The largest of the possible solutions is the aluminum cap which is 1.5” in width, and the balsa caps which are also 1.5” in width. With this in mind, a distance of ⅛” will be given between each slider, and the sliders and the walls, resulting in a 5¼”x5¼” enclosure.

However the enclosure is made, there needs to be a gap from the bottom of the enclosure to the surface of which it sits upon. This is to allow for extra space for the wires from the sliders to fall through and turn outwards. A gap of 1½” should be sufficient. To allow for the sliders to have full range of motion while sitting within the enclosure, while not exceeding the highest point of the enclosure, the enclosure (excluding the bottom gap) should be a height of roughly 7½”. For added safety, an extra .1” should be added, resulting in a final dimension of 5¼”x5¼”x7⅗”.

2.2.10 Enclosure Material Selection

Due to size being the only restriction put on the enclosure, the material selection became a process of deciding which is easier to work with, and which is more robust. The materials at our disposal were aluminum, balsa wood, and 3D printed PVC.

Figure 8 - Enclosure Material Selection: Aluminum and 3D printed frame.

Aluminum while quite robust, is difficult to work with in terms of creating an enclosure. Sheet metal has to be bent and cut into shape, and then epoxied together with little room for error, given that epoxy will need to be removed and the parts reassembled if there is an error. While not exactly a difficult process, it is extremely undesirable and potentially time consuming.

Balsa while quite easy to work with, is simply not robust enough. To be interacted with with hours on end, it is a concern for larger pieces of balsa. The weight of the balsa was useful for the casing of the pixels, however, having such a lightweight enclosure could be the cause of easy to avoid accidents due to how easy it would be to move at such light weights.

3D Printed materials are rather lightweight, and the drawings, specifications, and dimensions could all quite easily be done in solidworks with high printing precision. Although Low density 3D Printing from VanDerZwaag was chosen for the final solution due to high precision and low cost, an aluminum case had to be used due to small errors and the ability to reprint.

2.2.11 Choice of Microcontroller

In order to allow the PC interface to interact with the physical device, a microcontroller is needed to receive and send serial commands and control digital pins connected to the various system components. To properly achieve these goals, it is necessary for the microcontroller to be fast enough to process all necessary commands and calculations, while also being physically capable of controlling the components with sufficient input and output pins. It is important to calculate the pin requirements of a microcontroller in order to make the correct decisions. Based on the chosen components, the following pins are required:

For each pixel:

- 2 digital output pins for H-Bridge direction control
- 1 PWM pin for motor power control
- 1 analog input for slider potentiometer feedback
- 1 digital input pin for capacitive touch input

For the LEDs, multiplexed as described later:

- 6 PWM outputs for two groups of RGB
- 5 digital output pins to address led groups

1 digital out for capacitive touch send pin to measure RC constant

Summing and multiplying for nine pixels and LEDs, the requirements are as follows:

- 24 digital output Pins
- 15 PWM Pins
- 9 Analog Inputs
- 9 Digital input Pins

For the purposes of this project, three competing microcontrollers were considered. Their relevant specifications are summarized in the following table.

Table 7 - Relevant specifications of competing microcontrollers

Name	**Speed**	**Analog In**	**Digital I/O**	**PWM Pins**	**Cost**
Raspberry Pi[8]	900 MHz	0	40	0	$35
Arduino Uno[7]	16 MHz	6	14	6	$30
Arduino Mega[9]	16 Mhz	16	54	15	$50

Based on the table, it appears that only the Arduino Mega has sufficient pins to satisfy the requirements. However, it is also possible to use the Arduino Uno if multiplexing techniques are employed, although issues with multiplexing mentioned in later sections made this solution unfeasible for the prototype. The raspberry pi would have been beneficial to use with a projector

due to HDMI input if it was used, but does not provide PWM outputs. As such the Mega as the only justifiable microcontroller choice. A $25 Arduino Mega clone was used.

2.2.12 Method of Programmatic Control

Interfacing with the microcontroller had to be optimized to allow a fast and robust two way channel between the microcontroller and the maker's app. The format of the communication protocol, the baud rate and the API for the maker are all important considerations.

Serial I/O Format

The serial commands sent between the Arduino and PC needed a common standard protocol in order to communicate information reliably. A single command is structured as:

[Pixel ID] [Command ID] [Other Data]

The pixel ID denotes which pixel the command is intended for, while the command ID establishes what kind of command the message is. The data which follows is specific to the type of command. For example, a valid command is presented below:

1C255000000

This command addresses the pixel with ID 1, and sends the command ID 'C' which is defined as "change colour". All characters following that are the RGB colour values, which range from 000 to 255 for each colour.

In this format, the minimum number of characters to convey all necessary information is used. Instead of using delimiter characters to separate different sections, the protocol is designed so that the first character is always the pixel ID, and the second character is always the command ID. By minimizing characters, the volume of serial traffic is reduced and more commands can be sent per second without overflowing the Arduino serial buffer. The "other data" section varies in length, but this does not present a problem because it is placed at the end of the message after all the special characters and can simply be read as "character 3 until the end of the message".

Serial baud and I/O rates

Both the Arduino and PC serial interfaces are capable of supporting multiple baud transmission rates. A higher rate means that more characters can be sent per second. Higher rates are usually prone to transmission error which results in lost messages, so the group needed to find the fastest rate with a 100% success rate. Through trial and error testing, the rate 115200 was chosen as it

was determined that this was the highest rate both sides of the serial link could support before errors began to occur.

Serial over Web Technologies

In order to facilitate serial communication on the PC side and establish communication between this and an HTML powered front-end interface, two major technologies were considered. The first was a web application running locally on the laptop which would communicate with the Javascript page using web requests. The chosen language for this server was Python, because it included a simple serial interface to interact with the Arduino. In effect, the server would act as a "relay" to send messages between the Javascript interface and the Arduino. The competing option was to use the Google Chrome serial API to communicate directly between the Javascript interface and Arduino, sidestepping the need for a relay.

In comparing these two options, the main decision factors were speed and complexity. Using the Chrome API allowed the serial interface to consist of two components rather than three, which eliminates a major point of failure and makes the system much more robust. Additionally, the speed of communication using the two systems was measured, and the web server approach introduced a delay of up to 10 ms for each message. The Chrome API had a delay of less than 2 ms. As such, the API was clearly the better choice, and was selected for the final prototype.

Javascript Hardware API

Due to the fact that the device was targeted towards makers, the Javascript interface code needed to be designed in a modular way so that the core device functionality could be easily added to any web application. To accomplish this, a reusable library was written which provided an easy-to-use API to interface with the device itself. This allowed many interface prototypes to be quickly developed using a common core library, significantly reducing the complexity of the codebase and development time.

2.2.13 Method of Initial Interaction for Maker

Referring back to the design statement, the objective of the design solution is to create a programmable surface for makers to design, develop and test haptic interfaces. In order to provide makers with a way to interact with the platform and design on it, the physical platform was to be accompanied by a web application that interfaces with the physical matrix. In order to

provide value to the user, the web application needed to be intuitive and easy to use. Specifically, it needed to provide users with a way to set the position and colour of each individual pixel.

Developer Page

In order to satisfy these requirements, an interactive three dimensional model was generated using HTML, CSS and JavaScript. The position of each pixel can be set by clicking on the pixel and dragging vertically. Similarly, the colour of each pixel and can set by clicking on the pixel and dragging it horizontally. Moving horizontally cycles through a collection of colours that have been tested to work on the RGB LEDs. In addition, the colours cycle in a gradient, so that the user can intuitively know whether he/she should move left or right to reach a desired colour. This setup page also allows a user to graph the position pixels and their step response.

The web application also includes several examples of the applications of the platform to provide makers with inspiration and show them what is possible. Examples include an interactive soundboard, an audio visualizer, a game and a simple form. Additional features include auto connection, a startup animation and a status message. Although these are simple features, they contribute to a pleasant user experience. Screenshots of all pages can be found in later sections.

2.2.14 Defining Haptic User Interactions

In order to design effective interfaces using the haptic device, modes of user interaction must be defined for each pixel that can then be incorporated into plans for new kinds of interfaces. Such interaction modes can be defined as resistance (or force feedback) profiles along the position.

Button Presses have a specific feel where there is more resistance at the start and end of a button press. The especially high resistance at the end is the haptic confirmation of a click.

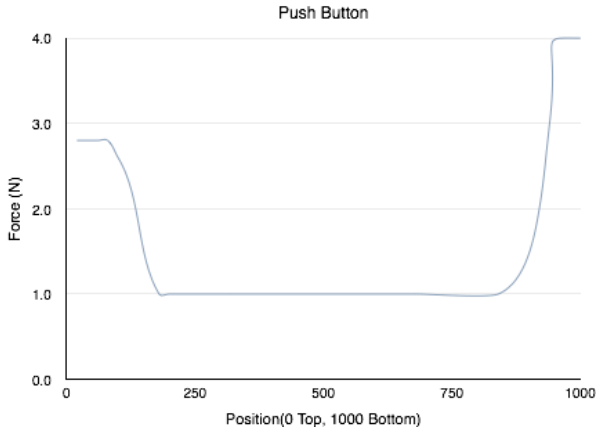


Figure 9 - Resistive force vs position of a button press mapped using a force sensor.

Touches represent capacitive touches instead of presses. Since this can be done at any position and does not impact the position, they do not require resistance modeling.

Slides allow users to move the pixel to a desired position physically, indicating an equivalent desired value to the interface. As the chosen linear potentiometers are the basis of sliders PID controls can be entirely turned off to achieve slide interactions. The k_D term can be set to nonzero to mimic more stiff sliders, and sliders can also be made stiff near the top and bottom of the allowed range to indicate the end of the slidable region.

Toggles are buttons that switch between position states upon each stage. In order to mimic toggles, the press interaction can be used followed by setting a desired position representing a toggle state and turning on full PID controls. For more precise mimicking of toggle haptics, the resistive action can be turned off when a pixel is pushed past a low threshold, and then PID controls can be reactivated with the new desired position. Such haptic feedback allows users to know when the button has been toggled.

2.3 Optimization of Selected Design

2.3.1 Final Pixel Cap Design

Due to the to the design of the PID controls and modelling of button presses, the weight of the caps to the pixels need to be as low as possible. This is because the added weight acts as an external disturbance on the PID control system. This was seen very clearly when a heavier cap, such as the aluminum or even the 3D printed cap was placed on the slider. When these caps were on the sliders, the slider control caused the downward motion of the slider to be far faster than the upward motion. Accounting for this added weight is extremely complicated. Given that the the balsa design had very little effects on this motion, the balsa wood is to be the final material over the 3D printed caps.

However, given that the balsa wood is strong vertically but not horizontally, there is a concern that the balsa wood is not robust or strong enough to withstand user interaction for prolonged periods of time. The weakest points on the caps are the glued corners. To increase the strength of these points, tape was wrapped around the caps. However, as not to remove from the visual aesthetics, simple masking tape or duct-tape will not do. Aluminum tape was donated to our group. Aluminum tape gives the wonderful illusion of the caps actually being aluminum.

Upon testing the strength, the glue on the balsa wood could withstand 10 pounds of weight directly on the side of the cap. With the aluminum tape, a max of 20 pounds was tested without failure. This was deemed as a reasonable strength and no further strength testing was conducted, especially given that the majority of the force would be from the top.

As for the cap tops, similarly, the lightest of the options was chosen. However, due to these tops not being capacitive, a new method of touch sensing must be employed. Luckily, because the caps have now been wrapped in a conductive aluminum tape, and due to the fact that the cap tops are also held on via the aluminum tape, the capacitive touch is connected directly to the cap. Now the user just has to touch the tape on the cap, which is quite easy. Also, the cap tops diffuse the light of the LED inside which allow for the light to spread across the entire top.

2.3.2 Final Enclosure Design

As discussed in section 2.2.7, both an aluminum casing and a 3D printed one are to be built. The 3D printed case designs can be found in section 2.4.1. The base in which the sliders sit within are specifically designed such that they can sit in any orientation desired. There are also notches on four parts of the base which the sliders sit upon. These notches were designed specifically to be cut out for wires to pass through. Only one notch per slider needs to be cut, and the other three remain to provide structural stability.

The 3D printed frame design is quite simple, as seen in section 2.4.1. It is simply a box with legs. However one modification to the design was made; there is a lip on the inside of the frame. The base which the sliders sit on top of will slide into the frame and rest on the lip. Thin aluminum, wrapped in packing tape to prevent conductivity, will be placed between the caps as dividers to prevent them from touching. This is both to prevent friction between the caps as well as to prevent the touch sensing from hopping between caps.

An aluminum version of the frame was also created. This aluminum frame is 6”x6”x6”. This frame also uses the 3D printed base that holds the sliders. The aluminum frame is held together with both epoxy and aluminum tape. Paper is used along the insides of the aluminum frame to prevent capacitive touch from jumping between caps. Also, the increased size is a precaution, just in case the 3D printed frame is too small, or the sliders are not on straight.

The aluminum frame does not have ‘feet’ to raise the it like the 3D printed frame. However, a cardboard box has been cut to allow for wires to move from the bottom of the frame

outwards onto the breadboards. With the frame and sliders on top of the cardboard box, the cardboard box “bows” due to the low strength of the cardboard. So, it was decided that the box would be wrapped in a thin aluminum. Thin aluminum was chosen over the 1/16” aluminum because it is far lighter, making the entire unit still easy to move.

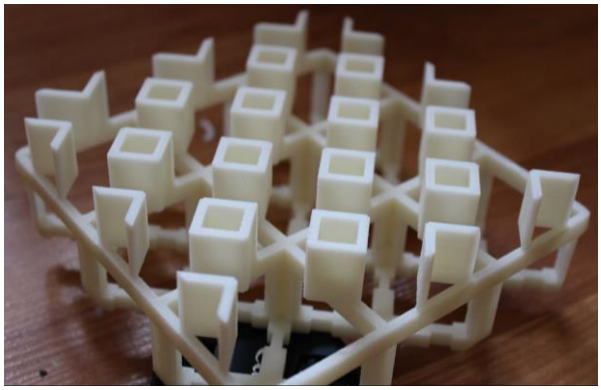


Figure 10 - The 3D printed base.

2.3.3 Electrical Optimization

As evidenced by the pin requirements in section 2.2.11, there is a large amount of wiring (over 200 connections) in the final solution. As such many optimizations were made in order to allow easy wiring, debugging and testing of the solution.

6-Pin Female Headers were stuck to the bottom of each slider with wire soldered into the contact points available on the slider. This allowed easily plugging each slider into a 6-pin jumper wire to connect it to the breadboard. The female headers can also be used with PCBS.

The Breadboard Layout itself was optimized using the fritzing app before wiring up. The H-bridges are placed close to the arduino, and the connection points for the pixels are located on the other end of the breadboard. Wires between the pixels and H-Bridges/Arduino are placed on the outer rows of the breadboard bent outwards, leaving ample space in the inner rows to plug in jumper wires from the pixels. This can be seen in the figure below.

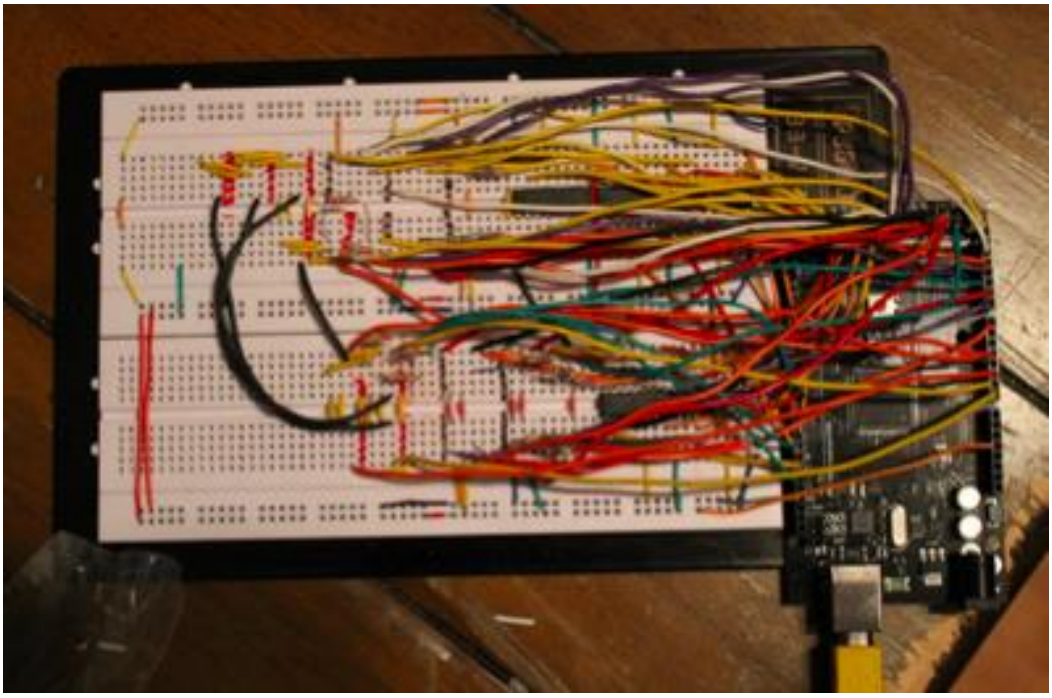


Figure 11 - Breadboard wiring, with inner rows left clear for plugging in jumpers.

2.3.4 LED Multiplexing Optimization

In order to reduce the number of pwm outputs required to control 27 LEDs (9 pixels x 3RGB), a grid multiplexing system was used. The LEDs can be divided into groups, where each LED in the group shares ground connections. The PWM output connections are connected to one LED from each group. The groups are addressed by setting the ground pin of exclusively the desired group to zero. The desired colors of the leds in the addressed group can then be sent over PWM. By quickly cycling through each group, the LEDs create an illusion of staying on.

It is important to realize that each LED only gets to turn on for one cycle per period, where the period consists the same number of cycles as groups. As such a high number of groups makes the flickering of the lights perceivable. On the other hand, the larger the number of groups, the less PWM pins are required to address each light in a group. On the Arduino Mega specifically, if 9 PWM outputs are reserved for the sliders, 6 are left for the lights. As such, 6 PWMs can be used to address 5 groups of 2 RGB LEDs each for a total of 10 RGB LEDs. As well, five groups is low enough that flickering is not perceivable at high clock speeds.

Another way to multiplex the LEDs is to use an LED driver chip that keeps each LED always on, and updates the PWM values periodically. Such a driver, specifically the TLC5940 was tested,

but was not compatible with the common cathode LEDs used. As such the solution used grid based multiplexing rather than driver based multiplexing. Specifically, five addressable grounds and six driving PWM outputs are used.

2.3.5 Optimizing Touch Circuit

The touch sensor works on the principle of measuring the time constant in an RC circuit. Here the circuit consists of a digital pin on the arduino sending an output signal through a 10MOhm resistor to the aluminum tape (sensor). The sensor is then connected to the input pin, this time a digital input pin. The output pin oscillates between high and low (5V,0V) and the time for the input pin to reach the same value is recorded. The time difference allows one to calculate the time constant of the RC circuit made up of the sensor and the resistor. As a hand approaches the sensor the capacitance changes which in turn changes the time constant. This input is buffered to smooth out any spikes. The input can then be interpreted as the hands distance to the sensor and a threshold is set to determine “touch”.

The touch electrode was also heavily optimized as the touch surface greatly affects the precision, accuracy and implementation of touch sensor package. The most notable difference arises from the surface area of the electrode - the larger the surface are the more sensitive the system (able to detect from further away). The size and weight for the electrode had to be optimized so that it take the least amount of space, minimized weight and remains attachable. The built in electrode is simply a small metal plate measuring 2cm by 1cm. This size offers good response for direct contact touches; however this contact point is hidden underneath the button cap. Attaching a wire to this surface to an external surface was investigated but this would decrease the amount of light from the LED a user could see and therefore was not chosen.

The chosen approach was to attach a 3cm x 3cm square of Aluminum tape to the metal plate so that the larger surface area allows for indirect touch sensing (sensing from underneath the cap), is hidden from view, does not add significant weight or friction to the pixel and is easy to install.

The other major component (the R in RC) can also be optimized to yield the best sensing results. Due to the fact that most of the restraints were on the electrode choice, the resistor was chosen after optimizing the electrode.

A larger resistor offers creates greater sensitivity in the system and allows the capacitance change to be measured for a body further away. A resistance value of 10MOhm was chosen to correspond to the 4x4cm square of Aluminum tape.

2.3.6 Optimizing Arduino Code

The Arduino code operated using an execution “loop”, which meant that the software defined a sequence of operations for the loop to complete, and when all tasks were finished the loop sequence simply repeated again from the beginning. These tasks included things like reading current pixel positions from the analog input, calculating the next control action with the PID loop, or updating LED colours. Optimizing this code consisted of ensuring the loop completed as fast as possible so that it could be executed more often, keeping things like the current pixel position as up-to-date as possible.

Due to the fact that the Arduino has very limited computational resources, it was found that performing too many periodic tasks on the same time interval would cause each one to not be performed often enough. Every task would need to be processed and completed before the next period could begin. In order to prioritize more important operations and make them occur more often, each individual category of task the Arduino periodically completed was assigned a different counter to determine how often it should occur. These counters were incremented during each execution loop, and when they reached their target thresholds their respective task would be executed. In effect, the counters would cause the tasks to run every certain number of loops, rather than run everything on each loop. The threshold for each counter varied between tasks, to ensure that two tasks never ran together in the same execution loop iteration. Through repeated trials, each counter value was determined in order to allow sufficiently often repetition for each task without detracting from the performance of the others. In effect, this implemented a priority queue execution style rather than a linear execution of processes. The result of this implementation in the final solution meets all the speed requirements set earlier.

2.3.7 PID Optimizations

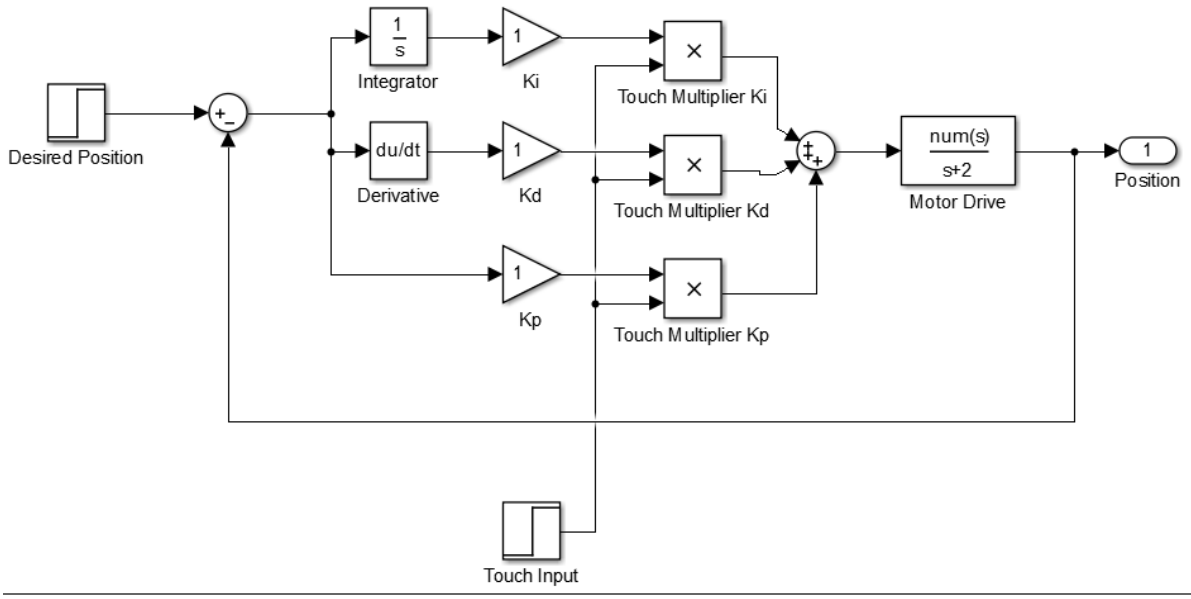


Figure 12 - Block diagram of the control system.

The control loop implemented above is a modified version of the traditional PID controller in that it includes a binary feedforward loop for touch input modeled as disturbance. The purpose of including the touch input is to modulate the individual gains for the PID loop to change the feel of a pixel (modeling a button). For example, when the user is pressing down the on the pixel the objective would be to give simulated resistance but not to fight the users input. This would be achieved by setting Ki and Kp to 0 while keeping a positive value of Kd. This is achieved using the multiplier blocks in the control loop above.

Through extensive testing, the optimal PID gain values for a general use-case were found to be the following:

Kp = 10 Ki = 0.02 Kd = 0.2

The values were found through a semi-automated iterative process where each iteration involved changing the gain values and measuring performance. The sequence was:

1) Roughly tune Kp
2) Finely tune Ki to minimize steady state error
3) Finely tune Kp to minimize 0-90 rise time and overshoot

4) Finely tune Kd to minimize oscillation and settling time

Detailed simulations were avoided due to the complicated interactions of friction in the casing and non-linear torque response of the motors. It was decided that informed trial-and-error testing would be a more effective and rapid approach to tuning the PID coefficients.

Another set of coefficients was developed to simulate “stiff” feedback in the motor. This involved increasing the derivative coefficient term in order to dampen the response and make it more difficult for users to press the pixel downwards. With these coefficients, the motor provided a stronger resistive force against counter-motion.

2.3.8 Non Linear Potentiometer Correction

The potentiometer that is supplied with the linear actuators unfortunately does not have a linear relationship between absolute position and resistance. At the ends of the potentiometer the values become non linear. To compensate for this a non linear mapping was used when converting resistance (voltage input) to absolute position. The values for the non linear mapping were obtained through the supplied manufacturer's data chart, displayed in the following figure:

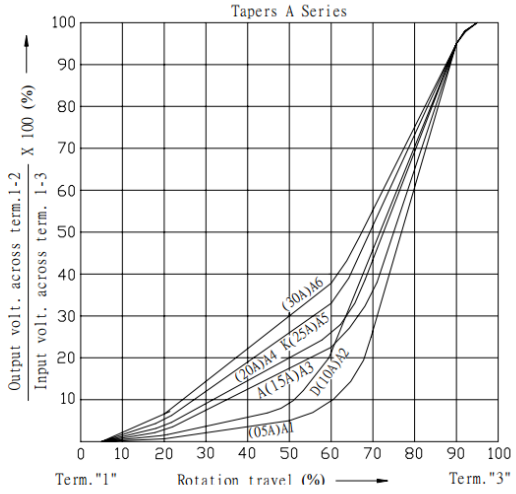


Figure 13 - Non Linear position mapping[6]

2.3.9 Optimizing User Interface

Besides the dev page, several demos were designed in order to investigate and demonstrate the new kinds of user interaction made possible by the device. Each demo demonstrates a different

kind of possible interaction. The included demos have each been optimized to display their respective aspect of the device, and together display all key aspects of the device.

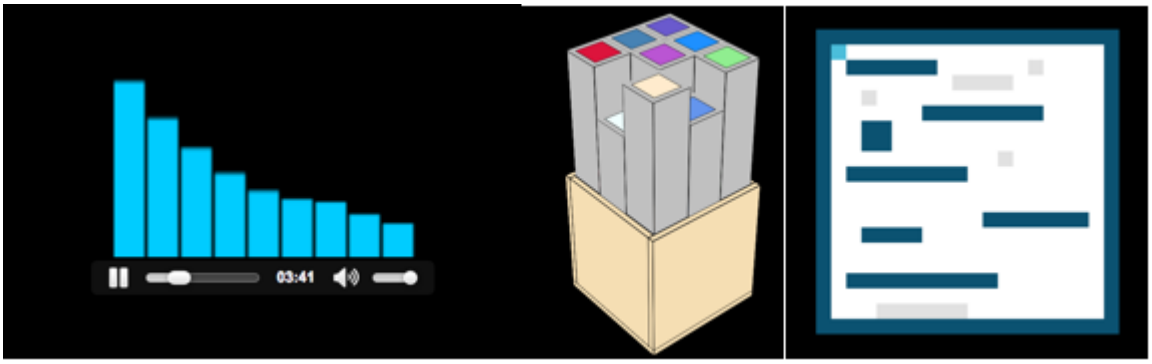


Figure 14 - Included interfaces: Audio Visualizer, Sound Board, and Game.

Audio Visualizer uses the frequency spectrum of songs to create a music player which "visualizes" the music as it plays. The spectrum is divided into nine sections, the the amplitudes of each frequency are averaged inside each section to create nine bars with heights determined by this average amplitude. The height of the bars is then sent to the haptic device, with each height determining the height of a corresponding pixel. As the music plays, the pixels quickly change heights to match the music. Additionally, the LEDs in each pixel change colour to match the current height of the pixel. The purpose of this demo is to demonstrate how the device can be used for different kinds of physical feedback in reaction to events on the user interface. It shows that the system can rapidly and accurately adjust positions and LED colours in order to physically convey information that was previously confined to a purely digital domain.

Sound Board is designed to demonstrate how the haptic device can aid in tasks where the user's full visual attention is not directed towards the interface they are interacting with. In this case, the interface is envisioned to be used by DJs and other performing artists who need to play pre-recorded "samples" at specific times to construct songs on-stage. To facilitate this, each button on the device plays a different song sample when pressed. As the sample starts, the pixel moves to its lowest position, then moves back up at a rate determined by the length of the sample. At the moment the sample is finished playing, the pixel returns to its full height. This functionality allows DJs performing on stage to feel the current state of the samples they are playing, rather than having to look at a display in order to know when to start the next sample or perform other tasks. The physical feedback provided by this interface aids in reducing the cognitive load

experienced by performers by separating the information into different human sensory modalities (touch and sight). As a result, the different sources of information do not consume as many attentional resources as if they were all concurrent visual modalities.[10]

Game was designed to illustrate the device's applications for contextual controls and haptic feedback. Each button on the device was mapped to a direction to create an interactive controller for a simple game. Pressing each button would move the player's current location in that corresponding direction. However, when the in-game player tile was next to a wall and therefore could not move in that direction, the buttons mapped to those directions were moved down to be flush with the surface and disabled. This immediately indicates to the user which controls are available and which are disabled. When the player moves away from the wall, the previously disabled controls move back up.

The haptic feedback feature was demonstrated by having "rough" and "normal" tiles which the player can move over. When moving onto a rough tile, the PID feedback would be adjusted to make the slider more stiff to press. This gives the player a feeling of greater mechanical difficulty when walking on these tiles, which cognitively maps to the feeling of walking on rough ground and gives the interface another dimension of interactivity.

Form was included as a demo in order to demonstrate the use of buttons and sliders, particularly their various states. The form begins with three sliders protruding from the surface as well as 2 disabled buttons. When the user changes the position of the sliders, the date on the right side of the interface updates accordingly. In addition, the 'CANCEL' button rises and changes from grey to red to signal to the user that he/she can now press this button. If the user chooses to press this button, the entered information is lost and the form returns to its initial state. Only when the user enters the day, month and year using the sliders does the 'SUBMIT' button rise and change from grey to green.

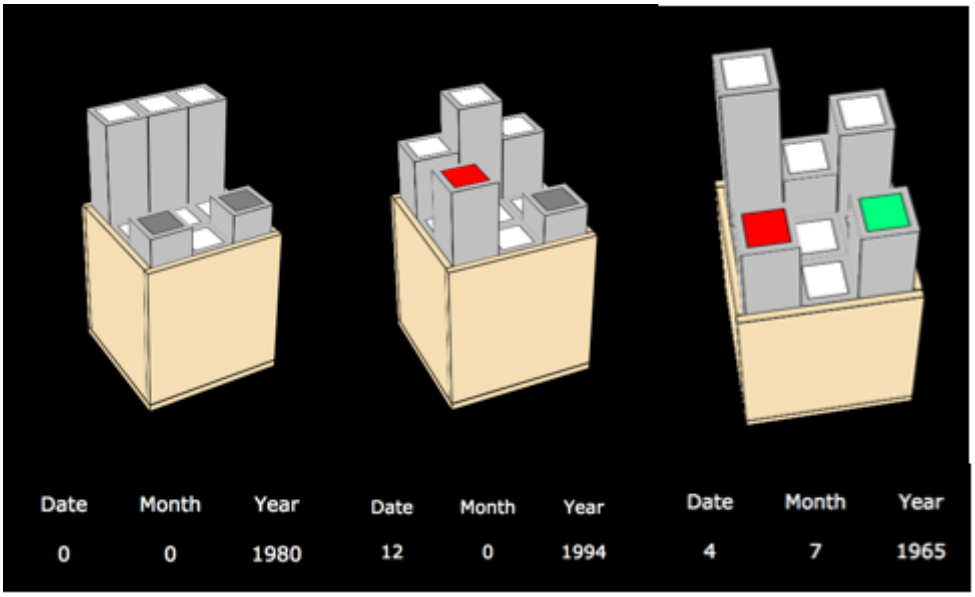


Figure 15 - 3 Stages of the form: Empty, In Progress and Filled

In all stages, both position and colour are used to indicate to the user the state of the button, employing redundancy gain.[11] Disabled buttons are characterized by their grey colour and low position. They hint to the user that these buttons are unavailable and pressing on them will not have any resulting action. On the other hand, enabled buttons are characterized by higher positions and colours that help indicate their purpose. Negative actions such as 'CANCEL' are associated with the colour red while positive actions such as 'SUBMIT' are associated with the colour green. This colour scheme is in accordance with the principle of consistency.[11]

2.4 Fabrication Specifications and Final Detailed Design

2.4.1 Technical Drawings

The first design drawn up was the base. This base was to be placed inside the frame and sit on a lip inside the frame, as discussed in section 2.2.7. The base had nine square 'insets' for which sliders could sit inside. Each 'inset' had four directions in which the slider could sit, with walls on each corner to keep the sliders upright. This design of four possible sitting directions was to allow the sliders to sit in whichever direction made the insertion of the slider easier. The bottom of the base actually sat ¾" below the lip. This was so the wires could have more freedom of movement, allowing for easier placement.

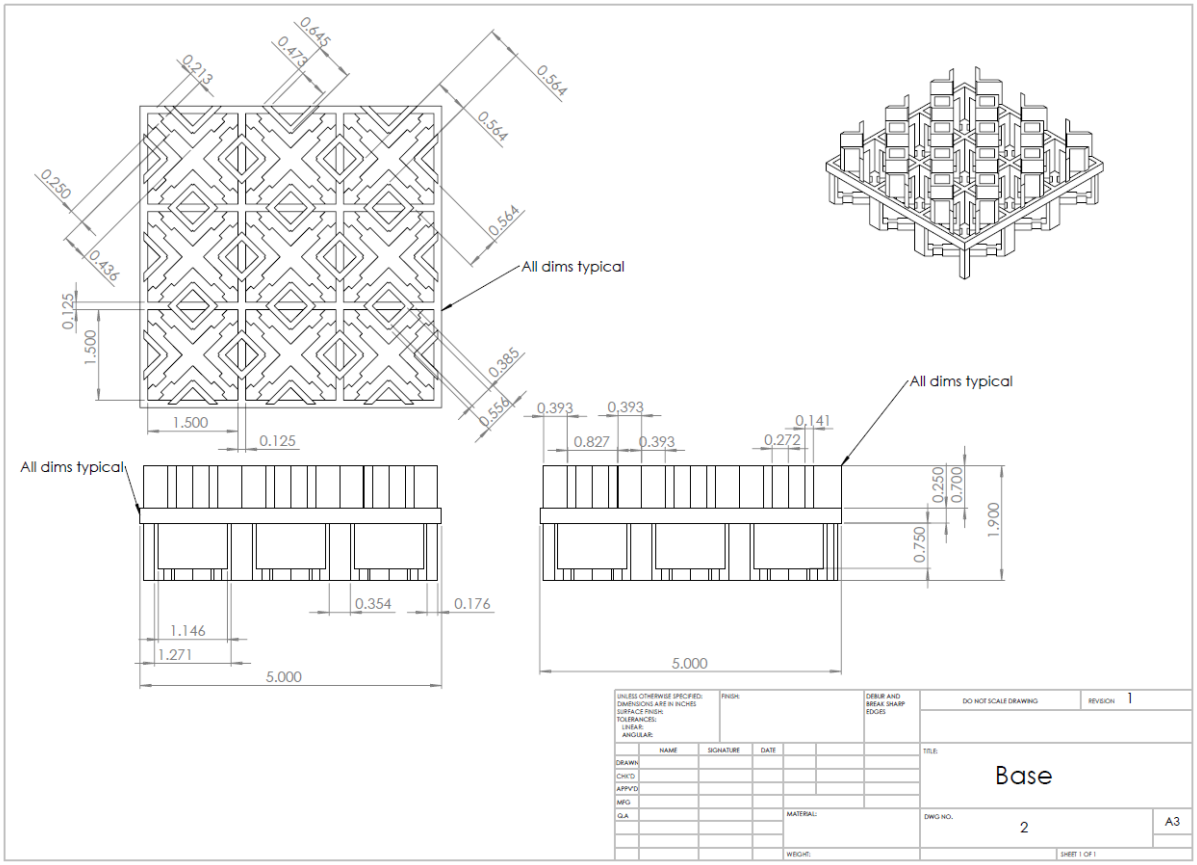


Figure 16 - Technical drawing of the base.

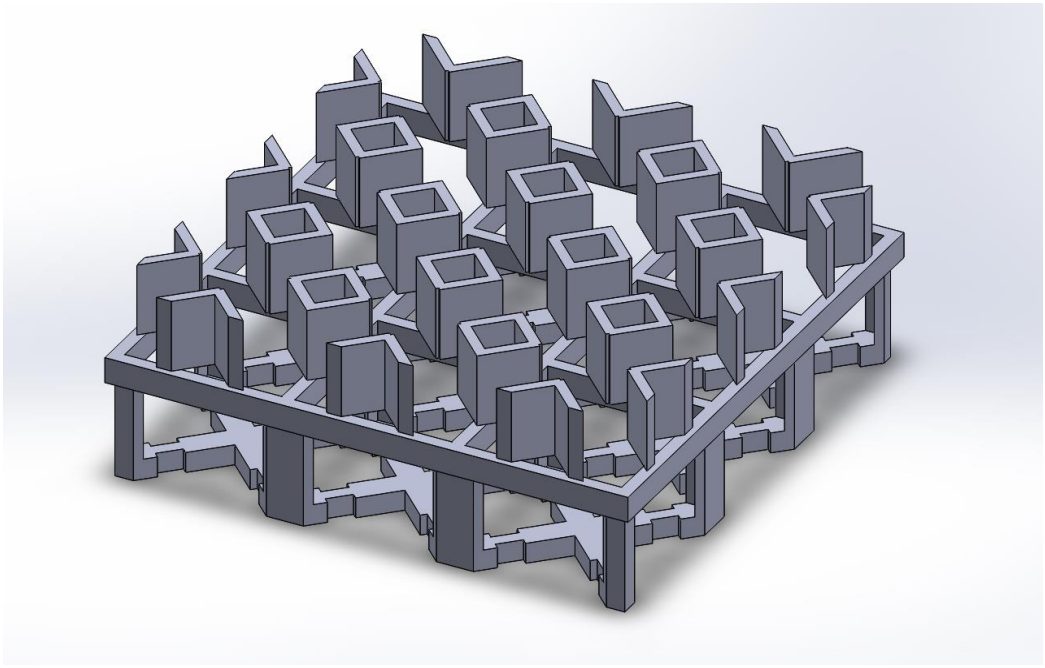


Figure 17 - Three dimensional model of the base.

The final result of the base was quite complicated. However, due to the modularity of the sitting positions of the sliders as well as the notches which to cut out for the wires as discussed in 2.2.7, the complexity of the base was necessary. This model was sent out to be 3D printed in Brantford at the 3D Print shop, and was picked up the next day after its completion. This completed print can be seen in section 2.3.2.

The pixel cap design is described in section 2.2.6. Things to note from the technical drawing is that there is an addition of ridges on the tops of the caps to help users pull the caps upwards if needed. The addition of ridges will greatly help the usability of these caps, given that some UI examples provided are things such as volume control and other uses where the user would need to move the cap up and down.

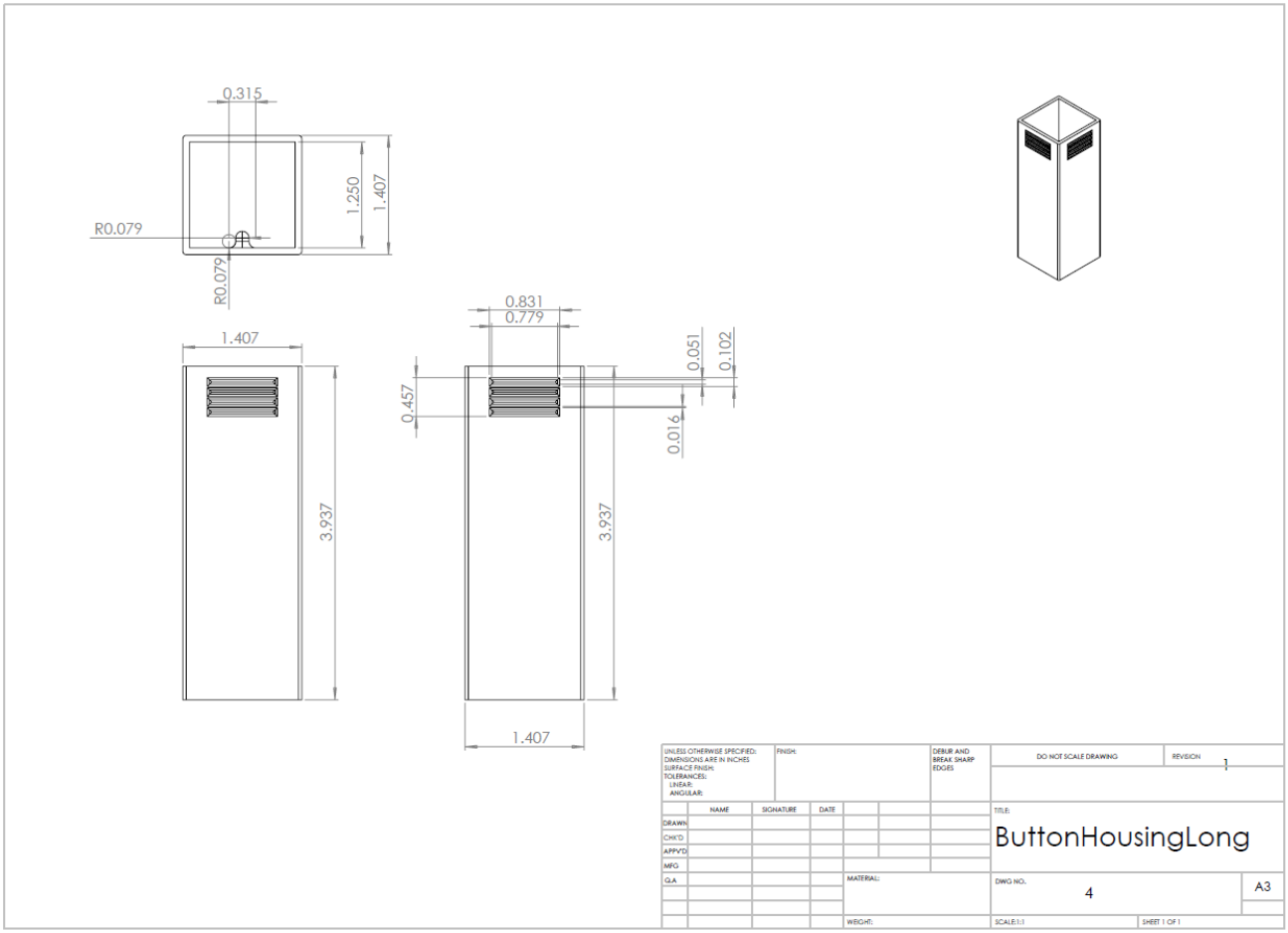


Figure 18 - Technical drawing of the pixel encasing.

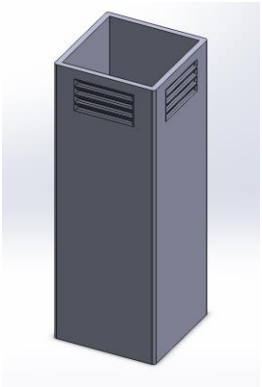


Figure 19 - Three dimensional model of the pixel encasing

This model was then sent to an architectural firm, ThinkForm, in Waterloo to be printed by a Makerbot Z18 printer. The final result can be seen in section 2.2.6.

The frame design is also described in section 2.2.7. There was concern that the frame would be slightly too small, however due to the nature of 3D printing, only one attempt could be made. As such, the design presented below was final, and sent to VanDerZwaag printing in Waterloo.

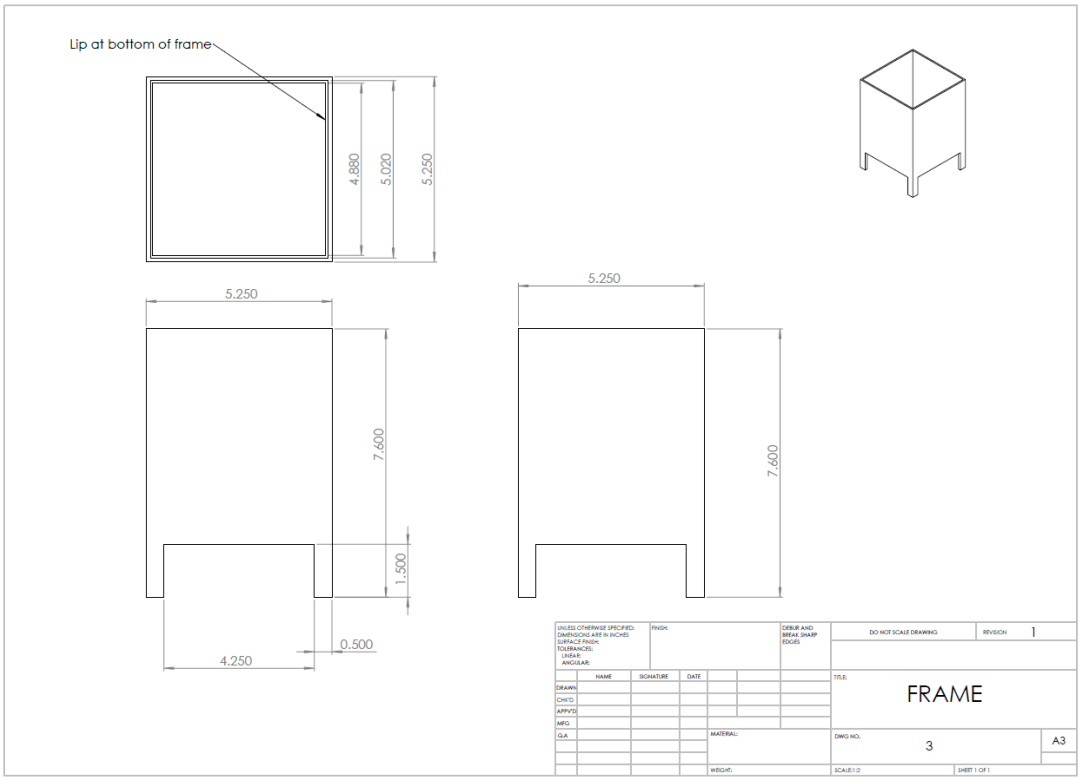


Figure 20 - Technical drawing of the frame.

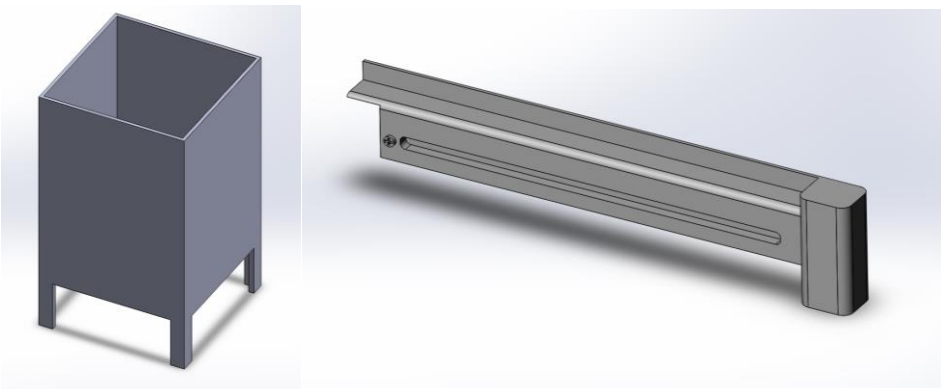


Figure 21- Three dimensional model of the frame and slider

The following model of the slider was created to more easily test the designs in SolidWorks to verify sizing and dimensions. This model was used to confirm the sizing of the pixel caps, the frame, and the base. Measurements from the real slider were taken with a micrometer and put into SolidWorks. The following model was produced.

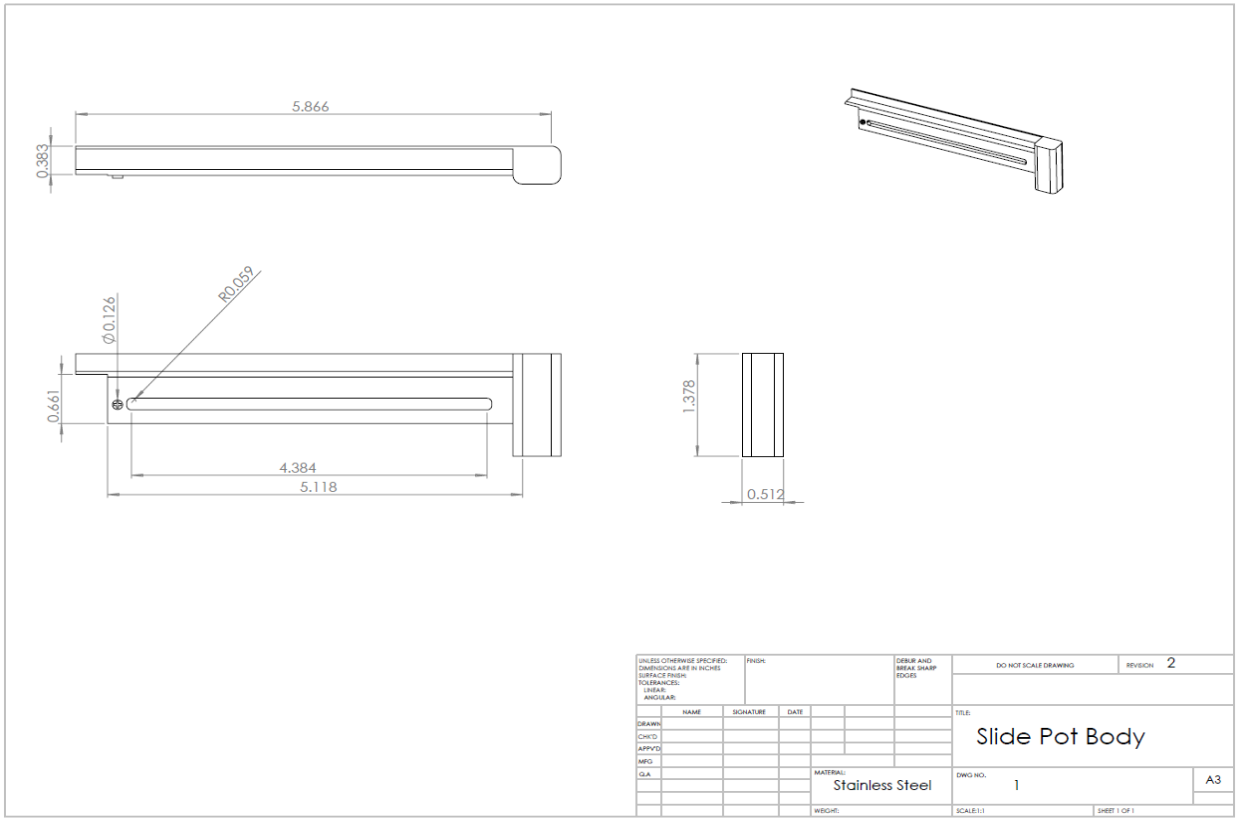


Figure 22 - Technical drawing of the motorized slide potentiometer.

2.4.2 Electrical Schematic

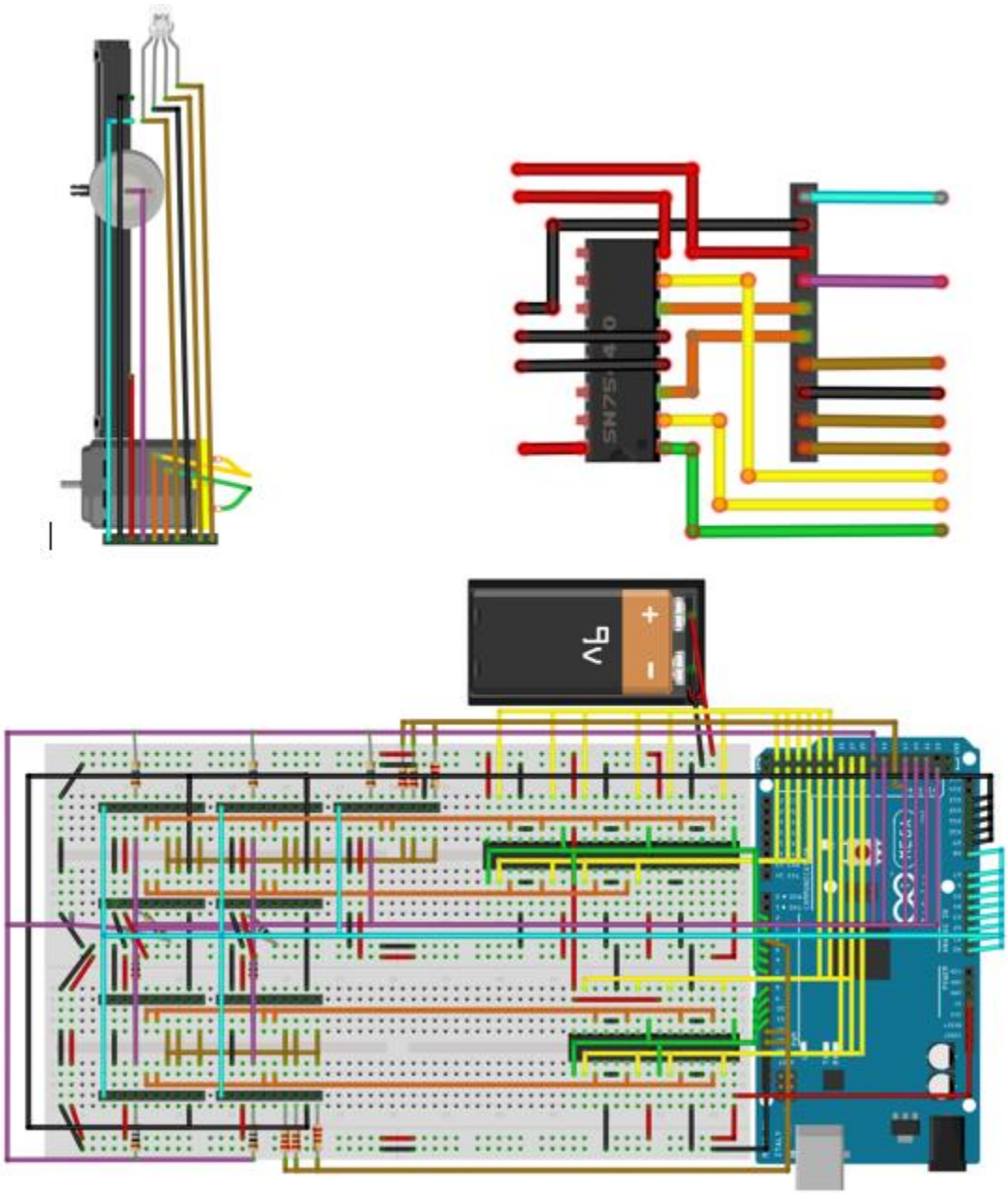


Figure 23 - Complete final wiring schematic

(Top Left) Singular Pixel, Motorized Potentiometer, Touch and LED wired into a 10-Pin Header.

(Top Right) Singular 10 pin header connecting to H-Bridge, meant to act as a wiring legend.

(Bottom) Full Breadboard and Arduino schematic with 9 pixel ports and 5 H-bridges

2.4.3 Bill of Materials

Table 8- Bill of materials for the pixel cap portion of Haptic

Bill of Materials					
Product: Haptic					Date: 2015-03-30
Assembly: Pixel Cap					
Item #	Part #	Quantity	Name	Material	Source
1	1	4x(3cmx6cm)	Cap Base	Balsa Wood	Micheal's (Arts & Crafts Store)
2	2	75 cm^2	Cap Surface	Aluminum Tape	Home Depot
3	3	3cm x 3cm	Cap Top	Diffusive Plastic	Staples
4	4	3cmx3cm	Capacitive Touch Electrode	Aluminum Tape	Home Depot
5	5	2cm	Capacitive Touch Electrode Wire	Copper	KW Surplus

Table 9 - Bill of Materials for the base and electronics of Haptic

Bill of Materials					
Product: Haptic					Date: 2015-03-30
Assembly: Base and Electronics					
Item #	Part #	Quantity	Name	Material	Source
1	1	1	Base	3D Printed	3D Print Shop Brantford
2	2	1	Frame	3D Printed	3D Print Shop Brantford
3	3	4	Frame Inserts	Balsa Wood	Micheal's (Arts & Crafts Store)
4	4	1	ATMega 256 Chip	Integrated Circuit Chip	Sparkfun
5	5	5	H Bridges	Integrated Circuit Chip	Sparkfun
6	6	6	220 ohm Resistors		Sparkfun
7	7	9	10M Resistors		Sparkfun
8	8	27	Flyback Diodes		Sparkfun
9	9	1	Printed Circuit Board	Silicon Wafer	PCB Fab Shop
10	10	9	Common Annode Diffusive RGB		Sparkfun
11	11	9	Motorized Linear Potentiometer		Sparkfun
12	12	9	Caps		See Cap BOM
13	13	90	Assorted Wires, Headers, Solder		KW Surplus

2.4.4 API Specification

The following functions are available in the Javascript API which allow developers to utilize the haptic hardware in their own interface designs. The API needs to serve two main purposes, one is to abstract serial communication effectively behind the scenes, and the other is to make it easy for the developer to invoke any function of the hardware with as little calls as possible. The API make use of Javascript callback functionality to indicate when position, force or touch input values are updated. As such, when the microcontroller does not notice a change in position, force or touch, the callback will simply not be evoked saving processor time.

Table 10 - List of functions available to developers in the web application and their purpose

Function Name	**Purpose**
updateColor(ID, R, G, B)	Allows developers to change the colour of a certain pixel by referencing its ID and providing desired RGB colour values.
updateDesiredPos(ID, position)	Allows developers to set the desired position of a certain pixel, which the PID controller will attempt to match. Used by referencing the pixel ID and position (between 0 and 1000).
onUpdate()	Triggered when the device sends updated information to the computer. Developers can use this to update their interface to reflect changes in the state of the physical device.
updatePIDPreset(ID, preset)	Enables the ability for developers to set the desired PID coefficient values from a list of predefined presets. This functionality is used in the “Game” UI demo to show how the stiffness of a button can be adjusted to provide haptic feedback in different situations.

3 Design Progression

3.1 Evolution of Design Solution

Through the course of the project, the design evolved significantly in order to better satisfy the design goals while still confining the work to the scope of a 3B design project. What follows is a summary of the progression the design went through as it evolved throughout the process.

3.1.1 Pre Slider Prototypes

Shape Memory Alloy were placed underneath a cloth screen that would push up on the cloth to simulate the rise of a button or user interface block. A novel and untested approach was to use Shape Memory Alloys as the actuators, specifically, Nitinol wire. Nitinol is a nickel titanium alloy that can change shapes based on its temperature. At temperatures below 90C the wire is malleable, once the temperature rises above 90 degrees celsius the wire reverts back into its "memory" shape. To set the "memory" shape the wire must be heated to 300 degrees celsius in the shape that is to be set. Temperature of the wire was raised by running current through it, as the wire has quite a high internal resistance. Such a technique was also to be used in operation.

The wire's memory was set to a 90 degree elbow so that when current is applied it will take this form. The wire is placed under the sheet at the desired location and then actuated when needed. Unfortunately this method was proved unseable due to the fact that the temperature the Nitinol wire heated up to caused browning and burning of the cloth with limited actual actuation.

Solenoid were also tested to push up on the screen. The solenoid was of the type "active pull" meaning that when the solenoid is on, it pulls the rod into itself. When turned off, a spring pushes the rod back out. This is placed under the cloth and the actuation is controlled. Unfortunately this method also proved to be too hot for use in a consumer device as well as consuming too much power per pixel.

A Cloth Screen prototyping allowed fulfilling the goals of the first prototype: to test the benefit of haptics and tacticity in rich displays and to validate the need for the idea. The interfaces were printed onto a cloth, which was then upholstered onto a wooden frame. The linear actuation devices were placed behind the cloth in order to simulate haptic feedback. Although the cloth provided a rich display, it did not provide user interaction, nor was it programmable. This approach worked well for the first prototype, but required iteration for the second one.

Figure 24 - Cloth screens upholstered on wooden frames for prototype 1.

Even with very limited actuation from the solenoid and shape memory wire, tests using the cloth screen strongly indicated that users find and press buttons faster and more accurately when the buttons were raised. This was true even in interfaces that made it hard to visually decipher the vertical position. This was especially true in tests where the user's visual cognition was occupied by tasks like driving in a simulation, or blindfolded.

3.1.2 One Slider Prototype

The goal of the second prototype was to demonstrate the functionality of a single pixel. The printed cloth display was traded in for a diffused RGB (tri-colour) LED. The idea was that when number of pixels is extended to form a matrix, the colour of each pixel can be individually controlled, as it is in capacitive touch screens.

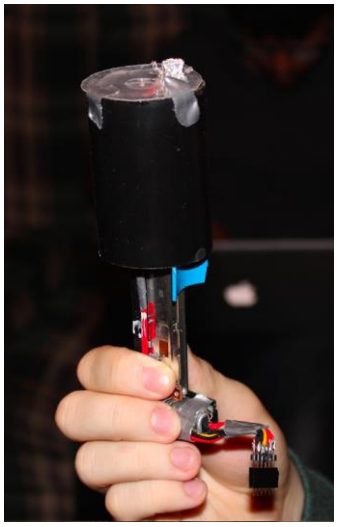


Figure 25 - Functional pixel for prototype 2.

The one slider prototype consisted of the cut ABS tubing described above in section 2.2.6. A small blue piece of plastic was epoxied to the side of the slider underneath the pixel cap top to prevent the slider from pushing through the top when it was moving downwards. Unfortunately, with this design, the cap of the pixel still slid downwards when unpowered. This meant that for the pixel to remain in an ‘up’ state, the slider had to remain powered.

In the original design plan, user input was gathered using an external “Leap Motion” device, which is capable of determining the current state of a user’s hand such as the position of each finger. The intent was for this information to enable complex user interactions commonly seen on multi-touch touch screens, such as twisting and scaling movements using two fingers.

Figure 26 - Leap Motion device receiving user hand positions.[12]

However, preliminary testing with the device showed that it did not provide sufficient accuracy and speed to satisfy the user input needs of the project. Finger locations were often misreported, making it very difficult to decipher user interactions.

Additionally, it was decided that given the large scope of the project, it would be beneficial to scale back the amount of user interactions the design was capable of handling, limiting it to just basic touch. Twisting and scaling motions were reserved for future work, outside the scope of the project.

Once the Leap Motion was abandoned, the user input method was replaced with a simple capacitive touch system. Multiple options to achieve this functionality, such as pre-made transparent capacitive touch layers and custom circuits using conductive material were considered. After the analysis described in section 2, a custom circuit was chosen for the final prototype. This consisted of aluminum tape to create a conductive surface, which was connected to a touch input receiver on the linear actuator component.

In order for the Arduino to receive and interpret the capacitive touch signals, software needed to be written which could deal with the type of electrical information received from the circuit. The original code and circuit sent a constant 5V signal through a 10 MOhm resistor to the aluminum tape (sensor) which acted as a small capacitor. The sensor was then connected to an analog input pin on the Arduino. As a person's hand came close to the aluminum tape the capacitance of the sensor increased which changed the voltage level going into the input pin. The input was buffered to smooth out spikes and then analyzed. If the values changed by a set threshold then a user touch was interpreted and the boolean value changed.

3.1.3 Four Slider Prototype

In the second physical prototype, four sliders were connected and operating with Arduino control. The base and frame had not yet been completed so each slider was mounted on directly onto the breadboard using the headers, supported by loose wooden blocks.

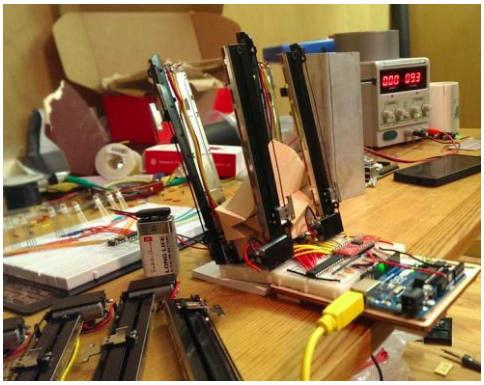


Figure 27 - Four slider prototype circuit with uncapped sliders.

The full wiring diagram using and Arduino Uno for this solution is presented below. This prototype also tested the viability of the TLC5940 PWM driver, but found that it does not work with common cathode LEDs. It performed well with the PID controls, still leaving it as an option for further expansion. The prototype also tested the viability of an analog multiplexer for input, which was found to work flawlessly, but not used when the Uno was replaced by the Mega.

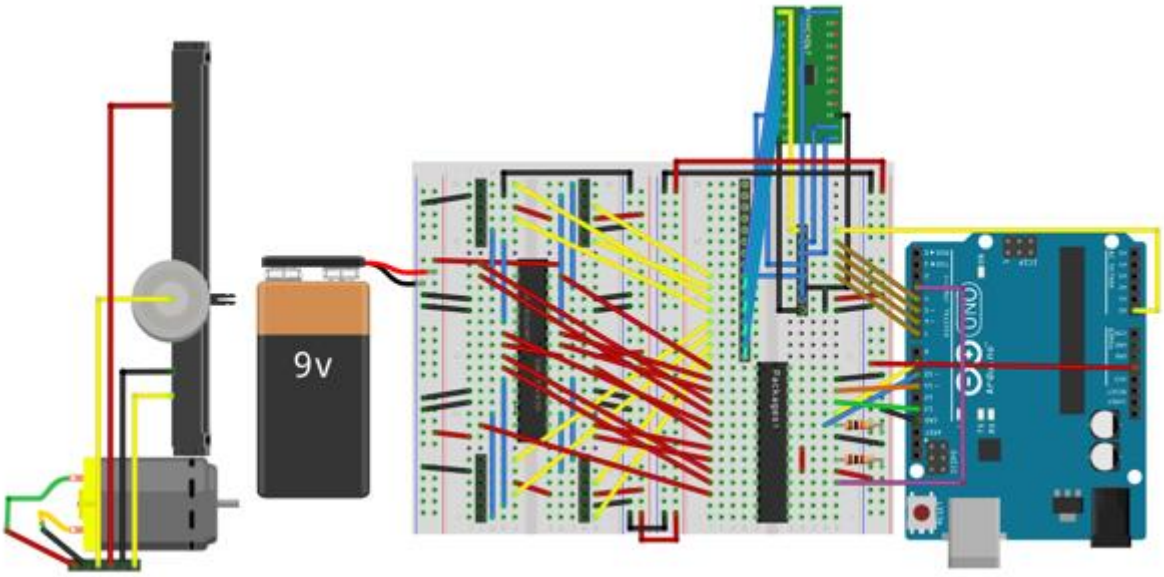


Figure 28 - Complete wiring diagram of four-slider prototype circuit

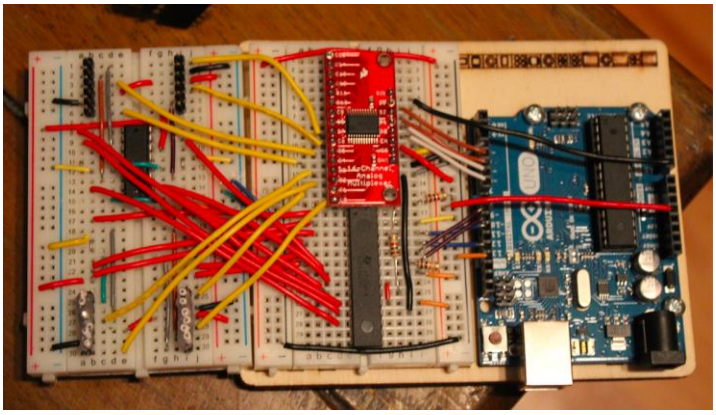


Figure 29 - Four-slider prototype circuit, including Arduino Uno and multiplexing circuitry

A third party capacitive touch sensing library (CapSense) for the Arduino is used for the four slider prototype. Here the circuit consists of a digital pin on the Arduino sending an output signal through a 10MOhm resistor to the aluminum tape (sensor), which is connected to an input. The output pin oscillates between high and low (5V, 0V) and the time for the input pin to reach the same value is recorded. High values of this time indicate a higher capacitance, indicating a touch. Unfortunately time based measurement significantly slowed the microcontroller operation.

3.1.4 Nine Slider Prototype

The final prototype consisted of nine distinct sliders, each with caps and working RGB LEDs. The Arduino Uno was abandoned in favor of an Arduino Mega with no additional multiplexing. By this time, the base had been completed and the pixels were installed and connected to the circuit. These aspects will now be discussed in further detail.

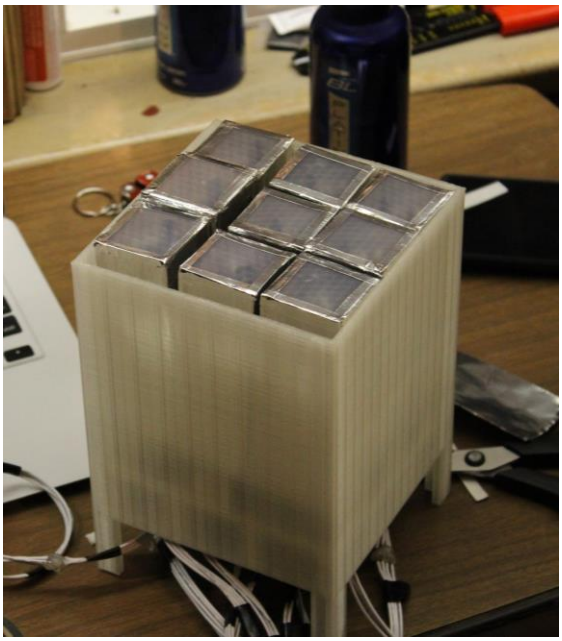


Figure 30 - Nine-pixel prototype with plastic casing installed.

With the introduction of the Arduino Mega and the removal of multiplexing circuitry, the circuit became more straight-forward. The only external integrated circuits required in addition to the Arduino were now simply the H-Bridges to drive the motors. The TLC PWM driver and analog multiplexer were no longer required. A complete diagram of the wiring is included in section 2.4

Section 2 covers the progression of the encasings and the frames, however, this is to reiterate the design decisions and small pitfalls encountered along the way. Before the 3D printed caps or balsa wood caps were completed, many hours were put into creating aluminium caps. These caps were cut out of aluminum sheets of metal and then bent into place using a metal brake. Given that the dimensions of the cap were rather small, bending and cutting these caps with a full sized brake was rather difficult and time consuming. Nonetheless, nine caps were bent into shape and epoxied shut, as seen below.

Figure 31 - Completed aluminum cap with epoxy sealing the edge.

Once the epoxy set, the holes were drilled out of the aluminum caps to reduce weight. Unfortunately this caused the epoxy to come off, and it needed to be reapplied after the holes were drilled. All nine caps were completed and tested. Unfortunately, the aluminum caps were deemed too heavy and then other materials were considered.

Figure 32 - Some completed aluminum caps, with holes cut.

As previously discussed, both balsa wood and 3D printed caps were tested in conjunction. Due to the extremely lightness of the balsa wood, coupled with the wrapping of aluminum tape, balsa

wood was chosen. Fortunately, the 3D printing of the caps was a free service, and as such no money was lost in this decision, nor was too much time.

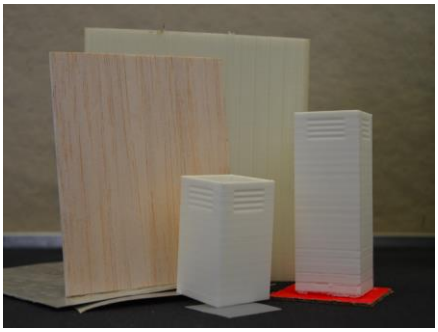


Figure 33 - Display of materials used for caps.

Once the actual completed caps were placed on the sliders, a few things needed to be changed. First, the LED position was originally on the inside of the caps, taped to the side. This was so that regardless of the position of the slider, the LED would remain in the same spot on the slider, thus providing consistent brightness. However, due to the movement of the LED along with the cap, the wires scraped along the base during movement causing a lot of friction. It was then decided that the LED would be placed on the slider and the brightness levels were tested at different slider positions. It was found that the difference in brightness from a fully raised slider vs a fully lowered slider with these new LED positions was negligible, and so the LEDs were moved for each slider.

The evolution of the base and frame followed similar routes as the caps. An aluminum frame was made, but it was deemed not aesthetically pleasing. Being made of aluminum, it also needed to be covered for the pixel caps to not touch the sides and cause incorrect touch detection. Once the 3D printed base and frame were completed, the sliders were placed inside of the base, and the base was slid into the frame. This resulted in an incredibly aesthetically pleasing design with the addition of the balsa wood inserts to prevent the sliders from touching.

Figure 34 - Completed frames with pixels in: 3D printed frame. Backup Aluminum frame

Unfortunately, due to friction being too high, the outer 3D printed frame had to be scrapped and the the aluminum frame on top of a cardboard box was used instead, as described in section 2.2.7. The final design was not nearly as aesthetically pleasing, however it did work and that was what was most important at the time.

This aluminum frame design became the final design that was presented. It was non ideal, but in the 3D printed frame the parts simply couldn't slide up and down, which was necessary. As such it was finally decided to scrap the 3D printed frame. However, the 3D printed base is still in use in the final design.

The CapSense library mentioned previously takes about 30ms to acquire a value. During this time the Arduino can not do anything and is stuck waiting for CapSense to finish. For four sliders this delay was not noticeable; however when using nine sliders the delay caused by CapSense was noticeable enough to cause problems with LED flicker and PID control. To combat this, the group wrote a different implementation from scratch which avoided the timing delay issue. This involved using counters in the Arduino's main execution loop rather than have a sub-loop run for the buffering of the CapSense library. Due to the caps of the pixels being wrapped in aluminum, the capacitive touch part of the slider was wired to the outer frame of the pixel, and this was used to detect touch on any point of the pixel casing.

For the final prototype, the use of a projector and camera system was intended to be used. However, since the value of the design solution was tactile and haptic feedback and not display, it was decided that the mathematical computation required to use a projector and camera system

for the display was outside the scope of this project. As a result, an RGB LED was installed in each pixel. If the development of this design solution is continued in fourth year, the use of a projector and camera system will be reconsidered.

3.2 Notable Technical Challenges and Resolutions

Given the complexity of the design project, many unexpected challenges were encountered during development. Each of these is detailed below.

3.2.1 Friction in the Case

A very significant problem was caused by the unexpected amount of friction in the encasing. When the nine sliders were installed and tested with the PID control system, it was found that the case caused too much drag for the motors to effectively move the sliders up and down. The problem was mitigated by removing the sliders and placing them in a temporary enclosure that allowed more range of motion. However, this fix was not permanent and future work on the project would include redesigning the encasing to reduce friction. The 3D printed base and outer frame designs were well done, however they simply needed to be slightly larger. Due to time and money constraints, these could not be 3D printed in time and as such the aluminum frame was used in the final design.

3.2.2 LED Placement Inside Encasing

A similar but distinct problem was that the wires connecting the LEDs inside each slider cap to the Arduino were moved up and down as the sliders moved, causing their own friction which hindered the movement. This problem was solved by moving the LED attachment point from the inside of the slider cap to the top of the stationary actuator base. In doing so, the LED wires no longer needed to move up and down when the sliders did, and the friction was therefore eliminated.

3.2.3 LED Flicker Patterns

It was also found that all the tasks which needed to be executed by the Arduino cumulatively took too long. This was evidenced by symptoms such as flickering LEDs, caused by the fact that their colours are updated every execution “loop”. When the loop took longer than expected, the LEDs were updated less frequently than required to eliminate the appearance of flickering. In order to resolve this problem, the various tasks in each loop had to be prioritized, so that less

important operations occurred on every few loop iterations rather than all of them. This allowed more important tasks like updating LEDs and calculating PID actions to occur more frequently.

3.2.4 Serial Buffer Overload

Another problem which occurred consisted of the Arduino serial interface failing to read or write new commands when too many messages were being sent back and forth. This problem was relatively easy to solve, as it was found that the Arduino's internal serial buffer, which contains incoming messages before they are read and dealt with by the code, was becoming full. The cause of this was simply that the Arduino code the group developed was not reading messages and clearing out this buffer fast enough. In order to solve this problem, a similar method to the LED flickering was used, whereby the number of commands that were read in each execution loop was increased, outpacing the rate at which the buffer was filled by incoming messages.

3.2.5 Fitting pixels into base

Difficulty was encountered during the process of inserting the pixels into the 3D printed base component. It was discovered that this task was quite difficult due to the design of the base and consumed large amounts of time. Additionally, with the pixels fully installed it was difficult to access the wires connecting them to the Arduino, which presented further challenge when electrical issues with the wiring connections necessitated removing the pixel from the base so the problems could be fixed. These problems were mitigated temporarily by not using the plastic base and casing together, which also eliminated the friction casing issue described above. Similar to that case, this is not a permanent solution and future work on the project would require a redesign of the base and casing system to allow easier installation, removal and maintenance.

3.2.6 Current Surges

Another problem was discovered when all nine sliders were connected to the Arduino for the first time. The combined current usage of the motors occasionally causes an electrical surge to occur through the Arduino and the computer's USB port. The major symptom of this problem is that the computer will automatically disable its USB port in order to protect itself from the excess current, and in occasional extreme cases, it will shut-off altogether. The specific cause of this problem is currently unknown, and as such the technical challenge has not yet been resolved. This is an issue which would need to be resolved for any future work on the project. One way to fix this is to introduce flyback diodes on all output pins on the arduino.

3.2.7 Isolating Touch Contacts

The touch contacts on each pixel consisted of the aluminum taping covering the exterior of each cap. In order for capacitive touch to function correctly, the tape on each cap needed to be isolated so that two caps did not come in contact and create an electrical connection. This proved to be unexpectedly difficult, as it was found that the balsa wood inserts created to separate the sliders were too thick to fit comfortably between the pixels, causing excess friction between them. The only thinner material available was metal sheeting, which would have caused electrical conductance to occur. In order to isolate the pixels while still eliminating the friction caused by the inserts, the metal was covered in a layer of transparent tape. This process did not appreciably increase the thickness, but allowed the inserts to become non-conductive and prevented interference with the operation of capacitive touch.

3.2.8 Outer Touch Connectivity

Once the capacitive touch contacts were attached to the outer aluminum casings of the pixel caps, it was found that the aluminum tape was not conductive on the underside. This meant that the aluminum did not create a completed circuit all the way around the caps. To mitigate this, first, conductive thread was taped around the outside of the caps. This worked, however, WD40 was sprayed on the pixel caps to decrease friction. The WD40 caused the tape holding the thread to lose adhesiveness and it slid off. Instead, conductive paint was applied at all corners of contact of the aluminum tape, as seen below. This completed the circuit around the caps and the touch sensing was working after this.

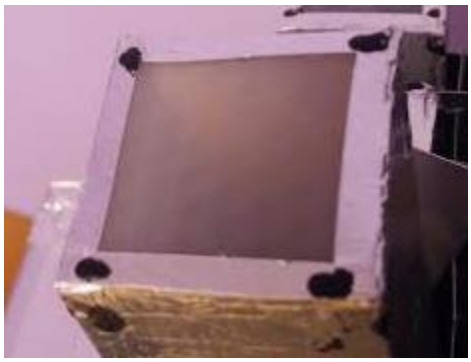


Figure 35 - Capacitive paint added to aluminum casing to allow for a completed touch circuit.

4 Design Testing and Validation

4.1 Benchmark Testing Protocols

Once the final prototype was constructed, testing took place in order to determine how effectively each design requirement was satisfied. Many of these tests were simply measurements that needed to be taken, such as the available range of motion for requirement A1. Each distance measurement was taken using a standard ruler, while timing measurements were conducted using internal timing and testing scripts for the arduino where possible and a human-operated stopwatch when not possible.

User testing was conducted with the device setup for two distinct interfaces, “flush” and “raised”, which are shown as 3D representations in the below figures.

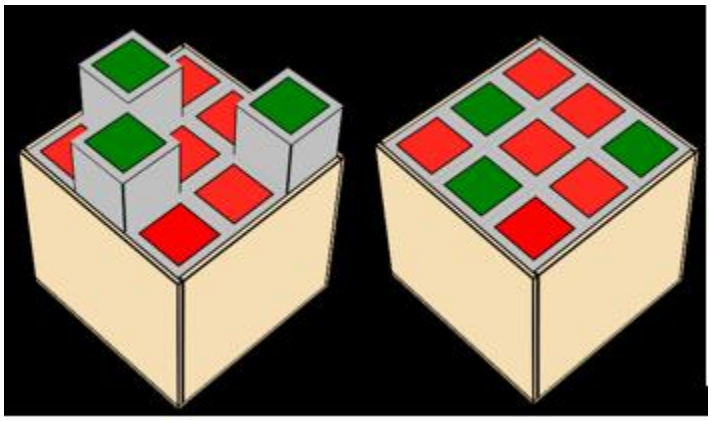


Figure 36 - 3D representation of interface used for “raised” and “flush” interface tests.

Each testing procedure described in section 1.5 was conducted using the raised interface, and compared to the results observed with a “standard” touch screen interface that was simulated using the flush surface interface with no raised controls. As seen from the figure, several of the pixel regions are raised. These sections are defined as the targets the user must hit, and in the touch screen case the targets are in the same place with the modification that the areas are now flush with the surface. In the speed test, users had to touch all the indicated controls on both the flush and non-flush surface in succession, as quickly as possible. These results were measured using a computer script that received inputs from the device and measured their times, and compared to each other.

For the accuracy test, the user was asked to touch one of the selected control regions as quickly as possible and hold their finger at the spot that was touched. A group member then measured the distance from their finger to the center of the target region. Again, the measured results were compared for the case where the region was raised versus when it was flush.

The data requirements described in section 1.5 govern whether or not the flow of information through the system is correct. These requirements were tested throughout the development of the project, as they were necessary for basic functionality. As such, the integrity and accuracy of the information had no formal testing procedure after the prototype was complete. In the course of project development, the information was made to be correct through continuous iteration and improvement. The quantitative measurement for this category of requirements was simply the time it took for information flow to occur through the system, which was measured programmatically within the code responsible for this functionality.

4.2 Results

The results gathered from the testing procedure explained in section 4.1 are concisely shown in the table below. Please note that some requirements may span more than one benchmark.

Table 11 - Summary of results from benchmark testing

Requirement	**Benchmarks**	**Goal**	**Results**
Pixel Actuation			
A1	Distance from lowest to highest achievable point	>2cm	15cm
A3	Response time of touch input	<20ms	15ms avg 30ms max
A4	Max force provided from actuator	>1N	1.275N (2.256N Peak)
A4	PID Settling time	<40ms	35ms
A6	PID Overshoot	<2mm	1.5mm (max)

A6	PID steady state error	<1mm	1.5 mm
A5	Actuation full stroke speed	<500ms	300ms
Display			
D3	LED color range	24bit Colour	24bit Colour
D3	LED flicker	None	Minor
D3	LED brightness	2000 mcd	2533 mcd (avg.)
D4	Frame	8 hours	24 hours
D5	Cap weight is low enough to be actuated	Achieve requirement A5	True
D5	Cap defection	2cm	2.5cm
D1	Processing time delay	Real time	Real time
Data			
C1	Accuracy of determining and reporting over serial touch versus press	>90%	80%
C2	Time for pixel actuation to reach target set by gui	<500ms	400ms
C2	Accuracy for pixel actuation to reach target set by gui	2mm	1.5mm
C2	Time to report back current position	50ms	20ms
C3	Report user force	True	True

C3	Time to send force data	<50ms	20ms
C4	Time to respond to force data	<50ms	20ms
User and GUI			
B1	Percent time saved for user to detect control vs using touch screens	100%	120%
B2	Percent increase in interaction speed vs using touch screens	15%	50%
B3	Percent increase in accuracy of interactions vs using touch screens	20%	25%

4.3 Discussion

As seen from the above results, most benchmark requirements that were listed have been fully satisfied except for PID steady state error. This indicates that the solution was highly successful, and further PID tuning would completely satisfy every requirement. In many cases, the final design significantly outperformed the suggested requirements. The maximum range from highest to lowest point was over five times larger than required, which enabled much more dynamic interfaces with more variation in height. Additionally, the other PID control requirements were outperformed entirely, resulting in a much more responsive and fluid performance.

All user performance goals were consistently exceeded, validating the design goals of improving the quality and accuracy of user interacts. It was shown that physical haptic interfaces significantly decrease the time it took for users to find and activate controls, and reduce the cognitive load on the user attempting to operate the interface.

The display goals were also satisfied, which simply meant that required information could be adequately displayed to users. During the final presentation, users were able to clearly see what was being displayed, and how to interact with it.

All data requirements were also satisfied, which ensured that the performance of the overall system did not detract from the experience. Information was able to flow through the system at a high rate, preventing noticeable delays between user interaction and system response.

Overall, the system performed exceptionally well, meeting nearly all of the design requirements. Therefore, the current design has been fully validated as satisfying the situation of concern.

5 Recommended Design Modifications

After building and testing prototype three, the team has gathered recommendations that can be made to the design to both further improve the project and decrease faults/shortcomings. The recommendations are divided into categories and listed below.

5.1 Linear motion of pixels

5.1.1 Low friction Enasing

Create a encasing with lower friction between the pixel walls and case so that the linear motion is consistent for every pixel.

5.1.2 Rigid connection

Make the connection from the pixel wall (interior) to the actuator considerably more rigid and aligned to ensure each pixels motion is parallel to its neighbors.

5.1.3 Extended pixel support

Extend the support for each so that they are not just secured at the base. This should ensure that each pixel’s motion is parallel to its neighbors and there will be no collisions.

5.1.4 Use more powerful motors

Replace the motors in the linear actuator with ones which have higher torque and power ratings while keeping within the same physical dimensions of the original motors.

5.2 Display and input

5.2.1Increase pixel density

Increase the pixel density so that images and large text can be presented on the entire display.

5.2.1 Use prebuilt capacitive sensors

Use prebuilt capacitive pads so that each pixel can accept “swipe” and “twist” gestures.

5.3 GUI Interface

5.3.1 Optimize user interaction of pixel grid

Optimize the way the user selects and modifies the pixel’s position on the GUI in the diagnostics page.

5.4 General

5.4.1 Decrease height

Decrease height of the entire unit so that it can be integrated into smaller places.

5.4.2 Custom build serial cables

Create serial (bound/packaged) cables so that the connections are more robust and so that the wiring is easier to debug.

5.4.3 Custom PCB

Print a custom PCB to minimize use of wires and make the electronics more reliable and make a smaller package.

5.4.4 Isolate power circuit

Isolate the power circuit which controls the motors from the logic circuit and communication ports.

6 Technical Summary

The design statement created at the beginning of the project was to “create a programmable surface for makers to design, develop, and test haptic interfaces.”. This goal was achieved

through the completion of prototype three. In detail the objectives achieved were: Shifting a surface into a shape configured by a dynamic user interface; displaying an interface; responding to user interaction and providing dynamic force feedback. The prototype achieved these objectives by having individual pixels that linearly move vertically under power of motorized potentiometers; display 24 bit color each; have capacitive touch capability; and can provide resistive force to the user. The device was tested using the procedures outlined in section 1.5, and was found to perform admirably, satisfying all design goals and proving its success as a solution to the situation of concern.

Part B - Project Review

7 Overall Satisfaction with Final Design

Overall, the team is satisfied with its final design. The team made significant progress in addressing the design statement, and went through significant design iterations and engineering analysis in order to arrive at the final solution. Unfortunately, the team is dissatisfied with that gaps that exist between the final design and what the envisioned final design was meant to be.

The next few sections cover the things that were done particularly well:

7.1 Functionality

The team was able to implement a collection of impressive functions including continuous and responsive colour display, capacitive touch, and sensing and actuation of vertical position and force. Furthermore, all functionality was encased in a self-contained form factor.

7.2 Accountability

A team champion was assigned to each major component. This proved extremely useful throughout the design and development process because all functionality was accounted for.

The next few sections cover the things that were not done particularly well:

7.3 Aesthetics

The pixels were not being held vertically upright as neatly as the team would have liked. The 3D printed base was able to arrange the pixels nicely, but unfortunately offered too much friction and thus had to be abandoned. The aluminum frame that replaced it was not ideal, and should be improved in future iterations.

7.4 Consistency, Reliability and Testing

Although each pixel could perform many functions, not all 9 pixels could perform all functions. Unfortunately, certain functionality would fail for certain pixels and it with difficult to determine the cause. More extensive testing throughout the development process would have been valuable.

To further improve the final design, the following steps could have been taken during the design implementation process:

7.5 Preparation and Foresight

Preparing for the various prototype demonstrations and the final symposium, the team realized the importance of anticipating unexpected challenges and being prepared to combat them. The day of the dry run for the symposium, the wiring of the circuit came loose during transport and time had to be spent frantically debugging it. Luckily, the loose connection was found in time and the live demonstration could be shown. However, the team learned that there is much value in taking a video of a working prototype, so that if the prototype unexpectedly stops working, the video can be shown to demonstrate the proof of concept.

7.6 Bottom-Up Approach

A bottom-up approach involves "progressing from small or subordinate units to larger or more important units, as in an organization or process." [13] The value of such an approach became apparent to the team when the team struggled to move from a state where all components were working independently to one where all components were working together. In order to alleviate this hardship, the team should have transitioned from a small working prototype to a slightly more complex working prototype and so on. For instance, the could could have first focused on getting a 2x2 matrix working well, and only then extending it to a 3x3.

7.7 Incremental Testing

Perhaps the biggest lesson the team learned was the value of incremental testing. Throughout the development process, the team worked on developing a component for all 9 pixels, and then testing it. Instead, the team should have tested each component for one single pixel, and if successful, implemented it for all pixels. For example, 9 aluminum encasings were prepared, which took significant time and material. Once all encasings were made, they were tested, and the team realized that aluminum is not a suitable material for the encasing as it is too heavy. The team could have saved significant resources by simply testing incrementally.

8 Initial Research and Design Planning

In general, the team predicted most of the challenges with the software portions of the project quite accurately. This can be explained by the software development backgrounds of many of the group members, making this the strongest area of collective group knowledge. All software goals were reached long before the target deadlines, and all issues were adequately resolved. Most technical challenges were already anticipated as they arrived, and delays caused by unexpected ones did not critically endanger the development schedule as software progress was always far ahead of the other project components.

Unfortunately, the same cannot be said for the other components of the project, namely the electrical and mechanical portions. The group collectively lacked experience in both these areas, and as a result the complexity and time required to complete tasks in this space were often underestimated. Technical challenges were much less well-anticipated, and unexpected difficulties caused significant delays in progress. Many components were not completed until very late in the design implementation process, and did not have time to be properly integrated and tested with the rest of the system. Furthermore, some components that were not modifiable after completion were completed far too late such as 3D printed parts. Once a part was printed, more money and time had to be spent printing a new one instead of modifying the current one.

In the future, the group should be more aware of areas where members lack expertise. All tasks that fall under these categories should be given extra time in the implementation schedule, as the group has a tendency to underestimate the length and complexity of developing these components. Additionally, the target deadline for components should be well ahead of the final symposium in order to provide ample time for integration into the main system and thorough testing. Also, in the future the rough deadline should be treated as the final deadline as much as possible. In the rough deadline, only a portion of the project was submitted. However, this deadline should have been a completed rough version of the project, so the time between then and the final submission could be spent fixing small problems and perfecting the design, rather than building the entire design in that time.

9 Technical Skills and Resources

In the Team Contract phase of the project, the team recognized that team members have unique skills but that the team's software skills heavily outweigh the team's hardware and electrical skills. The team identified that where skills are lacking, it would seek the help of experts in the area such as professors and graduate students. Specifically, the team identified the team's supervisor, Orion Bruckman as a valuable resource in the electrical and hardware domain.

The team accurately identified the strengths and weaknesses of its members and the need for outside guidance. In fact, the team discussed many technical challenges with Orion, who was able to provide valuable insight. He advised the team to use motorized potentiometers early on in the design process. Unfortunately, the team was determined to first test shape memory alloys, and embraced motorized potentiometers as the best approach after realizing shape memory alloys do not work well. Although this proved a great learning experience, in the future, the team should factor in expert advice early on in order to save time.

Although the team identified professors and graduate students as valuable resources, it did not seek their assistance. In future design projects, the team should make an effort to approach experts in order to design a more advanced and elegant solution.

10 Team Communication and Time Management

10.1 Flexible Schedules

The flexible schedules of all team members greatly helped the team's time management. Instead of conducting weekly meetings, meeting were held when they were needed. Some weeks, meetings occurred 4-5 times in the week, while other weeks only had 1-2 meetings. Whenever a meeting needed to be called, group members were flexible and willing to adjust their schedules to accommodate the meeting.

10.2 Workspace

The living arrangements of the team greatly benefited the team. Four of five teammates live together and so the living room of their house was dedicated as a workspace for the design project. As a result, the team did not have to work around the schedule of the Engineering 5

machine shop, or transport the materials to and from a workspace. This freed up teammates time in order to be focused on actual design and engineering work. The team's living arrangements also allowed the team to have impromptu meetings. Unfortunately, the one team member that did not live in the same house missed out on these meetings or had to spend time commuting. Also, given that the hardware components were built and completed in a space in the house containing workshop hardware, one team member also missed out on valuable team communication while building hardware components.

10.3 Communication Methods

Facebook chat was used as a prime method of communication. Fortunately, all team members check their messages often and so were always up to date with the latest developments.

10.4 Team Manager

The team elected to have one group member serve as a team manager, which gave them the responsibility of keeping track of deadlines, managing team communications and organizing meetings. This proved to be extremely beneficial, as it aided the group in staying on track and ensuring all major checkpoints were reached in a timely fashion.

10.5 Estimation

Unfortunately, the team consistently underestimated the amount of time and work required to develop a certain component. In future design projects, in order to better manage time, the team should make realistic estimations and heavily pad them to give leeway for unexpected challenges that arise.

11 Project Summary

The term has been an intensive one filled with various design iterations and extensive engineering analysis. Below is a compilation of the lessons the team has learned throughout this journey that would be beneficial for future design projects. These lessons learned are in addition to the improvements suggested above to initial research and design planning, technical skill considerations, resource allocation, team communication and time management.

11.1 Realistic Expectations

The team was extremely ambitious throughout the entire design project. The team was adamant on designing a surface that does not lose any functionality already present in a capacitive surface and extending functionality with haptic feedback. Instead, the team should have focused on perfecting haptic feedback, and adding in capacitive touch and display only if time permitted. As a result, many hours were spent on the prototype, and especially the final solution.

11.2 Transparency and Relatability

The team was passionate about the design space, but relaying the value of the design solution was difficult. People agreed that touch screens lack haptic feedback but had trouble visualizing the application of the prototype to real world situations and getting added value. The UI demos attempted to demonstrate these applications, but it was found that this was often not enough for people to get a clear picture of the possibilities of the device.

11.3 Engineering Analysis

With the guidance of the supervisor and professors, the team was able to apply a much higher level of engineering analysis than previous years in Systems Design Engineering. Every component added to the project was fully analysed, and compared to all possible alternatives. In the end, the skills gained from this required higher level of engineering analysis were extremely useful and powerful, and will be carried forward into our careers as engineers.

12 References

[1] MarkeWatch, "New Honeywell Technologies Including Touchscreens In Cockpit Help Gulfstream's Newest Aircraft Take Flight," October 2014 [Online]. Available:

http://www.marketwatch.com/story/new-honeywell-technologies-including-touchscreens-in-cockpit-help-gulfstreams-newest-aircraft-take-flight-2014-10-14. [Accessed April 1].

[2] Microsoft, "Beyond Tapping and Sliding", August 5 [Online]. Available:

http://research.microsoft.com/en-us/news/features/haptics-080514.aspx. [Accessed April 1].

[3] Tactus, "Tactus," April 2015 [Online]. Available:

http://tactustechnology.com/. [Accessed April 1].

[4] LED Shoppe, "4 Legs Tri-Color 5mm Red & Green & Blue LED X 20pcs," April 2015 [Online]. Available:

http://ledshoppe.com/index.php?route=product/product&product_id=87 [Accessed April 1].

[5] TestStandard, "ABS Material Data Sheet," April 2015 [Online]. Available:

http://teststandard.com/data_sheets/ABS_Data_sheet.pdf [Accessed April 1].

[6] "Resistance Taper List," April 2015 [Online]. Available:

http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Components/General/TAPER.pdf [Accessed April 1].

[7] Arduino, "Arduino Board Uno". Availabile:

http://arduino.cc/en/Main/arduinoBoardUno [Accessed April 7, 2015].

[8] Raspbery Pi, "Raspberry Pi 2 Model B", November 2014 [Online]. Available:
http://www.raspberrypi.org/products/raspberry-pi-2-model-b/ [Accessed April 1 2015]

[9] Arduino, "Arduino Board Mega 2560". Availabile:
http://arduino.cc/en/Main/arduinoBoardMega2560
[Accessed April 7, 2015].

[10] John Hopkins University, “Principles of cross=-modal competition: Evidence from deficits of attention,” October 2003 [Online]. Available:

http://web.jhu.edu/cogsci/templates/images/rapp/RappHendel2003.pdf [Accessed April 1]

[11] Adas Geek, "13 Principles of Display Design", October 2009 [Online] Avaible: https://adasgeek.wordpress.com/2009/10/12/the-13-principles-of-display-design/, [Acceessed April 1, 2015]

[12] Leap Motion, “Leap Motion Hands” September 2015, [Online] Available: http://www.robotshop.com/media/files/images2/leap-motion-3d-motion-gesture-controller-10-large.jpg, [Accessed April 1]

[13] "Bottom-up." The Free Dictionary. Farlex, n.d. Web. 06 Apr. 2015. http://www.thefreedictionary.com/bottom-up. [Accessed April 1]