Affordable Portable MIDI Keyboard Synthesizer

Sujay Murali
Richard Engel
David Gutzwiller

TA: Akshat Sanghvi

9 Feb 2023

Table of Contents

1 Introduction

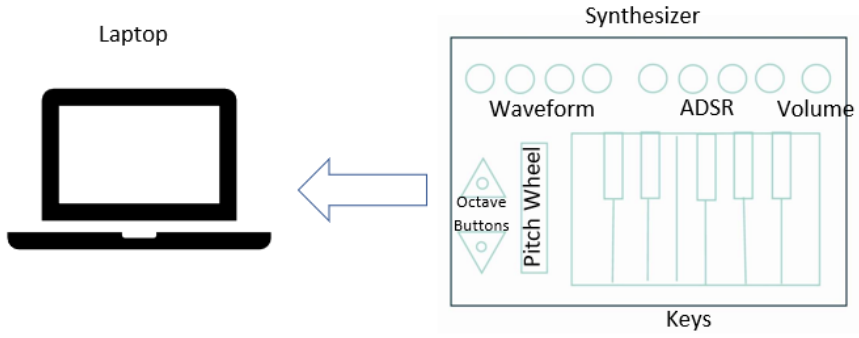
1.1 Problem

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.

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.

1.3 Visual Aid



1.4 Requirements List

Input Functionality

Every form of input should be fully functional. Each key must both register which note is played and the velocity that the key is pressed. 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

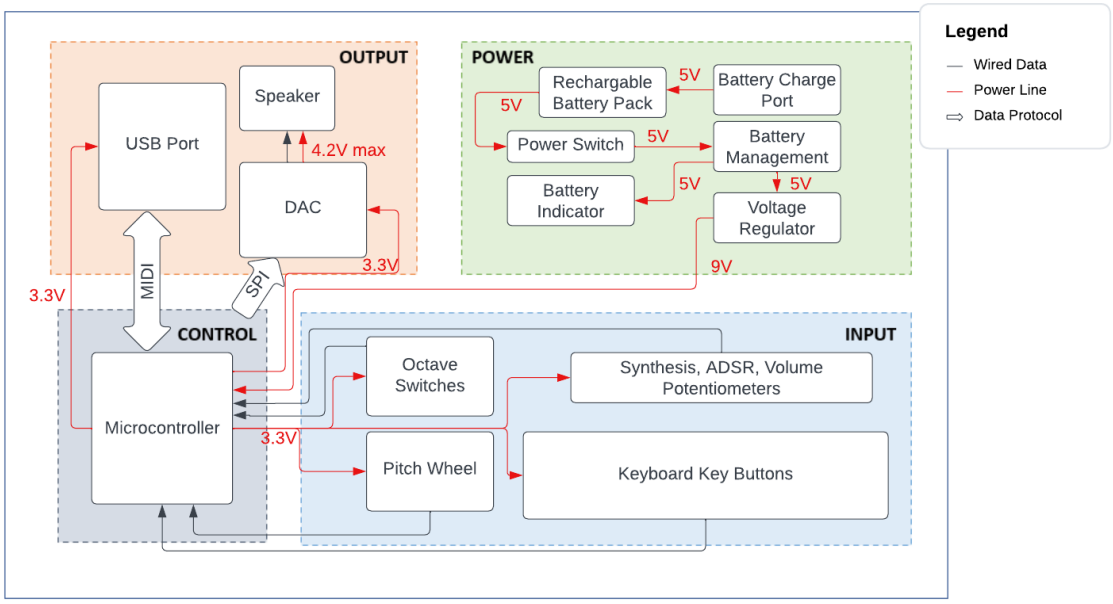
Output should be working properly. The built-in speaker should be able to output analog sound that is digitally synthesized within the keyboard. The USB port should be able to transmit MIDI messages from the keyboard, and this should allow the keyboard to communicate with DAWs on laptop or desktop devices.

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



2.2 Subsystem Overview

2.2.1 Power

- The power subsystem consists of a rechargeable battery that can deliver power to the Arduino Uno microcontroller. The battery will be started by a power switch. The charge will then be passed along to a battery management system that will monitor the conditions of the battery to make sure it is operating safely and properly. Power is then passed to an LED bar graph which will indicate battery life, and a voltage regulator which will control the voltage that is passed on to the microcontroller.

  2.2.1.1 Rechargeable Battery Pack

  - The battery that we have chosen to use for this project is the USE-18650-2600PCBJST from US Electronics, which is rated at 2600 mAh and 3.7V. This will provide power to the system.
  - *Requirements: Supply a steady voltage to the system in order to power everything.*

2.2.1.2 Battery Charge Port

- For the battery charging port, we are choosing the BQ21080 linear charger IC from Texas Instruments. This is tolerant of up to 25V input and can controls the voltage delivered to the battery in order to maximise battery life while managing the device temperature.
- *Requirements: Charge the battery while maximizing its operational life and monitoring the battery temperature.*

2.2.1.3 Battery Management System

- The battery management system will control the battery to ensure that it is operating at a safe voltage. The system will also include a failsafe in case the battery overheats. We have yet to decide on a specific battery management system, but we want to make sure it meets these requirements.
- *Requirement 1: Regulate the battery's operation within 5V.*
- *Requirement 2: Trigger a failsafe to shut down off the battery if it overheats or acts erratically.*

2.2.1.4 Voltage Regulator

- The voltage regulator will take the battery voltage input and output and maintain a constant voltage to the Arduino Uno.
- *Requirement: Maintain a constant 9V output to the microcontroller.*

2.2.1.5 Power Switch

- The power switch will connect the battery to the battery management system. When it is switched on, the circuit will be complete. When it is not, the circuit won't be, and the system will be off.
- *Requirement: Allow the circuit to run when in the on position, turn off the circuit when in off position.*

2.2.1.6 Battery Life Indicator

- We have chosen to use an LED bar graph indicator from Sparkfun, the COM-09937 ROHS. This 10-segment LED bar graph will indicate the level of charge that the battery has left.
- *Requirement: The lithium battery charge is displayed on the LED in 10% increments.*

2.2.2 Input

- The input system consists of a collection of buttons and potentiometers, all with their own function within the system. All of the data collected from these sensors will be sent off to the control system for processing.

  2.2.2.1 Keyboard Key Buttons

  - The keyboard layout is made up of one octave worth of keys, or 13 individual keys. Each key will be pressed down by the user, and it will activate two buttons that are spaced apart. The buttons will not only sense which key is pressed, but the timing of when each button is pressed by the key will also be used to determine the velocity that the key is pressed.
  - *Requirement: The buttons for each key should both effectively detect when the key is pressed and also effectively send timing data so velocity can be calculated.*

  2.2.2.2 Synthesis, ADSR, Volume Potentiometers

  - Besides the keyboard, there will be a number of knobs that will act as potentiometers to control other aspects of the instrument. There will be four knobs controlling the waveforms used for synthesis, four more knobs controlling the ADSR envelope of these notes, and one knob controlling output volume. All of these will change voltage and send this information to the microcontroller for processing.
  - *Requirement: Each knob should be functional and send the correct signal to the corresponding function in the microcontroller.*

  2.2.2.3 Octave Switches

  - There will also be two additional octave changer buttons, that will each have a button component that sends data to the microcontroller. This data will communicate whether to change the pitch of the keyboard up or down by an octave. This should work both for the analog audio output and the digital MIDI output.
  - *Requirement: Both octave changing buttons function as they should. The up button changes the audio and MIDI data up by an octave, and the down button changes audio and MIDI down by an octave. This should be functional within reason, it wouldn't make sense to try and produce sounds outside the audible range of hearing.*

  2.2.2.4 Pitch Wheel

- The final form of input will be a pitch bend wheel which will be bought off of the shelf. This pitch bend wheel will also act as a potentiometer, bending the pitch of the current note either up or down by a maximum of a whole tone.
- *Requirement: This should successfully result in a note being smoothly "bent" by up to a whole tone in either direction.*

2.2.3 Control

- The control unit 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.

2.2.3.1 Microcontroller

- The microcontroller will be driven by an Arduino Uno, which will collect data from all of the switches and potentiometers in the input subsystem. The Uno should also power the input subsystem. The Arduino Uno will then be able to process this data and send it off to both a USB port via MIDI and a Digital to Analog Converter via SPI.
- *Requirement 1: Collect data from the keyboard buttons and calculate key velocity*
- *Requirement 2: Collect data from the potentiometers and calculate the respective parameter values for output.*
- *Requirement 3: Take 9V input and provide 3.3V output to the input subsystem*
- *Requirement 4: Format the stored data into both MIDI messages for the USB output and SPI for the audio output.*

2.2.4 Output

- The output system 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 is also a DAC being used to convert data from the microcontroller to analog audio.

2.2.4.1 Digital to Analog Converter

- The DAC will receive data from the microcontroller subsystem in SPI format, and it will be able to process this data to output to the speaker. The specific DAC that we have decided to use is Texas Instrument's PCM5252. This due to its SmartAmp and programming capabilities. We can program the DAC with necessary filters as well to eliminate any unwanted frequencies or voltages from the output.

- *Requirement 1: Receive SPI messages from the microcontroller and output this to the speaker at a volume decided by the volume knob.*
- *Requirement 2: Implement any necessary filters for voltage and frequency control.*

2.2.4.2 Built-In Speaker

- The output speaker should take analog sound data from the DAC and be able to project it at an adequate value. The speaker component we've selected is the Micro Round 8 Ohm Mylar Speaker from Jameco ValuePro.
- *Requirement: Successfully output analog sound delivered from the speaker down to 100Hz.*

2.2.4.3 USB Port

- This will be a USB-A port that will allow the user to connect the keyboard to a computer. This USB port will be able to receive MIDI messages from the microcontroller and send these messages to the computer.
- *Requirement: Successfully deliver MIDI messages from the microcontroller to communicate with DAWs on the user's computer or laptop.*

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 1: 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.

3 Ethics and Safety

The biggest safety hazard is electrocution and making sure 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, 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.

We will make sure to avoid any safety hazards and follow ethical guidelines as described by the ACM code of ethics. As professionals, we will work towards being honest and trustworthy with our ideas and presenting them in a fair, honest capacity. 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. 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.

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.