Heads Up Display for MIDI Keyboard

ECE 445 Fall 2018

Design Review

Andy Woodruff

Alexa Hirsch

Table of Contents

Introduction

Objective

Learning the piano can be a difficult thing because in order to do it yourself you are required to know something about how the notes you are playing sound. Without this knowledge, and without someone there to correct finger placement, it can be hard to determine conveniently if you are playing the right chord. The internet has many resources that make any individual capable of teaching themselves the piano or looking up chord structures. Websites are available that allow you to put in the different notes you are playing, or even click the keys you are pressing on a piano keyboard graphic, and will return the chord that it makes [6, 7, 8]. However, this is inconvenient to work with as several steps are required in the process and can provide a roadblock to a beginner who gets frustrated.

Our proposed solution is to bring these resources that are available on a separate technology directly to the instrument. Heads Up Display technology is being used to bring access to information away from separate technologies and into the way we interact with the physical world. Using the information from the MIDI (Musical Instrument Digital Interface) keyboard, we will be able to determine, as the user is playing the keyboard, each chord that they play and then use the HUD technology to place the chord being played onto a music stand so the user can see the data on their preferred place on their sheet music while they are playing.

Background

There are many MIDI interface software products currently on the market. These include simple recording programs, pitch bending, sustain pedals, and various sound effects. Additionally, there are web services designed to translate key presses, guitar fret positions, or note names, into chord names. These web services, along with many forum posts looking for ways to determine chord names while playing them, are right at the top of any web search for “how to tell what chord I’m playing.” Until now, these products have been largely separate. That is, MIDI interface devices and largely limited to purely storing or manipulating MIDI data, rather than analyzing it, and music analysis software, even when given a MIDI interface, tends to assume the user is displaying the information on a computer [10]. Our product is thus unique in offering a hardware tool to view the chords played on a MIDI instrument in real time.

High-Level Requirements

- The projector must accurately display the chord name sprites with enough clarity that the average user misreads fewer than one in fifty displayed chords.
- The total time the system takes to read MIDI input, determine the chord being played, and pipe that information to the display should be less than the average visual reaction time, which is approximately 250ms.
- The user must be able to record an arbitrary number and length of MIDI files in real time, only limited to the file size and SD card space, simply by pressing a single button to begin and end recording.

Design

Block Diagram

This MIDI interpretation and data display device can be separated into: a control unit, a projection module, and a display module. The control unit will be responsible for interfacing the the MIDI input from the keyboard via the USB cable with the data interpretation, user input, and data storage with the SD card. The projection module will be the first step of data display of the control unit's chord name output. This consists of the pixel LCD which will show the chord names as a mirrored image, and the fresnel lens which will be placed as to bring this image to the desired dimensions. The display module, a combination of an adjustable light valve and the reflective glass, is the Heads Up Display itself which will be placed on the user's music stand over sheet music. The reflective glass will be placed in front of the light valve with respect to the light coming from the projection module, and the light valve will be immediately behind it so as to adjust the brightness of the display without causing an additional reflected image.

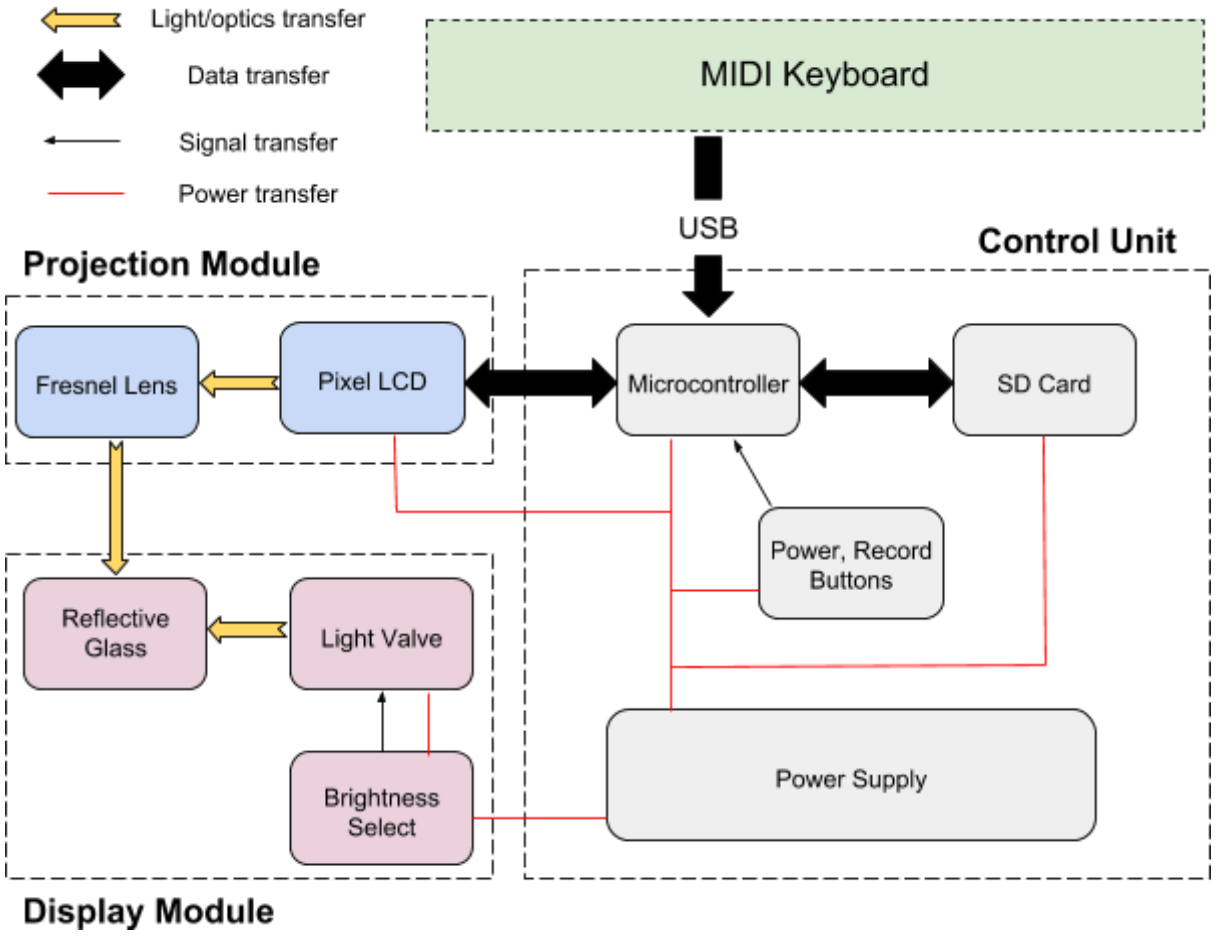

Figure 1. Block diagram

Physical Design

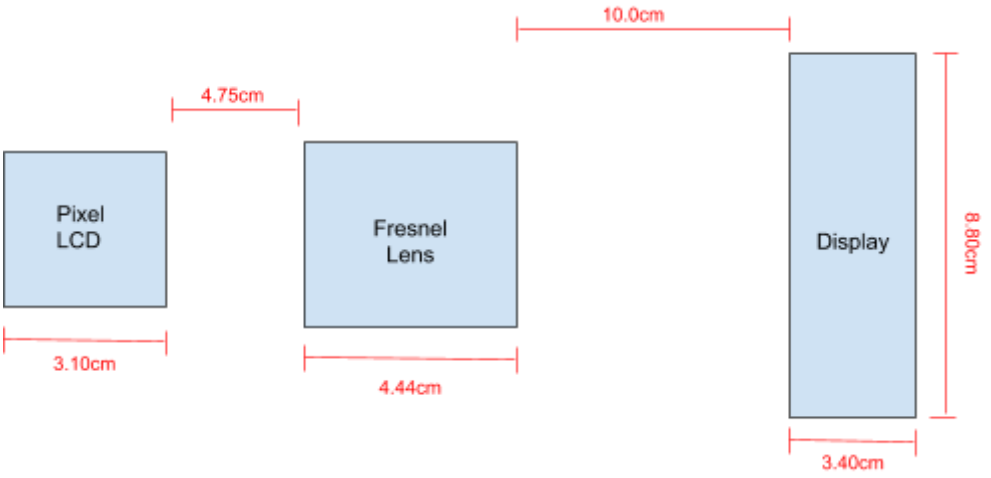

Figure 2. Spacing measurements of optical components within enclosure

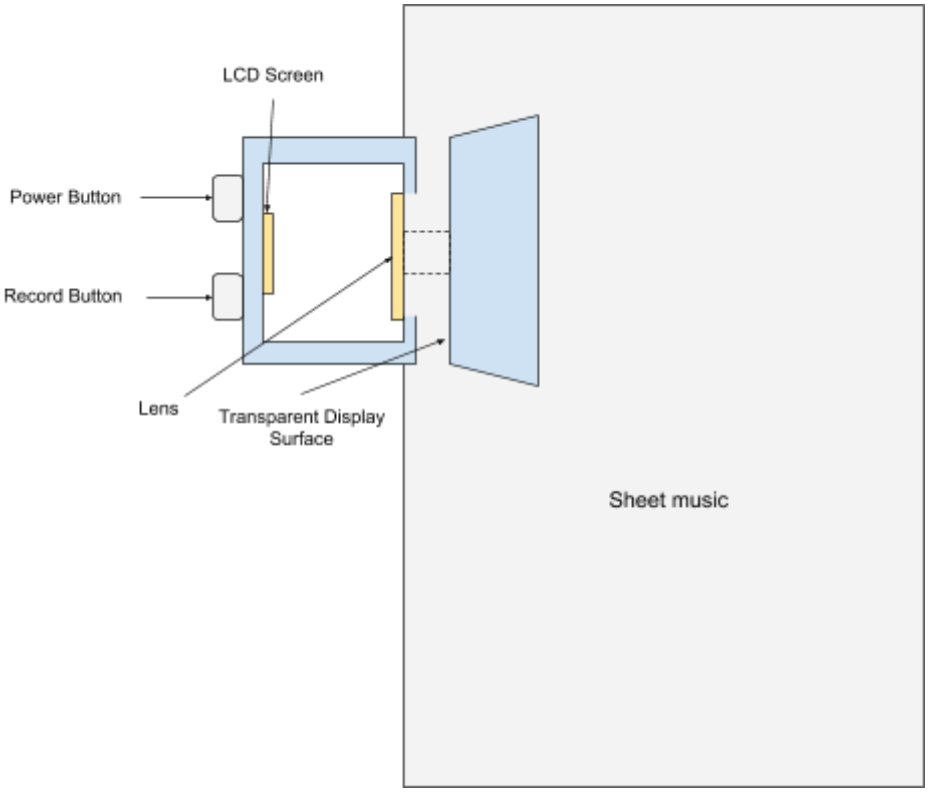

Figure 3. Physical design showing optics components and enclosure

Functional Overview

Control unit

Microcontroller

We have chosen to use an ATMEGA32U4 microcontroller. This will receive the communication via the USB from the keyboard, send data to the SD card, receive inputs from the Record and Power buttons, send voltage input data to the battery status LED, and send information to the pixel display in the Projection Module.

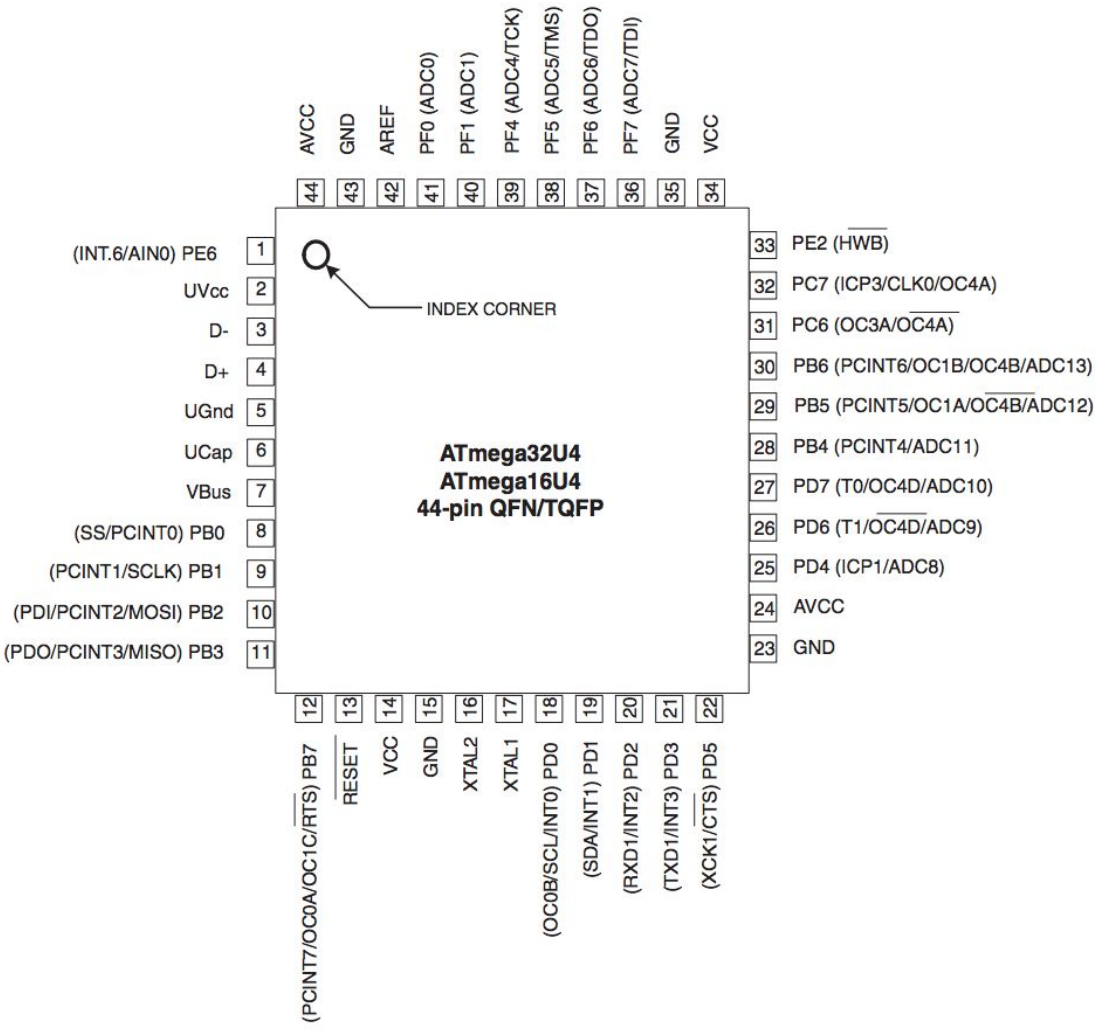

Figure 4. Pin schematic for ATmega32U4 microcontroller [1]
See Appendix A for pin allocations

Requirements	Verifications
The microcontroller must be able to: *1) Read MIDI over USB in real time* *2) Determine chord names from note values in under 200ms. (This time requirement, combined with the maximum display time of 50ms referenced below, totals to a 250ms delay between the user playing a chord and the user seeing the chord name. This is within a standard deviation of the average human visual reaction time as cited by multiple sources, and thus should appear as nearly instantaneous to the user.)* *3) Send a display command to the LCD screen in under 50ms.* *4) Write MIDI packet data to the SD card at 3125 bytes per second or faster. (This is the speed at which MIDI data is sent, according to the standard implementation. If it wrote slower, our recording time would be limited by the microcontroller's RAM, which we find unacceptable.)* *5) Write multiple, differently named MIDI files to the SD card sequentially.*	1. Attach a display to the microcontroller, have it display the raw MIDI data as soon as it's received. Use a camera to record a notes being played and displayed for one minute. If the delay between the audio and the note display is within 10ms of the display lag, and is the same at the beginning and end of the test period, this qualifies as "real time." 2. Within the software, get a global time as soon as the MIDI packet is received and compare it with the global time as soon as the chord determination is complete. Average over 50 trials. 3. Within the software, get a global time as soon as the chord calculation is complete and compare it with the global time as soon as the LCD frame buffer write is complete. Average over 50 trials. 4. While this is almost guaranteed by hardware and protocol specs, we can again compare a global time retrieved at the time of MIDI packet retrieval, to a global time retrieved after writing said packet to the SD card. Dividing the packet size by this time gives us the data rate. Average over 10 minutes. 5. Record one minute of play, stop the recording, wait 30 seconds, and repeat this process. Repeat 10 times, then wait 10 minutes, and repeat. Success if all 20 files are correct and distinct.

SD Card Reader

An SD card reader will be placed in the main body of the device for saving MIDI data while in record mode. It will not be needed to write data to the device, only read. For our device we

will be using an SD micro card 8-pin push & push memory card connector with a PCB mounting style.

We will be interfacing with the SD card in SPI mode. SPI is a simple synchronous protocol which only has a chip select as bus protocol and no maximum bus rate. Communication speeds achieved by interfacing “real-time” MIDI data with an SD card reader can be up to 10mbps. This is much higher than the speed required for the “real-time” data processing that we have defined in the microcontroller requirements in the figure below.

Requirement	Verification
The SD card must be capable of storing MIDI data as fast as it is sent through the USB connection from the keyboard. This equates to writing 3125 bytes per second in continuous operation.	1. Tests 4 and 5 from the microcontroller should adequately prove this.

Power and Record Push Buttons

These buttons will be placed on the main body of the device for enabling recording mode. When this button is “on”, MIDI data from the keyboard will be stored in the SD device. In “off” this function will be turned off.

Requirements	Verifications
1) *The Power button should have a failure rate below 1%* 2) *The Record button should have a failure rate below .5% (A record failure wastes more of the users time than a single failure to turn off/on, and thus is a more negative user experience.)*	1. Power the device on, allow the screen to begin displaying, and power the device off. Repeat this 500 times. Success if < 5 failures to turn on / off. 2. Program the record button to increment a counter, rather than begin recording. Press the button 1000 times. Success if count is 996 to 1000 inclusive. (This is a test of the hardware sending a digital signal correctly.)

Power Supply

Component	Specifications
Microcontroller	Operating voltage: 2.7-5.5V
Pixel LCD Screen	Operating voltage: 3-5V
Light Valve	Operating voltage: 0-5V
SD Card	Operating voltage: 3.3V
Lithium Ion Polymer Rechargeable Battery	Output voltage: 3.7V 2500 mAh
3.3V Voltage Regulator (LD1117V33)	Maximum DC Input: 15V

Figure 5. Device voltage requirements

Each of the components that will draw power from the supply can operate at either 3.3V or 5V, except for any interaction with the SD card as SD cards are strictly 3.3V devices. However, the light valve requires a supply of 5V for its full range of opacity control, as it adjusts its brightness linearly in proportion to a 0-5V range. For the purpose of this device it is not necessary to attain complete opacity of the display and we wish to regulate the power consumption created by the microcontroller, LCD and light valve, and require SD card functionality. Prioritizing this, we have decided to output 3.3V from the power supply to the rest of the device and regulate this output with a 3.3V voltage regulator to stabilize battery output for the microcontroller. The voltage regulator will connect the battery (Vin) to the microcontroller and light valve (Vout) as shown in Figure 8.

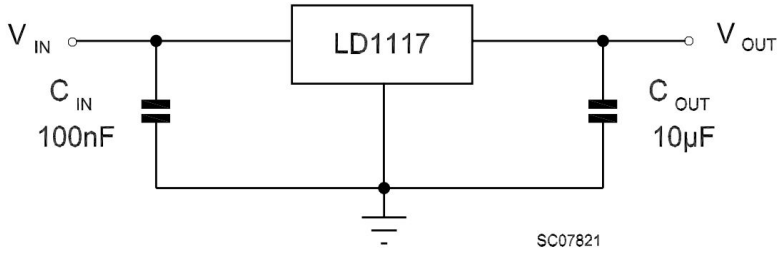

Figure 6. 3.3V voltage regulator integration circuit [3]

Requirements	Verifications
1) *The power supply must output steady 3.3V to the circuit while receiving the variable voltage from the rechargeable battery.* 2) *The power supply must be capable of running without recharging the batteries for at least an hour with consistent play and record mode on.*	1. Measure voltage output of battery and voltage regulator simultaneously. Observe the voltage values output from the voltage regulator as the input from the battery is 3.7, 3.6V, 3.5V, 3.4V. Verified if regulator output is 3.3V for all input values. 2. Measure the current pull at the output of voltage regulator while record mode is on and notes are being played (i.e. chord names are being calculated) and compare against the mAh of the battery to establish a theoretical battery life for the device. This is verified if the theoretical life is 2 hours or more. If not, we will physically test by playing consistently with record on for 1 hour while observing power supply voltage output.

Projection Module

Pixel LCD

We have chosen to use the Adafruit 1.54" 240x240 Wide Angle TFT LCD Display for the pixel LCD display. This will receive output from the microcontroller and will display a mirrored image of chord names that have been determined by the device's MIDI interpreter. As calculated in Fisher's article, we end up with a 41 PPI display after our magnification step. This will be visibly pixelated, but not difficult to read.

Requirement	Verification
The LCD display must display mirrored chord name sprites.	1. Simple visual comparison between

The chord name images on this display must be readable at 41 pixels / inch, easily distinguished from 3 feet away.	sprite stored on LCD's SD card and image displayed on screen. 2. Again, visual inspection, preferably by multiple users. Must be able to accurately read off 50 chords per user with one or fewer misreads per user.

Fresnel Lens

A fresnel lens from a magnifying glass will be used to project the small image of the pixel screen into a larger display on the reflective glass a small distance away. This will be placed between the pixel LCD and the reflective glass.

The focal length (f) of the fresnel lens used in this device is 10.1cm.

Resolution Calculations:

Screen to lens (s)	Min: 2.32 cm Max: 4.86 cm
Screen to display	Min: 6.93 cm Max: 10.27 cm
Lens to display	Min: 4.61 cm Max: 5.41 cm

Figure 10. Optics components testing distances

Enclosure prototype construction:

We intend to develop the simple enclosure for our optics testing prototype with dimensions that will accomodate distances closer to the maximum measurements made above:

- Screen to lens ~4.75 cm
- Screen to display ~10 cm
- Lens to display ~ 10 cm - 4.75 cm = 5.25 cm

This enclosure prototype will be made of cardboard and easily alterable in order to achieve a screen reflection that meets our basic criteria for resolution and the size of image on display. The primary importance of the enclosure is to cut off light reaching the display from the room and sources other than the LCD screen so it only shows the image from the LCD screen. This prototype enclosure will be clipped onto the side of the music stand.

The production enclosure will be designed based on the successes and failures of the prototype. In general, it should have very similar dimensions, and will most likely be largely 3d printed. It should allow for easy access to the user interface buttons and dial, support the

display surface via a transparent arm, and clip onto the user's music stand from left side. It should not slip during operation, and should be as small as possible while housing the microcontroller, battery, and projection unit. In addition, this housing should allow the user to remove the battery for charging, and to easily access the SD card.

The focal length of a fresnel lens is related to the screen-lens distance and magnification factor by:

$$f = \frac{s}{(1-1/m)}$$

Where f is the focal length, s_1 is the object-lens distance, and m is the magnification determined by the ratio:

$$m = \frac{desired\ object\ size}{actual\ object\ size}$$

Based on component testing for desired image size and quality for the display, we determined the values above. Using these values we can determine the exact magnification factor that will be achieved for our initial prototype:

$$m = \frac{1}{1-s/f} = \frac{1}{1-(4.75/10.1)} = 1.89$$

Since we know that the height of our pixel LCD display is 3.1 cm, the measured height of the chord names being displayed on the HUD will be:

$$virtual\ object\ height = (actual\ object\ height)(m) = (3.1)(1.89) = 5.85cm$$

Requirement	Verification
Fresnel Lens *Any distortion from the lens must be low enough to be compensated for in chord name sprite design. By rough estimate, this means no pixels from the test pattern should be displaced by more than 10 pixels on the display image, and the mean pixel displacement should be 3 or lower. This would correspond to a maximum distortion of 20% and a mean of 6% (assuming 50 x 50 pixel sprites).*	**Fresnel Lens** Display the reference image described in Tolerance Analysis. Photograph the resulting image on the display screen. Overlay the two images and count pixel displacements.

Display Module

Reflective Glass

The reflective glass that will act as the display for the user will be a dielectric beam splitter mirror. This is used in teleprompters, and has a reflective side with tint-free mirror coating and a backside with anti-reflective coating to prevent a double image. The image of the live-updating chord name projected from the projection module will be shown here.

Requirement	Verification
The reflective glass must be large enough that an easily readable size of lettering can be displayed from the projector module.	Have the glass cut to 6 cm by 3.5 cm

Light Valve

We have chosen to use a controllable shutter glass liquid crystal light valve to control the transparency of the display. This will be attached to the beam splitter mirror so that the mirror is in between it and the projection from the fresnel lens. The brightness of the valve will be connected to the “Brightness Select” user input potentiometer and has an input range of 0-5v (in which 5v input will cause complete opacity and 0v will cause complete transparency).

Requirement	Verification
The light valve cannot receive an input above 5 volts.	Covered by brightness select test.

Brightness Select

The Brightness Select is a potentiometer that will be adjusted by the user in order to control the transparency of the heads up display and change the display quality. This will be connected directly to the light valve and will not go through the microcontroller.

Requirement	Verification
The Brightness Select must be limited to controlling an input to the light valve between 0 and 3.3v.	Measure the voltage across the light valve while sweeping the full range of the single-turn potentiometer. At fixed furthest left rotation the voltage should measure 0v and at the fixed furthest right rotation the voltage should measure 3.3v.

Software Mapping

MIDI:

First the microcontroller receives the MIDI packets from the keyboard. It then reads them and keeps track of which notes are being played. It also treats notes that were only released 100 ms ago as being still active, to avoid erroneous calculations on chord release. Then it compares the currently playing notes with a chord reference table. The notes being played are treated as one chord, with a closeness score for each chord in the table. It scores points for each note that matches with a candidate chord based on how fundamental to the candidate chord the notes are. Conversely, mismatched notes lose points according to how strongly they contradict the candidate chord. The candidate chord with the highest closeness score wins, with ties going to the least complex candidate chord. Complexity will be determined when the reference table is built, based on music theory.

Storage:

Upon boot, the system prepares a new file name for the next MIDI file it will store. Once the record button is pressed, it initializes a file with that name, and writes the MIDI file header. Once a packet that contains at least one note is sent, the system writes that and each subsequent packet to the file as they are received. Once the record button is pressed a second time, the system sends a flush command which ensures all of the data has moved from any buffers into proper file storage. It then closes the file and prepares a new file name for the next MIDI file it will store.

Tolerance Analysis

Image tolerance

For most video projection systems, the two main calibrations are focus and convergence. Focus is self evident, and convergence is the overlap of the RGB components of the image. We will define a “good” convergence to mean that the RGB components overlap perfectly, and a “poor” convergence to mean that there exist visibly distinct overlapping segments.

For our setup, focus is fixed at time of assembly, and rather than having separate RGB channels to converge, we have pre-integrated RGB pixels that might diverge due to the Fresnel lens. Additionally, said lens may introduce distortion. To test and correct for these, we will project a test pattern onto our display, and compare it with a reference image (Figure 10). This test pattern will identify color divergence by having square regions of solid red, green, blue, cyan, magenta, yellow, and white, on a black background. While we will be

generating our own test pattern to specifically suit the needs and resolution of our display, it should look something like Figure 11 below.

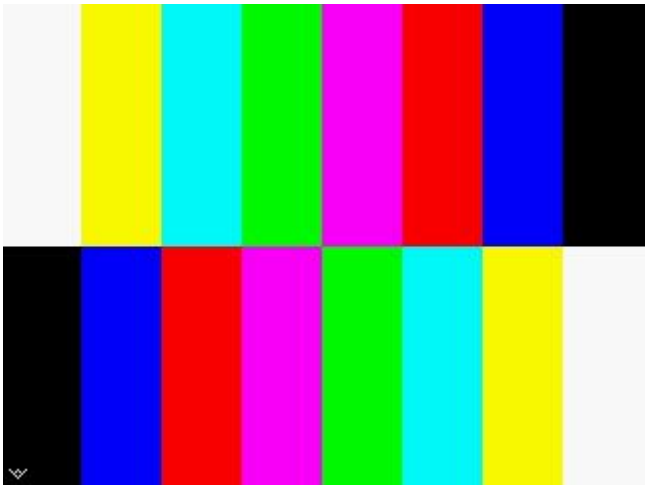

Figure 7. Sample reference image

If two colors are refracted at different relative angles, their respective joint color will see “fringing”, as will the white segment. If the aberration is within 6% (measured by apparent pixel displacement), no action is required. If it is greater than 6% but only between two colors, we can simply avoid pairing those colors. If it is greater than 6% and between each color, we can run in strict RGB, and the user won’t have any issues.

Distortion will be measured by our test pattern being a grid. We can then compare the projected grid to the source grid and directly measure the displacement of each square from its intended location. If this distortion is variable, it means we need to make the lens more rigid, which can be solved with the use of a rigid transparent backing. If it is not variable, and is instead due to lens defects, we can programatically account for it (provided it’s within listed thresholds) by projecting an anti-distorted image. When distorted, this image will come out in our originally intended shape.

As for focus, the same test pattern again can be used due to its sharp corners and clear contrast between colored blocks. Here we simply look for blurring at edges, and adjust the relative positions of the LCD screen, lens, and display surface until it disappears. All in all, the user should be able to clearly read lettering of 50 pixels tall at a distance of 3 feet.

Cost

Light Valve(s)	$15.00
Reflective Backing	$15.99

LCD Display	$19.95
Fresnel Lens(es)	$5.99
Memory card connector	$2.52

Figure 8. Current costs table

If we assume those component costs will triple by the end of the project, that gives us a total of $180 worth of materials. Based on projected work loads, this will take around 20 hours per partner per week, and we have already put in approximately 50 combined hours. At an estimated $25 per hour, this comes out to $9,250 in labor. Adding the suggested 2.5x fudge factor, this all ends up at a total of $23,600. This somewhat explains the massive reported R&D budgets from most major firms.

Scheduling

Oct 7	Alexa: SD card storage software Andy: MIDI-in complete (can read individual MIDI note information and process)
Oct 14	Andy + Alexa: Complete rough draft enclosure + arduino prototype Andy + Alexa: Finalize all components for PCB Alexa: Enclosure design, completing SD card storage Andy: Chord-identifying algorithm
Oct 21	PCB Round 1 Alexa: Finish PCB layout for round 1 Andy: Enclosure CAD (Initial LCD reference image display development)
Oct 28	Alexa + Andy: PCB Testing (beginning integration of memory storage and MIDI processing software and hardware components)
Nov 4	PCB Round 2 Andy: Enclosure printing + optics assembly testing (reference image displaying on Heads Up Display screen) Alexa: PCB alterations/improvements

	design
Nov 11	Alexa + Andy: PCB 2 testing, software/hardware assembly off of prototype
Nov 18	Fall break: Slack week for additional issues
Nov 25	Mock demos (major subsystems constructed) Andy + Alexa: Testing/debugging

Figure 9. Scheduling table

Risk Analysis

Over-arching the many modular components of this device are two main sections: the electronic components and the optical components. While each of these function very independently, the device as a whole depends on the fact they will be linked together successfully. Therefore the risk of this project lies in one of these sides failing.

Firstly is the greatest risk in terms of probability: that the projection system fails in the key points that we have established in our individual component requirements. Though the optics consist of several components, a “successful” image (an image that meets our requirements in magnification and resolution) is only produced by each of these components having individual material quality and functioning successfully together.

The electronic side of this device is controlled almost completely via the microcontroller. If the main processing module is not able to display correct chord values with the expected speed then this project will not be successful in any capacity. We expect that this has a much lower probability of risk than the optical aspect of the project, but the degree of failure is much more severe here. A poor display can still function as a display, but the data on the display is either correct or incorrect.

We have only a few other examples of people using a setup similar to our own. This isn’t necessarily bad, but if we run into difficulties we won’t have many resources for troubleshooting this exact configuration.

Ethics and Safety

As for point one on the IEEE code of ethics regarding public welfare, MIDI is a freely available data protocol, and we were unable to find any other examples of our product on the market. It is technically possible for someone to plug the device into someone else's instrument and use it to record their performances without their knowledge or permission, but given the size and shape, that seems unfeasible.

On point nine, public safety, the biggest risk to the end user comes from the fact that we plan on using teleprompter glass and two LCD components (the screen and the light valve) for part of the display which could break and produce sharp edges. If testing shows this to be too great of a risk, we do have plans for an alternative display using a plastic reflective surface instead.

Under point six, our own safety risks are no greater than any other computing electronics projects; we aren't working with any high voltage equipment, so the most dangerous thing we might encounter is a soldering iron.

On point two, neither of us have any conflicts of interest as we're not currently employed by nor invested in anyone in the music production equipment marketplace.

We have done our best on point three, finding making honest estimates based on good and reliable data, and will do our best on point seven, accepting honest technical criticism. The other points seem to be either not applicable or require no special effort to maintain.

References

[1] Atmel, “8-bit AVR Microcontroller with 16/32K Bytes of ISP Flash and USB Controller,” ATmega32U4 datasheet, 2010. Available: https://www.pjrc.com/teensy/atmega32u4.pdf [Accessed Oct 2, 2018]

[2] A Davis, F Kuhnlenz, “Optical Design Using Fresnel Lenses”, Optik & Photonik, 2007. Available: https://www.orafol.com/tl_files/EnergyUSA/papers/Optical-Design-Using-Fresnel-Lenses.pdf. [Accessed Oct. 1, 2018]

[3] STMicroelectronics, “Low Drop Fixed and Adjustable Positive Voltage Regulators,” LD1117V33 datasheet, Dec 2005. Available: https://www.sparkfun.com/datasheets/Components/LD1117V33.pdf [Accessed Oct 4, 2018]

[4] Tim Fisher, “How Many Pixels in an Inch?”, Lifewire, August 23, 2018 Available: https://www.lifewire.com/how-many-pixels-in-an-inch-4125185 [Accessed Oct 4, 2018]

[5] Wurth Electronics, “Micro SD Card Connector- Push & Push- With Card Detection 8 pins,” Available: https://www.mouser.com/datasheet/2/445/693071010811-336082.pdf [Accessed Oct 3, 2018]

[6] ScalesChords, “Chord Identifier (Reverse Chord Finder),” [Online]. Available: https://www.scales-chords.com/chordid.php [Accessed Sept 19, 2018]

[7] 8notes.com, “Piano Chord Finder,” [Online]. Available: https://www.8notes.com/piano_chord_chart/chord_finder.asp [Accessed Sept 19, 2018]

[8] gootar.com, “Piano,” [Online]. Available: http://www.gootar.com/piano/index.php [Accessed Sept 19, 2018]

[9] IEEE Code of Ethics. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[10] KVR Audio, “Midi Chord Analyzer,” [Online]. Available: https://www.kvraudio.com/product/midichordanalyzer-by-insert-piz-here [Accessed Sept 19, 2018]

[11] “Midi and Music”, Available: https://www-user.tu-chemnitz.de/~heha/petzold/ch22d.htm [Accessed Oct 3, 2018]https://www-user.tu-chemnitz.de/~heha/petzold/ch22d.htmmidi file "3125" bytes

Appendix

Appendix A

ATmega32U4 pin allocations

Pin	Function	Connection
2	UVcc USB internal regulator input supply voltage	Power supply
3	D- Negative data rail (external 22 ohm resistor in serial)	USB port
4	D+ Positive data rail (external 22 ohm resistor in serial)	USB port
5	UGnd USB ground	Power supply (ground)
6	UCap USB internal regulator output supply voltage (external 1µF capacitor in serial)	Power supply (ground)
7	VBus USB data connection (external 10µF in serial)	USB port
8	PB0 TFT Reset	LCD display
9	PB1- SCLK Clock	SD card connector, LCD display
10	PB2- MOSI Master slave input	LCD display
11	PB3- MISO Master slave output	LCD display
14, 24, 34, 44	VCC/AVCC	Power supply

15, 23, 35, 43	GND	Power supply
18	PD0 DAT0 (connector data line)	SD card connector
19	PD1 DAT1 (connector data line)	SD card connector
20	PD2 DAT2 (connector data line)	SD card connector
21	PD3 CD/DAT3 (card detection)	SD card connector
22	PD5	Power button
25	PD4 CMD (command/answer)	SD card connector
26	PD6	Record button
27	PD7	Battery status LED
28	PB4 TFT SPI Chip Select	LCD display
29	PB5 TFT SPI Data / Command	LCD display
30	PB6 LCD SD Chip Select	LCD display

Appendix B

SD card connector pin descriptions

Pin	**Function**	**Description**
1	DAT2	Connector data line 2
2	CD/DAT3	Card detection
3	CMD	Command/Answer
4	Vdd	Voltage
5	CLK	Clock

6	Vss	Ground
7	DAT0	Connector data line 0
8	DAT1	Connector data line 1

Appendix C

LCD pixel screen pin descriptions

Pin Name	Function
BL	Backlight control. Active High, PWM capable
MISO	SPI Master In Slave Out pin (SD card only; TFT display is write-only)
SCLK	SPI clock input
SI	SPI Master Out Slave In pin (sends data from the microcontroller to the SD card and/or TFT)
TF CS	TFT SPI chip select
SD CS	SD card chip select
DC	TFT SPI data or command selector
RT	TFT reset pin. Active Low
Vin	3 - 5 Volts in (goes through regulator)
3Vo	3.3 V (can function as input or output)
GND	Ground