MIDI Controller Sequencer

ECE 445 Final Report -- Spring 2019
Team 2: Devin Alexander (dbalexa2),
Martin Lamping (mdl3),
Nathan Zychal (nzycha2)
TA: Christopher Horn
Date: 4/9/2019

Abstract

This report explains the process of creating a MIDI Controller Sequencer. The MIDI Controller Sequencer was designed for the senior design course (ECE 445) at the University of Illinois at Urbana Champaign. Throughout this document, the software, hardware, methodology for testing, safety precautions, and physical considerations for the project are all explained in detail. Design choices, conceptual / initial design ideas, and design influences for each portion (if any) are also mentioned. The MIDI Controller Sequencer was designed with the user in mind, and as such, an attempt was made at creating an easy to understand, intuitive interface. The sequencer will allow users to produce any type of music while also offering a new, unique, and modern way to do it.

Table of Contents

1 Introduction

1.1 Purpose

Modern sequencers are almost exclusively software based and most frequently employ the use of DAWs (Digital Audio Workstations) to output MIDI data. The software sidedness and in some cases non-intuitive control interface of current software sequencers is a major proponent that inspired the creation of this MIDI controller sequencer. The MIDI Controller Sequencer design itself is unique and has not been previously implemented in the same way. The motorized potentiometers, PCB design modularity, LED code structure and physical considerations all emphasize its' unique qualities. The purpose of this MIDI Controller Sequencer project is to create a dynamic physical interface that responds, alters, and reports musical parameters to the user(s) in real time, all while MIDI data is being output. This MIDI Controller Sequencer fills the gap modern software based sequencers have created. The sequencer does this by allowing parameters to be changed with dials, buttons, DC motors, and a plethora of other intuitive and easy to use interfaces.

1.2 Functionality

There are many high-level project functionalities that were taken into consideration while designing the sequencer. Firstly and most importantly, the output of the system must be a MIDI data as a sequence of notes at a consistent tempo. MIDI OUT data is generated and adjusted based on user input sequencer values such as the tempo, root note, scale, and pitch. Second, the potentiometer slider positions can be set manually and also can be moved into position by the motors. When the motor moves the slider, the final slider position is also determined by the sequencer status values. The position of the slider in combination with the sequencer status values will set the MIDI OUT data and in turn produce the desired output sound. Third, the current step in the sequence of notes will be automatically indicated by a dedicated LED placed below each motorized potentiometer slider. Fourth, the ability to turn off / on a step in the sequence is afforded to the user. Steps in the sequence can be turned off / on by pressing a dedicated button below the desired step. Lastly, the sequencer status information is output accurately and in timely manner to the LCD screen to be displayed to the user(s). Each of these high-level project functionalities play a separate and crucial role in the successful operation of the MIDI Control Sequencer and meeting the project's purpose.

1.3 Subsystem Overview

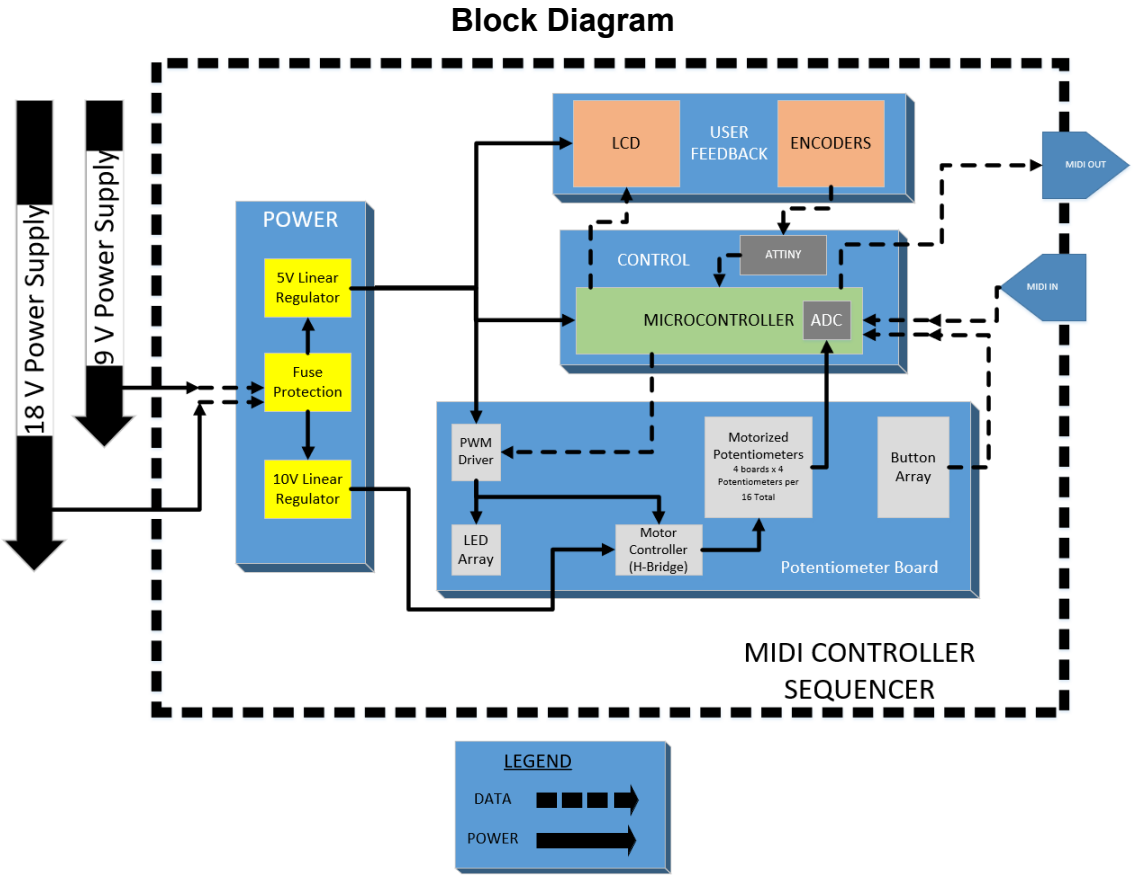

Figure 1: The above figure shows the block diagram. The block diagram consists of four modules: the power, control, potentiometer and user interface. Note the system has four ports: independent 18 V and 9 V power supplies as well as a set MIDI In and Out ports.

Subsystems:

Power Distribution Board - This board will be used to distribute appropriate amounts of power to all other boards in the MIDI Controller. Has inputs from both an external 18 V DC source and also an external 12 V DC source.

Table 1: Power Distribution Board Interconnects:		
Connection From:	**Connection To:**	**Purpose:**
Power Distribution Board	Potentiometer Board	Supplies *both* an 18 V DC and 12 V DC power source connection.
Power Distribution Board	Microcontroller / MIDI Board	Supplies *only* a 12 V DC power source connection.
Power Distribution Board	Auxiliary Board	Supplies *only* a 12 V DC power source connection.

Microcontroller / MIDI Board - This board contains the ATmega2560 which controls most of the project's functionality. The ATmega2560 reads analog and reads/writes digital values, services

interrupts, interacts with serialized MIDI channels, and provides communication to ICs on aux boards using the I2C protocol.This board also contains the MIDI IN, and MIDI out communication ports and accompanying circuitry.

Table 2: Microcontroller/MIDI Board Interconnects:		
Connection From:	**Connection To:**	**Purpose:**
Microcontroller / MIDI Board (ATmega2560)	Microcontroller / MIDI Board (ATtiny85)	Communicates with the ATtiny85 using the I2C communication protocol after the ATtiny85 requests to be read via hardware interrupt.
Potentiometer Board	Microcontroller / MIDI Board (ATmega2560)	Receives analog input values from each motorized potentiometer, this information will in turn be supplied to the MIDI circuitry and then MIDI THROUGH and MIDI OUT ports will be set through software.
Microcontroller / MIDI Board (Button Pins)	Microcontroller / MIDI Board (ATmega2560)	Receives digital input values from the buttons connected to the Microcontroller / MIDI Board.
Auxiliary Board	Microcontroller / MIDI Board (ATmega2560)	Receives digital input values from the buttons connected to the Microcontroller / MIDI Board.
Microcontroller / MIDI Board (ATtiny85)	Microcontroller / MIDI Board (ATmega2560)	The ATtiny85 sends “interrupts” to the ATmega2560 serially, in the form of writing a digital pin value high (this is connected to a hardware interrupt pin on the ATmega2560).
Microcontroller / MIDI Board	Potentiometer Board	Supplies I2C signals to the Potentiometer Board for the motor and LED PWM drivers.
Microcontroller / MIDI Board	Auxiliary Board	Supplies I2C signals to the Auxiliary Board for the LED PWM drivers.
Power Distribution Board	Microcontroller / MIDI Board	Connects to Power Distribution board through *only* a 12 V DC power source connection.

Potentiometer Board - This board was made with a modular design, and as such each board contains sufficient space for four motorized potentiometers. Four of these boards are used in the final design. Here, both input Power levels from the Power Distribution Board are regulated down. This board drives the motors, and step LEDs.

Table 3: Potentiometer Board Interconnects:		
Connection From:	**Connection To:**	**Purpose:**
Potentiometer Board (Button Pins)	Microcontroller / MIDI Board (ATmega2560)	Supplies digital values from buttons (on board) to the ATmega's digital pins on the Microcontroller / MIDI Board.
Potentiometer Board	Microcontroller / MIDI Board	PWMs connect to Microcontroller / MIDI Board through I2C data bus lines.
Power Distribution Board	Potentiometer Board	Connects to the Power Distribution Board through *both* the 18 V DC and 12 V DC power source connections.

Auxiliary Board - This board houses additional buttons and LED pins as well as additional LED PWM drivers. This board was *potentially* used in the event that there was time to implement additional functionality to the MIDI Controller Sequencer.

Table 4: Auxiliary Board Interconnects:		
Connection From:	**Connection To:**	**Purpose:**
Auxiliary Board	Microcontroller / MIDI Board	PWMs connect to Microcontroller / MIDI Board through I2C data bus lines.
Auxiliary Board	Microcontroller / MIDI Board. (ATmega2560)	Supplies digital values from buttons (on board) to the ATmega's digital pins on the Microcontroller / MIDI Board.
Power Distribution Board	Auxiliary Board	Connects to Power Distribution board through *only* a 12 V DC power source connection.

2 Design

2.1 Equations & Simulations

Timing Calculations

- Estimations for timing specific tasks shown in a table in the Appendix.

Detent Calculations

-Many of our calculations were code based due to the logic controlling aspect of the microcontroller. These are shown in the Appendix.

2.2 Design Alternatives

Table 5: Design Alternatives Explanation and Results			
Design Issue:	**Corrective Action Taken:**	**Overall Results:**	**Quantitative Explanation:**
Mounting PCBs to each other.	Included drill holes into MIDI, Potentiometer and Auxiliary PCB design.	Mounting between boards worked successfully with hex standoffs.	N/A
Keeping ground planes common while isolating noise between planes for both the 18 V DC and 12 V DC power supplies on the Power Distribution Board.	Decided to make two ground planes on the Power Distribution Board connected only at the “star point” (ferrite bead).	Allowed for safer testing and fewer shorts throughout the project.	N/A
The original Potentiometer Board design was incorrectly designed with the 10 V linear regulators as the electromotive potential between the potentiometer.	In the board redesign, the 10 V linear regulator output was switched to supply power to the motor driver instead.	The 10 V linear regulator supplied power to the correct components.	N/A
The original Potentiometer Board design used a library with an incorrect 5 V linear regulator pad configuration.	In the board redesign, the 5 V linear regulator pad configuration was manually changed (using EAGLE) to the correct configuration.	Redesigned Potentiometer Board’s 5 V linear regulator regulated voltage levels correctly.	N/A
MIDI IN tempo code with arduino UNO testbench not reading in data correctly.	Altered the code and used the 6 times per beat MIDI tuning clock command to obtain correct tempo.Additionally the MIDI cable direction was changed.	Arduino UNO correctly read input MIDI data.	N/A
Polling for the tempo and setting the tempo based off of that was too asynchronous, and resulted in an atonal sequence.	Hardware interrupts were used to calibrate the tempo and output notes at the correct frequency.	Sequences were played with correct timing.	N/A
Motors for the motorized potentiometers on one and a half boards could not be controlled. This was due to shorting issues connecting VCC and GND for some I.C.s .	Most components that overheated due to shorting were replaced. However some I.C.s could not be.	Some of the motor functionality for the device could not be implemented due to software constraints	6 motors could not be controlled due to lack of H-bridge supply
The motors that did run successfully on the MIDI controller Sequencer resulted in instability in potentiometer slider position (at times) .	The code was rewritten several times in an attempt to reduce the instability in potentiometer slider position.	This issue could not be fixed.	Using *only* the wiper blade voltage as the determining factor for potentiometer slider position was not adequate. If a future attempt was made for accurate potentiometer slider positioning, a PID algorithm should be implemented.

2.3 Design Description & Justification

Component selection was one of the most important parts of the design process. One of the most important components that needed to be selected was a microcontroller. We decided to use the Atmega 2560 microcontroller. From the data sheet:

Table 6: Comparison between: Atmel Atmega640/V-1280/V-1281/V-2560/V-2561/V. [1]

Device	Flash	EEPROM	RAM	General Purpose I/O pins	16 bits resolution PWM channels	Serial USARTs	ADC Channels
ATmega640	64KB	4KB	8KB	86	12	4	16
ATmega1280	128KB	4KB	8KB	86	12	4	16
ATmega1281	128KB	4KB	8KB	54	6	2	8
ATmega2560	256KB	4KB	8KB	86	12	4	16
ATmega2561	256KB	4KB	8KB	54	6	2	8

Using the above table, we decided the ATmega2560 is the best option for this application. We need 16 ADC channels and at least 2 USART pins. This cuts down the options between the ATmega 2560, ATmega 640 and ATmega1280. The reason we chose the ATmega 2560 is because it has a large amount of flash storage. This gave us room to add more software functionality at a later time.

The second most important component to find was the sixteen linear, motorized potentiometers. These were selected by the ability to perform pretty well and with the cheapest motorized pots we could find on the market. Due to how expensive these specialized components are, we had to select these really carefully.

For most board to board interfaces we chose to use 0.1’’ (inch) spaced pin headers to house jst xh connector housings. This allows for easy interchange of board to board connections as well as good debugging points. For the power distribution board, we thought it would be best to use screw jacks for easy interchange, because they are quite robust and are designed for power applications.

The ATtiny85 will act as an intermediary between the encoder dials and the main microcontroller. These will be nice to act as an I2C data buffer and interrupt handler. These ATtiny’s will also help reduce the software complexity on the Atmel 2560 microcontroller which will help with debugging the code. Various voltage regulators were selected to deliver reasonable power values to various components across the device. These dropout regulators were selected based on the restrictions of the components drawing power from them.

We also designed and implemented a power distribution board that will disperse power to the other various boards in the project. It is this board that will house the circuit protection and makes sure over current scenarios were handled safely.

Modularity was also a key design component. This module design would allow for unit testing of individual components and subsystems. Unit testing ensured that when subsystems were connected for the first time, that they would work accordingly. This design technique helped shorten the time it took to fabricate and test working circuitry. In only a few instances this did not actually help mitigate integration issues and these situations were mostly self inflicted.

Faceplate Design and Development

The faceplate design went through many iterations and design improvements. All of these improvements fell into two different categories, 1) adding more functionality for future project versions or 2) adjusting fitment issues. Below are the two different versions of the faceplate design in CAD and the third image is the final, fabricated version 2 of the faceplate.

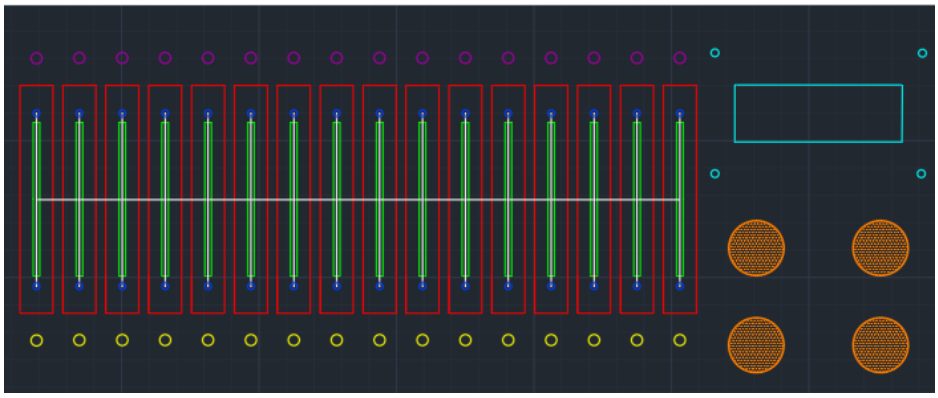

Figure 2: The above figure show the birds eye view of Team 2's first faceplate design. Note that this design didn't have adequate cutouts for all buttons and LEDs we wanted in the final design. This design was useful to help reteach CAD development and we used many of these measurement in future versions.

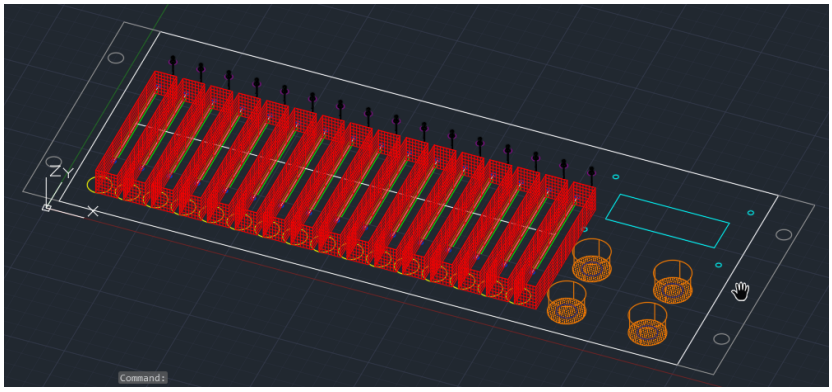

Figure 3: Shows a 30° offset 3D view of version 1 of the faceplate.

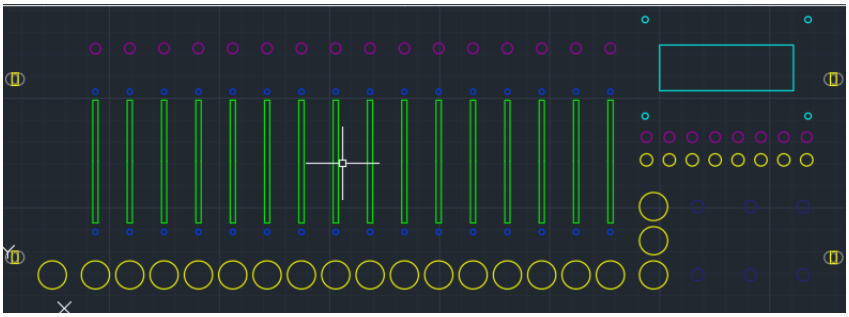

Figure 4: The above figure shows V2 (version 2) of the faceplate. In this version the number of button locations along the bottom of the PCB was increased to the preferred amount of one per potentiometer. This faceplate was also design to fit a 4U rack mount.

Figure 5: The figure above is the final, V2 faceplate post fabrication. It was laser cut from a 3mm thick blue acrylic stock.

2.4 Subsystem Diagrams & Schematics

Note: The following subsystem Diagrams and Schematics detail the parts of the initial design we got working by the time of demo. More was designed and built and can be found in the Design Document.

Software Flowchart

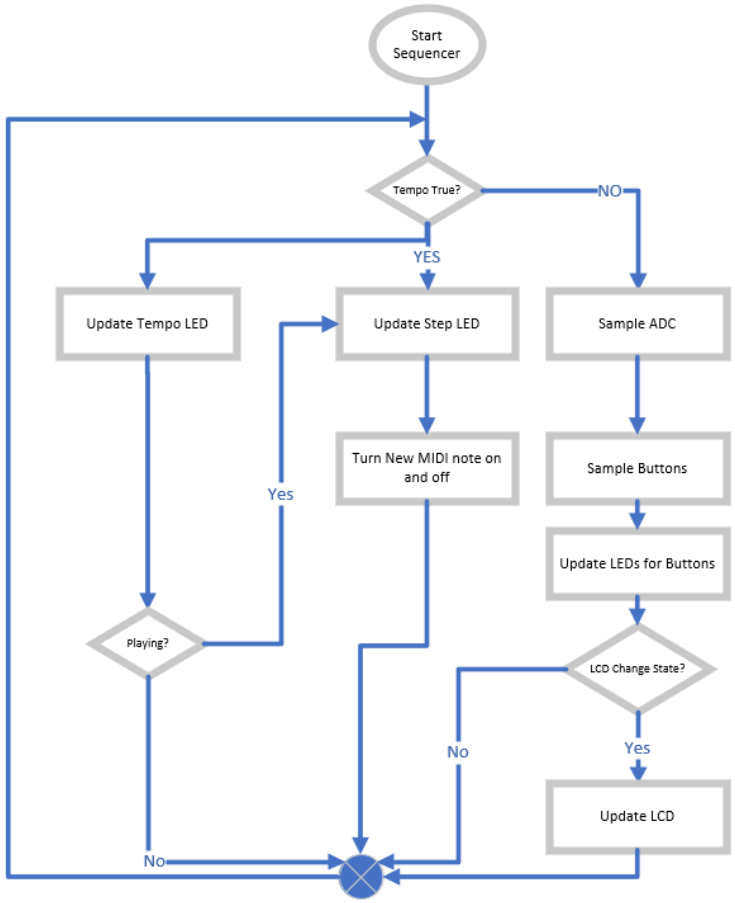

Figure 6: The flowchart above details the software system that was designed for the MIDI sequencer. It show how timing sensitive features took priority in the completed project. Less timing sensitive tasks were completed only after timing sensitive tasks/interrupts were serviced.

Potentiometer Module Schematics

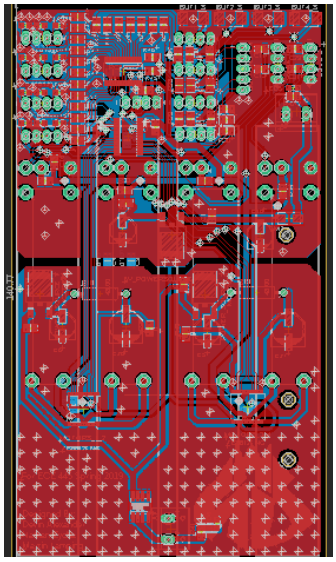

Figure 7: Potentiometer Board PCB with planes on the front and back(4 total).

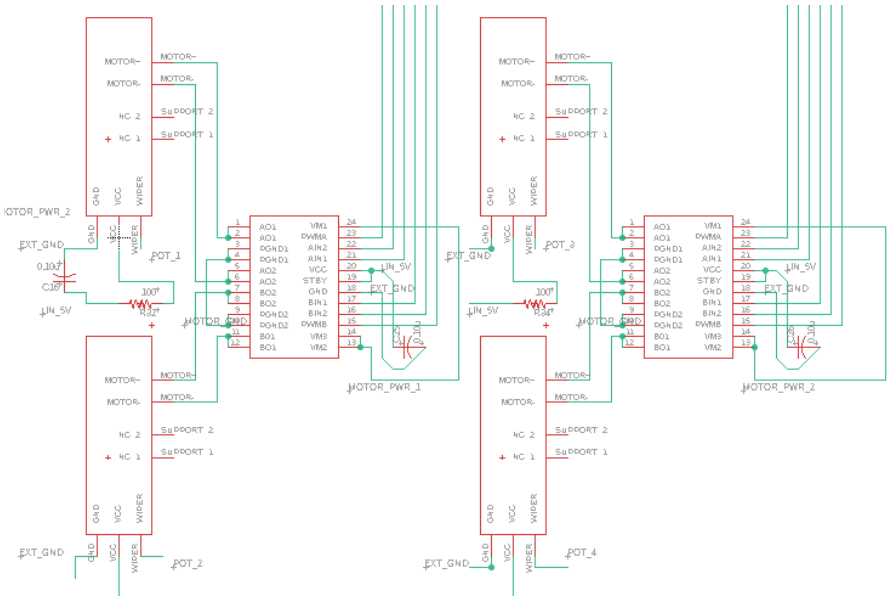

Figure 8: Portion of potentiometer board schematic showing the 4 potentiometers connected to their respective H-bridges.

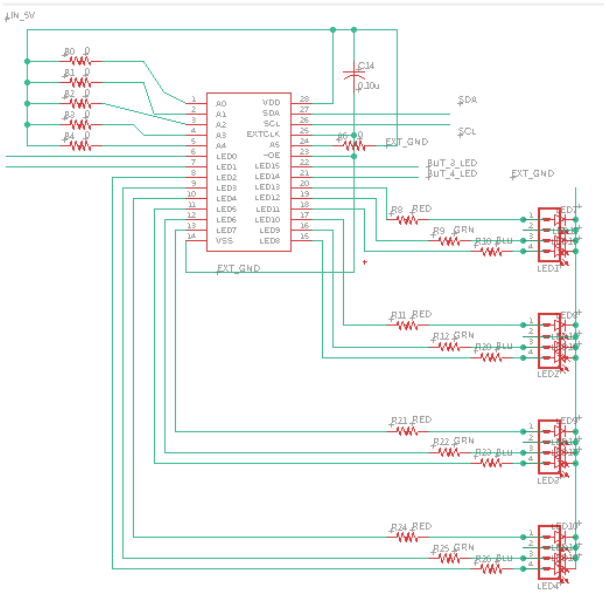

Figure 9: Portion of potentiometer schematic connecting LED JST connectors to PWM driver.

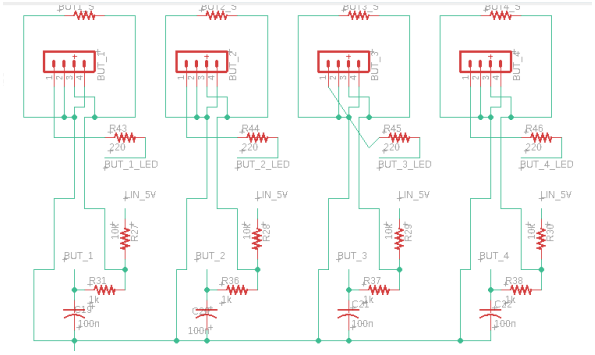

Figure 10: Portion of potentiometer schematic connecting button JST connectors to RC low pass filters.

Control Module Schematics

Microcontroller

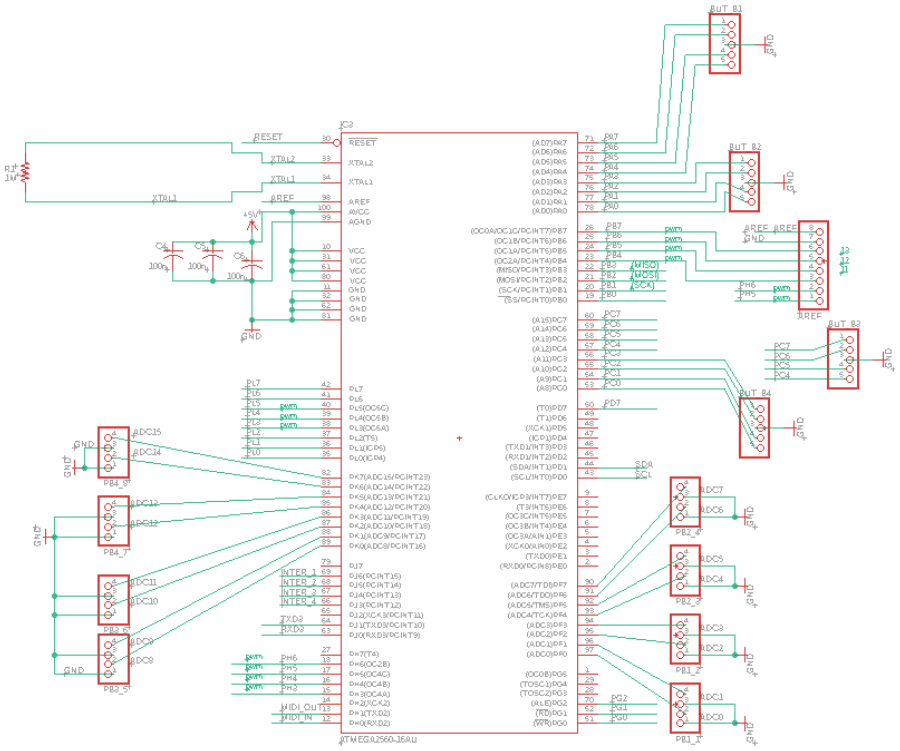

Figure 11: The above figure shows the AtMega 2560 microcontroller connections used on our final control module iteration.

Voltage Regulation

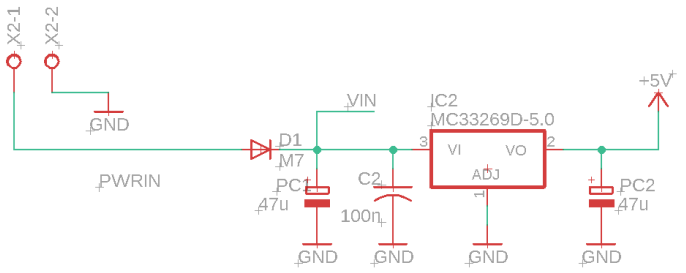

Figure 12: Power Circuitry on controller board. 5 V dropout regulator with capacitors to reduce voltage transients on the control board is used to power the ICs and made available via external pinouts.

MIDI Out

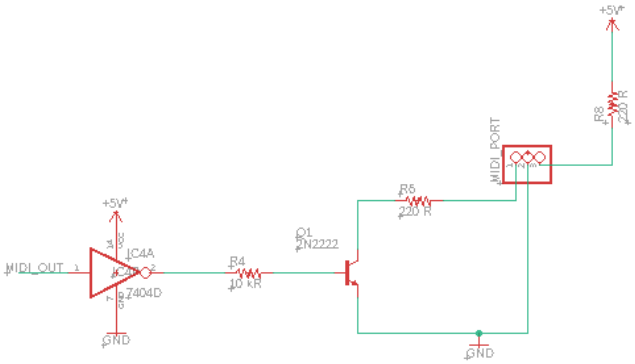

Figure 13: The above figure show the MIDI out circuit. It uses a standard 7404D inverter IC and 2N2222 bipolar junction transistor (BJT). The 7404D (like all other IC's used in the project) uses 5 V VCC. The MIDI port selected is a standard MIDI Port and will be mounted external to the board for durability concerns and connected via a jst connector.

MIDI In

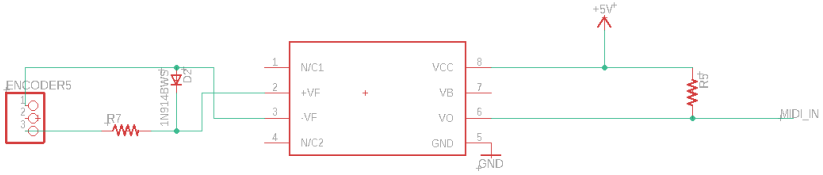

Figure 14: The above figure shows the MIDI In circuit. It utilizes a 6N139ND opto isolator from SD Micro. It was on of the more expensive parts that we almost ran out of during the fabrication portion of the different control modules versions. A standard 1N914 surface mount diode was selected for current management.

External Oscillator

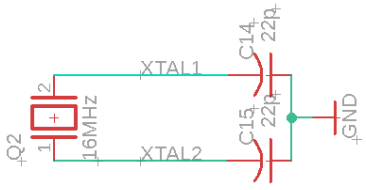

Figure 15: External Crystal Oscillator Circuit

Reset Circuitry

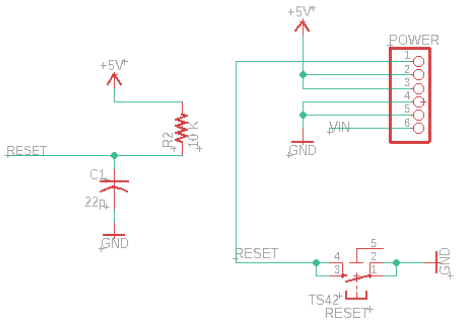

Figure 16: Reset Circuitry. Common reset circuitry used on the ATMega 2560.

Miscellaneous Circuitry

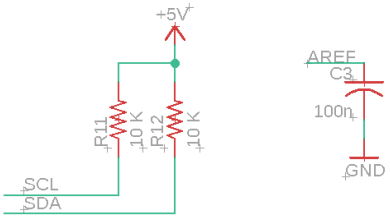

Figure 17: **Other Circuitry**. This circuitry was necessary from the proper implementation of the SCL and SDA lines. It is these two lines that comprise the I2C data bus we needed to utilize for inter board communication. The external AREF pin needed a 100 nF capacitor to smooth the voltage input and help mitigate potential surge or high voltage ripple situations for ADC readings.

Control PCB

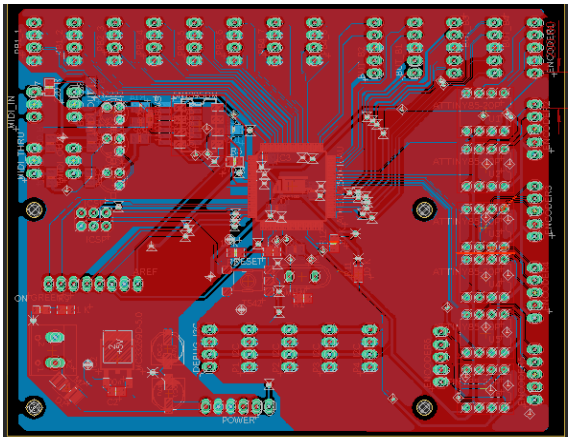

Figure 18: Control PCB: the above PCB holds the circuits detailed above. Furthermore it also has an array of six ATTiny85 ports and many interboard jst connector ports.

Power Distribution Circuitry

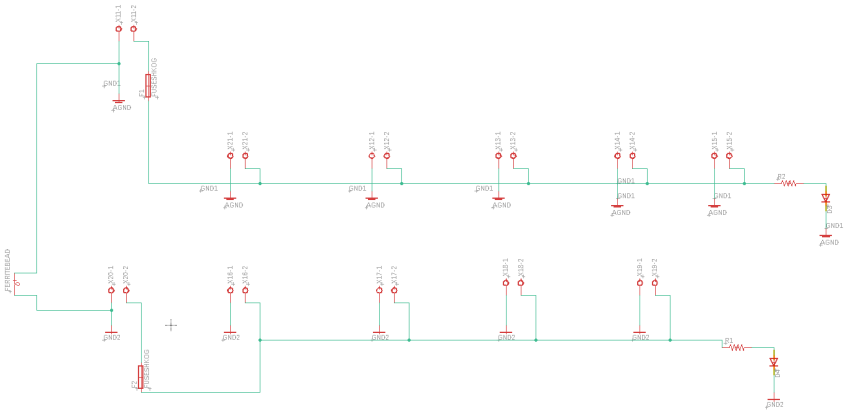

Figure 19: Power Distribution Circuitry with circuit protection: Circuit schematic detailing the power distribution circuit. This circuit has diodes that show when the specific power bus was being used for safety reasons for reverse voltage. It also has fuse protection in case of over current scenario.

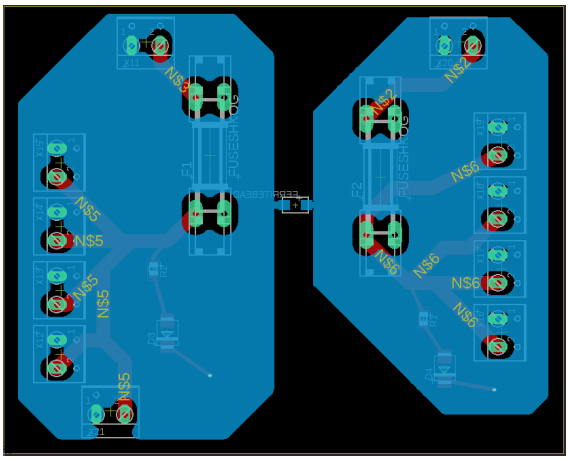

Figure 20: Power Distribution PCB: The above PCB diagram shows the power distribution circuit. This circuit was tasked with distributing power sent from the external power supplies and routing to the individual modules. It also includes circuit protection to protect the individual modules from over current using fuses.

3 Cost & Schedule

3.1 Cost

Labor

An entry level engineering job makes a starting wage of around $40 / hr or around $76,800 per year (pre-tax). With these numbers in mind, the project cost roughly: $25,600 (with project duration assumed consistent at 4 months.)

Table 7: PROJECT COST TABLE:			
PART:	**MODEL(DIGIKEY):**	**QUANTITY:**	**Total COST:**
10 V Linear Regulator	BAJ0CC0FP-E2CT-ND	8	12.16

Gen Purp 75 V 200 mA Diode	1N914BWTCT-ND	1	0.17
270 Ohm Resistor	RMCF0805JT270RCT-ND	1	0.10
10 K Ohm Resistor	RMCF0805JT10K0CT-ND	22	0.45
1 K Ohm Resistor	RMCF0805JT1K00CT-ND	16	0.35
600 Ohm Ferrite Bead	732-1620-1-ND	1	0.22
0.1 uF Ceramic Capacitor	1276-1286-1-ND	69	1.50
220 uF Aluminum Capacitor	493-2098-1-ND	8	2.37
180 Ohm Resistor	RMCF0805JT180RCT-ND	20	0.40
47 uF Aluminum Capacitor	PCE3908CT-ND	16	5.96
100 Ohm Resistor	RMCF0805JT100RCT-ND	1000	3.95
5 V 1.5 A Linear Regulator	497-7255-1-ND	5	3.50
6Ch Inverter	296-38210-1-ND	1	1.89
5 V 800 mA Linear Regulator	LM1117IDTX-5.0/NOPBCT-ND	10	13.22
8 Pin DIP Socket	AE9986-ND	6	1.08
PVC Pipe	N/A	10 ft	1.68
Opto-Isolator	6N139SDMCT-ND	5	8.65
ATmega 2560	ATMEGA2560-16AU-ND	1	12.21
P-Ch Mosfet	FDS6681ZCT-ND	4	8.80
Fuse Holder	486-1261-ND	2	3.76
Fuses (10 A, 3 A)	AGC Type	2	1.50
RGB LED Pack	N/A	24	8.99
110 V / 220 V AC to 18 V Transformer	Aiposen	1	17.94
Hex Standoff Pack	N/A	1	11.99
Black Plastic LED Holder Pack	N/A	1	5.19

Header Pack	N/A	1	5.00
60mm Motorized Potentiometers	688-RS60N11M9A0E	16	350.00
20 Detent Rotary Encoders	N/A	6	8.99
16mm LED Backlit Buttons	N/A	20	11.08
3 Pin XH JST 2.54mm Connector	N/A	4	1.99
4 Pin XH JST 2.54mm Connector	N/A	63	24.99
5 Pin XH JST 2.54mm Connector	N/A	18	12.99
6 Pin XH JST 2.54mm Connector	N/A	4	2.93

Grand Total= Labor + Cost (Note: Labor is per person)
$77,346.00 = (3 x $25,600.00) + $546.00

3.2 Schedule

This portion of the project was similar to that of the schedule displayed in the submitted "Revised Design Document", as such it will be reported in the Appendix below.

4 Requirements and Verification

Due to the expansive nature of our RV tables, the RV tables that weren't changed or were met successfully were placed in the Appendix. Tables that were not met are placed below.

Table 8: Encoder Requirement and Verification Table

REQUIREMENT:	VERIFICATION:
1.) Enough detent for active cycling.	A. Create list of all desired outputs that encoder needs to cycle through. B. Connect *every* encoder output to Arduino board pins. C. Use digitalRead(pin) syntax to read *every* pin output with Arduino IDE software. D. If detent of encoder is less than or greater than the number of desired outputs, ensure that the encoder cycles through the list.
1.) Tempo encoder has a range of 60 bpm - 140 bpm (1 Hz - 2.33 Hz.)	A. Turn encoder to minimum tempo of 60 bpm. B. Connect minimum frequency output of encoder to oscilloscope. C. Check that minimum frequency selection operates at or below 60 bpm. D. Turn encoder up to maximum tempo of 140 bpm.

	E. Connect maximum frequency output of tempo encoder to oscilloscope. F. Check that maximum frequency output of tempo encoder operates at or above 140 bpm.

Table 9: Motorized Potentiometer Requirement and Verification Table	
REQUIREMENT:	**VERIFICATION:**
1.) Potentiometer ranges from 0 KΩ - 10 KΩ with a tolerance +/- 2 KΩ.	A. Configure individual potentiometer testbench. B. Apply known voltage across potentiometer. C. Measure output current as the potentiometer position is adjusted. D. Ensure that all potentiometers operate in at 10 KΩ +/- 2 KΩ.
2.) Ability to drive motors such that positions can be "quantized" to localized positions along potentiometer movement range.	A. Configure individual potentiometer testbench. B. Configure Microcontroller subcircuit with test code that will slide through all necessary positions. C. Ensure smooth transition between locations. D. Ensure lack of movement in between movements.
3.) Each individual motor needs to have noise from surrounding motors filtered out from the output voltage before being sent to ADC.	A. Connect output from the potentiometer to an oscilloscope. B. Configure microcontroller subcircuit that will slide through all necessary potentiometer positions. C. Inspect that the amplitude of any periodic spikes in voltage from the motor from that potentiometer (or nearby potentiometers) is less than 8 mV +/- 2 mV.
4.) Potentiometer sliders will need to be moved at a speed that seems fluid but does not harm users.	A. Configure individual potentiometer testbench. B. Configure microcontroller subcircuit with test code that will slide through all necessary note positions. C. Test out the motor controllers with human hands touching the knobs. D. Determine that movements do not cause discomfort.

5 Conclusion

5.1 Accomplishments

Overall, when looking back at the objectives and completeness, this project has shown much success. The final version of the project interacts both with MIDI IN, and MIDI OUT data. MIDI OUT and MIDI IN data is sent and read by our code and operates similarly to MIDI encoding standards. MIDI OUT sends output according a pitch, root note, tempo and scale which was listed as a project objective. MIDI OUT also sends MIDI active sensing and MIDI clock as specified in MIDI encoding. The tempo for a sequence of notes can be read in and set from MIDI IN, or set from a linear potentiometer slider (instead the planned use of an encoder) by the user. Tempo data is used to output notes at a consistent tempo through the use of interrupts. The motors for the motorized linear potentiometer sliders do work on some of the steps for the sequence however not as intended. For the final version of the project 10 of the 16 potentiometers have motor code that can snap the slider into

place (to a detent) when pushed by the user. Using the detent positioning motor functionality of the slider proved to be quite challenging, and the final slider position often oscillated about a point. In total there was some success with motor positioning, however its full potential functionality has not been reached. The objective to display information on the LCD screen for the user has also been reached. The LCD screen functions as intended, and displays the root note, pitch, scale and tempo (in real time).

5.2 Uncertainties

Throughout this project there were a couple prominent areas of unsatisfactory results. These areas presented mainly in the code / software portion of the design, and the motor aspect of the project. In the final version of the code / software, a working version of interrupts to be sent by the ATtiny had not been finished. The final code for the ATtiny interrupts was such that an interrupt was sent twice when the encoder was turned clockwise, and once when turned counter-clockwise. In a future version of the project, this issue could be fixed by accounting for if the gray-code output of the encoder was incrementing or decrementing. Based on the change in grey code, it could be determined which direction the encoder was turned in. If turned clockwise, a counter would be set and ensure that only one interrupt was set per turn.

Secondly, the intended functionality of the motors on some of the motorized potentiometers was not met. The motors did not function correctly due to both the hardware and software. Some of the H-bridges didn't function as intended and this caused some motors to run unpredictably. Some H-bridge ICs did not work due to inadequate soldering and the fact that a cheap package was chosen. Heat was not dissipated on these H-bridges and thus the ICs were prone to burning up and internal shorting. Other aspects of code that had not been finished include all of the auxiliary board functionality (which was a reach goal).

5.3 Future Work / Alternatives

There are many areas for potential improvement in the current MIDI Controller Sequencer design. Some, but not all of the reported reach goals outline the areas for improvement. Firstly, the micro-controller board would need to have input pins for the transmitting and receiving lines, also boot loader circuitry would prove a worthy addition, both of these are crucial in shortening debugging time. One aspect in the design that proved unruly was cable management, and as such, the use of ribbon cables in place of the JST connectors would greatly aid in organization. Another very important alternative would be to use a PID algorithm for intended functionality of motors instead of purely using the potentiometer slider position. For the LCD screen, a menu could be added for user navigation with the use of encoders.

5.4 Ethical Considerations

IEEE's code of ethics article 1 is, "to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, and to disclose promptly factors that might endanger the public or the environment." [5] Our group also strives to uphold IEEE's article 1 safety standards throughout our project. Synthesizers/sequencers generally operate at 15 V [6], these voltage levels are considered a safe range for users to operate the device at. Our synthesizer implementation will run at a similar voltage level, with a maximum operation voltage of around +18 V. Even with the device operating at safe voltage levels, there is always a potential for risk. To mitigate the risk of electrical shock, the device inputs and outputs, and even some passive elements will be clearly labeled and fuse / led circuitry was implemented to limit current through some of the boards.

Additionally, non-electrical issues taken into consideration are users jamming their fingers into the slider holes, sharp edges, or otherwise rough use of the device. The physical faceplate will be designed such that the holes are not large enough for a user to insert their fingers into. Sharp edges will either be noted by a warning label on the front of the faceplate or the edges will simply be ground down so the edges feel smooth. In following with Underwriter Laboratories, " We will create and maintain environmental, health and safety (EHS) work practices and secure work environments that enable employees to work injury free." [7] Significant amounts of soldering will pose a safety risk for the team but we will handle it accordingly. Carbon filtered fans were used to keep from solder fume inhalation during soldering. In keeping with the Underwriter Laboratories statement above, safety precautions such as personal protective equipment, adequate ventilation, and lighting will always be available and in use during the assembly stages of this project. Furthermore cleanliness and proper workspace organization will lessen the risk of accidents.

The main ethical dilemma that may need to be considered is the way in which our device is used. The types of music, sounds, and melodies, a user chooses to produce can have an effect on a listener's behavior and emotions. Another safety hazard/ ethical dilemma would be if a user operates the MIDI controller sequencer with malicious or harmful intent or otherwise forcing the device to operate under abnormal conditions (e.g. running the motors too often or quickly, forcing potentiometers into place, etc.).

References

[1] “Atmel Atmega640/V-1280/V-1281/V-2560/V-2561/V” Datasheet. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf. [Accessed: 27-Feb-2019].

[2] “MIDI,” code circuits construction. [Online]. Available: http://www.tigoe.com/pcomp/code/communication/midi/. [Accessed: 12-Mar-2019].

[3] “Arduino Mega 2560 Reference Design” arduino.cc [online] available: https://www.arduino.cc/en/uploads/Main/arduino-mega2560-schematic.pdf [accessed: 2-Mar-2019]

[4] Bermudez, Adam. “How to Reduce Electromagnetic Interference in Motor Drive Systems.” [Online] Available: http://www.quantumautomation.com/uploads/7/3/8/8/7388264/techcorner_46_-_reduce_electromagnetic_interference.pdf [Accessed: 5-Mar-2019]

[5] “IEEE Code of Ethics,” *IEEE - Advancing Technology for Humanity*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 22-Feb-2019].

[6] “What is Voltage? | The Synthesizer Academy,” *synthesis tutorial*. [Online]. Available: http://synthesizeracademy.com/what-is-voltage/. [Accessed: 22-Feb-2019].

[7] “Ethics and Compliance,” *UL - Empowering Trust*. [Online]. Available: https://www.ul.com/aboutul/standards-of-business-conduct/. [Accessed: 25-Feb-2019].

6 Appendix

6.1 Timing

Table A1: Timing

The table below shows the timing data that was calculated empirically. Tasks were timed in software and the results are as follows:

Task	Time (sec)
Sample ADC	108 µs - 1896 µs
Sample Button	108 µs -1896 µs
Change Step LED	3996 µs - 4004 µs
Change Motor	8 ms
Blink Tempo LED	Variable based on BPM
Service Encoders	~ 500 µs
Update LCD Screen	~ 26548 µs

Detent Calculations:

$$\text{Detent spacing} = \frac{1024}{Num\ Detents} \qquad \text{(eq 1)}$$

where : $Num\ Detents$ = 13 for one octave
= 25 for two octaves

6.3 Schedule

Table A2: Schedule

WEEK:	Nathan:	Devin:	Martin:
Week of 2/18	1. Design Document Check - 2/19 2. Begin initial potentiometer board PCB diagram/design in Eagle 3. Start soldering assignment	1. Design Document Check - 2/19 2. Start soldering assignment 3. Begin initial potentiometer board PCB diagram/design in	1. Design Document Check - 2/19 2. Start soldering assignment 3. Begin software design/debugging for 1 test potentiometer 4. Microcontroller

	4. Plan design review presentation details 5. Begin software design/debugging for 1 test potentiometer	Eagle 4. Plan design review presentation details 5. Begin software design/debugging for 1 test potentiometer	breakout board design 5. Plan design review presentation details
Week of 2/25	1. Design Review - 2/27 2. Finish potentiometer board PCB diagram/design in Eagle	1. Design Review - 2/27 2. Finish potentiometer board PCB diagram/design in Eagle	1. Design Review - 2/27 2. Continue work on microcontroller circuit
Week of 3/4	1. Teamwork evaluation 1 - 3/4 2. Soldering assignment due - 3/8 3. Finish microcontroller board PCB diagram/design in Eagle 4. Begin software design/debugging for microcontroller 5. Finish initial PCB designs and (microcontroller and potentiometer board) for PCBway	1. First round of PCBway orders must pass audit - 3/14 2. Submit final revisions to machine shop - 3/15 3. Begin software design/debugging for LCD and MIDI data 4. Improve/fix initial PCB design	1. Teamwork evaluation 1 - 3/4 2. Soldering assignment due - 3/8 3. Finish microcontroller board PCB diagram/design in Eagle 4. Begin software design/debugging for microcontroller 5. Finish initial PCB designs and (microcontroller and potentiometer board) for PCBway
Week of 3/11	1. First round of PCBway orders must pass audit - 3/14 2. Submit final revisions to machine shop - 3/15 3. Begin software design/debugging for	1. First round of PCBway orders must pass audit - 3/14 2. Submit final revisions to machine shop - 3/15	1. First round of PCBway orders must pass audit - 3/14 2. Submit final revisions to machine shop - 3/15 3. Begin software design/debugging for

	LCD and MIDI data 4. Improve/fix initial PCB design	3. Begin software design/debugging for LCD and MIDI data 4. Improve/fix initial PCB design	LCD and MIDI data 4. Improve/fix initial PCB design
Week of 3/18	1. Finish final PCB design 2. Finish software design/debugging for LCD and MIDI data	1. Finish final PCB design 2. Finish software design/debugging for LCD and MIDI data	1. Finish final PCB design 2. Finish software design/debugging for LCD and MIDI data
Week of 3/25	1. Individual progress reports due - 3/25 2. Final Round PCBway orders must pass audit - 3/28 3. Prepare for mock demo	1. Individual progress reports due - 3/25 2. Final Round PCBway orders must pass audit - 3/28 3. Prepare for mock demo	1. Individual progress reports due - 3/25 2. Final Round PCBway orders must pass audit - 3/28 3. Prepare for mock demo
Week of 4/1	1. Mock demo 2. Debug/improve all software applications	1. Mock demo 2. Debug/improve all software applications	1. Mock demo 2. Debug/improve all software applications
Week of 4/8	1. Prepare for mock presentation 2. Add “reach goal(s)” hardware and software functionality if possible	1. Prepare for mock presentation 2. Add “reach goal(s)” hardware and software functionality if possible	1. Prepare for mock presentation 2. Add “reach goal(s)” hardware and software functionality if possible
Week of 4/15	1. Prepare for mock presentation 2. Prepare final paper document	1. Prepare for mock presentation 2. Prepare final paper document	1. Prepare for mock presentation 2. Prepare final paper document
Week of 4/22	1. Mock presentation	1. Mock presentation	1. Mock presentation

	2. Prepare for final presentation 3. Prepare final paper document	2. Prepare for final presentation 3. Prepare final paper document	2. Prepare for final presentation 3. Prepare final paper document
Week of 4/29	1. Final presentation 2. Final paper due - 5/1 3. Lab notebook due - 5/2 4. Teamwork evaluation II due - 5/2	1. Final presentation 2. Final paper due - 5/1 3. Lab notebook due - 5/2 4. Teamwork evaluation II due - 5/2	1. Final presentation 2. Final paper due - 5/1 3. Lab notebook due - 5/2 4. Teamwork evaluation II due - 5/2

Requirement and Verification Tables

-These RV tables were met in the finished version of the project.

Table A2: LCD Module Requirement and Verification Table	
REQUIREMENT:	**VERIFICATION:**
1.) Make sure that the contrast of the LCD display is such that the message can be seen on it from multiple viewing angles.	A. Configure microcontroller test circuit with LCD interface. B. Send known text to the LCD from microcontroller. C. Provide an analog voltage to the pin on the display controller that is responsible for controlling the contrast. D. Adjust contrast until viewable from top, sides and bottom with viewing angle of roughly 45 degrees from the z axis.
2.) LCD backlight brightness is great enough to see the text on the screen in a dark environment.	A. Configure microcontroller test circuit with LCD interface. B. Send known text to the LCD from microcontroller. C. Take test setup into dark room. D. Make sure that the voltage going to the LCD I2C to parallel backplate has the correct voltage to power the LED backlight. E. Ensure backlighting is bright enough
3.) Real time display of messages	A. Configure microcontroller test circuit with LCD interface. B. Supply the LCD with a series of known messages. C. Increase the refresh rate of that.
4.) The LCD display can not take up too many pins on the microcontroller.	A. To save IC digital pins for use of connecting to other modules, we will be using a Parallel to I2C BUS converter chip to free up digital pins for other requirements.

5.) LCD screen will run off a 5 V regulator +/- 0.5 V.	A. Connect linear regulator to 12 V power supply. B. Using multimeter measure the output of the linear input to the LCD screen and confirm voltage is 5 V +/- 0.5 V.

Table A3: 9 V Linear Regulator Requirement and Verification Table	
REQUIREMENT:	**VERIFICATION:**
1.) Outputs at a voltage level of 9 V with a tolerance of +/- .1 V at a maximum current output of 500 mA.	A. Set up linear voltage regulator mock up circuit with 500 mA load. B. Connect known 12 V power supply to voltage regulator. C. Connect voltage regulator to 24 Ω in order to pull 500 mA. D. Test output is operating within acceptable range of 8.9 V -9.1 V and the output current does not exceed 500 mA.

Table A4: 5 V Linear Regulator Requirement and Verification Table	
REQUIREMENT:	**VERIFICATION:**
1.) Outputs at a voltage level of 5 V with a tolerance of +/- .4 V at a maximum current output of 1500 mA.	A. Set up linear voltage regulator mock up circuit with 1500 mA load. B. Connect known 12 V power supply to voltage regulator. C. Measure output is operating within acceptable test range of 4.6 V - 5.4 V and the output current does not exceed 1500 mA.

Table A5: 12 V Power Supply Requirement and Verification Table	
REQUIREMENT:	**VERIFICATION:**
1.) Provides +/- 12 V with a tolerance of +/- .1 V .	A. Set up mock test circuit with rated resistor. B. Apply 12 V power supply to rated resistor. C. Use digital multimeter to test that the voltage drop across the resistor does not exceed 12.1 V.
2.) Supplies 4.5 A +/- 2A	A. Set up mock test bench circuit with resistive load of 1.10 Ω (comparable to circuit). B. Calculate current across resistor.

Table A6: Microcontroller Requirement and Verification Table	
REQUIREMENT:	**VERIFICATION:**
1.) Supports an I2C frequency of at least 100 KHz.	A. Input I2C test signal using computer and USB input port of breakout board. B. Measure output frequency and confirm it reaches at least 100 KHz +/- .5 KHz using oscilloscope.

2.) Crystal oscillator will run off a 5 V regulator +/- 0.2 V.	A. Connect linear regulator to 12 V power supply. B. Using multimeter measure the output of the linear oscillator and confirm voltage is 5 V +/- 0.2 V. C. Apply output of 5 V linear oscillator to the input of the crystal oscillator. D. Confirm normal operation of 16.000 MHz output of crystal oscillator via oscilloscope.
3.) Samples voltage levels at 1000 KHz.	A. Input square wave with peak to peak voltage 3.3 V from function generator. B. Connect microcontroller output to oscilloscope. C. Ensure that microcontroller is capable of sampling signal at 1000 KHz by measuring period of output sample.
4.) Microcontroller must be capable of reading a sample of at most 10 ms +/- 5 ms.	A. Set up microcontroller test bench and read signal values via oscilloscope. B. Provide microcontroller with known analog signal. C. Make sure that the known signal and values read into the microcontroller at the required 10 ms +/- 5 ms match.
5.) Samples voltage levels at 1000 KHz.	A. Input square wave with peak to peak voltage 3.3 V from function generator. B. Connect microcontroller output to oscilloscope. C. Ensure that microcontroller is capable of sampling signal at 1000 KHz by measuring the period of the output sample.
7.) ADC being internal to the microcontroller.	A. Check the datasheet for the microcontroller to see if there are any pins being utilized for analog to digital conversion. This will reduce design complexity.

Table A7: PWM LED Driver Requirement and Verification Table

REQUIREMENT:	VERIFICATION:
1.) I2C bus on the microcontroller, that functions at a frequency of 100 KHz.	A. Connect mockup PWM driver circuit to oscilloscope. B. Run data through the I2C bus. C. Measure the frequency at which I2C signal oscillates. D. Check If the signal oscillates at a frequency within +/- .1 KHz of the required 100 KHz.
2.) PWM driver will run off a 5 V regulator +/- 0.2 V.	A. Connect linear regulator to 12 V power supply. B. Using multimeter measure the output of the linear oscillator and confirm voltage is 5 V +/- 0.2 V. C. Apply output of 5 V linear oscillator to the input of the crystal oscillator.
3.) The LED driver provides pulses that are able to drive LEDs at 20 mA per channel.	A. Assemble microcontroller and LED driver subcircuit. B. Use multimeter to check the LED's channel current is within +/- 5 mA of the required 20 mA.

Table A8: PWM Motor Driver Requirement and Verification Table	
REQUIREMENT:	**VERIFICATION:**
1.) I2C bus on the microcontroller, that functions at a frequency of 100 KHz.	A. Connect mockup PWM driver circuit to oscilloscope. B. Run data through the I2C bus. C. Measure the frequency at which I2C signal oscillates. D. Check If the signal oscillates at a frequency within +/- .1 KHz of the required 100 KHz.
2.) PWM driver will run off a 5 V regulator +/- 0.2 V.	A. Connect linear regulator to 12 V power supply. B. Using multimeter measure the output of the linear oscillator and confirm voltage is 5 V +/- 0.2 V. C. Apply output of 5 V linear oscillator to the input of the crystal oscillator.

Table A9: Button Array Requirement and Verification Table	
REQUIREMENT:	**VERIFICATION:**
1.) Output digital HIGH (1) when pressed or digital LOW (0) when pressed depending on if the button is active high or active low.	A. Configure button testbench on breadboard. B. Use a multimeter to read voltage across button when it is pressed. C. Check if voltage is above the critical point of 2.5 V within +/- .2 V.
2.) Debounce buttons.	A. Build a lowpass filter with capacitor in parallel with the button and resistor in series. B. Connect oscilloscope to the output of button. C. Apply 3 V input to button. D. Map button activation on oscilloscope. E. Ensure that the button remains in the high position with tolerable bounce (listed in **2.3.12** summary above).

Table A10: LED Array Requirement and Verification Table	
REQUIREMENT:	**VERIFICATION:**
1.) The LEDs are RGB and can be changed to any color in the the RGB colour space .	A. Cycle through absolute RGB values. B. Ensure correctly display R, G and B C. Variation in LEDs will be judged by running the R , G, and B values. through the standard 256 color value ranges.
2.) The LEDs must be driven correctly by PWM signals generated by an external	A. Set up I2C connection protocol between microcontroller and LED array. B. Ensure an I2C data signal from the Microcontroller has individually addressed PWM channels. C. Test I2C communication with all three color channels on each LED individually.

Table A11: MIDI Ports Requirement and Verification Table	
REQUIREMENT:	**VERIFICATION**
1.) Successfully outputs MIDI data with industry standard bit rate of 31.25 Kbaud ~ 3 bytes / ms.	A. Mock up microcontroller test board. B. Connect MIDI input and output data lines to UART port (solder). C. Supply MIDI port with known sequence of MIDI data from computer DAW. D. Check that data is being sent at a rate of 31.25 Kbaud~ 3 bytes / ms.
2.) Ports connected to microcontroller on two of the four UART ports.	A. Mock up microcontroller test board. B. Connect MIDI input and output data lines to UART port (solder). C. Supply MIDI port with known sequence of MIDI data from computer DAW. D. Check that data is being transferred to the microcontroller ensure successful connection.
3.) Provide a means to communicate with external MIDI devices.	A. Mock up microcontroller test board. B. Connect MIDI output port to computer DAW software. C. Flash know MIDI sequence that probes the ‘status bit’ onto microcontroller and output through MIDI output port. D. Using a DAW (digital audio workstation), check that the MIDI encoding standard “status bit” is sent with regularity.
4.) MIDI will output on, off, duration, velocity, and tempo data.	A. Synchronize the MIDI output data to that of a MIDI input device to verify if sound is coming out of a MIDI module. B. If status byte of the MIDI output data, is flashed high every 1 ms +/- 1 ms for each parameter.
5.) MIDI input will only be used for syncing with external tempo data and nothing else.	A. Mock up microcontroller test board. B. Connect MIDI input and output data lines to UART port (solder). C. Supply MIDI port with known tempo data from computer DAW. D. Ensure correct synchronization of the MIDI input clock with that of a MIDI output via the DAW using a MIDI out port on a digital interface for testing.