Affordable Portable MIDI Keyboard Synthesizer

Design Document

Group 54:
Sujay Murali
Richard Engel
David Gutzwiller

TA: Akshat Sanghvi

23 Feb 2023

Table of Contents

1 Introduction

1.1 Problem

One desirable quality for musical production instruments is portability. For many production setups, it can be difficult for musicians to take all of their gear with them, so it's convenient for them to own a robust portable synthesizer keyboard. However, here is where people will often run into another major issue: the cost. It can sometimes cost hundreds or even thousands of dollars to purchase a portable keyboard instrument that is comfortable to use. There are cheaper options out there, but often those cheaper options have to be stripped back and can be uncomfortable to use due to sacrifices needing to be made in key feel. These issues can be especially bad for new musicians who are looking to get into creating music, or for musicians who can’t afford the high costs. In short, after looking at similar products on the market and different synthesizers that are commonly used, there are not a lot of options at a cheaper price range with easy portability and good functionality.

1.2 Solution

Our proposal is for a low-cost keyboard synthesizer. The instrument is both simple enough to save on cost, but also has enough features to be highly versatile for musicians, more so than similar products on the market. The keyboard would feature one octave of range with an octave changer and pitch bend wheel, along with input knobs for volume, waveform synthesis, and ADSR envelope. These features would be enough to create a versatile instrument that is affordable and fun to use. The battery of the device would last for at least 3 hours of playing time, allowing for easy portability and use in different settings.

1.3 Visual Aid

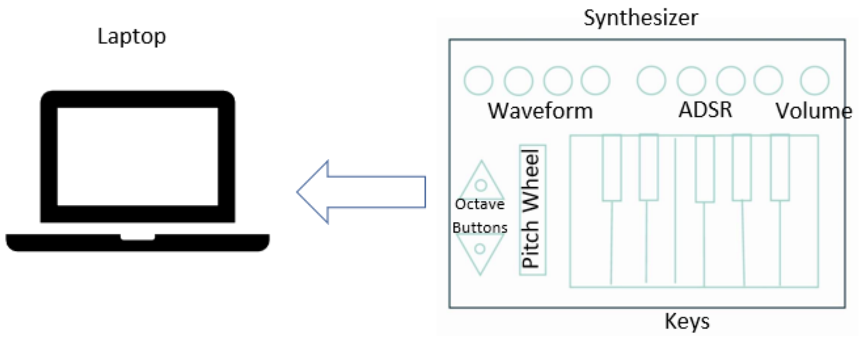

Figure 1: Visual aid of keyboard device layout and device for output.

Our visual aid represents the connections between the different components of our keyboard, as well as how our keyboard outputs MIDI data to an external device. The synthesizer will have input buttons to change the waveform, the attack, decay, sustain, release, and the notes themselves. It contains other functionalities such as a pitch wheel and a set of buttons to change the current octave that the notes are playing.

1.4 High Level Requirements

- **Input Functionality** - Every form of input should be fully functional. Each key must register which note is played, each potentiometer knob should fully control its respective parameter, both octave switches should change the octave of the keyboard's note range, and the pitch wheel should effectively shift the pitch up and down by one whole note.

- **Output Functionality** - Output should be working properly. The built-in speaker should be able to output analog sound that is digitally synthesized within the keyboard. The USB port should be able to transmit MIDI messages from the keyboard, and this should allow the keyboard to communicate with DAWs on laptop or desktop devices.
- **Power Functionality** - The built-in battery should be capable of providing power to the microcontroller within the keyboard. The battery should also last at least three hours on a complete charge.

2 Design

2.1 Block Diagram

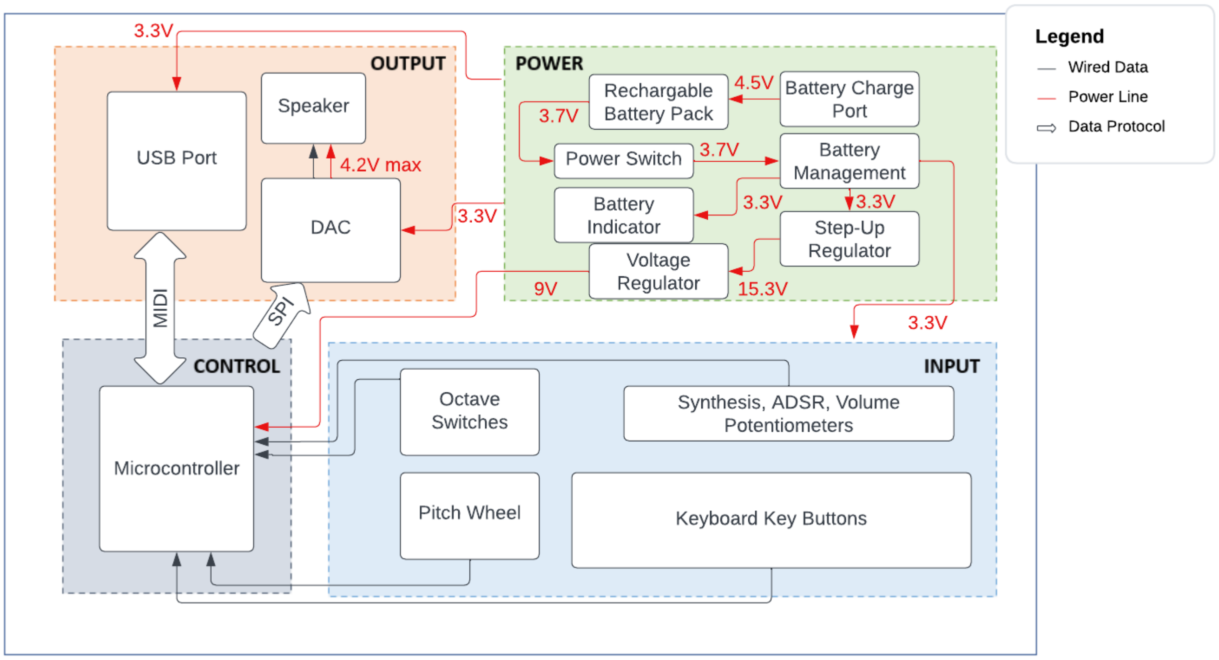

Figure 2: Synthesizer Block Diagram

2.2 Functional Overview & Block Diagram Requirements

2.2.1 Power Subsystem

The Power Subsystem is responsible for providing power to the control subsystem, while maintaining a constant voltage and keeping battery operation safe. This subsystem will consist of a battery charge port, a rechargeable li-ion battery, a rocker power switch, a battery management IC, an LED bar graph battery indicator, and a voltage regulator. The battery that we have chosen to use is the USE-18650-2600 PCB JST from US Electronics, which is rated at 2600mAh and 3.7V. For the charge port, we will use the BQ25180 IC from Texas Instruments, which can tolerate up to a 25V input voltage and output an optimized voltage to the battery between 3.6V and 4.65V. It also has a maximum charge current of 1A and integrated fault detection for the battery. For the switch, we will be using the CWSB11AA2F from NKK Switches, which is rated at up to 6A and 250V. For the LED bar graph, we will use the COM-09937 from SparkFun Electronics, which will display the battery life in 10% increments. For the battery management system, we will use the ISL94212INZ from Renesas Electronics America Inc. This IC will control the battery to ensure that it is operating at a safe voltage. It includes failsafes in case the battery overheats. From here, the battery management IC provides 3.3V to power the input and output subsystems. However, the control subsystem requires a higher voltage. To achieve this, we will use the MAX632ACPA from Maxim, which is a step-up voltage regulator with a maximum output voltage increase of +18V. This will step up the input voltage by +12V to about 15.3V. Then we will use another voltage regulator, the S-1212BC0-V5T2U from ABLIC Inc. This regulator can take input voltages between 3.0V and 36V and step down to a precise output voltage between 2.5V and 16.0V. This regulator will be able to maintain a 9V output to the Microcontroller Subsystem.

Table 1: Power Subsystem Requirements and Verification

Requirements	**Verifications**
● The power system successfully supplies 9V of power to the microcontroller	● Ensure that when the battery is not powered on that the Vin to the microcontroller is 0 Volts ● Ensure that when the battery is turned on the microcontroller Vin is greater than or equal to 9 Volts
● The battery life can last at least 3 hours from a full charge	● Measure a 9 Volts or greater as input to the microcontroller at any moment within 3 hours of turning the battery on
● LED bar graph can display battery life in 10% increments	● Monitor the LED bar graph as the battery drains

2.2.2 Input Subsystem

The Input Subsystem consists of a collection of buttons and potentiometers, all of which have their own function within the system. The data collected from these sensors will be sent off to the Microcontroller Subsystem for processing. The input layout will consist of nine knobs, thirteen piano keys, an octave up and octave down button, and a pitch wheel. Each keyboard key will have a molded silicone button with conductive ink, in order to register when a key is pressed. There will also be a switch PCB with conductive ink printed on it, which will short as each button is pressed. If there is time, a second button will be implemented for each key in order to implement velocity detection. This technique is shown in Figure 3, which shows the button design for an older MIDI keyboard. The nine knobs are split into three different functions. There will be four knobs controlling the level of a basic waveform for sound synthesis, these waveforms being sine, square, sawtooth, and triangular. The next four knobs will control the ADSR envelope, allowing for more customization of the synthesized sounds. The final knob will control the system volume that is sent to the built-in speaker. Each knob will consist of a 10kOhm linear rotary potentiometer (COM-09939) and a black knob (COM-09998) mounted onto each potentiometer. Both of these parts would be from Sparkfun. The two octave-changer buttons will alter both the current octave of the synthesized audio and the current octave of the MIDI data output to the computer device. Each octave switch will activate an E-Switch TL2202EEEZB push button. The pitch bend wheel will bend both the pitch of the synthesized audio and the pitch of the MIDI data either up or down by at most one whole tone. This pitch bend wheel will be bought off of the shelf from Syntaur. We will be using a Roland replacement pitch bend wheel (Roland part number 5100042238, Syntaur part #10391).

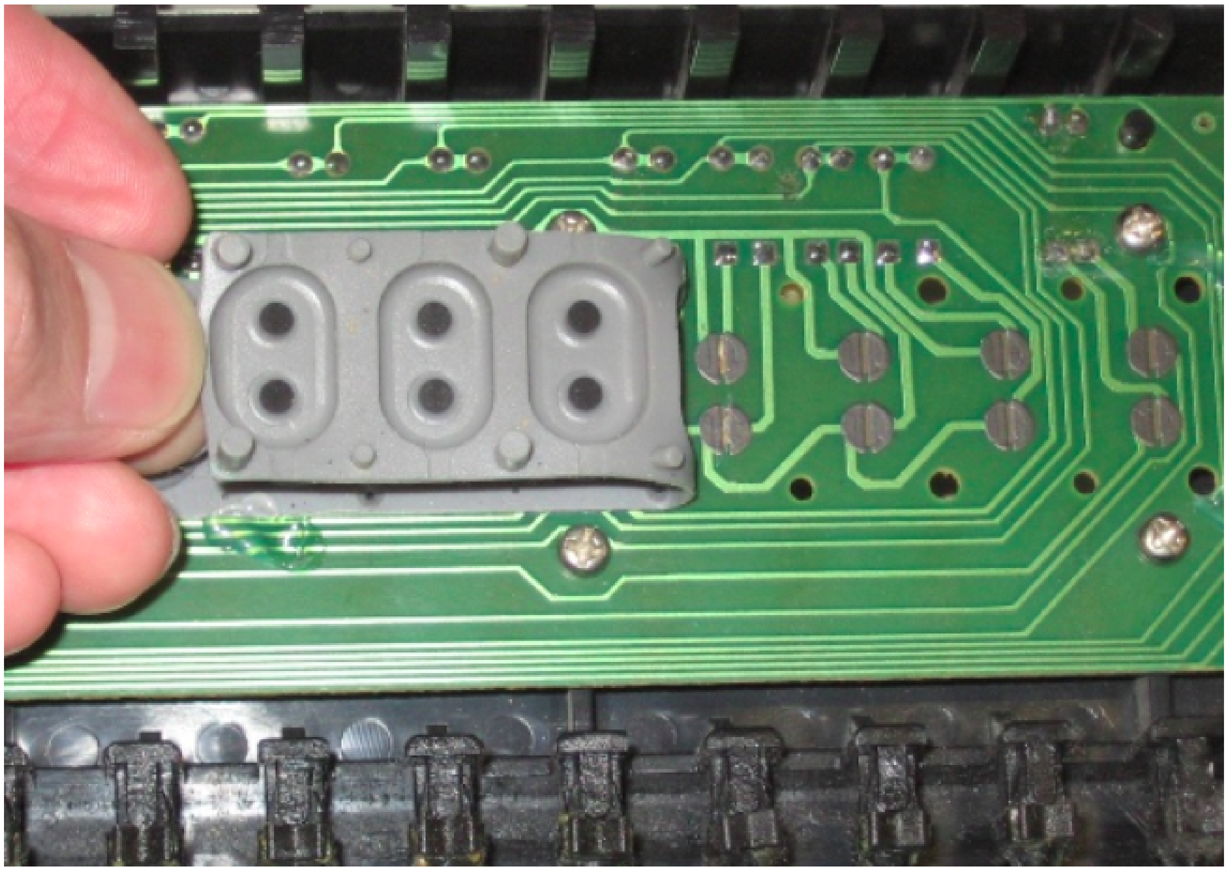
Figure 3: Rubber Buttons with Conductive Ink in Old MIDI Keyboard

Table 2: Input Subsystem Requirements and Verification

Requirements	**Verifications**
● Each piano key should send a distinct signal (13 total signals) to the Microcontroller Subsystem	● Check using the Arduino that each signal is received and distinct
● Each knob should send a variable voltage signal by adjusting resistance up to at least 8k Ohms	● Once we have a current decided, measure the max voltage through the potentiometer using a voltmeter. We can then divide this measurement by the current to figure out the maximum resistance.
● Octave changers and pitch bend wheel should change both the synthesized audio and the MIDI output by one octave or one whole tone either up or	● Verify functionality on the synthesized notes through examination of the speaker audio

down respectively.	● Verify proper MIDI functionality through examination of the MIDI messages output from the microcontroller

2.2.3 Microcontroller Subsystem

The Microcontroller Subsystem will collect data from the various sensors and potentiometers of the input system and process it. Once the data is processed, it will be sent off to the output subsystem, formatted in both MIDI and SPI. For our microcontroller, we have decided to use an Arduino Mega2560 ATmega2560 A000067, which has more than enough analog and digital input pins for all of the input sensors. We will program this microcontroller to take sensor data from the Input Subsystem and use this data to calculate the correct output data. The microcontroller will perform digital audio synthesis and translate this data into SPI to send to the DAC, and it will also generate MIDI messages based on the input to send to the USB port.

Table 3: Microcontroller Subsystem Requirements and Verification

Requirements	**Verifications**
● The microcontroller successfully performs digital audio synthesis and outputs the data in SPI form	● Sensor data is processed and converted to SPI by using an oscilloscope with a high-impedance multimeter. We check the SPI bus signals on the oscilloscope for different input signals to ensure that the differences are accurately reflected.
● The microcontroller successfully generates MIDI messages based on the input	● Connect the microcontroller to a Windows computer via USB. Using MIDI Ox, see if MIDI messages are being translated through the microcontroller onto the computer

2.2.4 Output Subsystem

The Output Subsystem will be able to take output data from the microcontroller and either output MIDI signals through USB or analog audio through a built-in speaker. There will also be a Digital to Analog Converter used in order to convert data from the Microcontroller Subsystem to analog audio. The DAC we have chosen is the PCM5252 from Texas Instrument. This DAC features a SmartAmp feature and programming capabilities. This converter will receive data from the Microcontroller Subsystem in SPI format and process this data to output to the speaker. We can also program the DAC with necessary filters to eliminate any unwanted frequencies or voltages from the output. The built-in speaker that we have chosen to output audio data from the DAC is the Micro Round 8 Ohm Mylar Speaker from Jameco ValuePro. For USB output, we will use an Allied Components International AUSB1-4600 USB-A 2.0 port. This USB port will be able to receive MIDI messages from the microcontroller and send these messages to the computer.

Table 4: Output Subsystem Requirements and Verification

Requirements	**Verifications**
● Speaker can produce sounds between 150Hz and 5000 Hz	● In a largely volume-free room, point speaker towards microphone that is plugged into laptop software. Use this to measure (with ~1% error) frequency response of speaker at lowest and highest setting.
● Speaker can produce sounds up to at least 60 dB at full volume.	● Computer software like TrueRTA or REW can test volume readings of our speaker very accurately, so we will do the same microphone testing with our driver to check the volume
● USB port sends decipherable MIDI signals to the connected computer device	● Using a Windows software called MIDI-Ox, we can check MIDI devices connected to a Windows computer via

	USB, press keys, and verify that signals are being communicated

2.3 Tolerance Analysis

One of our three main initial requirements is that the battery has to last three hours on a complete charge. This is going to require some clever engineering; we need to be able to provide power to all of the modules and ensure that the sound quality is not blunted as the charge on the battery decreases. By effectively analyzing the power requirements of each component, we can mitigate this constraint and provide enough power to each of the subsystems as necessary. As there are larger, much more expensive products with similar battery lives, this is a feasible process.

Table 5: Power Draw Calculation

Components	Arduino	Charge Indicator	DAC
Current Draw (A)	25m	200m	25m
Voltage Draw (V)	9	3.4	3.3
Power Draw	0.225W	0.68W	0.0495W

From this analysis, we see that at most, the components of this system will draw about 250 mA of current. With a battery capacity of 2600 mAh, this should result in a minimum battery life of about 10 hours. This gives us a lot of room to work with to reach our goal, fortunately. We just have to be sure not to go overboard. We are powering the arduino, the output DAC, and the charger indicator with a battery listed at 3.7 V.

Additionally, after looking at how to perform frequency response and volume response calculations, we realized that in testing the outputs of our keyboard for accuracy, such as the frequency and volume, there is a certain percent error [7]. This is because the room we measure in will undoubtedly have some background noise, and we do not have hundreds of dollars to spend on high tech microphones and computer software to give us very accurate results on our audio output. We can mitigate this as much as possible by combining low-frequency response of near field measurement (distance between speaker and microphone is as small as possible) with high-frequency response of far field measurement in a non-ideal room (that is as quiet as we can possibly make it). We will place the speaker on a stand roughly in the center of the room to minimize sound waves bouncing off the walls. We will use Ableton EQ eight software to measure the frequency oscillations, and as we might expect around a 1% error with this software due to mic latency and limitations of the program, we will take that into account when measuring our output readings and adjust as necessary.

We also have to take into account the fact that our product has to project the correct volume and pitches. Regarding the pitch, we feel like the best course of action would be to have our keyboard sound from c3-c6 (c3 is the lowest frequency it projects and c6 is the highest). C3 has a frequency of about 130.81 Hz and c6 is 1046 Hz [7]. If we include any more, we risk going outside of the frequency range that our speaker is able to project, and other notes might cause discomfort to the listener. Thus, while coding the Arduino, we will be careful to stick to this range of notes at first. Next, we need to worry about the volume. We can use a second sensor for the keys as velocity detection, so the speed at which the key is pressed determines the loudness or softness of the note. This will take care of any extraneous issues we might encounter when it comes to the volume and pitch of the notes.

2.4 Cost Analysis

The total cost of the project is the sum of the prices in table #2 which is $141.49, not counting the cost of labor. The labor cost per team member can be calculated with $40/hr * 2.5 * 60 hours. The resulting labor cost for the total 3 members comes out to be $18,000. Additionally, a 5% shipping fee and a 10% sales tax fee added to the total components' price results in a total components cost of $162.71. Thus, the total cost of the project is $18,162.71. We are estimating 60 labor hours each to complete the design, building, and testing portions of the project.

Table 6: Components Cost Table

Description	**Manufacturer**	**Quantity**	**Extended Price**	**Link**
ARDUINO MEGA2560 ATMEGA2560	Arduino	1	$48.40	Link
SWITCH PB DPDT 0.1A 30V	E-Switch	2	$0.84	Link
SWITCH ROCKER SPST 6A 250V	NKK Switches	1	$3.27	Link
2.6Ah, 3.7V, Li-Ion, 18650, JST	US Electronics Inc.	1	$9.49	Link
IC BATT MFUNC MULTI 1-12C 64QFP	Renesas Electronics America Inc	1	$11.64	Link
MAX632ACPA Step-up Voltage Regulator	Maxim Integrated	1	$7.14	Link
S-1212BC0-V5T2U Linear Voltage Regulator	ABLIC Inc.	1	$1.28	Link
10 SEGMENT LED BAR GRAPH - BLUE	SparkFun Electronics	1	$2.44	Link
1 PORT USB CONNECTOR	Allied Components International	1	$1.10	Link
Rotary Potentiometer - 10k Ohm, Linear	SparkFun Electronics	9	$9.45	Link
Black Knob - 15x19mm	SparkFun Electronics	9	$9.45	Link
Pitch Bend Wheel 5100042238	Roland	1	$19.95	Link
BQ25180 programmable linear charger	Texas Instruments	1	$2.78	Link
PCM5252 32-Bit Stereo Differential-Output DAC	Texas Instruments	1	$11.91	Link
Micro Round 8 Ohm Mylar Speaker 93 dB 1.5 Watt 1.5" Wires	Jameco ValuePro	1	$2.35	Link
Total Cost			$141.49	

2.5 Schedule

Table 7: Schedule for Project Progression

Week	**Task**	**Member(s)**
2/19-2/25	**Work** on Design Document **Work** on Teammate Contract **Order** Components **Begin** PCB Design	Everyone Everyone Everyone Everyone
2/26-3/4	**Write** Software for Key Functionality **Write** Software for Volume Knob Functionality **Write** Software for Pitch/Note Functionality **Breadboarding** if components arrive	Sujay David Richard Everyone
3/5-3/11	**Receive** Microcontroller, write Software for Translating into MIDI Data **Receive** and process all the data from input sensors **Build** power circuit with battery and switch	Sujay, Richard David Everyone
3/19-3/25	**Solder** the PCB components **Test** PCB	Richard Sujay
3/26-4/1	**Debug** Code for input and microcontroller, checking for proper functionality **Assemble** DAC for proper MIDI output **Test** speaker and audio output	Sujay, David Richard Everyone
4/2-4/8	**Debug** PCB switches, continue to debug code **Connect** with computers and analyze output for accuracy **Integrate** components and begin testing power	Sujay, David Richard Richard, Sujay
4/9-4/15	**Continue** testing/debugging I/O system as necessary **Prepare** for mock demo	Everyone Everyone
4/16-4/22	**Mock Demo** **Debug/Test** Subsystems	Everyone Everyone
4/23-4/29	**Final Demo**	Everyone
4/30-5/6	**Final Presentation/Paper**	Everyone

3 Ethics and Safety

The biggest safety concern is ensuring that we are following proper safety precautions while working with these electrical components. We need to make sure that we do not run everything (modules, pedals) through some sort of external mixer at once, as we could have stray voltage. Some voltage controlled filters such as low pass filters are used in synths to protect from this, and our DAC is programmable to add filters so that we can remediate this issue.

Another safety concern that this project poses comes from using a rechargeable Lithium-Ion battery. We will follow all of the specifications for this battery in order to preserve its health and the safety of the user as well as reduce the risk of battery failure or overheating.

There are a few documents that we can use as a resource to help us abide by the safety guidelines that we have set. As we are using a lithium ion battery, we will employ the Harvard Lithium-Ion Battery Safety Guidelines Document [5] which explains the dangerous properties and safety measurements. We will not solder near the battery and keep from external heat sources to avoid thermal runaway. We will create housing that protects the battery from other heat sources to prevent injury to developers and users.

We will make sure to avoid any safety hazards and follow ethical guidelines as described by the ACM code of ethics, including building and designing robust and usable systems [1]. As such, we will robustly test to make sure that our MIDI output is correct and that the keyboard is functional (correct input and output) to ensure that it is presentable to consumers. Additionally, as professionals, we will work towards being honest and trustworthy with our ideas and presenting them in a fair, honest capacity, as specified by the ACM code of ethics [1]. We will also take steps towards mitigating harm as much as possible, by adding filters to our power sources and using the current voltage/mAH values to ensure no stray voltage and electrical hazards.

Additionally, we have to make sure that we are allowing ideas and input from one another and focusing on delivering for our target audience, taking action not to discriminate and prevent fair participation. Lastly, as described in the ACM code of ethics, we will accept professional review at all stages, taking any constructive criticism and working to improve as a result [1]. We will each hold each other accountable in order to ensure that we all abide by the safety and ethical standards of this school and the state.

Arduino Uno: https://www.digikey.com/en/products/detail/arduino/A000066/2784006

Pushbutton Switch:
https://www.digikey.com/en/products/detail/e-switch/TL2202EEZB-GRN-ACT/16020087
Potentiometer: https://www.sparkfun.com/products/9939
Black Knob: https://www.sparkfun.com/products/9998
Pitch bend wheel: https://www.syntaur.com/Part-10391-Pitch-bend-wheel-assembly-Roland-

Rocker Switch:
https://www.digikey.com/en/products/detail/nkk-switches/CWSB11AA2F/671494
Battery:
https://www.digikey.com/en/products/detail/us-electronics-inc/USE-18650-2600PCBJST/15781482
Battery Management IC:
https://www.digikey.com/en/products/detail/renesas-electronics-america-inc/ISL94212INZ/3877119
Charger IC: https://www.ti.com/product/BQ25180#features
Voltage Regulator:
https://www.digikey.com/en/products/detail/ablic-inc/S-1212BC0-V5T2U/14664155
LED Bar Graph:
https://www.digikey.com/en/products/detail/sparkfun-electronics/COM-09937/7319641

DAC:
https://www.ti.com/product/PCM5252#:~:text=The%20PCM5252%20is%20a%20monolithic,in%20a%20small%20QFN%20package.
Speaker:
https://www.jameco.com/z/40R-Jameco-Valuepro-Micro-Round-8-Ohm-Mylar-Speaker-93-dB-1-5-Watt-1-5-Wires_2234134.html
USB:
https://www.digikey.com/en/products/detail/allied-components-international/AUSB1-4600/14321121

References

[1] “ACM Code of Ethics and Professional Conduct.” *Code of Ethics*, Association of Computing Machinery, 2018, https://www.acm.org/code-of-ethics.

[2] Texas Instruments, “PCM5252 Purepath™ Smart Amp 4.2-VRMS DirectPath™, 114-dB Audio Stereo DifferentialOutput DAC with 32-bit, 384-kHz PCM Interface,” PCM5252 Datasheet, November 2014.

[3] Sparkfun, “Model No.: YSLB-10251B5-10,” 10 Segment LED Bar Graph - Blue Datasheet.

[4] Adomaviciute, Lukas, et al, “Bluetooth Enabled E-Walker,” ECE 445 Project Proposal, September 2022.

[5] *Laboratory safety guideline - ehs.harvard.edu*. (n.d.). Retrieved February 23, 2023, from https://www.ehs.harvard.edu/sites/default/files/lab_safety_guideline_lithium_ion_batteries.pdf.

[6] Open Music Labs, “How does a MIDI keyboard work?” 2016. http://www.openmusiclabs.com/learning/digital/input-matrix-scanning/keyboard/index.html.

[7] *Analog | Embedded Processing | Semiconductor Company | ti.com*. (n.d.). Retrieved February 24, 2023, from https://www.ti.com/lit/an/slaa641/slaa641.pdf

[8] *Tuning*. Frequencies of Musical Notes, A4 = 440 Hz. (n.d.). Retrieved March 24, 2023, from https://pages.mtu.edu/~suits/notefreqs.html