Stepper Symphony

Corinna Chen, Maria Wang, Dev Bhatia

Abstract:

Stepper Symphony explores the integration of stepper motors and a programmable microcontroller to create an unconventional musical instrument. By controlling the speed and frequency of the spinning motors, the project produces musical compositions. Our system includes a preprocessing pipeline that converts single-channel MIDI files into control signals for the motors, which are mounted on a custom-designed chassis to optimize sound output. The final product demonstrates the potential for blending engineering and art in sound design.

Introduction:

Stepper Symphony seeks to explore an unconventional approach to creating music - using stepper motors as the sole sound-producing devices. The concept stems from the observation that stepper motors, when operated at specific frequencies, emit tones that can be directly correlated with musical notes. The chassis design for this project highlights the unique usage of motors and calls for the audience to think outside the box when it comes to artistic mediums.

The primary objective of this project is to develop a programmable array of motors capable of performing simple musical compositions. Key challenges include converting MIDI data into motor control signals, ensuring accurate pitch reproduction, and creating a structurally sound chassis to house the motors and wiring. Ultimately, the project aims to merge engineering with art, demonstrating how mechanical devices (motors) can serve as unique musical instruments.

Methodology:

Hardware Design

Hardware Implementation began with setting up a single stepper motor using an Arduino Uno, an A4988 stepper motor driver, and a breadboard. Initial testing revealed issues with the power supply and driver functionality. The motor, requiring 12-24v, initially stuttered when powered by a 5v USB connection. After switching to the ACT Lab power supply, the team identified wiring errors and replaced a faulty motor.

Once a single motor was operational, we scaled up the system to four motors. Each motor was mounted on a custom-designed truck-shaped wooden chassis, which housed the wiring, breadboards, and Arduino. The chassis design focused on stability and sound amplification by ensuring that the motors had ample surface to vibrate against. The motor mounts were crafted from bent sheet metal, introducing challenges in achieving precise fits due to material variability.

Software Design

The software workflow involved three primary stages:

1. Midi file processing which included:
    a. Converting polyphonic MIDI to monophonic MIDI
    b. Generating a CSV file containing three columns: MIDI note, time-on, time-off
2. Frequency Conversion
    a. MIDI frequencies converted into sound frequencies, which correlates to the motor speed
3. Motor Control

a. The arduino code implemented a round-robin system to distribute notes across the four motors. By varying the motor speeds, we achieved specific pitches corresponding to musical notes.

The source code is publicly available at https://github.com/dbhatia00/motorSymphony.

Assembly and Testing

The team tested the system using simple tunes like *Twinkle Twinkle Little Star*, sourced from popular MIDI websites (https://bitmidi.com/ and https://onlinesequencer.net/). Early iterations faced difficulties due to sparse and unclear documentation. Through iterative debugging, we refined the timing and pitch output. Scaling the system to four motors required additional effort to synchronize motor outputs and ensure consistent sound quality.

Our project involved both hardware and software components, which were often completed in parallel. Our hardware journey began with Corinna trying to set a motor up using a stepper driver, a breadboard, and an Arduino. After getting it wired up and creating some code to test that the motor can spin, she tried to run it but the motor would not spin. We initially believed that this was due to a lack of power as the power source was her computer which supplies 5 volts while the motor required 12 - 24 volts. We then met up and tested using a power supply and got the motor to stutter. After moving to the ACT lab, the team then found that some of the wiring was done wrong and the stepper motor was dead. Once corrected, the motor spun successfully. As this was occurring, Maria and Dev worked on the software side. Maria used the CSV file that Dev outputted to convert the MIDI notes into frequencies, and finally output it into a txt file that contains a C struct and can then be easily pasted into the arduino code and make playing and processing easier.

After getting one motor to spin and create sound, Dev and Maria worked on getting it to be able to fully play a simple tune, Twinkle Twinkle Little Star. This process was complicated because the few sources that they were referencing had messy and confusing code causing confusion in the relationship between the pausing and pitch output. They ultimately overcome this obstacle by correlating the length of pauses between signals and pitches, which successfully played out. While this was occurring, Corinna expanded the hardware set up by wiring 3 more motors and stepper drivers to the breadboard and worked on the design of the chassis to house all the electrical components. To design the chassis, she first completed a rough sketch of the chassis with estimated dimensions of the truck. Then using CAD, she was able to alter the design and dimensions, while taking into account the width of the wood and ensuring that the portions looked accurate.

After the motor was set up to accommodate all four motors, Dev worked on scaling a song from one motor to 4 motors. To finish up the hardware side, Maria and Corinna worked on creating the housing for our motors. Corinna designed the truck, used the band saw to cut the pieces of wood, and glued them together. They both worked on the motor holders which was difficult as bending sheet metal is not the most precise way to ensure the motors fit perfectly as

we had to take into account the thickness of the metal and small variations in motor size. After getting 4 holders, Corinna drilled holes into the wood and used nuts and bolts to attach the motor holders to the chassis and threaded the wires from the motors inside of the truck. We then tested the design to hold the 4 motors, an Arduino, and a breadboard with appropriate connections. After successful sound output from the four motors on the chassis design, Maria painted the truck in a ghibli theme. Finally, we were successful in getting two songs to play on the truck: "Jingle Bells Rock" and the "Super Mario Bros. Theme Song".

Results:

The project successfully demonstrated the ability to produce recognizable musical compositions via the stepper motors. Key findings include the following:

1. Melody reproduction: The motors achieved sufficient accuracy for single-channel monophonic melodies. Higher-pitched notes required scaling down due to hardware limitations. The inclusion of note timing allowed for consistent rhythm reproduction.
2. Scalability: The system performed reliably with four motors, though adding more motors may require a more complex power management strategy.
3. Structural Design: The truck chassis provided an effective housing for the components without dampening the volume.

Figures:

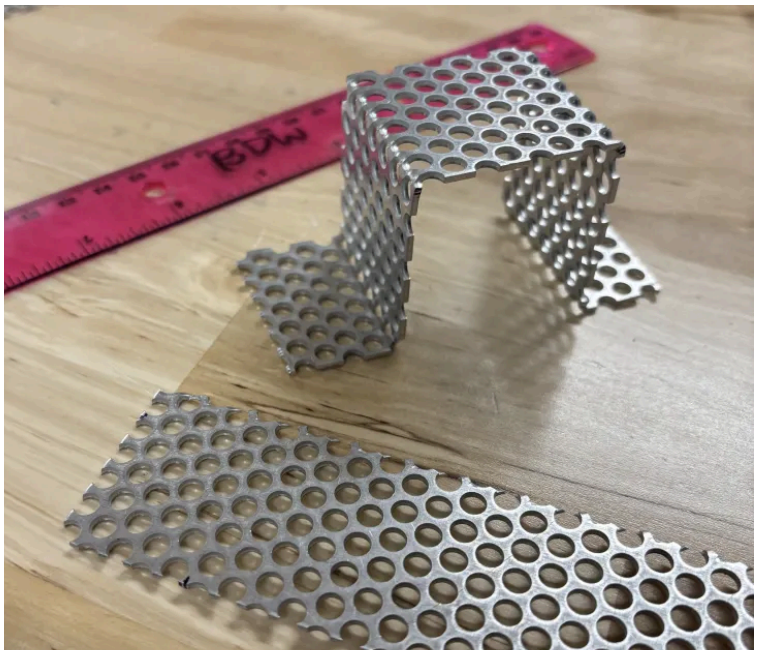

Figure 1: Motor holders made out of sheet metal. The sheet metal was cut using a metal shear and bent using a finger breaker.

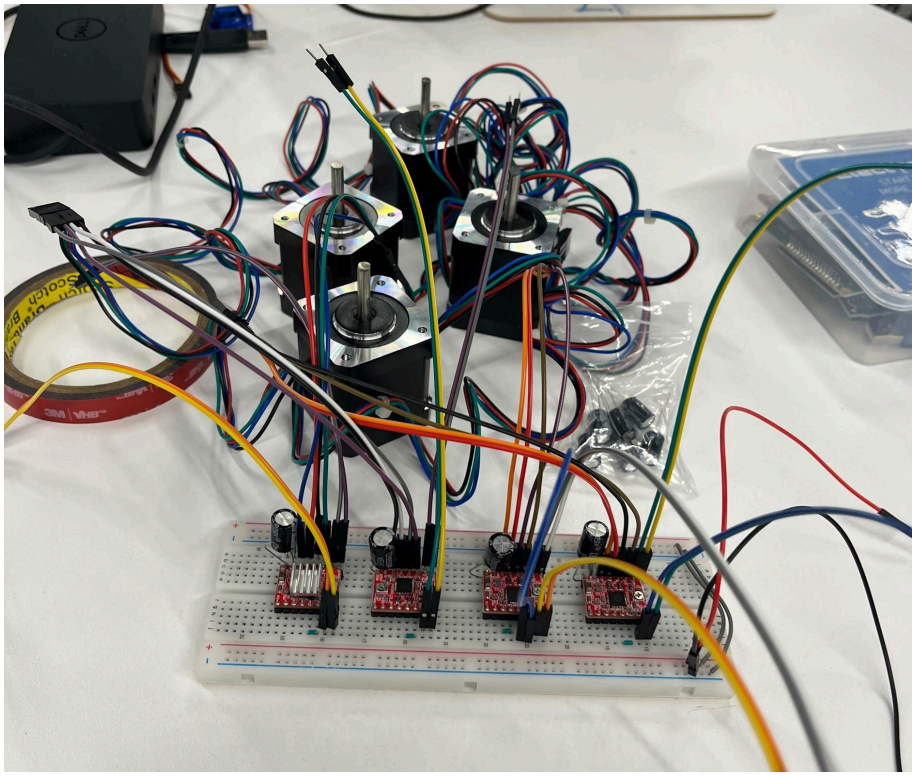

Figure 2: An image of the breadboard set up after scaling one motor and stepper driver up to 4 motors and stepper driver.

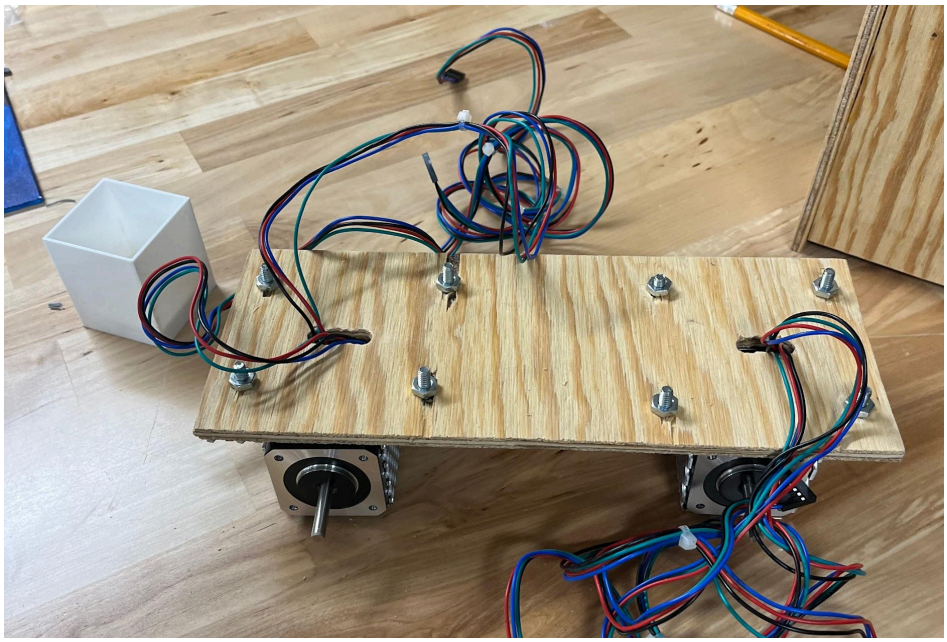

Figure 3: Image showing how the motor wires were fed up into the chassis and how the motors were attached to chassis through nuts and bolts.

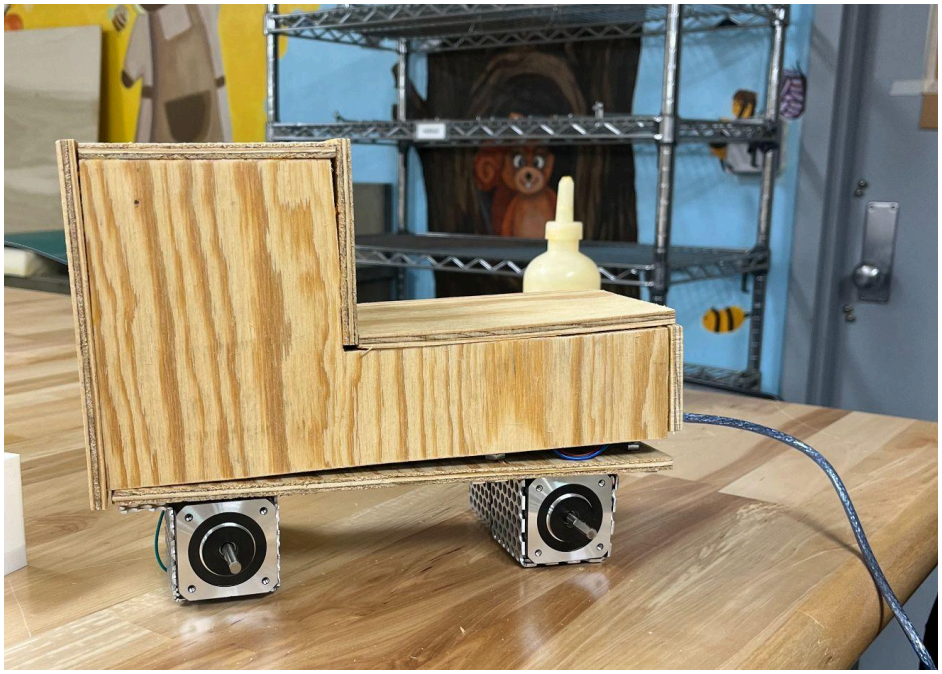

Figure 4: The completed chassis build encasing electronics inside the chassis.

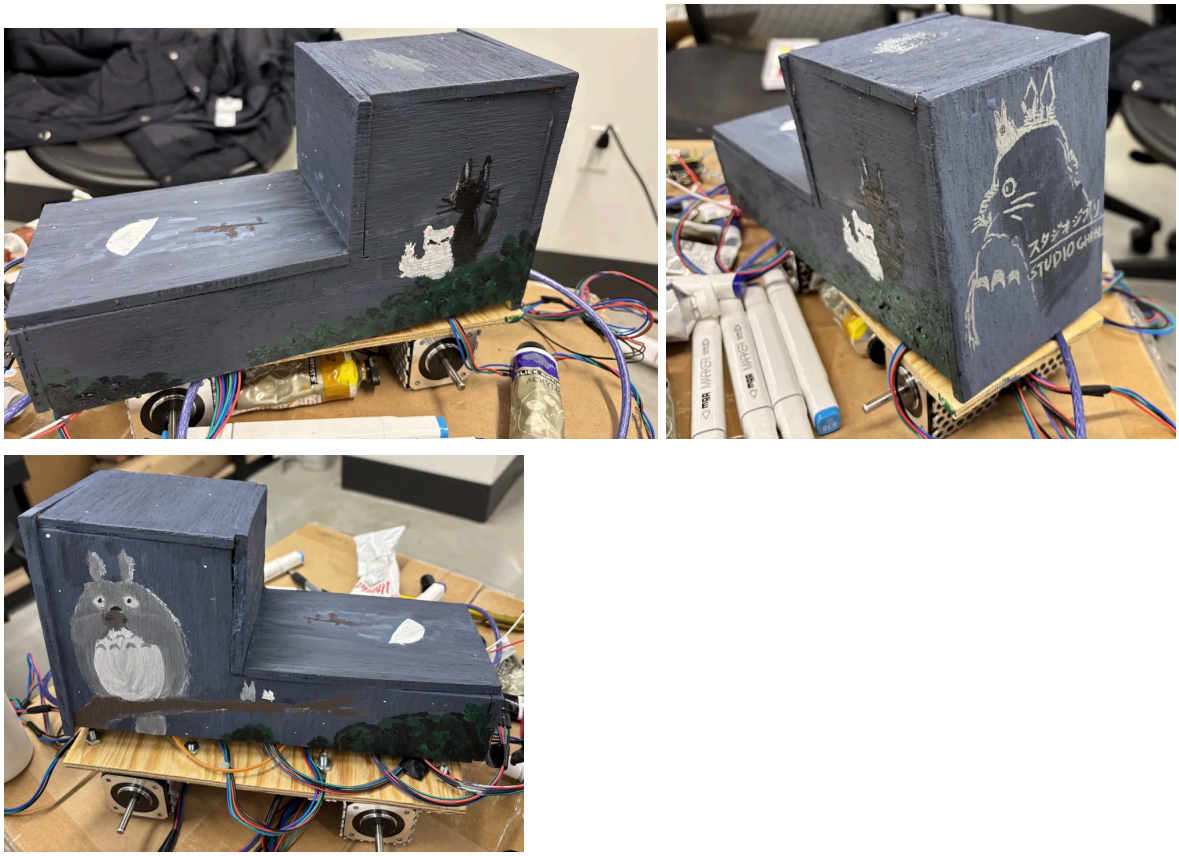

Figure 5-7: Painted chassis with a Totoro theme.

IMG_7333.mov - A video of our project playing Jingle Bells Rock.

IMG_7206.mov - A video of our project playing the Super Mario Bros. Theme Song.

Discussion:

Originally the scope of the project involved using 6 motors. However, due to budget constraints and hardware failures, the final implementation used four. Despite these limitations, the system demonstrated the potential for a scalable, programmable motorized instrument. The preprocessing scripts enhance the modularity of the tool, allowing for other users to directly transform a single-channel midi file into the microcontroller instruction set.

The project highlights the intersection of engineering and art, offering a unique, open-source perspective on sound generation. By repurposing industrial components for creative expression, Stepper Symphony opens new possibilities for educational tools, artistic installations, and experimental music.

Conclusion:

In the future, we can scale up the project to use 5 or 6 motors like we initially wanted. We could also expand the amount of octaves the sound our motors can produce. We can also explore more methods that might allow us to allow concurrent motors playing.

References:

Kostyukov, K. (2020, January 27). Make Music with Stepper Motors!. Autodesk Instructables. https://www.instructables.com/Make-Music-With-Stepper-Motors/

NEMA 17 bipolar 59ncm(83.55oz.in) 2A 42x42x48mm 4 wires W/ 1M Cable & Connector. StepperOnline Datasheet. (2020, October 30). https://www.omc-stepperonline.com/nema-17-bipolar-59ncm-84oz-in-2a-42x48mm-4-wires-w-1m-cable-connector-17hs19-2004s1?srsltid=AfmBOopo0vGr3xDJ11beI3qRop9eZo4FZRkqtVIOG6mhx5MAwWC71BNZ

Staff, L. E. (2023, January 16). In-depth: Control Stepper Motor with A4988 driver module & arduino. Last Minute Engineers. https://lastminuteengineers.com/a4988-stepper-motor-driver-arduino-tutorial/