Project Proposal

MIDI Music Box

Jeremy Lee [jeremyl6]

Sean Liang [shliang2]

Tyler Shu [tmshu2]

Introduction

Problem

Making music using MIDI Devices commonly found in modern day music devices can tend to be inaccessible to beginners - with a large amount of equipment (and subsequently research) required to listen to music created on MIDI Devices, which typically do not include any forms of playback options, access to music making becomes difficult. Creating a Cost AND User-Friendly attachment specifically geared towards playback would eliminate the need to purchase expensive software and hardware needed to process and translate MIDI signals into usable data, enabling users to more easily break into music making at any level.

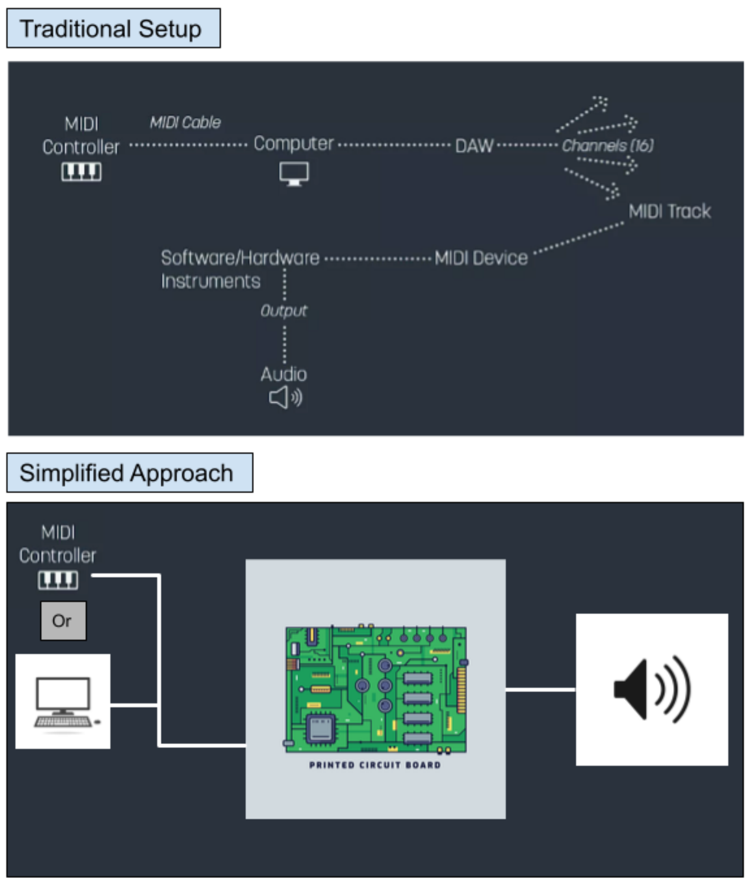
Solution

We want to implement a form of music box (playback device) that would be able to interface with standard USB ports to either play a recording provided by the user via a file or play signals live from a MIDI device. The project will achieve such using four subsystems, connected together to receive, process, and output relevant data:

1. **The Signal Processing Processing and Conversion Subsystem**
   This subsystem will primarily be responsible for taking in MIDI data via USB input directly into the microcontroller/processor, which will be a Raspberry Pi 3. The data will be processed and translated into a digital signal to be passed into a Digital-to-Analog converter, which will convert the Digital Signal into an Analog Signal for usage within the Output Control Subsystem.
2. **The Power Output and Conversion Subsystem**
   This subsystem will be primarily responsible for powering the rest of the board. The subsystem will employ conversion techniques to provide multiple levels of voltage in order to power various modules throughout the project.
3. **The Output Control Subsystem**
   This subsystem will be primarily responsible for providing variability to the output for user control. The subsystem will provide a method for the user to determine what type of signal will be output, and provide a method for the user to adjust the intensity of the noise produced by the project. Tactile inputs (like switches and dials) will be provided to achieve such.
4. **The Output Subsystem**
   This subsystem will be primarily responsible for amplifying the sound signals into perceivable noise. The subsystem will consist of an amplifier circuit connected to the Output Control Subsystem, which will determine the amount of amplification done to the waveform. The subsystem will also produce the waveform via a speaker.

Visual Aid

The following visual aid provides an idea of what the device will look like when utilized. The image contrasts the traditional method of setting up a MIDI system with our simplified version. In the image, the device is represented by the "Printed Circuit Board":



High-Level Requirements

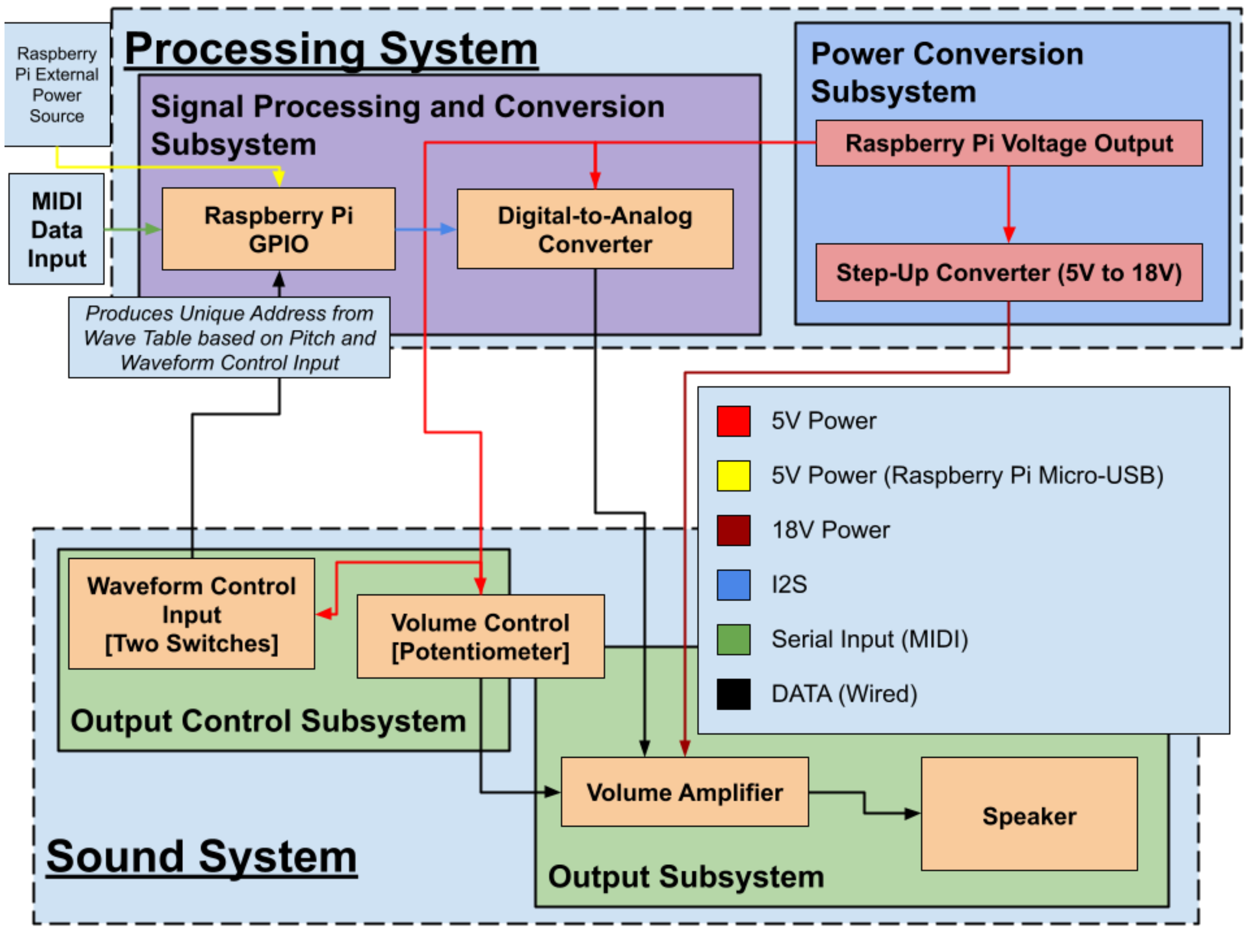
The project must be able to exhibit the following characteristics in order to solve the problem. These requirements are included to provide realistic expectations and goals for the project. The requirements are listed below:

1. The project must be able to synthesize at least **four different tones** (waveforms), including (but not restricted to): **Triangle, Square, Sine,** and **Sawtooth**.
2. The project must be able to produce at least **eight note polyphony** - that is, the project must have the ability to combine at least eight different musical parts together to form a harmony.
3. The project must be able to produce **pitch** in the frequency range **C2 (65.4 Hz) - C5 (523.5 Hz)**, with additional capability of reaching up to **15,000 Hz** including harmonics for synthesized tones.
4. The project must be able to drive a speaker up to **20W**.

Design

Block Diagram

The following is the Block Diagram for the project described. The board will consist of two main systems and four subsystems. A legend is included in the Block Diagram dictating the types of connections expected within the project.



Subsystem Overview

The project will include four subsystems - the **Power Output and Conversion Subsystem (POCS), Signal Processing and Conversion Subsystem (SPCS), Output Control Subsystem (OCS),** and **Output Subsystem (OS)**. Each subsystem and their relationship to the other subsystems will be provided below. Further quantitative detail regarding each subsystem will be provided in the **Subsystem Requirements** section below.

Power Output and Conversion Subsystem

The **Power Subsystem** will be primarily responsible for generating the power necessary to drive the operations of the device. Components of the subsystem include the Raspberry Pi 3 and a Step-Up Converter (such as the QA053C DC DC CONVERTER). The subsystem will utilize the voltage pins on the Raspberry Pi to provide voltage that will either be directly supplied to other components, or stepped up to a higher voltage via a Step-Up Converter like the QA053C, and then directed to the appropriate components. This subsystem connects to all other subsystems in the design - lower voltage components include the DAC and Waveform Control Switches, and higher voltage components include the Volume Amplifier and Speaker.

Signal Processing and Conversion Subsystem

The **Signal Processing and Conversion Subsystem** will be primarily responsible for receiving the external MIDI input and translating the input into the I2S protocol required by the Digital-to-Analog Converter. Components of the subsystem include the Raspberry Pi and a DAC (such as the PCM1753 IC DAC/AUDIO 24BIT 200K 16SSOP). The subsystem will utilize the processing power of the Raspberry Pi and its I/O ports to receive the MIDI input, and utilize further processing and its GPIO Pins to output I2S data to the DAC, such as the PCM1753. The subsystem will be powered by the POCS and Raspberry Pi External Power Supply, take input from an external MIDI Data Source via the onboard Serial Connection, take input from the Waveform Control Input located within the OCS, and output to the Volume Control located within the OCS.

More precisely, the MIDI input provides information regarding the state of a note - whether the note is on or off, its pitch, and its velocity. Using the EEPROM on the Raspberry Pi, samples of the four previously-described waveforms are preloaded with bit resolution high enough for audio-grade sampling. The Raspberry Pi will calculate how many samples are needed to jump throughout the wavetable to find the sample representing the target waveform. To achieve polyphony, several sample values are obtained and linearly added, which is then sent as a digital signal in I2C format to the DAC, where it is then processed into an Analog Signal to be sent out to the Volume Control Component of the OCS. Additional processing such as filter application or value attenuation may be implemented depending on the project's progression.

Output Control Subsystem

The **Output Control Subsystem** will be primarily responsible for determining primary aspects about the waveform produced, specifically the type of waveform produced by the SPCS and the volume of noise produced by the OS. Components of the subsystem include a pair of switches (such as the 100SP1T1B1M1QEH SWITCH TOGGLE SPDT 5A 120V) and a potentiometer (such as the 3310Y-001-103L POT 10K OHM 1/4W PLASTIC LINEAR). The subsystem takes physical user input via its switches, switching between high and low, to determine which of the four waveforms to produce, with each of the combinations of the switches representing a specific waveform. Additionally, the subsystem will also take input from the DAC within the SPCS and pass the input (as an analog signal) through the potentiometer, where the waveform will be dampened by the amount of resistance set by the user within the potentiometer and passed over to the Volume Amplifier located within the OS. The subsystem is powered by the POCS.

Output Subsystem

The **Output Subsystem** will be primarily responsible for amplifying and producing the waveform directed by the external MIDI Input. Components of the subsystem include the Volume Amplifier (such as the LM386 IC AMP CLASS AB MONO 700MW 8DIP) and a Speaker (such as the WSPH-1805W SPEAKER 8OHM 1W TOP PORT 86DB). The subsystem will take input from the Volume Control located within the OCS which will come in a waveform, amplify the waveform and transmit the waveform to the speaker, where the waveform will subsequently be played out loud. The subsystem is powered by the POCS, utilizing its higher voltage output to enable the amplifier to amplify at higher levels.

Subsystem Requirements

The quantitative requirements of each subsystem are listed below. A description is included for each previously mentioned subsystem.

Power Output and Conversion Subsystem

The **Power Output and Conversion Subsystem** will be the primary power provider of the overall project. The subsystem powers all other subsystems in the project, making it essential to the operations of the project. The subsystem will utilize the Raspberry Pi's onboard GPIO 5V Pin to power its various components - the 5V pin will power the DAC within the SPCS, Waveform Control Input within the OCS, and be passed to the previously mentioned Boost Converter, which will convert the voltage from 5V to 15V in order to power the Volume Amplifier and Speaker contained within the OS. The essential requirements of the subsystem are included below:

- The subsystem should be able to output both 5V of power across one line, and 18V of power across another line.
- Failure or insufficient minimum voltage will result in the failure of the DAC within the SPCS, as well as the Waveform Control Input and Volume Control within the OCS.
- Failure of insufficient maximum voltage will result in the failure of the Volume Amplifier and Speaker within the OS. Without sufficient voltage, clipping may occur within the amplifier causing the sound produced to be inaccurate.

Signal Processing and Conversion Subsystem

The **Signal Processing and Conversion Subsystem** will be the primary processing unit used to interpret our input data, converting it from a Digital Waveform to an Analog Waveform. The input MIDI data will indicate to the processor the status of a given note (whether the note should be on or off), as well as 7-bit values for pitch and velocity. Using the EEPROM on the Raspberry Pi 3, a 1024 samples of each waveform will be preloaded into the Raspberry Pi's memory, with 16-24 bit resolution, which will require up to 12.288 kB of storage. The processor will employ a Ping-Pong buffer, which will simultaneously read and process MIDI input via the Serial Port and output I2S data to the DAC. For polyphony, up to 8 sample values would be obtained and then linearly added, sent as a digital signal into the DAC.

The sampling rate of the DAC was determined to be 32Khz, with the need for a 6.155MHz Clock to be used as the System Clock and a 1.536MHz to be used as the Bit Clock. These clock values were determined by analyzing the corresponding Datasheet for the PCM1753. The DAC will also need to contain 16-24 bit resolution, matching the resolution output by the Raspberry Pi, as well as requiring the output to be in I2S format. The SPCS will receive 5V power from the POCS, as well as power the Raspberry Pi via an external source which will supply the Raspberry Pi's Micro-USB port. The DAC will need to be able to output waveforms with frequencies up to 15KHz to the Volume Control of the OCS via wired data, as dictated by the High-Level Requirements. The essential requirements of the subsystem are included below:

- The Raspberry Pi must be able to output 5V via its 5V pin.
- The Raspberry Pi must be able to read Serial input from its Serial Ports utilizing the MIDI Protocol, look up a corresponding wave dictated by the MIDI Input, and output a waveform in I2S format
- The DAC must be able to read data in I2S format from the Raspberry Pi and output an analog waveform corresponding to the MIDI Input Data at the rate dictated by the System Clock, Sampling Frequency, and Bit Clock.
- The DAC must be able to output waveforms with frequencies within the target range, which can be anywhere up to 15KHz.

<u>Output Control Subsystem</u>

The **Output Control Subsystem** will be the primary interface for the user to interact with the project's user, enabling the user to determine the type of waveform synthesized, as well as giving the user control over the level of volume output by the speaker. Analog signals will be received from the DAC within the SPCS and connected to the Volume Control, where the signal will be fed through a potentiometer and output to the Volume Amplifier within the Output Subsystem via wired connection. Additionally, the Waveform Control Input will output data via wire to the GPIO Pins of the Raspberry Pi with both a high and a low voltage, influencing the type of waveform output. The subsystem also takes 5V power input from the POCS. The essential requirements of the subsystem are included below:

- The Waveform Control Input must be able to influence the type of waveform output by the SPCS, and be able to represent all four waveforms defined by the High-Level Requirements.
- The Volume Control must be able to receive sound waveforms from the SPCS's DAC and adjust its amplitude accordingly. The Volume Control must be able to completely reduce the sound or enable the waveform to pass at maximum amplitude.

<u>Output Subsystem</u>

The **Output Subsystem** will be the primary sound production unit within the project. The subsystem will contain the Volume Amplifier and a Speaker. The Volume Amplifier must be able to take input via wired data from the Volume Control located in the OCS, and amplify the waveform. Additionally, the volume amplifier must be capable of outputting the inputted waveform with a gain ranging between 20 and 200, as dictated by the amplifier's design. The speaker must be able to output frequencies within the defined range. Notably, due to the Amplifier's higher voltage requirements, the subsystem is powered by the POCS's 18V connection. The essential requirements of the subsystem are included below:

- The Volume Amplifier must be able to receive Analog Waveforms via wired data from the Volume Control within the OCS, and output an amplified waveform with gain running between 20 and 200, to the speaker.
- The Speaker must be able to output frequencies within the range defined by the High-Level Requirements, which reach up to 15Khz.

Tolerance Analysis

The usage of the Raspberry Pi 3 innately poses a risk to successful completion of the project - without the operations of the microprocessor, the project is unable to take its desired input. However, with the known documentation of the MIDI Protocol as well as inclusion of several Serial Ports and GPIO Pins, it's likely that the microprocessor will be able to read the incoming data in either form. The microprocessor likely contains capability to read at the desired speeds as well - MIDI Protocol transmits at a rate of 31.25 Kbps (Wright, 2023) with 8-bit serial transmission: the Raspberry Pi offers speeds typically in the range of < 1MHz - 25MHz (Dsadmin, 2020), indicating that from a speed perspective the microprocessor is fast enough to read the Input and Output at the rate dictated by the DAC and the MIDI Protocol. The Raspberry Pi 3 also contains 1GB of Internal RAM and contains a dedicated 5V Pin for output (*Raspberry pi 3*), indicating that the microprocessor is capable of storing the appropriate waveforms and their samples, as well as powering the rest of the board as intended.

Ethics and Safety

While not particularly prevalent, Ethics and Safety issues continue to be a concern during the development of the project, as well as after its completion. Possible breaches of Ethics may stem from the standards of integrity and ethical conduct practiced by the engineers working on the project. According to the IEEE Code of Conduct, engineers should "commit [themselves] to the highest ethical and professional conduct and agree to uphold the highest standards of integrity, responsible behavior, and ethical conduct in professional activities". In the context of such, the personal integrity of the engineers as well as the conduct must be monitored, as requirements and deadlines may drive project members to breach the provided ethics through cutting corners. To avoid this, proper documentation and group accountability should be implemented, as well as a sufficient design plan and timeline should be implemented to discourage members from breaching the code of ethics.

A Safety issue that may stem from the completion of the project may be the misuse of the project to damage the physical wellbeing of another person - a person may misuse the project and damage their own hearing or another's, either through the prolonged exposure to the maximum volume the device can output, or through the installation of a much more powerful speaker in the place of the intended device. Methods to prevent this are innate in the device's design - the project is not designed to produce more than the allowed volume range to prevent the user from damaging their own hearing. OSHA regulations state, "Exposure to loud noise kills the nerve endings in our inner ear…the employer must reduce noise exposure through [various methods]…to attenuate occupational noise", indicating that exposure to loud noise should be avoided if possible.

An additional safety concern may stem from the possibility of electrocution - though highly unlikely, the operation of an electronic component without sufficient voltage regulation may result in the electrocution and subsequent injury of an individual. Voltage regulation and proper securement of the project will prevent the possibility of such, and subsequently will be included in the final design.

The possibility of ethical and safety issues will always be present in the development process of projects, as well as after their development. By enforcing group accountability amongst members and working with ethics in mind, these issues can be addressed as they arise, in the proper setting and with the proper actions. Ethical and Safety issues will continue to be assessed with the development of the project.

Citations

Dsadmin. (2020, June 29). *GPIO I/O clock speed*. Digital Shack. https://digitalshack.org/gpio-i-o-clock-speeds/#:~:text=Absolute%20Max%20speed%20on%20Raspberry,and%20most%20cases%20much%20less.

Raspberry pi 3. Components101. (n.d.). https://components101.com/microcontrollers/raspberry-pi-3-pinout-features-datasheet

Wright, G. (2023, January 24). *What is MIDI (Musical Instrument Digital Interface)? – TechTarget definition*. WhatIs.com. https://www.techtarget.com/whatis/definition/MIDI-Musical-Instrument-Digital-Interface#:~:text=The%20MIDI%20protocol%20uses%208,data%20rate%20and%20is%20asynchronous.