

# ECE 44x Project Document

Group 33: Electromechanical Orchestration Device

Nicholas Kim, Kexin Liu, Liam Warner  
November 22nd, 2023

<b>Executive Summary</b> .....	<b>5</b>
1. Project Overview.....	6
1.1. Description.....	6
1.2. Team Contacts and Protocols.....	6
1.2.1. DEI Statement.....	7
1.2.2. Communication Analysis.....	8
1.3. Gap Analysis.....	8
1.4. Timeline and Task List.....	10
1.5 References and File Links.....	13
1.5.1 References (IEEE).....	13
1.6 Revision Table.....	14
<b>2. Impacts and Risks</b> .....	<b>15</b>
2.1. Design Impact Statement.....	15
2.1.1. Introduction.....	15
2.1.2. Public Health, Safety, and Welfare Impacts.....	15
2.1.2.1. Noise Pollution.....	15
2.1.2.2. Electrical, Mechanical, and Software Safety.....	15
2.1.3. Cultural and Social Impacts.....	16
2.1.3.1. Cultural Significance of Musical Instruments.....	16
2.1.3.2. Artificial Intelligence and Robotics Replacing Human Musicians.....	16
2.1.4. Environmental Impacts.....	16
2.1.4.1. Metal Mining and Production.....	16
2.1.5. Economic Factors.....	16
2.1.5.1. Employment Impact.....	16
2.1.5.2. Consumer Behavior Changes.....	17
2.1.6. Conclusion.....	17
2.2. Risks.....	18
2.3. References and File Links.....	19
2.3.1 References.....	19
2.4. Revision Table.....	20
<b>3. Top-Level Architecture</b> .....	<b>21</b>
3.1. Block Diagram.....	21
3.2 Block Descriptions.....	22
3.3. Interface Definitions.....	27
3.4 References and File Links.....	30
3.5 Revision Table.....	30
<b>4. Block Validations</b> .....	<b>31</b>
4.1 Solenoid Block Validation.....	31
4.1.1 Description.....	31
4.1.2 Design.....	31

4.1.3 General Validation.....	33
4.1.4 Interface Validation.....	33
4.1.5 Verification Plan.....	35
4.1.6 References and File Links.....	35
4.1.7 Revision Table.....	35
4.2 Mallet Driver.....	35
4.2.1 Description.....	35
4.2.2 Design.....	36
4.2.3 General Validation.....	38
4.2.4 Interface Validation.....	40
4.2.5 Verification Plan.....	43
4.2.6 References and File Links.....	44
4.2.6.1 References (IEEE).....	44
4.2.7 Revision Table.....	45
4.3 Music Performance Code.....	45
4.3.1. Description.....	45
4.3.2. Design.....	46
4.3.3. General Validation.....	47
4.3.4. Interface Validation.....	49
4.3.5. Verification Plan.....	51
4.3.6. References and File Links.....	51
4.3.6.1 References.....	51
4.3.7. Revision Table.....	52
4.4 MIDI/Autonomous Performance Library.....	52
4.4.1 Description.....	52
4.4.2 Design.....	53
4.4.3 General Validation.....	54
4.4.4. Interface Validation.....	55
4.4.5. Verification Plan.....	57
4.4.6. References and File Links.....	57
4.4.6.1. References.....	57
4.4.7.1 Revision Table.....	58
4.5 Microcontroller.....	58
4.4.1 Description.....	58
4.4.2 Design.....	59
4.4.3 General Validation.....	60
4.4.4. Interface Validation.....	61
4.4.5. Verification Plan.....	64
4.5.6. References and File Links.....	65
4.5.6.1. References.....	65
4.5.7. Revision Table.....	66

4.6 Power Supply.....	67
4.6.1 Description.....	67
4.6.2 Design.....	67
4.6.3 General Validation.....	69
4.6.4 Interface Validation.....	70
4.6.5 Verification Plan.....	75
4.6.6 References and File Links.....	76
4.6.6.1. References.....	76
4.6.7. Revision Table.....	77
4.7 DC Buck Converter Validation.....	77
4.7.1 Description.....	77
4.7.2 Design.....	78
4.7.3 General Validation.....	80
4.7.4 Interface Validation.....	81
4.7.5 Verification Plan.....	83
4.7.6 References and File Links.....	84
4.7.6.1. References.....	84
4.7.7 Revision Table.....	84
4.8. Distance Sensors Block Validation.....	84
4.8.1. Description.....	84
4.8.2. Design.....	85
4.8.3. General Validation.....	85
4.8.4. Interface Validation.....	85
4.8.5. Verification Plan.....	86
4.8.6. References and File Links.....	86
4.8.7. Revision Table.....	87
<b>5. System Verification Evidence.....</b>	<b>88</b>
5.1. Universal Constraints.....	88
5.1.1. The system may not include a breadboard.....	88
5.1.2. The final system must contain a student designed PCB.....	88
5.1.3. All connections to PCBs must use connectors.....	89
5.1.4. All power supplies in the system must be at least 65% efficient.....	90
5.1.5. The system may be no more than 50% built from purchased 'modules.'.....	91
5.2 Requirements.....	91
5.2.1. Autonomous Performance.....	91
5.2.2. Dynamic Range.....	93
5.2.3. Human Interactivity.....	95
5.2.4. Latency.....	96
5.2.5. MIDI Input.....	97
5.2.6. Power Supply.....	98
5.2.7. Tempo.....	100



5.2.8. Thermal Management.....	101
5.3 References and File Links.....	102
5.3.1. References.....	102
5.3.2. File Links.....	102
5.4 Revision Table.....	102
<b>6. Project Closing.....</b>	<b>103</b>
6.1. Future Recommendations.....	103
6.1.1. Technical Recommendations.....	103
6.1.2. Global Impact Recommendations.....	103
6.1.3. Teamwork Recommendations.....	104
6.2. Project Artifact Summary with Links.....	105
6.2.1. PCB Materials.....	105
6.2.2. Code Materials.....	107
6.2.3. Power Supply, Sensors, and Miscellaneous Electrical Materials.....	108
6.2.4. Mechanical and Enclosure Materials.....	111
6.3. Presentation Materials.....	112
6.4. References and File Links.....	112
6.4.1. References (IEEE).....	112
6.4.2. File Links.....	112
6.5. Revision Table.....	113

## Executive Summary

The purpose of this project is to design and build an electromechanical orchestrion device, which is a robot that plays a steel tongue drum in real-time using actuators controlled by a human interactable interface. The team has set up goals, tasks, and a project timeline. The electrical and mechanical engineering teams have started researching and brainstorming different actuators and how they can best meet our project requirements both musically, electrically, and mechanically. We learned that solenoids can play notes quickly but lack musical dynamics capabilities. We haven't found a better cost-conscious alternative to solenoids. Our group is currently prioritizing response time of the actuators. We have also researched pre-existing orchestrion devices (that play different instruments) to gather ideas for our implementation. We found that many inexpensive and simple systems use solenoids, but more complex systems use brushless DC motors which have greater dynamic capabilities but are very expensive. After research, we started generating different actuator and tremolo designs, which were presented and narrowed down to a select few. The team has now been able to 3D-print and prototype one solenoid actuator design and two different tremolo designs. The solenoid design nearly meets our tempo requirement (how many hits per second it can achieve) but does not meet our dynamics requirements yet. The tremolo designs were not very successful due to the body of the drum making most of the sound. Not much sound comes out of the sound hole on the bottom of the drum, which is what the tremolo was designed to modify. After some research, a microcontroller was selected for the project. A USB MIDI test was achieved using the code library we will use and a similar microcontroller we had available. Real-time MIDI data was read from a DAW and the microcontroller lit specific LEDs based on which notes were being played. Currently, the team is continuing to prototype additional solenoid and DC motor actuator concepts, reconsidering how to achieve a tremolo effect, and investigating the best method for connecting the microcontroller to the solenoid/motor driver circuit. Additionally, we are beginning to brainstorm aesthetics for the system by considering visual themes and lighting aspects.

# 1. Project Overview

The purpose of this document is to detail all tasks and processes regarding the ECE44x Engineering Design Project for Group 33: Electromechanical Orchestration Device. Section 1 will describe the project and keep a record of team dynamics. Section 2 will assess the potential impacts and risks of the project. Sections 3-5 will document the project's top-level architecture, define its blocks and validation methods, and provide system verification evidence. Section 6 will evaluate future recommendations and provide a summary of project materials. This document is meant to be comprehensive and cover all electronic aspects of the orchestration.

## 1.1. Description

The Electromechanical Orchestration Device is a multidisciplinary project between electrical and mechanical engineering students sponsored by Chet Udell, an award-winning electronic instrument designer and assistant professor at Oregon State University. An Orchestration is a machine or robot designed to perform music autonomously and in collaboration with human performers. The goal of this project is to design and build a system that performs on a steel tongue drum using methods of electromechanical actuation for the purpose of advancing the synthesis between engineering and the musical arts. The electrical engineering team is responsible for designing the electrical parts of the system that powers and controls the solenoids which actuate the tongues on the drum. A microcontroller will interpret external USB MIDI data to actuate the solenoids which hit the tongues at varying dynamic levels. Musical Instrument Digital Interface (MIDI) is a standard protocol for transmitting music data in real-time or storing it in files. The user may plug in a computer to control the orchestration via digital audio workstation (DAW) software, or any other MIDI output capable software of their choosing. The project will also incorporate sequence generation algorithms to play the drum autonomously with minimal human interaction. The entire system will be powered by the microcontroller's USB connection and a single standard wall outlet.

## 1.2. Team Contacts and Protocols

The contacts and roles of each member involved in the project are listed below. If any part of this document is unclear, anyone is welcome to reach out for further clarification using our information below.

Electrical and Computer Engineering Team:

- Nicholas Kim
  - Email: [kimnich@oregonstate.edu](mailto:kimnich@oregonstate.edu)
  - Project role: Solenoid actuator control
- Kexin Liu
  - Email: [liukex@oregonstate.edu](mailto:liukex@oregonstate.edu)
  - Project role: Power Supply
- Liam Warner
  - Email: [warnerli@oregonstate.edu](mailto:warnerli@oregonstate.edu)
  - Project role: Real-time MIDI decoder

Mechanical Engineering Team:

- Gianluca Rianda
- Luke Lutnesky
- Liam Hodge

Project Partner:

- Chet Udell

The Team Protocols Table below defines agreed-on team protocols and standard quality metrics for work quality and communication. By following these protocols, we as a team will be more successful and less prone to miscommunication.

TABLE I  
Team Protocols Table

<b>Team Protocols</b>	<b>Quality Metrics</b>
Text communication	Prioritizes communication and encourages quick messages and responses
ECE Team weekly meeting	Face-to-face communication, collaboration, and discussion. Includes creation of a weekly agenda to be covered in the meeting.
Helping other teammates	Prevents blocking issues and ensures all aspects of project are on track
Team meals	Team bonding and natural project discussion
ME Team weekly meeting	Face-to-face collaboration with ME team
Project partner and full team weekly meeting	Frequent updates for project partner and full team presence
Basecamp communication with project partner	Separate site for project partner to receive messages from team

### 1.2.1. DEI Statement

We will ensure that our project brings people together, respects and celebrates different cultures, and is able to be interacted with by everyone. Within our team, we will respect our different backgrounds and preferences for how each of us work, accommodate differences in our communication styles, and ensure that everyone's voice is heard. Throughout this project's timeline, we will hold ourselves accountable for any discrepancies between our actions and this statement, and seek to correct them in a timely manner.

### 1.2.2. Communication Analysis

We have several methods of communicating with our project partners, Dr. Chet Udell, as well as the mechanical engineering team and their capstone professor Layne Clemen. We meet with the ME team and Chet on Fridays from 9-10am. During Fall term, the ME team meets Mon/Wed from 3-6pm, which Professor Clemen attends when needed. Chet gives general guidelines for what information he wants from our team each meeting, and expects us to deliver that information at the next meeting, or another time if specified. Chet expects us to present our findings via discussion during the meeting. If we need to save or share research papers, websites, or other digital media related to the project, we do so via a shared Google Drive which Chet can access, or via Basecamp, which is a project planning/communication platform that Chet set up for us. This is also the primary way to contact Chet online, since email is not his preferred method. Contacting Professor Clemen can be done via email, and is mostly the mechanical engineering team's responsibility.

### 1.3. Gap Analysis

Our project was conceived in response to the high demand for innovative, technologically driven music performance. This intersection between technology, art and music aims to attract an audience seeking visual and aural experiences. The robotic steel tongue drummer system bridges this intersection gap, combining the rich tones of traditional instruments with precision and innovation of modern robotics.

Incorporating a microcontroller with MIDI USB to the robotic performer system can turn it into an interactive art piece where users can write pre-defined sequences or play the drum in real-time using a DAW. This is a primary need for our project partner Dr. Udell.

The capacity of the system to adapt and play changing sounds suggests that users may desire to play the orchestrion along with other instruments or generate algorithmic sequences of notes. Our system provides a greater musical capabilities beyond basic steel tongue drum playing for this purpose.

An interactive robotic drum system can be used to educate others about music, technology, and mechanical design all at the same time.

Research indicates that there is a market demand for robotic performance devices. There is a growing popularity for electronic musical instruments and robotic music performers, as shown by large concert venues booked by a band comprised only of robots [1]. Technology can be integrated in music performance especially as visual augmentation, allowing visual and body language expressivity when the performer may be in a remote location [2]. Music playing robots may also be used as educational tools in mechatronics or music technology.

This orchestrion system stands out in a market full of new music technology because it is being designed to be mostly 3D-printable, which makes it more appealing and accessible to a wide

range of users. Most electromechanical music robots are not meant to be reproduced, and if they are, their exact construction is held as a company secret such as the construction of ROLI's seaboard [3].

Our primary project stakeholder is Dr. Chet Udell. Udell is an award-winning electronic instrument designer who wants to use the orchestrion for his own musical performances or art installations. His focus is to apply new technology and the growth of DIY learning to explore "design, interaction, performance, and STEAM (Science, Tech, Engineering, Arts, & Math) education in musical and scientific instrumentation" [4]. Secondary stakeholders include professors Don Heer and Layne Clemen who instruct the ECE and ME senior capstone project courses. It is in their interest that their respective teams of engineering students succeed on this project. End users of the project may include musicians and producers who would use the system for general music performance and production.

For our design considerations, we decided to implement the USB MIDI protocol as input to our system because it facilitates easy integration with Digital Audio Workstation software and other standalone MIDI controllers. In addition to live MIDI input, the orchestrion will also be able to use predefined algorithms to generate its own note sequences with minimal human input. This will allow the orchestrion to be used as a standalone art piece if desired, and opens up capabilities for environmental sensors to be added to the project as stretch goals. These sensors would be used to change the qualities of the music (such as tempo, timbre, dynamics, and rhythm) in real-time as users interact with the drum or the environment around the drum changes. We also decided to focus on solenoid-based actuation methods since solenoids provide low-latency actuation for the steel tongue drum which is crucial for percussion instruments.

## 1.4. Timeline and Task List

Below in figure 2 is a draft of the project timeline, spanning over three academic terms. It shows project phases and large-scale deadlines for the team to follow.

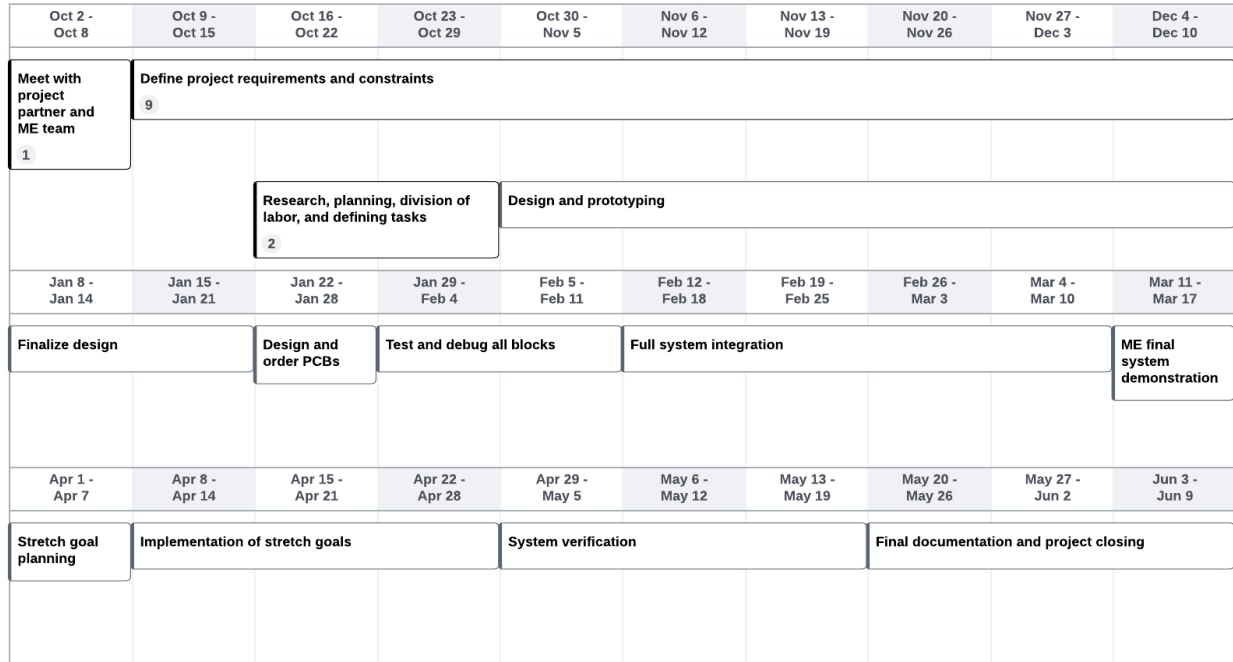


Fig. 1. Project Timeline

The Task List tracks project-related tasks and reflects the current state of the project.

TABLE II  
Task List

Description of Task	Impact Risk	Expected Work Hours	Due Date (Week #)	Champion	Actual Work Hours
Research previously implemented orchestrions	10	6	10/20	All	3
Research ESP32 Feather capabilities	8	2	10/20	Liam	1.5
Research different actuators	9	2	10/20	Nick	3
Research rotational motors	9	2	10/20	Liukee	2.5

Sketch possible designs for "sculpture"	5	1.5	10/27	All	6
Prototype/design amplifier for actuators	7	12	Week 5	Nick	4
Prototype/design control for tremolo motor	7	12	Week 5	Liukee	2.5
Programming MIDI Controller w/ LED demo	6	10	Week 7	Liam	2
Assist ME team with solenoid prototyping	4	2	Week 8	Nick	4
Report to Chet on TPIC, serial in/parallel out options	5	2	Week 8	Nick	2
Research I2C versus SPI protocol for I/O expansion	3	2	Week 9	Liam	1
Research options for driving solenoids that meets current requirements (>600mA)	5	2	Week 9	Nick	2
Design BLDC motor control for mallet prototyping	7	3	Week 9	Nick	3
Figure out how to read frequency and/or period of FG signal on Arduino and test with signal generator in Dearborn	5	3	Week 9	Liukee	3
Test the BLDC motor with an oscilloscope and create a table showing RPS for each duty cycle input	6	3	Week 9	Liukee	3
Create 9 output live USB MIDI demo with Feather M4 Express and figure out how to set it up as a USB host	5	3	Week 9	Liam	3
Flesh out Senior Design deliverables (requirements,	6	3	Week 9	Liam	3



interface definitions, verification, block definitions, block diagram)					
Research MIDI sequence generation algorithms	3	2	Week 10	Liam	2
Update autonomous code to change algorithm based on parameters	6	3	Week 12	Liam	5
Update MIDI Input code with Note timer functionality	10	3	Week 14	Liam	3
Document functions in MIDI/Autonomous Library	5	2	Week 15	Liam	2
Design and Order PCB	10	20	Week 16	Nick	24
Update code to ignore MIDI notes with velocity of zero and have 5 ms more of solenoid on time for velocity level 1	3	1	Week 18	Liam	1
Test Power Supply	9	2	Week 18	Liukee	2
Assemble PCB	10	10	Week 18	Nick	12
Solder JSTs	8	3	Week 18	Liam and Nick	3
Test Full System	10	5	Week 18	All	6
Write MIDI sequences to perform at MIME expo	6	2	Week 19	Liam	3
Research Distance Sensors	9	3	Week 19	Liukee	3
Assist MEs with MIME expo	8	5	Week 20	Liam and Nick	5

Add sensor-only control mode in code	8	6	Week 25	Liam	
Integrate note generation with performance function	6	6	Week 25	Liam	
Lay out connection from PCB to distance sensors	9	4	Week 22	All	4
Revise/improve 1st version PCB layout	3	2	Week 24	Nick	2

## 1.5 References and File Links

### 1.5.1 References (IEEE)

- [1] P. Neely, "Musical Robots take the stage for harmony, not domination," NPR, <https://www.npr.org/sections/alltechconsidered/2013/09/14/216482747/musical-robots-take-the-stage-for-harmony-not-domination> (accessed Nov. 21, 2023).
- [2] J. Anderson, "Communication in music: Using robots to physicalize remote performances," Michigan IT News, <https://michigan.it.umich.edu/news/2023/07/09/communication-in-music-using-robots-to-physicalize-remote-performances/> (accessed Nov. 21, 2023).
- [3] "Seaboard Rise 2: A brief history of reinventing the piano," ROLI, [https://roli.com/stories/seaboard-rise-2-a-brief-history-of-reinventing-the-piano?gad\\_source=1&gclid=CjwKCAiAx\\_GqBhBQEiwAIDNAZuMWjRCIVozP47dI1I-tZOrIIYw8T0cTUVxPgYswQ9KBRwehm5JyDxoC5fYQAvD\\_BwE](https://roli.com/stories/seaboard-rise-2-a-brief-history-of-reinventing-the-piano?gad_source=1&gclid=CjwKCAiAx_GqBhBQEiwAIDNAZuMWjRCIVozP47dI1I-tZOrIIYw8T0cTUVxPgYswQ9KBRwehm5JyDxoC5fYQAvD_BwE) (accessed Nov. 21, 2023).
- [4] "Instrumentation." Chet Udell. <https://www.chetudell.com/instrumentation> (accessed Nov. 22, 2023).

### 1.6 Revision Table

10/18/2023	Liam Warner: Created document and added draft of DEI statement
10/19/2023	All: Filled out sections 1.1-1.6
11/6/2023	Nicholas Kim:

	<ul style="list-style-type: none"> <li>● Removed document instructions</li> <li>● Created Overview</li> <li>● Revised Description</li> <li>● Added section descriptors and figure labels</li> <li>● Removed empty References section</li> </ul>
11/12/2023	Nicholas Kim: Fixed Revision Table format
11/21/2023	Liam Warner: <ul style="list-style-type: none"> <li>● Fixed table and figure formatting (IEEE)</li> <li>● Added to / modified Gap Analysis</li> <li>● Edited other sections especially introduction and executive summary <ul style="list-style-type: none"> <li>○ Added project description to Student Portal website</li> </ul> </li> </ul>
11/22/2023	Nicholas Kim: <ul style="list-style-type: none"> <li>● Updated Week 8-9 tasks</li> <li>● Reformatted project timeline</li> <li>● Revised Gap Analysis</li> </ul>
03/14/2024	All: added tasks completed this term
4/26/2024	All: Updated task list
5/16/2024	All: Updated task list

## 2. Impacts and Risks

### 2.1. Design Impact Statement

#### 2.1.1. Introduction

The purpose of the design impact assessment is to address potential impacts of the project design in areas relating to safety, culture, the environment, and the economy. It is the responsibility of the engineers to identify potential harms resulting from the design and to present those concerns to affiliated members of the project as well as preventative measures to reduce the risk of harm.

Research was conducted across four general categories to assess the risk of the orchestrion design and the findings are reported in the following sections. Section 2 assesses the impact of noise pollution and general engineering safety. Section 3 reports the cultural significance of the steel tongue drum used in the design as well as the impact of artificial intelligence on music performance. Section 4 addresses the environmental impact of sourcing the materials found in the design. Section 5 discusses the economic impacts of robotic performance in the music industry.

#### 2.1.2. Public Health, Safety, and Welfare Impacts

##### 2.1.2.1. Noise Pollution

The orchestrion will be striking the drum autonomously via musical algorithms. The sounds produced by this process may have negative public health impacts in the form of noise pollution on the area in which it operates. According to Environmental Health Perspectives, high levels of sound lead to negative impacts such as sleep disturbance and decline in cognitive performance [1]. Owners of the device should be cognizant of the surrounding environment and time of day during which it will perform to limit noise disturbances to others.

##### 2.1.2.2. Electrical, Mechanical, and Software Safety

Ensuring proper circuit connections during the design phase is essential to preventing electric shock. The use of the proper voltage levels must also be considered in order to avoid overheating, which can result in short circuits and possible fires. Other crucial factors are the way components are designed and the material chosen, such as avoiding sharp edges. Furthermore, ongoing safety depends on frequent inspections, assessments, and maintenance. This includes looking for signs of wear and tear and making sure all safety precautions are current. The design control software may be susceptible to mistakes and hacking, which could result in dangerous conduct. Strong protection measures, regular software updates to fix new bugs, and thorough system testing should be put in place to make sure it can handle problems that come up unexpectedly [2].

### 2.1.3. Cultural and Social Impacts

#### 2.1.3.1. Cultural Significance of Musical Instruments

One impact to consider when designing around certain musical instruments is their cultural significance. Many instruments are closely related to certain ethnic groups, religions, or countries. Our project essentially builds a robot to play a steel tongue drum. The steel tongue drum is based on the wooden slit drum, which is tied to many cultures across Africa, Southeast Asia, and Oceania. However, the modern steel tongue drum was first developed in the late 1900s using propane tanks by American Jim Doble. [3] It is safe to assume building a robot to play a modern steel tongue drum wouldn't have any negative impacts to the cultures that slit drums come from.

#### 2.1.3.2. Artificial Intelligence and Robotics Replacing Human Musicians

The area of robotics is quickly developing, especially in terms of artificial intelligence and human-like robots. This applies to music as well, where AI and robots are becoming increasingly capable of performing complex music accurately. The idea that technology can replace the human element of music can be terrifying. Can artificial intelligence or robots make music that evokes emotion? In the case of AI, it is important that its use in music is respectful of human creativity [4]. Many artists use AI to “get the creative juices flowing”, analyze target audiences, and assist in social media analytics. In general it may be a good principle to design robots/AI that augment human creativity and productivity rather than replace it. This applies to our project as well, where its intent is not to replace human performers, but to be more of an interactive art piece which a performer can exercise varying levels of control over.

### 2.1.4. Environmental Impacts

#### 2.1.4.1. Metal Mining and Production

Many of the components of this orchestrion, namely the steel tongue drum and the solenoid actuators, are made of steel. These metals may be sourced by mining which contributes to toxic waste in river systems. According to an article from Science, an estimated 23 million people live in areas polluted with toxic waste as a result of metal mining [5]. Research also shows that the production process of metals has negative impacts on the environment by contributing to greenhouse and acid rain gas emissions [6].

While the materials required for this project are marginal, awareness of the environmental impact of sourcing these materials may be appropriate if producing similar models in high quantities.

### 2.1.5. Economic Factors

#### 2.1.5.1. Employment Impact

The advancement of orchestrion and other similar devices may affect the job scene for musicians. These robots can play instruments that would otherwise require a human performer, which might mean fewer opportunities for professional musicians. Places like restaurants or

shopping malls might start using these robots because they may be more reliable, cheaper, and easier to control in the long run than hiring a live band.

These musicians may need to look for different kinds of jobs in the music industry, such as music production or teaching. The advancement of orchestrions could open up new job positions for programming and maintaining these devices.

Despite these advancements, there's something about humans playing music that a robot just can't replace. People still value real musicians for their unique art and the vibe they bring to performances. While these robots might shake things up a bit, there's still going to be a need for human musicians and their unique talents and creation skills. [7]

#### 2.1.5.2. Consumer Behavior Changes

Orchestrion is anticipated to attract consumer investment due to its unique features and advantages not seen in traditional drums. This preference for our technology is due to its user friendly design, constant performance, and new functions that distinguish it from conventional instruments. Because of the novelty and attractiveness of the orchestrion, consumers may reallocate a portion of their entertainment expenditure to it, thereby affecting sales of other musical equipment. As technology advances and software updates become more frequent, consumers may find themselves upgrading their equipment on a regular basis to keep up with the latest improvements, which may result in a progressive increase in cost. Consumer comments and reactions are critical in shaping the progress of our design. Positive user experiences can create a good chain reaction, while negative feedback provides important insights for the enhancement and improvement of the product. [8]

#### 2.1.6. Conclusion

In conclusion, there are more risks and impacts than one may think that are related to creating a robot that plays a musical instrument (orchestrion). As previously discussed, the replacement of human musicians and their creativity by robots and AI is concerning and has many economic and social impacts. Creation of music robots also may soon have an impact on musician jobs. A good guiding principle is to aim not to replace musicians, but to augment their creative abilities and expression. Robots may be able to play instruments in ways that humans cannot, which can be exciting. As with many medium-large scale electromechanical systems, ensuring safety of users is important when dealing with high speed actuators. We need to ensure that prior research and testing is done to minimize the risk of injuries or damage to the system related to high thermals, electric shock, or motor kinetics. Implementing safety shutdown features would also help minimize these risks. As with most products, where materials are sourced is also a concern. Doing our research to increase awareness of where our system's components are being sourced from will help us make ethical purchasing decisions, especially decisions that don't exploit the labor of low-income populations around the world. Overall, these ideas will give us an understanding of how our project affects others and will help us make smart design decisions as engineers both during our project and throughout our careers.

## 2.2. Risks

This table lists potential risks that are related to this orchestrion project, along with their corresponding action plans to help mitigate, reduce, or react to the risk in a constructive way.

TABLE II  
Potential Risks Table

Risk ID	Risk Description	Risk Category	Risk Probability	Risk Impact	Action Plan
R1	Delayed shipping on parts	Timeline	M	M	Contact shipping company and request expedited, reevaluate upcoming deadlines
R2	Carcinogenic materials or fabrication of parts	Health	L	M	Replace materials with safer ones and ensure safe procedures when 3d-printing [9].
R3	Task deadline missed	Timeline	H	M	Notify the project partner in advance. Evaluate a new deadline and get assistance from the team. Provide ample time to complete tasks.
R4	Parts catching on fire	Safety	M	H	Put out fire and redesign thermal control
R5	Tremolo disc ineffective on steel tongue drum	Technical	M	M	Evaluate new project requirement
R6	Microcontroller breaks	Technical	L	M	Take caution when designing high power circuits to protect microcontrollers. Protective circuits can mitigate risk [10]. Order new parts if an incident occurs
R7	Budget Overruns	Cost	M	M	Budget planning and management in advance

R8	Intellectual Property Issues	External	M	M	Ensure all the music materials have the necessary licenses or permissions to be used
R9	Supply Chain Disruption	Technical	M	M	Identifying appropriate substitutes or modifying the design to accept varying components
R10	Injury due to motors/actuators	Safety	L	L	Treat injury and write supplementary safety documentation

## 2.3. References and File Links

### 2.3.1 References

- [1] W. Psschier-Vermeer and W. F. Passchier, "Noise exposure and public health," *Environ. Health Perspect.*, vol. 108, pp. 123-131, Mar. 2000. [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1637786/>
- [2] Robot Manipulator Safety Rules. (n.d.). [Online] Retrieved from: <https://robotics.illinois.edu/robot-manipulator-safety-rules/#:~:text=Have%20a%20safety%2D buddy%20present,such%20as%20%E2%80%9CStarting%20robot%20motion%E2%80%9D>
- [3] "Steel tongue drum," Wikipedia, [https://en.wikipedia.org/wiki/Steel\\_tongue\\_drum](https://en.wikipedia.org/wiki/Steel_tongue_drum) (accessed Nov. 9, 2023).
- [4] Rolling Stone Culture Council, "The impacts and disruption of AI on music industry stakeholders," The Impacts and Disruption of AI on Music Industry Stakeholders, <https://council.rollingstone.com/blog/the-impacts-and-disruption-of-ai-on-music-industry-s takeholders> (accessed Nov. 9, 2023).
- [5] M. G. Macklin *et al.*, "Impacts of metal mining on river systems: a global assessment," *Science*, vol. 381, no. 1664, pp. 1345-1350, Sep. 2023. [Online]. Available: <https://www.science.org/doi/10.1126/science.adg6704>
- [6] T. E. Norgate, S. Jahanshahi, and W. J. Rankin, "Assessing the environmental impact of metal production processes," *Journal of Cleaner Production*, vol. 15, no. 8-9, pp. 838-848, 2007. [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S0959652606002320>



- [7] Swann, Peter. "The Robot Revolution: Understanding the Social and Economic Impact." Prometheus 36.1 (2020): 69-75. [Online] Available: <https://www.proquest.com/openview/3f158f8d5c3137b981e64f6040f7eebb/1?pg-origsite=gscholar&cbl=2036679>
- [8] Wissel, Tim. "Fairness and Freedom for Artists: Towards a Robot Economy for the Music Industry." (2021). [Online] Available: <https://repository.tudelft.nl/islandora/object/uuid:72a5c834-177b-4b3c-a6f8-8e69e65cfd4>
- [9] J. Kite-Powell, "New research shows fumes from 3D printers can create human health hazards," Forbes, <https://www.forbes.com/sites/jenniferhicks/2022/10/02/new-research-shows-fumes-from-3d-printers-can-create-human-health-hazards/?sh=26b51c3b48b6> (accessed Nov. 12, 2023).
- [10] Solutions Cubed, "Protecting inputs in digital electronics," DigiKey, <https://www.digikey.com/en/articles/protecting-inputs-in-digital-electronics> (accessed Nov. 12, 2023).

## 2.4. Revision Table

11/12/2023	Kexin Liu : Added "Risks"
11/12/2023	Kexin Liu : Added description and IEEE format of "2.2 Risks"
11/12/2023	Nicholas Kim: Added to Risks Table
11/12/2023	Liam Warner: Added to risks table, references
11/22/2023	Nicholas Kim: Revised R3 Action plan
4/26/2024	Nicholas Kim: Added Design Impact sections

### 3. Top-Level Architecture

#### 3.1. Block Diagram

Below figure 2 is a black box diagram of our system, showing the inputs and outputs seen on a level external to our system.

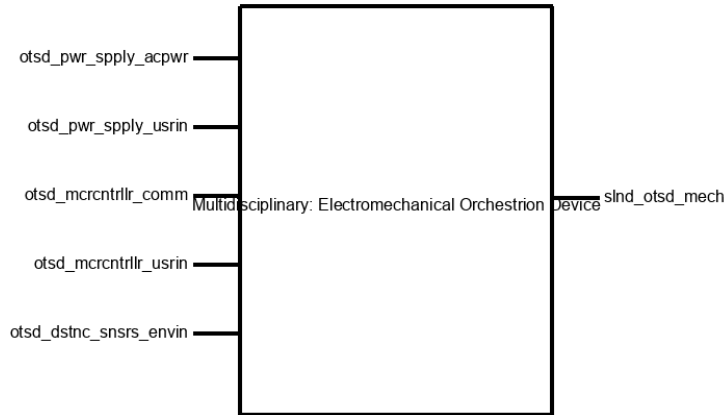


Fig 2. Black Box Diagram

Below in Figure 3 is a top-level block diagram of our system which shows how each of the blocks are connected, which interfaces go between each block, and the interfaces which interact with components outside of our system (external).

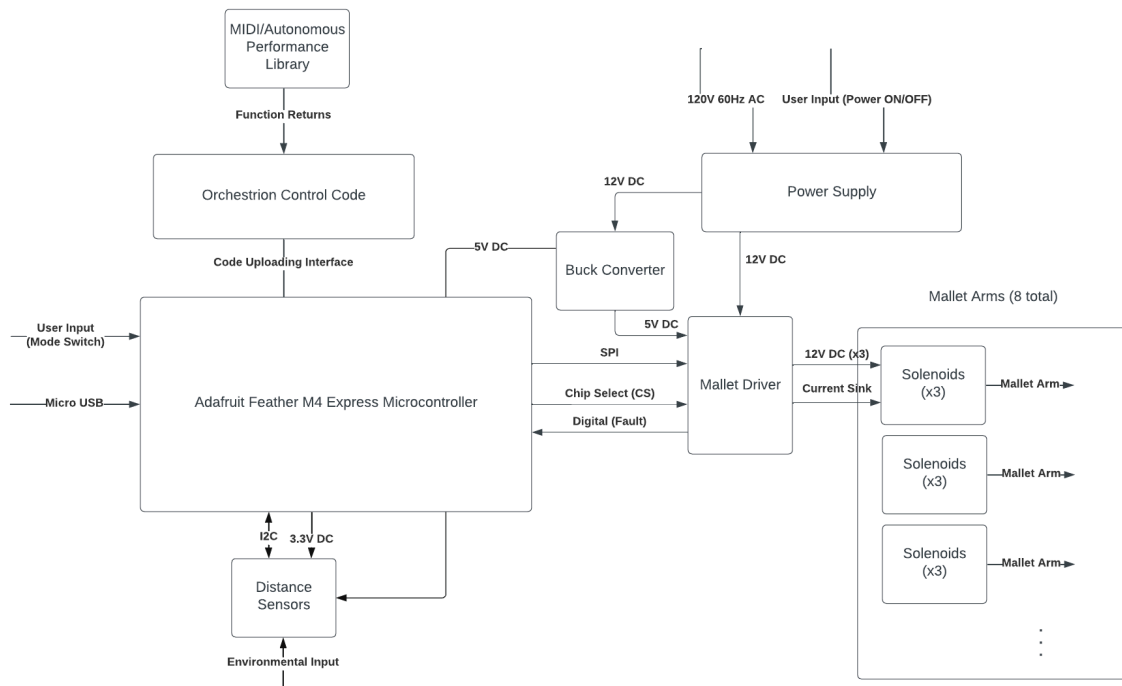


Fig 3. Top-Level Block Diagram

### 3.2 Block Descriptions

This block table describes each high-level subsection of our system and briefly describes what each block is responsible for. It also notes who is the champion for each block (responsible for its success).

TABLE III  
Block Description Table

Name	Description
<p>Music Performance Code Champion: Liam Warner</p>	<p>This code block is responsible for all MIDI input data processing, autonomous note sequence generation, and all SPI message configuration for the system. This especially ensures the system will meet our MIDI input and autonomous performance requirements. It works by configuring the microcontroller to be able to send and receive signals via the microcontroller's I/O pins to and from the mallet driver block and the various distance sensors in the system. All code is written in the Arduino IDE.</p> <p>It has two main modes of operation, autonomous mode and real-time MIDI mode. The code configures the microcontroller to read I/O pin 5 to determine whether the autonomous mode is on or off. If pin 5 is low, it is in MIDI mode which will process USB MIDI data in real-time over the usb connection, receiving signals such as note value, note velocity, and modulation value from the data packets being sent. These signals will be then processed and appropriate 8-bit SPI messages will be sent through the MOSI, SCLK, and CS pins. The messages are in a 0bXX00_0000 format, where the first two bits are "don't cares" and the next represent gates 5-0 for a given pre-fet driver TPIC chip [1]. For context, there will be four different CS (chip select) signals/pins, each of which enable and disable input for one of four separate and identical TPIC chips. Each TPIC drives two notes, which have 3 solenoids each. This makes for a total of 24 solenoids in the system. More details can be found in the mallet driver block description.</p> <p>The second mode is an autonomous performance mode, which uses Markov matrices and probability distributions to generate a song, or a sequence of notes, by using parameters set within this block. These parameters include time signature, a discrete "energy level" which can either be calm, moderate, or upbeat (0, 1, 2), and beats per minute (bpm). Each of these parameters can be adjusted within the code to change how the autonomous performance mode behaves. The autonomous mode will be turned on via an external switch which is checked with a digitalRead() function. As long as pin 5 is high, note sequences will be generated then</p>

	<p>performed. The music performance code will call functions from the MIDI/Autonomous code blocks to perform these tasks. The five functions that are called from within this block are shown in the interface properties section.</p>
<p>Mallet Driver Champion: Nicholas Kim</p>	<p>The mallet driver is a PCB that uses signals from the microcontroller to drive multiple solenoids and provide feedback. Active components within the block (AND gates and pre-FET driver) are powered by the DC buck converter. Active components are rated for a 5V supply and must be maintained from 4.5V-5.5V. The mallet driver receives SPI data from the microcontroller to change the state of a pre-FET driver (TPIC46L01) according to which solenoids to actuate. The serial communication consists of a serial clock, serial input, chip select (for each pre-FET device), and optional MISO for fault-interrupts. The serial clock must be under 10MHz and 3.3V logic signals from the microcontroller will be level shifted up to 5V logic using an 8-bit bidirectional voltage level shifter (TXB0108). Each pre-FET driver drives 6 gate voltages, which will control the MOSFETs (CSD88539ND) on the solenoid block. The mallet driver will also provide a fault signal indicating when an open-load or shorted-load is detected on any MOSFET drain line by comparing it to a fault-reference voltage.</p>
<p>Power Supply Champion: Kexin (Liukee) Liu</p>	<p>The electrical system includes a flexible 12V 30A DC Universal Regulated Switching Power Supply that can operate with either 110V or 220V AC input voltages, which may be chosen using a switch to align with local power requirements. The power supply effectively transforms alternating current (AC) into direct current (DC) at 12 volts, providing a maximum output of 360 watts to meet the system's needs. It features an ON/OFF input that allows users to manage the activation of power. The system's power distribution originates from a supply, splitting into two paths. One path supplies a DC Buck Converter which reduces the voltage from 12V DC to a stable 5V DC for low-voltage components. The other path directly powers 24 solenoids operating at 12V.</p>

<p>Solenoid Champion: Nicholas Kim</p>	<p>A solenoid is an actuator which uses electromagnetic induction to create linear mechanical motion. The electromagnetic field is generated by passing current through the solenoid coil. The magnetic armature generates a linear force which actuates the mechanical system. A current recirculation diode (rated for at least 3 times the solenoid peak current) is required to absorb inductive kickback. There will be 24 total solenoids to support 8 mallet arms, each using 3 solenoids. 3 levels of dynamic range will be achieved by firing 1-3 solenoids on an arm at a time. Cables will distribute 12V DC from the main PCB to the solenoids. The mallet driver block controls the MOSFET gate voltages. Setting the gate voltage above the threshold voltage (3V) will turn the MOSFET on and draw current through the solenoid to actuate it.</p>
<p>Microcontroller Champion: Liam Warner</p>	<p>This is an Adafruit Feather M4 Express microcontroller which acts as a processing unit for the Orchestrion. It interprets the USB MIDI data by way of the various code blocks in our system and outputs necessary signals to operate the actuators and tremolo via the SPI and/or digital I/O pins. It is powered via the 5V line in the micro USB input.</p>
<p>DC Buck Converter Champion: Kexin (Liukee) Liu</p>	<p>The DC Buck Converter is constructed with the TPS54232D microcontroller, which effectively reduces a 12V input to a 5V output, so enabling the operation of four-mallet drivers. On average, each driver consumes around 2.6mA, resulting in a cumulative consumption of roughly 12mA for all four drivers. During peak usage, each driver can sustain up to 4.2mA, resulting in a total consumption of close to 18mA. The inclusion of a 1.2A inductor guarantees reliable functioning, and the system is equipped with preventative mechanisms to prevent any potential hazards caused by sudden increases in voltage or current.</p>

<p>MIDI/Autonomous Performance Library Champion: Liam Warner</p>	<p>This code block encapsulates all algorithms and use of libraries that are responsible for the drum's ability to play autonomously and live via MIDI control. For the autonomous part of the library, the music performance code will have an autonomous mode variable that is "turned on" by an external switch. Then, the functions (properties of the interface) that handle music sequence generation will be called by the music performance code block, and the library will use parameters and predetermined probability arrays/matrices to start creating note sequences to be stored in an array as a song. The block crucially contains an autonomous_seq_generation function which generates a song to perform, and a perform_song function which is designed to take the generated song and perform it by formatting SPI messages to be sent to the mallet driver block at the appropriate times. This code block also supplies functions for formatting and sending SPI messages for incoming MIDI messages via the externally sourced USB MIDI library and via other simple functions which streamlines the code in the music performance block. The main function here is the readMidi() function, which is called in the Music Performance block and returns a Note struct which contains three parameters: note index, velocity, and duration. Additionally, there are two functions that deal with updating and checking timers for each note on the drum to ensure the solenoids are on for the appropriate amount of time.</p>
<p>Distance Sensors Champion: Kexin (Liukee) Liu</p>	<p>The SHARP distance sensor enhances the performance by using IR reflection to detect audience proximity within 20cm to 150cm. It dynamically adjusts the drumming speed based on the spectators' proximity, creating an interactive experience.</p>

### 3.3. Interface Definitions

This table describes all the interfaces between each of our top-level blocks described in section 3.2 and specifies the properties they are required to exhibit.

TABLE IV  
Interface Definitions

Name	Properties
dstnc_snsrs_mrcntrllr_comm	<ul style="list-style-type: none"> <li>● <b>Datarate:</b> At 20cm, 3V; At 150cm, 0.4V</li> <li>● <b>Vmax:</b> 3V</li> <li>● <b>Vmin:</b> 0.4V</li> <li>● <b>Vnominal:</b> between 0.4 V and 3.0 V</li> </ul>
otsd_dstnc_snsrs_envin	<ul style="list-style-type: none"> <li>● <b>Other:</b> Distance Range: 20cm to 150cm</li> <li>● <b>Temperature (Absolute):</b> Temperatures from -10°C to +60°C</li> </ul>
otsd_pwr_sply_usrin	<ul style="list-style-type: none"> <li>● <b>Timing:</b> Takes &lt;1 second to use</li> <li>● <b>Type:</b> Power ON/OFF</li> <li>● <b>Usability:</b> 10/10 Usability</li> </ul>
otsd_pwr_sply_acpwr	<ul style="list-style-type: none"> <li>● <b>Inominal:</b> 2.25A</li> <li>● <b>Ipeak:</b> 3.013A</li> <li>● <b>Vnominal:</b> 120V</li> </ul>
otsd_mrcntrllr_usrin	<ul style="list-style-type: none"> <li>● <b>Other:</b> Label indicating autonomous mode is next to switch</li> <li>● <b>Timing:</b> Takes &lt;10 seconds to flip switch</li> <li>● <b>Type:</b> Mode Switch</li> </ul>
otsd_mrcntrllr_comm	<ul style="list-style-type: none"> <li>● <b>Messages:</b> Note Value (0-127)</li> <li>● <b>Messages:</b> Velocity Value (0-127)</li> <li>● <b>Protocol:</b> USB MIDI</li> <li>● <b>Vnominal:</b> 5V</li> </ul>
msc_prfrmnc_cd_mrcntrllr_code	<ul style="list-style-type: none"> <li>● <b>Other:</b> Uses USB Micro B to USB A cable for uploading</li> <li>● <b>Other:</b> Source code file types include .ino and .h</li> <li>● <b>Other:</b> Upload Baud Rate: 921600</li> </ul>

mllt_drvr_slnd_asig	<ul style="list-style-type: none"> <li>● <b>Vmax:</b> 12V</li> <li>● <b>Vrange:</b> OFF: 0V-2.5V</li> <li>● <b>Vrange:</b> ON: 8V-15V</li> </ul>
mllt_drvr_mrcntrlr_dsig	<ul style="list-style-type: none"> <li>● <b>Logic-Level:</b> 3.3V (active LOW)</li> <li>● <b>Vmax:</b> Fault: 0.8V</li> <li>● <b>Vmax:</b> No fault: 3.3V</li> <li>● <b>Vmin:</b> No fault: 2V</li> </ul>
pwr_spply_slnd_dcpwr	<ul style="list-style-type: none"> <li>● <b>Inominal:</b> ON: 750mA per solenoid, ~18A</li> <li>● <b>Ipeak:</b> ON: 1A per solenoid, ~24A</li> <li>● <b>Vmax:</b> 13V</li> <li>● <b>Vmin:</b> 11V</li> <li>● <b>Vnominal:</b> 12V</li> </ul>
pwr_spply_dc_bck_cvrtr_dcpwr	<ul style="list-style-type: none"> <li>● <b>Inominal:</b> 19mA</li> <li>● <b>Ipeak:</b> 100mA</li> <li>● <b>Vmax:</b> 13V</li> <li>● <b>Vmin:</b> 11V</li> <li>● <b>Vnominal:</b> 12V</li> </ul>
slnd_otsd_mech	<ul style="list-style-type: none"> <li>● <b>Other:</b> 50ms max latency</li> <li>● <b>Other:</b> Capable of attaching to mechanical system</li> <li>● <b>Other:</b> 10mm stroke length</li> </ul>
slnd_mllt_drvr_asig	<ul style="list-style-type: none"> <li>● <b>Other:</b> Open-load fault detection: In OFF state, drain voltage should be greater than fault reference voltage (Drain &gt; 1.25V)</li> <li>● <b>Other:</b> Shorted-load fault detection: In ON state, drain voltage should be less than fault reference voltage (Drain &lt; 1.25V)</li> <li>● <b>Vmax:</b> 15V</li> </ul>
mrcntrlr_dstnc_snsrs_dcpwr	<ul style="list-style-type: none"> <li>● <b>Inominal:</b> 33mA</li> <li>● <b>Ipeak:</b> 50mA</li> <li>● <b>Vmax:</b> 7V DC</li> <li>● <b>Vmin:</b> 4.5V DC</li> <li>● <b>Vnominal:</b> 5V DC</li> </ul>



mrcntrllr_mllt_drvr_dsig	<ul style="list-style-type: none"> <li>● <b>Logic-Level:</b> 3.3V (active LOW chip select)</li> <li>● <b>Vmax:</b> LOW: 0.8V</li> <li>● <b>Vmax:</b> HIGH: 5V</li> <li>● <b>Vmin:</b> HIGH: 2V</li> </ul>
mrcntrllr_mllt_drvr_dsig	<ul style="list-style-type: none"> <li>● <b>Logic-Level:</b> 3.3V (active HIGH MOSI/SDI)</li> <li>● <b>Vmax:</b> LOW: 0.8V</li> <li>● <b>Vmax:</b> HIGH: 5V</li> <li>● <b>Vmin:</b> HIGH: 2V</li> </ul>
mrcntrllr_mllt_drvr_dsig	<ul style="list-style-type: none"> <li>● <b>Logic-Level:</b> 3.3V (active HIGH SCLK)</li> <li>● <b>Max Frequency:</b> 10MHz (max clock frequency)</li> <li>● <b>Vmax:</b> LOW: 0.8V</li> <li>● <b>Vmax:</b> HIGH: 5V</li> <li>● <b>Vmin:</b> HIGH: 2V</li> </ul>
dc_bck_cnvtrr_mllt_drvr_dcpwr	<ul style="list-style-type: none"> <li>● <b>Inominal:</b> 2.6mA per driver, ~12mA</li> <li>● <b>Ipeak:</b> 4.2mA per driver, ~18mA</li> <li>● <b>Vmax:</b> 5.5V</li> <li>● <b>Vmin:</b> 4.5V</li> <li>● <b>Vnominal:</b> 5V</li> </ul>
mdtnms_prfrmnc_lbrry_msc_prfrmnc_cd_data	<ul style="list-style-type: none"> <li>● <b>Messages:</b> Note read_midi ()</li> <li>● <b>Messages:</b> void perform_song (Note* song, int song_length, int bpm)</li> <li>● <b>Messages:</b> void update_note_timers(int note_inactive_arr[], Note cur_note)</li> <li>● <b>Messages:</b> Note* autonomous_seq_generation(Note* song, int energy_level, int song_length, int time_sig)</li> <li>● <b>Messages:</b> void check_note_timers(int note_inactive_arr[])</li> </ul>

### 3.4 References and File Links

[1] Texas Instruments, "6-Channel Serial And Parallel Low-Side Pre-FET Driver," TPIC46L01 datasheet, Nov. 1996 [Revised Aug. 2001].

[2] RobotShop, “24mm Brushless DC Gear Motor,”  
<https://www.robotshop.com/products/24mm-brushless-dc-gear-motor-12v-1000rpm>  
 (accessed Dec. 1, 2023).

[3] Adafruit Industries, “Adafruit Feather M4 Express”, Adafruit Feather M4 Express datasheet,  
 Nov. 2023

### 3.5 Revision Table

12/1/2023	Liam: added interface definition table from project portal and helped with formatting
12/1/2023	Nicholas Kim: DC Motor interface revision
12/3/2023	All: updated most interfaces and added extra DC power requirements
3/14/2024	Liam: updated interface definition table to match student portal
4/26/2024	Nicholas Kim: Added distance sensor block
5/16/2024	Nicholas Kim: Updated mallet driver block description to match portal

## 4. Block Validations

### 4.1 Solenoid Block Validation

#### 4.1.1 Description

The solenoid block provides the electrical system with the ability to actuate the mechanical system. Accelerating the mallet arm to strike the drum by electrical means is essential to the function of this electromechanical orchestration device.

The solenoid block receives an input voltage signal to change the motion of the mallet arm using a solenoid. A solenoid is an actuator which uses electromagnetic induction to move an armature. The electromagnetic field is generated by passing current through the solenoid coil. The armature generates a linear force which actuates the mechanical system. The solenoid is powered by an external 12V DC power supply and controlled by a CSD88539ND field-effect transistor (FET). An SK36A Schottky rectifier diode is used as a current recirculation diode to absorb inductive kickback.

There are 24 total solenoid blocks in the system to support 8 mallet arms using 3 solenoids each. Each mallet arm has three levels of dynamic range achieved by firing 1 to 3 solenoids on an arm on a given actuation.

#### 4.1.2 Design

This section contains diagrams and other artifacts of the solenoid block design. The black box diagram in Figure 1 defines the inputs and outputs of the solenoid block. Figure 2 shows the solenoid used in the block design. Figure 3 is a block diagram which shows block interfaces and interfaces between internal components.

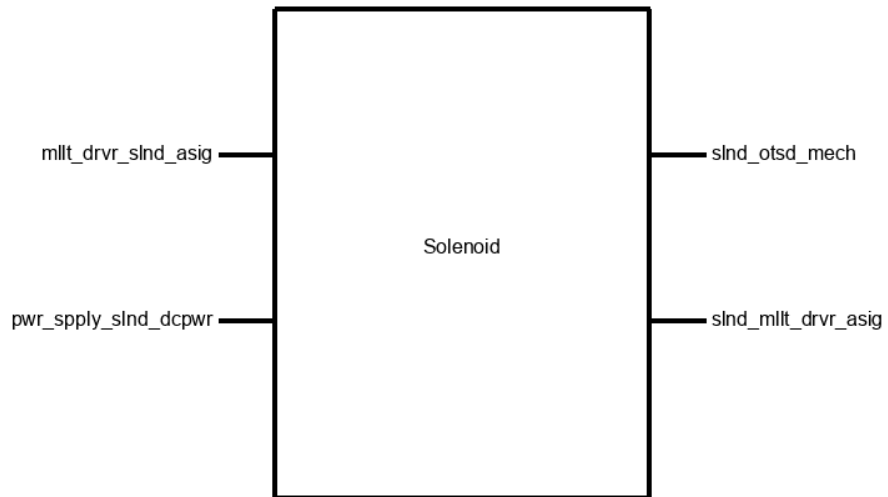


Fig 1. Black Box Diagram



Fig 2. Large push-pull solenoid [1]

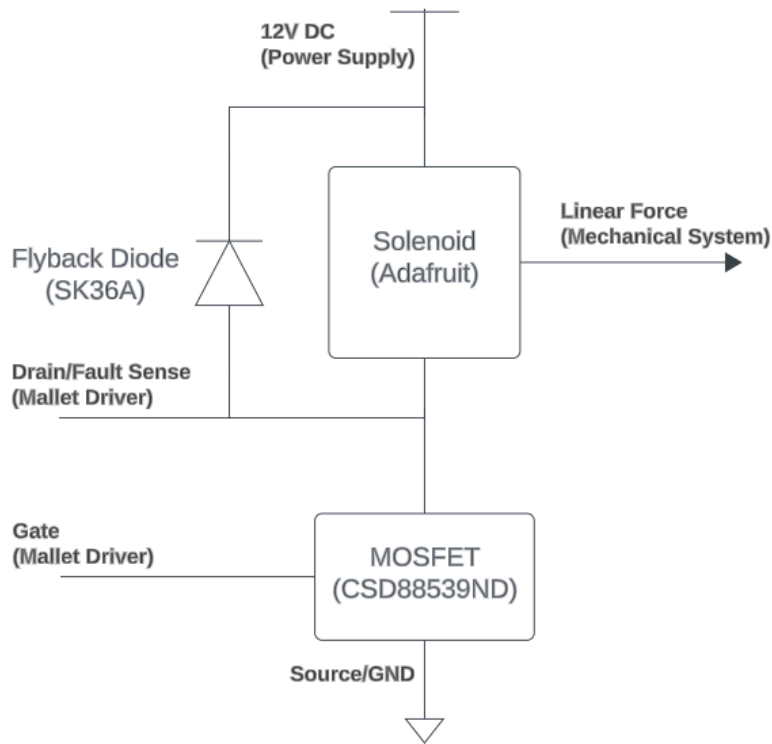


Fig 3. Solenoid Design Outline

#### 4.1.3 General Validation

The electrical part of the system needs to support a means of electromechanical actuation for the mechanical system to physically strike the drum. Solenoids were chosen for their relatively high force, low latency, high repetition speed, and ease-of-use. The particular solenoid in our design supplied by Adafruit was chosen for its availability and cost-efficiency [1].

Driving current through from the positive to negative terminal of the solenoid induces an electromagnetic field which moves a magnetic armature to produce a linear force. The Adafruit solenoid is rated for 12V DC with a coil resistance of ~12 ohms [1].

The fundamental design for solenoid control is using a FET as a switch to control current flow through a solenoid coil, as shown in Figure 3. The solenoid is actuated by applying voltage to the FET gate, allowing current to flow through the coil to apply an electromagnetic field on the armature. The switching device must be able to source enough current to support the upper limits of the solenoid to achieve low system latency and high armature speeds.

FET selection must take into consideration the drain-to-source voltage, threshold voltage, and current capabilities. The CSD88539ND has drain-to-source breakdown voltage at 60V and thus operational with a 12V supply. It is rated for 6.3A of continuous drain current, which is sufficient for the solenoid load which requires <1A to operate. The mallet driver block supplies a minimum gate drive voltage of 7V for a 12V VBAT supply, which is sufficient for the 3V threshold voltage on the CSD88529ND. While there are hundreds of suitable MOSFET options, the CSD88539ND is sufficient for the needs of the block [2]. Each CSD88529ND contains two FETs; 12 devices will be needed to support 24 solenoids.

The system must be protected from the large current running through the solenoid coil after being switched off at the mallet driver output channel. A current recirculation diode is added in parallel with the solenoid to absorb the current after the mallet driver output channel switches off. This diode should be rated for at least 3 times the nominal current. The SK36A Schottky rectifier diode supports an average forward current of 3A, which is 3 times the maximum observed current on the solenoid itself [3].

#### 4.1.4 Interface Validation

##### **mlt\_drvr\_slnd\_asig : Input**

Vmax: 20V (abs max MOSFET Vgs)	Absolute max Vgs [2]	CSD88539ND (FET on solenoid block) has 20V absolute max gate-to-source voltage [2]
Vrange: OFF: 0V-2.5V (below MOSFET Vth)	3V threshold voltage [2]	Pre-FET driver grounds gate output in OFF state [3]
Vrange: ON: 3.5V-12V (above MOSFET Vth)	3V threshold voltage [2]	7V min gate drive voltage for 12V VBAT [3]

##### **pwr\_spplly\_slnd\_dcpwr : Input**

Inominal: ON: 600mA	Initial testing showed 600mA sufficient for solenoid actuation	Power supply rated for 30A, sufficient for 24 blocks [4]
Ipeak: ON: 1A	Solenoid current peaking did not exceed 1A in testing	Power supply rated for 30A, sufficient for 24 blocks [4]
Vmax: 12.5V	Solenoid rated for 12V [1]	Power supply has over-voltage protection [4]
Vmin: 9V	9V solenoid min operating voltage [1]	Power supply rated for 12V [4]

##### **slnd\_otstd\_mech : Output**

Other: Hole to connect to mechanical part	Mechanical system uses rod to sync 3 solenoids together	Solenoid datasheet [1]
Other: Active linear force: 6N	Must produce appropriate force to actuate mechanical system	Solenoid datasheet [1]
Other: Stroke length: 10mm	Must achieve range to actuate lever arm to appropriate distance	Solenoid datasheet [1]

#### **slnd\_mllt\_drvr\_asig : Output**

Vmax: ON: 60V	Absolute max Vds [2]	VBAT supply to drain limited to 12V
Vrange: ON (shorted-load detection): 0V-1.25V	Pre-FET driver must detect 0V-1.25V at drain while ON, otherwise faults [3]	Rds(on) = 25 mOhm [2] << solenoid coil resistance = 12 Ohm [5]
Vrange: OFF (open-load detection): 1.25V-12V	Pre-FET driver must detect 1.25-12V at drain while OFF, otherwise faults [3]	No current in OFF state, drain approximately equal to 12V supply voltage

#### 4.1.5 Verification Plan

1. Connect solenoid block to 12V DC power supply
2. Apply 3.5-12V to MOSFET gate for 1 second to source current through the solenoid
  - a. Confirm solenoid operates at less than 1A and provides a full-stroke linear force at the armature
  - b. Confirm drain voltage within 0V-1.25V
3. Apply 0V-2.5V (under threshold) to MOSFET gate for 1 second to turn off
  - a. Confirm no current through solenoid and no actuation
  - b. Confirm drain voltage within 1.25V-12V

#### 4.1.6 References and File Links

[1] Adafruit Industries, “Large push-pull solenoid”, Adafruit large push-pull solenoid datasheet, Nov. 2023, <https://www.adafruit.com/product/413#technical-details> (accessed Dec. 7 2023)

[2] Texas Instruments, “CSD88539ND Dual 60 V N-Channel NexFET Power MOSFETs,” CSD88539ND datasheet, Feb. 2014 [Revised Dec. 2023], [https://www.ti.com/lit/ds/symlink/csd88539nd.pdf?ts=1706170848347&ref\\_url=https%253A%252F%252Fwww.mouser.kr%252F](https://www.ti.com/lit/ds/symlink/csd88539nd.pdf?ts=1706170848347&ref_url=https%253A%252F%252Fwww.mouser.kr%252F) (accessed Dec. 7 2023)

- [3] SMC Diode Solutions, “SK36A SCHOTTKY RECTIFIER,” SK36A datasheet, <https://www.smc-diodes.com/propdf/SK36A%20N0104%20REV.A.pdf> (accessed Mar. 14 2023)

#### 4.1.7 Revision Table

12/7/2023	Nicholas Kim: Initial outline (all sections)
1/24/2024	Nicholas Kim: Revised block description and interfaces
1/28/2024	Nicholas Kim: Moved MOSFET from mallet driver block to solenoid block; updated description, validation, and interface table
3/14/2024	Nicholas Kim: Updated all sections to reflect current design

## 4.2 Mallet Driver

### 4.2.1 Description

The mallet driver block provides the microcontroller with control of the mallet arms. Control of the mallet arms to strike the drum is essential to the function of this electromechanical orchestrion device. Each of the 8 mallet arms uses 3 solenoids; the mallet driver provides control of 24 independent solenoids to the available microcontroller output pins. The mallet driver block also provides an interrupt signal upon detection of an open-load or shorted-load fault; in the case of a broken solenoid coil or a short circuit across the solenoid, the microcontroller will be notified.

The mallet driver receives Serial Peripheral Interface (SPI) communicated by the microcontroller to drive field-effect transistor (FET) gate voltages on all solenoid blocks. Active components within the block are powered by a 5V DC buck converter. The mallet driver receives SPI data from the microcontroller to change the state of a pre-FET driver (TPIC46L01) according to which solenoids to actuate. The serial communication consists of a serial clock, a serial input, and chip selects for each pre-FET driver. The 3.3V logic provided by the microcontroller is level shifted to 5V logic using a voltage-level translator (TXB0108). Each pre-FET driver controls 6 FET gate voltages which each control a solenoid block. The mallet driver will indicate a fault to the microcontroller when an open-load or shorted-load incident is detected on any FET drain node using a quadruple AND-gate (SNx4HCT08).

### 4.2.2 Design

This section contains diagrams and other artifacts of the mallet driver design. The black box diagram in Figure 1 defines the inputs and outputs of the mallet driver. Figure 2 is a block diagram which shows block interfaces and interfaces between internal components. Figures 3 and 4 show the mallet driver section of the KiCad schematic and layout of the fully-integrated

PCB design. Note that there are 4 pre-FET drivers that function identically to support a total of 24 FETs.

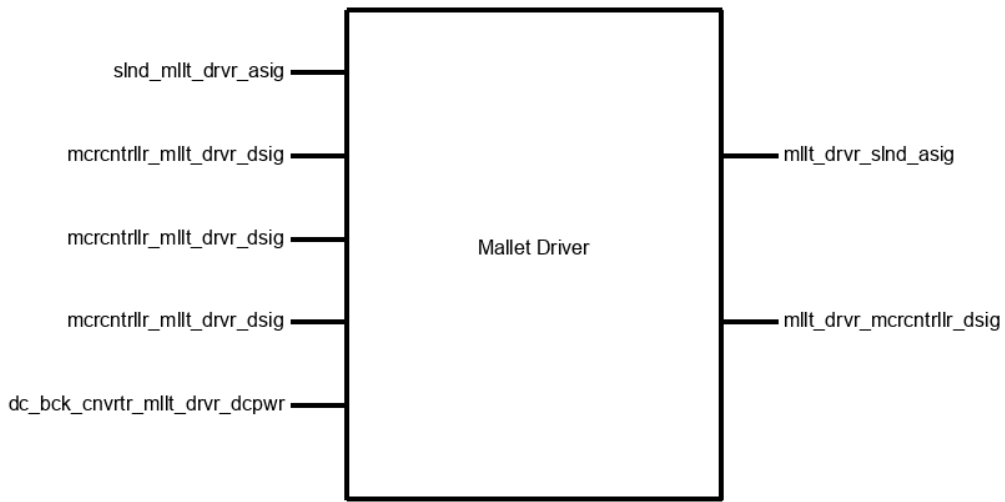


Fig 1. Black Box Diagram

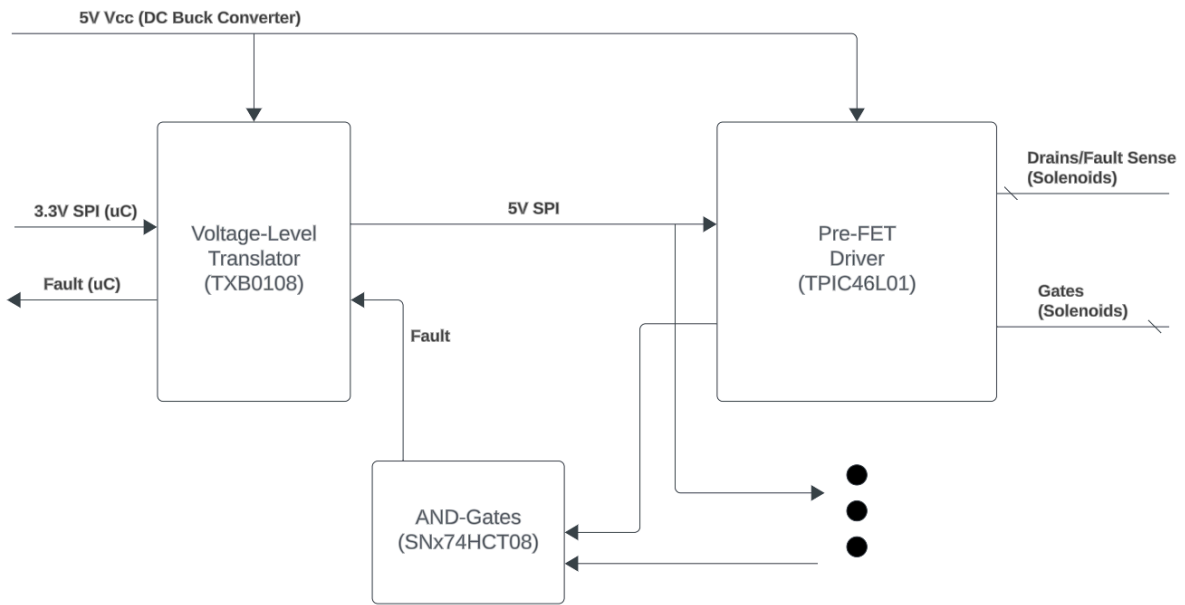


Fig 2. Mallet Driver Block Diagram



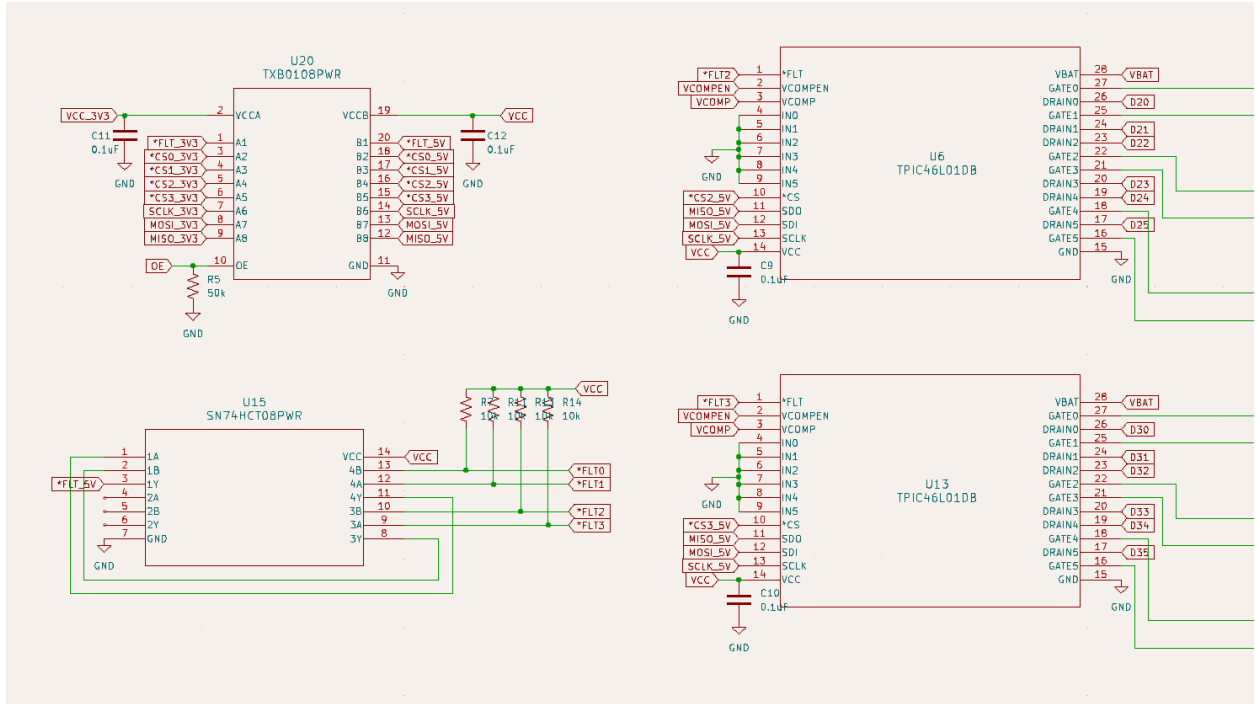


Fig 3. Mallet Driver Schematic

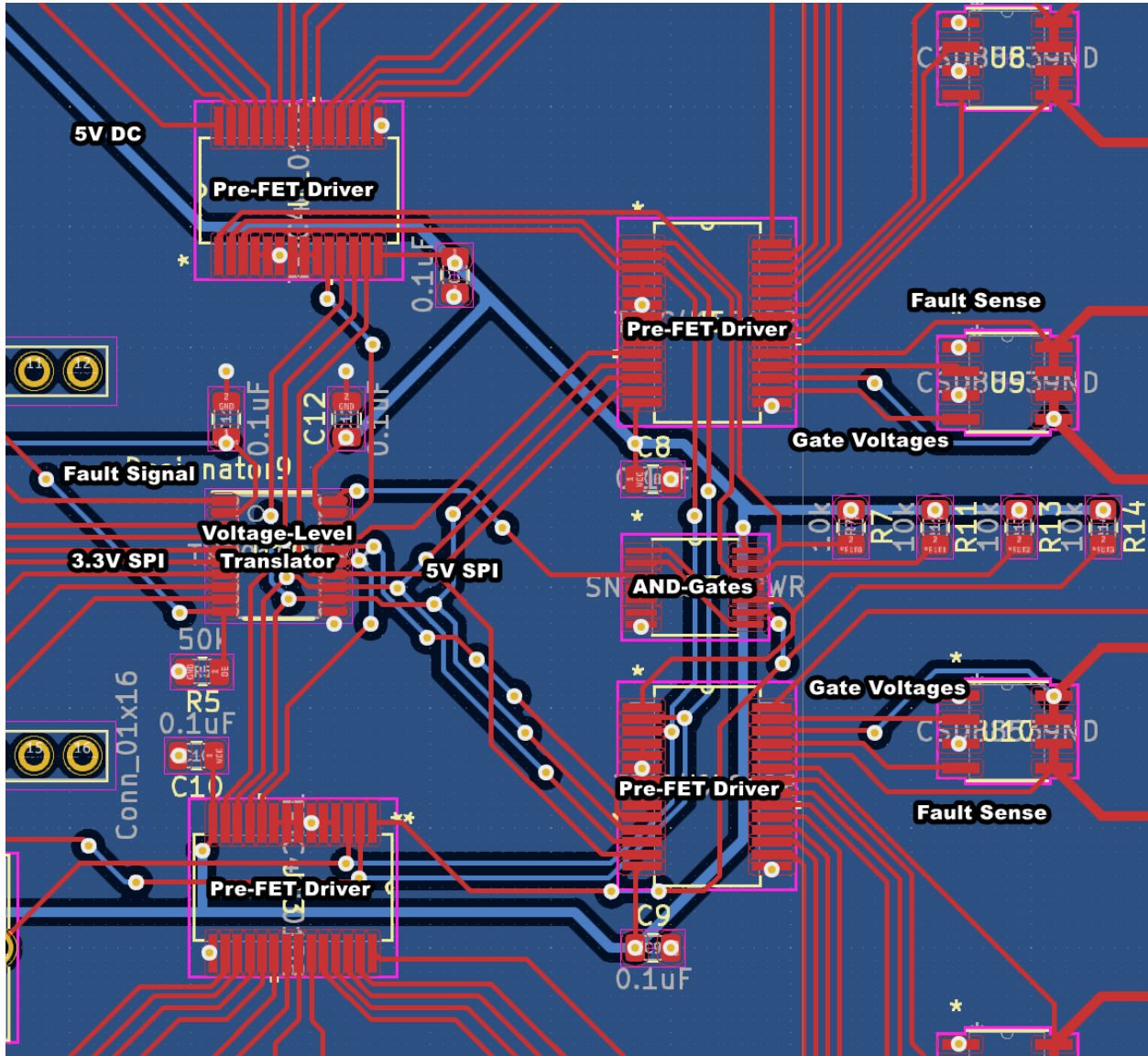


Fig 4. Mallet Driver PCB Layout

#### 4.2.3 General Validation

The design for the mallet driver block is centered around the TPIC46L01 pre-FET driver. The TPIC46L01 offers control of 6 FET gate voltages via SPI protocol and many other features for FET applications [1]. Each of the 8 mallet arms required by the system uses 3 solenoids; the mallet driver includes 4 TPIC46L01 devices to support a total of 24 FET gate voltages for 24 solenoids. The TPIC46L01 was selected to avoid microcontroller I/O constraints and for its relevant features for driving inductive loads, such as protection at the drain of the external FETs from large voltage spikes as a result of switching off inductive loads [1].

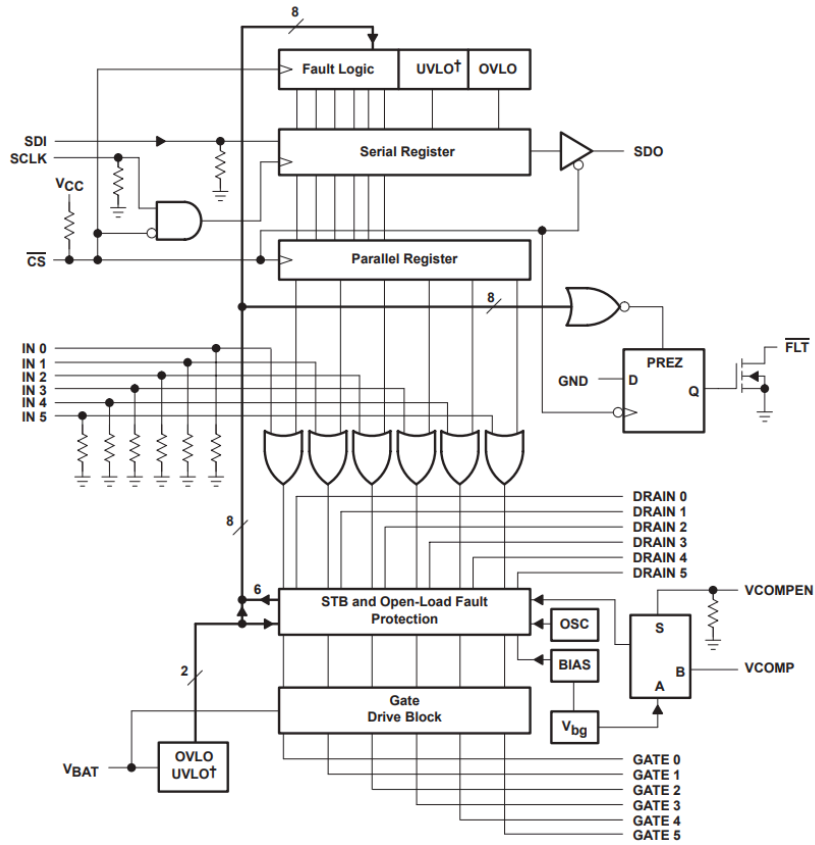


Fig 5. TPIC46L01 Schematic Diagram [1]

The TPIC46L01 requires 5V logic for its Serial Peripheral Interface. The TXB0108 translates 3.3V SPI from the microcontroller to 5V SPI [3]. The TPIC46L01 supports up to a 10MHz serial clock frequency [1]. The first 6 SPI data bits sent to the TPIC46L01 set the state of each gate drive voltage [1].

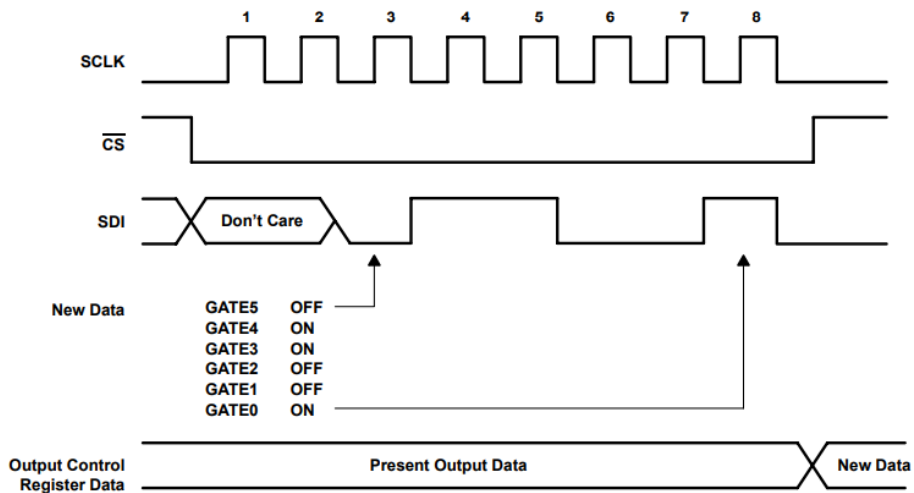


Fig 6. TPIC46L01 Serial Data Operation [1]

Each TPIC46L01 monitors the drain voltage of each FET it drives for open-load and shorted-load fault detection. Under normal conditions, the drain voltage will be tied to the solenoid block's supply voltage (VBAT) while the FET is OFF and tied to ground (GND) while the FET is ON. An open-load fault occurs when the TPIC46L01 detects a drain voltage below its 1.25V reference voltage while the associated gate is OFF. A shorted-load fault occurs when the TPIC46L01 detects a drain voltage above its 1.25V reference voltage while the associated gate is ON. The TPIC46L01 will pull its fault node to GND upon detection of either fault [1]. Under normal conditions, all 4 fault nodes are pulled up to 5V using external pull-up resistors. The SNx4CT08 performs an AND operation on all 4 fault nodes to produce an active LOW 5V logic fault-interrupt signal [2]. The TXB0108 translates this signal to 3.3V logic for the microcontroller [3].

The TPIC44L01 is a similar pre-FET driver which offers control of 4 FET gate voltages and may easily substitute the TPIC46L01 by using 6 devices with identical operation to support all 24 FET gates [4]. Other dissimilar devices such as the SN74HC595 8-bit shift register may be used to drive gate voltages using SPI [5]; however, the fault detection is not provided by the SN75364 and would need to be implemented into design by other means.

#### 4.2.4 Interface Validation

<b>Interface Property</b>	<b>Why is this interface property this value?</b>	<b>Why do you know that your design details for this block above meet or exceed each property?</b>
---------------------------	---	--

Table 1: Gate Voltage Output Interface Property Validation

**mllt\_drvr\_slnd\_asig : Output**

Vmax: 13.5V	TPIC46L01 has 13.5V max gate drive voltage for 12V VBAT [1]	TPIC46L01 has 13.5V max gate drive voltage for 12V VBAT [1]
Vrange: OFF: 0V-2.5V	CSD88539ND has 3V threshold voltage [6]	TPIC46L01 grounds gate output in OFF state [1]
Vrange: ON: 3.5V-12V	CSD88539ND has 3V threshold voltage [6]	TPIC46L01 has 7V min gate drive voltage for 12V VBAT [1]

Table 2: Fault Detection Output Interface Property Validation

**mllt\_drvr\_mrcntrlr\_dsig : Output**

Logic-Level: 3.3V (active LOW)	Microcontroller requires 3.3V logic [7]; TPIC46L01 grounds fault on detection [1]	Pull-up resistors (Fig 2, 3) hold TPIC46L01 fault lines at 5V; SNx4HCT08 outputs 0V when any fault line falls to 0V [2]; TXB0108 translates 5V logic to 3.3V logic [???
Vmax: No fault: 3.3V	Microcontroller requires 3.3V logic [7]	SNx4HCT08 has 3.3V max HIGH level output voltage [2]
Vmax: Fault: 0.8V	Microcontroller requires 3.3V logic [7]	SNx4HCT08 has 0.8V max LOW level output voltage [2]
Vmin: No fault: 2V	Microcontroller requires 3.3V logic [7]	SNx4HCT08 has 2V min HIGH level output voltage [2]

Table 3: Drain Sense Input Interface Property Validation  
**slnd\_mllt\_drvr\_asig : Input**

Other: Open-load fault detection: In OFF state, drain voltage should be greater than fault reference voltage (Drain > 1.25V)	TPIC46L01 will fault if drain voltage is less than 1.25V in OFF state [1]	In OFF state under normal conditions, the drain node is pulled up to solenoid supply voltage
Other: Open-load fault detection: In OFF state, drain voltage should be greater than fault reference voltage (Drain > 1.25V)	TPIC46L01 will fault if drain voltage is greater than 1.25V in OFF state [1]	In ON state under normal conditions, the drain node is pulled down to ground
Vmax: 15V	12V DC power supply has 15V max output voltage	TPIC46L01 has 60V max clamp voltage [1]

Table 4: SPI SCLK Input Interface Property Validation  
**mcrctrlr\_mllt\_drvr\_dsig : Input**

Logic-Level: 3.3V (active HIGH SCLK)	Microcontroller outputs 3.3V logic [7]	TXB0108 translates 3.3V logic to 5V logic [3] required by TPIC46L01 [1]
Max Frequency: 10MHz (max clock frequency)	TPIC46L01 has 10MHz max max SCLK frequency [1]	TPIC46L01 has 10MHz max max SCLK frequency [1]
Vmax: HIGH: 5V	Microcontroller outputs 3.3V logic [7]	TXB0108 has unbounded max HIGH level input voltage [3]
Vmax: LOW: 0.8V	Microcontroller outputs 3.3V logic [7]	TXB0108 has 0.8V max LOW level input voltage [3]
Vmin: HIGH: 2V	Microcontroller outputs 3.3V logic [7]	TXB0108 has 2V min HIGH level input voltage [3]

Table 5: SPI MOSI Input Interface Property Validation  
**mrcntrlr\_mllt\_drvr\_dsig : Input**

Logic-Level: 3.3V (active HIGH MOSI/SDI)	Microcontroller outputs 3.3V logic [7]	TXB0108 translates 3.3V logic to 5V logic [3] required by TPIC46L01 [1]
Vmax: LOW: 0.8V	Microcontroller outputs 3.3V logic [7]	TXB0108 has 0.8V max LOW level input voltage [3]
Vmax: HIGH: 5V	Microcontroller outputs 3.3V logic [7]	TXB0108 has unbounded max HIGH level input voltage [3]
Vmin: HIGH: 2V	Microcontroller outputs 3.3V logic [7]	TXB0108 has 2V min HIGH level input voltage [3]

Table 6: SPI CS Input Interface Property Validation  
**mrcntrlr\_mllt\_drvr\_dsig : Input**

Logic-Level: 3.3V (active LOW chip select)	Microcontroller outputs 3.3V logic [7]	TXB0108 translates 3.3V logic to 5V logic [3] required by TPIC46L01 [1]
Vmax: HIGH: 5V	Microcontroller outputs 3.3V logic [7]	TXB0108 has unbounded max HIGH level input voltage [3]

Vmax: LOW: 0.8V	Microcontroller outputs 3.3V logic [7]	TXB0108 has 0.8V max LOW level input voltage [3]
Vmin: HIGH: 2V	Microcontroller outputs 3.3V logic [7]	TXB0108 has 2V min HIGH level input voltage [3]

Table 7: DC Power Input Interface Property Validation  
**dc\_bck\_cnvtrr\_mllt\_drvr\_dcpwr : Input**

Inominal: 2.6mA per driver, ~12mA	Each of the 4 TPIC46L01 devices have 2.6mA nominal supply current [1]; TXB0108 and SNx4HCT08 have negligible supply currents [2]	Supply current determined by block components
Ipeak: 4.2mA per driver, ~18mA	Each of the 4 TPIC46L01 devices have 4.2mA max supply current [1]; TXB0108 and SNx4HCT08 have negligible input currents [2]	Supply current determined by block components
Vmax: 5.5V	Max supply voltage for TXB0108 [3], TPIC46L01 [1], and SNx4HCT08 [2]	Max supply voltage for TXB0108 [3], TPIC46L01 [1], and SNx4HCT08 [2]
Vmin: 4.5V	Min supply voltage for TXB0108 [3], TPIC46L01 [1], and SNx4HCT08 [2]	Min supply voltage for TXB0108 [3], TPIC46L01 [1], and SNx4HCT08 [2]
Vnominal: 5V	Nominal supply voltage for TXB0108 [3], TPIC46L01 [1], and SNx4HCT08 [2]	Nominal supply voltage for TXB0108 [3], TPIC46L01 [1], and SNx4HCT08 [2]

#### 4.2.5 Verification Plan

1. Interface with microcontroller block (required for SPI)
  - a. Connect microcontroller to SPI and fault signal I/O
2. Connect mallet driver to 5V DC power supply
3. Interface with solenoid block (required, otherwise open-load fault condition)
  - a. Connect gate drive to MOSFET gate, drain sense to MOSFET drain
  - b. Connect ammeter in series with supply and solenoid
4. Run SPI test on microcontroller to actuate GATE0
  - a. Any serial clock frequency <10MHz
  - b. Set chip select and serial input (HIGH on 8th clock pulse) to activate GATE0

- c. Confirm at least 600mA on solenoid and solenoid actuation
  - d. Test fault signal
    - i. Disconnect drain to cause open-load fault
    - ii. Confirm fault-interrupt voltage falls to LOW
  - e. Set chip select and serial input (all LOW) to deactivate all gates
5. Repeat steps 3 and 4 for all output channels

## 4.2.6 References and File Links

### 4.2.6.1 References (IEEE)

- [1] Texas Instruments, “6-Channel Serial And Parallel Low-Side Pre-FET Driver,” TPIC46L01 datasheet, Nov. 1996 [Revised Aug. 2001], [https://www.ti.com/lit/ds/symlink/tpic46l01.pdf?ts=1701936974753&ref\\_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTPIC46L01](https://www.ti.com/lit/ds/symlink/tpic46l01.pdf?ts=1701936974753&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTPIC46L01) (accessed Dec. 7 2023)
- [2] Texas Instruments, “SNx4HCT08 Quadruple 2-Input Positive-AND Gates,” SNx4HCT08 datasheet, Nov. 1988 [Revised Oct. 2022], <https://www.ti.com/lit/ds/scls063f/scls063f.pdf?ts=1706149182318> (accessed Jan. 24 2024)
- [3] Texas Instruments, “TXB0108 8-Bit Bidirectional Voltage-Level Translator with Auto-Direction Sensing and  $\pm 15$ -kV ESD Protection,” TXB0108 datasheet, Nov. 2006 [Revised Aug. 2020], [https://www.ti.com/lit/ds/symlink/txb0108.pdf?ts=1709917608320&ref\\_url=https%253A%252F%252Fwww.mouser.com%252F](https://www.ti.com/lit/ds/symlink/txb0108.pdf?ts=1709917608320&ref_url=https%253A%252F%252Fwww.mouser.com%252F) (accessed Mar. 8 2024)
- [4] Texas Instruments, “4-Channel Serial And Parallel Low-Side Pre-FET Driver,” TPIC44L01 datasheet, Nov. 1996 [Revised Aug. 2001], <https://www.ti.com/lit/ds/symlink/tpic44l01.pdf?ts=1709943806716> (accessed Mar 8. 2024)
- [5] Texas Instruments, “SNx4HC595 8-Bit Shift Registers With 3-State Output Registers,” SNx4HC595datasheet, Nov. 1988 [Revised Oct. 2022], <https://www.ti.com/lit/ds/symlink/sn74hct595.pdf?ts=1709902865074> (accessed Mar. 8 2024)
- [6] Texas Instruments, “CSD88539ND Dual 60 V N-Channel NexFET Power MOSFETs,” CSD88539ND datasheet, Feb. 2014 [Revised Dec. 2023], [https://www.ti.com/lit/ds/symlink/csd88539nd.pdf?ts=1706170848347&ref\\_url=https%253A%252F%252Fwww.mouser.kr%252F](https://www.ti.com/lit/ds/symlink/csd88539nd.pdf?ts=1706170848347&ref_url=https%253A%252F%252Fwww.mouser.kr%252F) (accessed Dec. 7 2023)
- [7] Adafruit Industries, “Adafruit Feather M4 Express”, Adafruit Feather M4 Express datasheet, Nov. 2023, <https://www.adafruit.com/product/3857#technical-details> (accessed Dec. 7 2023)



## 4.2.7 Revision Table

12/7/2023	Nicholas Kim: Initial outline (all sections)
1/24/2024	Nicholas Kim: Revised block description and interfaces
1/25/2024	Nicholas Kim: Replaced TPIC2601 with CSD88539ND
1/28/2024	Nicholas Kim: Moved MOSFET from mallet driver block to solenoid block; updated description, validation, and interface table
3/8/2024	Nicholas Kim: Revised all sections according to final design and instructor feedback

## 4.3 Music Performance Code

### 4.3.1. Description

This code block is responsible for all MIDI input data processing, autonomous note sequence generation, and all SPI message configuration for the system. This especially ensures the system will meet our MIDI input and autonomous performance requirements. It works by configuring the microcontroller to be able to send and receive signals via the microcontroller's I/O pins to and from the mallet driver block and the various distance sensors in the system. All code is written in the Arduino IDE.

It has two main modes of operation, autonomous mode and real-time MIDI mode. The code configures the microcontroller to read I/O pin 5 to determine whether the autonomous mode is on or off. If pin 5 is low, it is in MIDI mode which will process USB MIDI data in real-time over the usb connection, receiving signals such as note value, note velocity, and modulation value from the data packets being sent. These signals will be then processed and appropriate 8-bit SPI messages will be sent through the MOSI, SCLK, and CS pins. The messages are in a 0bXX00\_0000 format, where the first two bits are "don't cares" and the next represent gates 5-0 for a given pre-fet driver TPIC chip [1]. For context, there will be four different CS (chip select) signals/pins, each of which enable and disable input for one of four separate and identical TPIC chips. Each TPIC drives two notes, which have 3 solenoids each. This makes for a total of 24 solenoids in the system. More details can be found in the mallet driver block description.

The second mode is an autonomous performance mode, which uses Markov matrices and probability distributions to generate a song, or a sequence of notes, by using parameters set within this block. These parameters include time signature, a discrete "energy level" which can either be calm, moderate, or upbeat (0, 1, 2), and beats per minute (bpm). Each of these parameters can be adjusted within the code to change how the autonomous performance mode behaves. The autonomous mode will be turned on via an external switch which is checked with a digitalRead() function. As long as pin 5 is high, note sequences will be generated then

performed. The music performance code will call functions from the MIDI/Autonomous code blocks to perform these tasks. The five functions that are called from within this block are shown in the interface properties section.

#### 4.3.2. Design

The diagram in Figure 1 depicts a black box diagram which shows the interfaces between this block and other blocks within the system. The right interface represents how the code will be uploaded to the microcontroller while the left interface represents the various function calls and returns that happen between this block and the autonomous midi code library block.

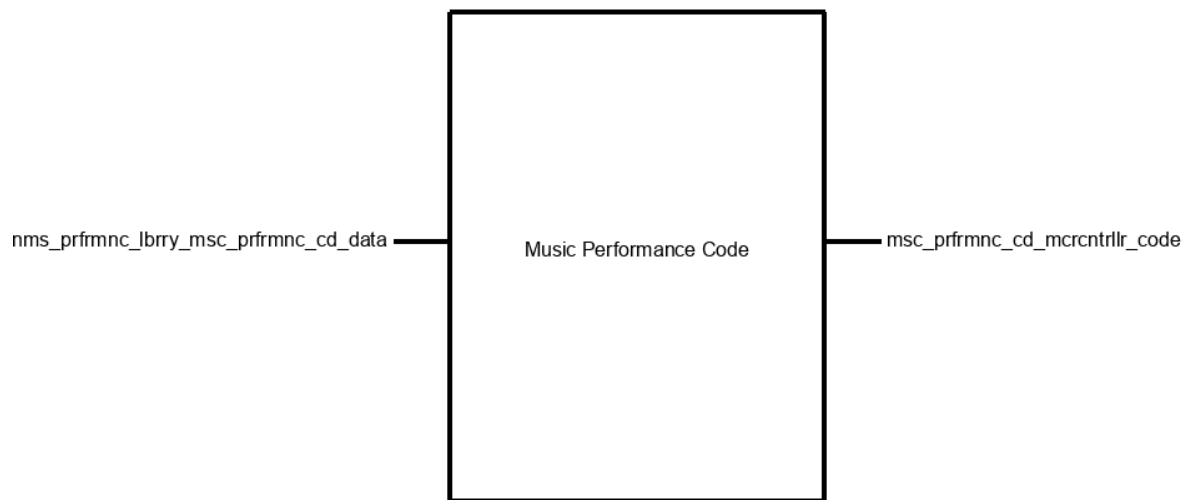


Figure 1: Music Performance Code Black Box Diagram

The pseudocode below outlines how the flow of the main loop works and what needs to be configured in the setup routine.

#### **Pseudocode:**

##### Setup/Initialization Routine:

1. begin serial at 115200 baud rate for checking output in serial monitor
2. call SPI begin function
3. configure AUTO\_PIN as input (pin 5)
4. configure FAULT\_PIN as input with internal pull-up resistor enabled (pin 9)
5. configure CS\_PIN0, ... , CS\_PIN3 as outputs (pins 10 to 13)
6. configure pin 6 (output enable) as output
7. set output enable and all CS\_PINS to HIGH logic level

##### Main Loop:

1. Check for and read any midi input data via read\_midi() function, store in Note struct
2. Update note timers by calling update\_note\_timers(note\_inactive\_arr, cur\_note) function

3. Check note timers by calling `check_note_timers(note_inactive_arr)` function
4. While `digitalRead(AUTO_PIN) == HIGH`
  - a. initialize `song_length`, `time_sig`, and `energy_level` variables
  - b. initialize bpm variable
  - c. allocate memory and initialize dynamic array for the note sequence aka song
  - d. initialize Prandom object for seeding the random number generator
  - e. call `autonomous_seq_generation(song, energy_level, song_length, time_sig, R)` function and store this back into `song`
    - i. note: this generates the sequence of notes, each having a `note_index`, `duration`, and `velocity`
  - f. call `perform_song(song, song_length, bpm)` function
    - i. note: this sends the SPI on/off messages to perform the song
  - g. delete `song` to free up allocated memory
  - h. wait 2 seconds to signal end of song

Figure 2: Music Performance Pseudocode

#### 4.3.3. General Validation

The initialization procedure handles configuring the input mode for all the required pins on the microcontroller as well as setting any initial values. In line 1, the baud rate for the serial monitor is relatively arbitrary, but matches examples seen in the MIDI USB library [2]. The SPI begin function initializes all default SPI pins needed for the communication protocol with the mallet driver TPIC chips. This ensures that SPI messages are able to be sent. The `AUTO_PIN` is defined as an input where it being HIGH (tied to 3.3V pin) or LOW (tied to GND) turns on/off the autonomous mode in this block. This ensures mode switching is configured correctly. As seen in line 4, The `FAULT_PIN` needs to be configured as an input with a pull-up because the TPIC chips will drive that logic level LOW when a fault is detected [1]. All CS pins are set as outputs so that they can be driven LOW or HIGH to tell the TPIC chips when to receive data [2]. They are set HIGH initially on line 7 to indicate no data is being sent yet. Pin 6 is the output enable pin which is configured as an output and is set HIGH in lines 6-7. This enables the TPICs to send output voltages [2]. The pseudocode initialization outlines the above behavior which ensures that the system is able to actuate the solenoids required to play the notes on the drum.

For the USB MIDI input mode, several libraries are used. An existing USB MIDI library obtainable through the Arduino IDE is used which configures the microcontroller to act like a USB MIDI device, enabling it to receive MIDI data packets via a USB connection to a laptop. The library is easy to use and is well-documented. While not shown in detail in this block, the contents of the `read_midi()` function are based on an example code snippet from the USB MIDI library which uses a `MidiUSB.read()` function to return a data packet [2]. For our system, we only care about the case when this data represents an event where a note is turned on, in which case a new `Note` struct is defined and populated with data (a note index, duration, and velocity). The populated note struct is then returned to the main loop. This behavior is handled by line 1 in the main loop of the pseudocode. The MIDI library contains other USB MIDI functions if we want to use them for future modifications. The `midi_read()` function calls many other functions within

the MIDI Autonomous Library block to handle sending SPI messages to the mallet block to actuate the solenoids.

One thing to note is that the solenoids need to be on for a fixed amount of time, which is handled via the `update_note_timers` and `check_note_timers` functions. Each of the 8 notes on the drum correspond to a certain timer and index in the `note_inactive_array` as seen in the pseudocode. The `update_note_timers` function called in line 2 of the main loop updates the timers corresponding to the locations in the `note_inactive_array` which contain a 1 (meaning that corresponding note is inactive). This means that notes which are currently being actuated (their solenoids are on) won't have their timers updated. Now, in line 3 of the main loop, the timers are checked to see if any notes which are active (have 0 in their corresponding index in the `note_inactive_arr`) have been on for longer than 60ms by subtracting their timer from the current program time via the `millis()` function. If so, the midi autonomous library block handles turning off the appropriate solenoids as that condition is met.

Another design that was considered and tested was to use simple delay functions to actuate the solenoids for 60ms rather than the timer array method shown above. The issue with this method is that the Arduino delay function uses busy-wait cycles, meaning that no other processing is done while the program is executing the delay. This makes it impossible to receive multiple MIDI messages "simultaneously", meaning the system would not be able to play two or more notes at the same time, since the first received note's wait period would need to be finished before the program jumps back to the main loop and checks for MIDI input again. Despite its shortcomings, one benefit of this method is that it was much simpler to implement since it didn't require updating/checking the note timers or recording which note is inactive via the `note_inactive_arr`.

Finally, in line 4 of the main loop pseudocode, the `AUTO_PIN` is checked and if it is HIGH, the program enters a while loop where the autonomous sequence generation is handled. First the necessary parameters are initialized such as `song_length`, `time_sig`, `energy_level`, and `bpm`. These are used later in the autonomous mode to generate and perform the song. Additionally, the song is allocated and the Prandom object is initialized, which ensures that the seed for the random number generation is different every time the program is initialized. Finally the song is generated via the `autonomous_seq_generation` function and subsequently performed by the `perform_song` function. These functions work by the principles of Markov matrices and various probability distributions [3].

#### 4.3.4. Interface Validation

<b>Interface Property</b>	<b>Why is this interface this value?</b>	<b>Why do you know that your design details <u>for this block</u> above meet or exceed each property?</b>
---------------------------	--	---

TABLE I:  
msc\_prfrmnc\_cd\_mrcntrllr\_code Interface Validation table

#### **msc\_prfrmnc\_cd\_mrcntrllr\_code : Output**

Other: Uses USB Micro B to USB A cable for uploading	These are the physical ports that connect the microcontroller to the PC for uploading the code.	The pseudocode above is designed to be implemented on an Adafruit Feather M4 Express microcontroller, which has a USB Micro B connector [6]
Other: Source code file types include .ino and .h	These are supported file types for the Arduino IDE, which crucially support the libraries used for this system [2] [4].	The pseudocode is designed to be implemented in a .ino file by using principles and libraries available to arduino [2] [4]
Other: Upload Baud Rate: 921600	This is the default upload baud rate as shown in the verbose upload output window [7].	When uploading the code onto the microcontroller, you can verify that the microcontroller is connected for uploading at 921600 baud

This table describes and validates the interface properties between the microcontroller and the Music Performance Block.

TABLE II:  
mdtnms\_prfrmnc\_lbrry\_msc\_prfrmnc\_cd\_data Interface Validation Table

#### **mdtnms\_prfrmnc\_lbrry\_msc\_prfrmnc\_cd\_data : Input**

Messages: Note read_midi ()	This returns a note struct for a given input data packet from USB MIDI protocol [2] which is used to update the note timers.	Makes use of the MidiUSB.read() function within the library along with other functions to receive USB MIDI data and determines/returns a note struct which stores the note_index (corresponds to a given note on the drum), duration, and velocity
-----------------------------	--	--

<p>Messages: void perform_song (Note* song, int song_length, int bpm)</p>	<p>This function needs the array of Note structs called song and song_length which represents the number of notes in the song. The bpm is also necessary so that the function knows how fast to play notes. It does not need to return anything, so it is void.</p>	<p>Once the song array is passed into the function, there are other function calls within this function to various SPI functions [4] which send on/off messages and in turn actuate the solenoids for a given note at appropriate times based on the duration of each note and the bpm, which are also passed into this function.</p>
<p>Messages: void update_note_timers(int note_inactive_arr[], Note cur_note)</p>	<p>This function needs the note_inactive_arr which represents which notes on the drum are currently being actuated, and the cur_note (note that was just received by MIDI input in order to know which timers to update and which timers to not.</p>	<p>This function then takes the array and cur_note and updates timers appropriately by making use of the millis() function and simple conditionals statements.</p>
<p>Messages: Note* autonomous_seq_generation(Note* song, int energy_level, int song_length, int time_sig)</p>	<p>This function needs the song array to put notes into, the energy_level to determine which markov matrix to use, the song_length to know how long the song is, and the time_sig to know how many phrases to generate and in which pattern. It returns the array back to the main function to be then performed.</p>	<p>This function makes use of Markov matrices, programmed probability distributions, and the probability library Prandom [8] to generate a song, varying each note parameter (note_index, duration, and velocity). This ensures that the song is different each time it is generated and does not rely on MIDI input.</p>
<p>Messages: void check_note_timers(int note_inactive_arr[])</p>	<p>This function needs the note_inactive array to know which timers to check (the notes that are flagged as active need to be checked) and to be able to set notes as active/inactive. It is void as it just modifies the passed array.</p>	<p>This function checks the note timers for all active notes and if they exceed the pre-set solenoid on time, it flags them as inactive again and calls other functions to send the SPI off message. It then returns to main</p>

This table describes and validates the interface properties between the MIDI/Autonomous Performance Library and the Music Performance Block.

#### 4.3.5. Verification Plan

1. Set up Digital Audio Workstation software on a computer to play each possible note, and for each note to play within each velocity range (0-39, 40-79, 80-127)
2. Read MIDI input data and print each returned Note via serial monitor (note velocity, duration and note\_index)
  - a. As each note is received, within the SPI message functions in the autonomous library, print the associated SPI message and chip select via serial monitor. See that the SPI protocol library is in use.
    - i. ex: 0bXX111000 with chip select 0 being HIGH corresponds to note\_index 0 with velocity 3 being returned from the midi function
3. Print the note\_inactive\_arr to ensure that the triggered note is now flagged as inactive before the timers are updated and checked via the appropriate functions
4. Repeat steps 1-4 to play all possible notes at all possible velocity ranges (8 notes \* 3 velocity ranges) = 24 times.
5. Flip the autonomous mode switch connected to pin 5.
6. Initialize all parameters to appropriate values (song\_length (multiple of time\_sig\*4), time\_sig (1-4), energy\_level (1-3), bpm (1-120)).
7. Allocate memory for song
8. Call autonomous\_seq\_generation function with said parameters
9. Print the parameters of each generated Note (note\_index, duration, velocity) in the song
10. Then the perform\_song function is called.
  - a. As each Note in song is performed, print the contents of the struct (note\_index, velocity, duration) and verify all content is present
  - b. Within SPI functions, print SPI message and being sent and chip select variable via chip select functions to ensure they are in the correct format
    - i. ex: 0bXX101000 and chip\_select == 0 for note\_index == 0, velocity == 2,
11. Repeat steps 6-10 again to ensure that a different song is generated

#### 4.3.6. References and File Links

##### 4.3.6.1 References

- [1] Texas Instruments, "6-CHANNEL SERIAL AND PARALLEL LOW-SIDE PRE-FET DRIVER", SLIS055B, 2001.  
<https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/1103/TPIC46L01%2C02%2C03.pdf> (accessed Jan. 25, 2024).
- [2] Arduino-Libraries, "Arduino-libraries/MIDIUSB: A Midi Library over USB, based on PluggableUSB," GitHub, <https://github.com/arduino-libraries/MIDIUSB> (accessed Jan. 25, 2024).
- [3] H. Jung, "Making music: When simple probabilities outperform deep learning," Medium, <https://towardsdatascience.com/making-music-when-simple-probabilities-outperform-deep-learning-75f4ee1b8e69> (accessed Jan. 25, 2024).

- [4] "SPI," SPI - Arduino Reference, <https://www.arduino.cc/reference/en/language/functions/communication/spi/> (accessed Jan. 28, 2024).
- [5] "Midi Basics," Acoustica, <https://acoustica.com/mixcraft-10-manual/midi-basics> (accessed Jan. 28, 2024).
- [6] Adafruit Industries, "Adafruit Feather M4 Express - Featuring ATSAMD51," Adafruit.com, 2019. <https://www.adafruit.com/product/3857>
- [7] "Use verbose output in the Arduino IDE ," Arduino, Jan. 29, 2024. <https://support.arduino.cc/hc/en-us/articles/4407705216274-Use-verbose-output-in-the-Arduino-IDE> (accessed Mar. 07, 2024).
- [8] "Prandom - Arduino Reference," www.arduino.cc. <https://www.arduino.cc/reference/en/libraries/prandom/> (accessed Mar. 08, 2024).

#### 4.3.7. Revision Table

01/25/2024	Liam Warner: Created Rough Draft of New Block configuration
01/28/2024	Liam Warner: Edited and finished rough draft
03/07/2024	Liam Warner: Made revisions based on feedback from rough draft assignment: <ul style="list-style-type: none"> <li>- Made direct statement at start of description</li> <li>- Divided content into more paragraphs</li> <li>- Created "line" numbers for the pseudocode section and added initialization section</li> <li>- Referenced design in validation section</li> <li>- Discussed alternate approach</li> <li>- Used better IEEE formatting</li> </ul>

## 4.4 MIDI/Autonomous Performance Library

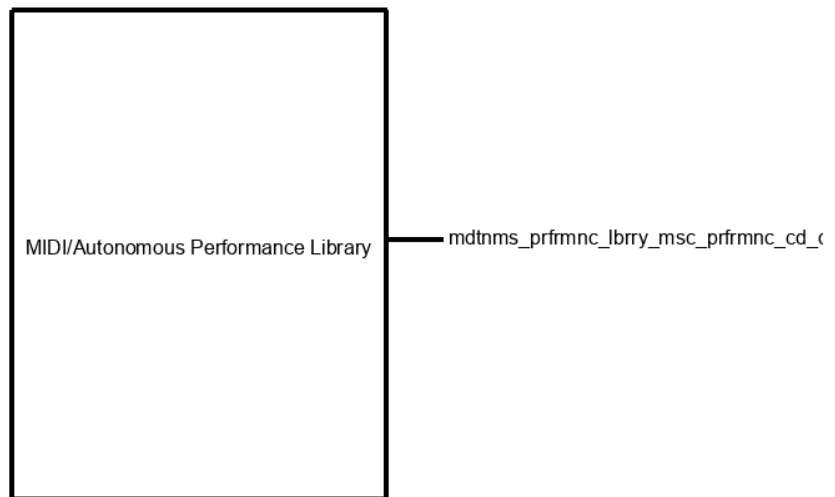
### 4.4.1 Description

This code block encapsulates all algorithms and use of libraries that are responsible for the drum's ability to play autonomously and live via MIDI control. For the autonomous part of the library, the music performance code will have an autonomous mode variable that is "turned on" by an external switch. Then, the functions (properties of the interface) that handle music sequence generation will be called by the music performance code block, and the library will use parameters and predetermined probability arrays/matrices to start creating note sequences to be stored in an array as a song. The block crucially contains an `autonomous_seq_generation` function which generates a song to perform, and a `perform_song` function which is designed to take the generated song and perform it by formatting SPI messages to be sent to the mallet



driver block at the appropriate times. This code block also supplies functions for formatting and sending SPI messages for incoming MIDI messages via the externally sourced USB MIDI library and via other simple functions which streamlines the code in the music performance block. The main function here is the readMidi() function, which is called in the Music Performance block and returns a Note struct which contains three parameters: note index, velocity, and duration. Additionally, there are two functions that deal with updating and checking timers for each note on the drum to ensure the solenoids are on for the appropriate amount of time.

#### 4.4.2 Design



#### Real-time MIDI Control Functions Pseudocode:

```
pitch_to_note_name(pitch)
    names = [all possible note names]
    note_name_index = pitch % 12
    return names[note_name_index]
byte get_spi_message(pitch, velocity)
    pitch determines chip select number and which 3 bits to address SPI message
    velocity determines which of the 3 bits to set as 1s/0s
    return spi_message
set_chip_select_high(pitch)
    get chip select number based on pitch
    set pin HIGH based on chip select number
set_chip_select_low(pitch)
    get chip select number based on pitch
    set pin LOW based on chip select number
NoteOn(channel, pitch, velocity)
    spi_message = get_spi_message(pitch, velocity)
    set_chip_select_high (pitch)
    send spi_message to mallet driver to turn solenoids on
```

```

    delay for a bit to allow solenoids to fully actuate
    send another SPI message to turn solenoids off
    set_chip_select_low (pitch)
ControlChange(channel, control, value)
    Serial.print(channel)
    Serial.print(control)
    Serial.print(value)

```

#### Autonomous Generation Functions:

```

struct Note {
    int note_index;
    float duration;
    int velocity;
};
const int available_notes[8] = {60, 62, 63, 67, 69, 72, 74, 75};
const float start_note_prob_array[8] = {0.35, 0.05, 0.05, 0.1, 0.05, 0.30, 0.05, 0.05};
const float next_note_prob_matrix_1[8][8] = {{0.2, 0.2, 0.2, 0.15, 0.05, 0.1, 0.05, 0.05},
        {0.35, 0.2, 0.2, 0.05, 0.05, 0.05, 0.05, 0.05},
        {0.2, 0.2, 0.2, 0.15, 0.05, 0.1, 0.05, 0.05},
        {0.35, 0.2, 0.05, 0.05, 0.05, 0.2, 0.05, 0.05},
        {0.05, 0.2, 0.05, 0.35, 0.05, 0.05, 0.2, 0.2},
        {0.1, 0.1, 0.1, 0.2, 0.05, 0.15, 0.1, 0.2},
        {0.05, 0.05, 0.05, 0.2, 0.05, 0.35, 0.05, 0.2},
        {0.05, 0.05, 0.05, 0.2, 0.05, 0.35, 0.2, 0.05}};

int getStartNoteIndex()
    determine starting note index via preset discrete probability distribution
    returns starting note index
int getNextNoteIndex(int currentRow)
    determine next note index from current note (corresponds to current row)

```

#### 4.4.3 General Validation

This block is responsible for the definition of all functions called by the Music Performance Code block. Its main purpose is to streamline the code presented in the Music Performance block, making it easy to modify the most important parameters for autonomous music generation, and to make it easy to understand how the USB MIDI data is being read. The library also contains a multitude of functions that are called by other functions in the library, further increasing modularity. This is generally good coding practice. The interface between this block and the music performance code represents function calls made by the music performance code block. All other processing is done internal to this block.

The MIDI control functions mask the MIDI data, sending SPI messages only when a note that matches a note on the drum is received. These notes include: C4, D4, Eb4, G4, A4, C5, D5,

and Eb5. These correspond to a value from 0-127 within the MIDI code, 60 is C4, 62 is D4, and so on, with sometimes a +/- 12 octave offset depending on the software being used. These values are also used to set specific chip select pins high via the chip select functions above.

The autonomous generation functions are a bit complex. The pseudocode above outlines their functionality. They work by taking in the main parameters from the autonomous\_seq\_generation function (energy\_level, song\_length, and time\_signature), and use them to generate a sequence, incorporating the markov matrix, and other discrete probability distributions used for velocity and duration. The generation process starts a new phrase if the end of a phrase (4-bar section within the song) is reached. The process also checks for large leaps between notes (4 or more notes away on the drum) and implements a simple resolution pattern to make the song sound less jarring. Overall, this creates a relatively pleasant and tame sequence which should be interesting to listen to.

One alternate method considered for autonomous generation is pure randomness, which does not sound musical at all. It does not reflect how humans may generate music, and lacks any melodic or rhythmic structure. Another alternate method considered is using machine learning models such as deep learning to generate the sequence, but this was deemed to be out of the scope for this project considering our timeline. Comparisons to the above method and deep learning show that using markov matrices actually implement better structure within the song as well [2].

#### 4.4.4. Interface Validation

Interface Property	Why is this interface this value?	Why do you know that your design details <u>for this block</u> above meet or exceed each property?
--------------------	-----------------------------------	--

TABLE I:  
mdtnms\_prfrmnc\_lbrry\_msc\_prfrmnc\_cd\_data Interface Validation Table

#### mdtnms\_prfrmnc\_lbrry\_msc\_prfrmnc\_cd\_data : Input

Messages: Note read_midi ()	This returns a note struct for a given input data packet from USB MIDI protocol [2] which is used to update the note timers.	Makes use of the MidiUSB.read() function within the library along with other functions to receive USB MIDI data and determines/returns a note struct which stores the note_index (corresponds to a given note on the drum), duration, and velocity
--------------------------------	--	--

<p>Messages: void perform_song (Note* song, int song_length, int bpm)</p>	<p>This function needs the array of Note structs called song and song_length which represents the number of notes in the song. The bpm is also necessary so that the function knows how fast to play notes. It does not need to return anything, so it is void.</p>	<p>Once the song array is passed into the function, there are other function calls within this function to various SPI functions [4] which send on/off messages and in turn actuate the solenoids for a given note at appropriate times based on the duration of each note and the bpm, which are also passed into this function.</p>
<p>Messages: void update_note_timers(int note_inactive_arr[], Note cur_note)</p>	<p>This function needs the note_inactive_arr which represents which notes on the drum are currently being actuated, and the cur_note (note that was just received by MIDI input in order to know which timers to update and which timers to not.</p>	<p>This function then takes the array and cur_note and updates timers appropriately by making use of the millis() function and simple conditionals statements.</p>
<p>Messages: Note* autonomous_seq_generation(Note* song, int energy_level, int song_length, int time_sig)</p>	<p>This function needs the song array to put notes into, the energy_level to determine which markov matrix to use, the song_length to know how long the song is, and the time_sig to know how many phrases to generate and in which pattern. It returns the array back to the main function to be then performed.</p>	<p>This function makes use of Markov matrices, programmed probability distributions, and the probability library Prandom [8] to generate a song, varying each note parameter (note_index, duration, and velocity). This ensures that the song is different each time it is generated and does not rely on MIDI input.</p>
<p>Messages: void check_note_timers(int note_inactive_arr[])</p>	<p>This function needs the note_inactive array to know which timers to check (the notes that are flagged as active need to be checked) and to be able to set notes as active/inactive. It is void as it just modifies the passed array.</p>	<p>This function checks the note timers for all active notes and if they exceed the pre-set solenoid on time, it flags them as inactive again and calls other functions to send the SPI off message. It then returns to main</p>

This table describes and validates the interface properties between the MIDI/Autonomous Performance Library and the Music Performance Block.

#### 4.4.5. Verification Plan

This is the same as the Music Performance Code block as it checks the same interfaces.

1. Set up Digital Audio Workstation software on a computer to play each possible note, and for each note to play within each velocity range (0-39, 40-79, 80-127)
2. Read MIDI input data and print each returned Note via serial monitor (note velocity, duration and note\_index)
  - a. As each note is received, within the SPI message functions in the autonomous library, print the associated SPI message and chip select via serial monitor. See that the SPI protocol library is in use.
    - i. ex: 0bXX111000 with chip select 0 being HIGH corresponds to note\_index 0 with velocity 3 being returned from the midi function
3. Print the note\_inactive\_arr to ensure that the triggered note is now flagged as inactive before the timers are updated and checked via the appropriate functions
4. Repeat steps 1-4 to play all possible notes at all possible velocity ranges (8 notes \* 3 velocity ranges) = 24 times.
5. Flip the autonomous mode switch connected to pin 5.
6. Initialize all parameters to appropriate values (song\_length (multiple of time\_sig\*4), time\_sig (1-4), energy\_level (1-3), bpm (1-120)).
7. Allocate memory for song
8. Call autonomous\_seq\_generation function with said parameters
9. Print the parameters of each generated Note (note\_index, duration, velocity) in the song
10. Then the perform\_song function is called.
  - a. As each Note in song is performed, print the contents of the struct (note\_index, velocity, duration) and verify all content is present
  - b. Within SPI functions, print SPI message and being sent and chip select variable via chip select functions to ensure they are in the correct format
    - i. ex: 0bXX101000 and chip\_select == 0 for note\_index == 0, velocity == 2,
11. Repeat steps 6-10 again to ensure that a different song is generated

#### 4.4.6. References and File Links

##### 4.4.6.1. References

- [1] Texas Instruments, "6-CHANNEL SERIAL AND PARALLEL LOW-SIDE PRE-FET DRIVER", SLIS055B, 2001.  
<https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/1103/TPIC46L01%2C02%2C03.pdf> (accessed Jan. 25, 2024).
- [2] Arduino-Libraries, "Arduino-libraries/MIDIUSB: A Midi Library over USB, based on PluggableUSB," GitHub, <https://github.com/arduino-libraries/MIDIUSB> (accessed Jan. 25, 2024).

- [3] H. Jung, “Making music: When simple probabilities outperform deep learning,” Medium, <https://towardsdatascience.com/making-music-when-simple-probabilities-outperform-deep-learning-75f4ee1b8e69> (accessed Jan. 25, 2024).
- [4] “SPI,” SPI - Arduino Reference, <https://www.arduino.cc/reference/en/language/functions/communication/spi/> (accessed Jan. 28, 2024).
- [5] “Midi Basics,” Acoustica, <https://acoustica.com/mixcraft-10-manual/midi-basics> (accessed Jan. 28, 2024).
- [6] Adafruit Industries, “Adafruit Feather M4 Express - Featuring ATSAMD51,” Adafruit.com, 2019. <https://www.adafruit.com/product/3857>
- [7] “Use verbose output in the Arduino IDE ,” Arduino, Jan. 29, 2024. <https://support.arduino.cc/hc/en-us/articles/4407705216274-Use-verbose-output-in-the-Arduino-IDE> (accessed Mar. 07, 2024).
- [8] “Prandom - Arduino Reference,” www.arduino.cc. <https://www.arduino.cc/reference/en/libraries/prandom/> (accessed Mar. 08, 2024).

#### 4.4.7.1 Revision Table

01/25/2024	Liam Warner: Created Rough Draft of New Block configuration
01/28/2024	Liam Warner: Edited and finished rough draft
03/12/2024	Liam Warner: Made edits to more closely match updated Music Performance Code Block <ul style="list-style-type: none"> <li>- Updated Verification plan</li> <li>- Updated Interface validation</li> <li>- Updated description</li> </ul>

## 4.5 Microcontroller

### 4.4.1 Description

This is an Adafruit Feather M4 Express microcontroller which acts as a processing unit for the Orchestrion. It interprets the USB MIDI data by way of the various code blocks in our system and outputs necessary signals to operate the actuators and tremolo via the SPI and/or digital I/O pins. It is powered via the 5V line in the micro USB input.

#### 4.4.2 Design

Below are several artifacts which outline the design considerations surrounding the microcontroller. The pinout diagram in Fig 1 indicates how many of which pin there are on the microcontroller, which is important for ensuring that the full capabilities of the mallet drivers and distance sensors are supported. The stacking header image in Fig 2 shows that the microcontroller can connect to the PCB and via the male headers and the female side can be used for any other additional connectivity.

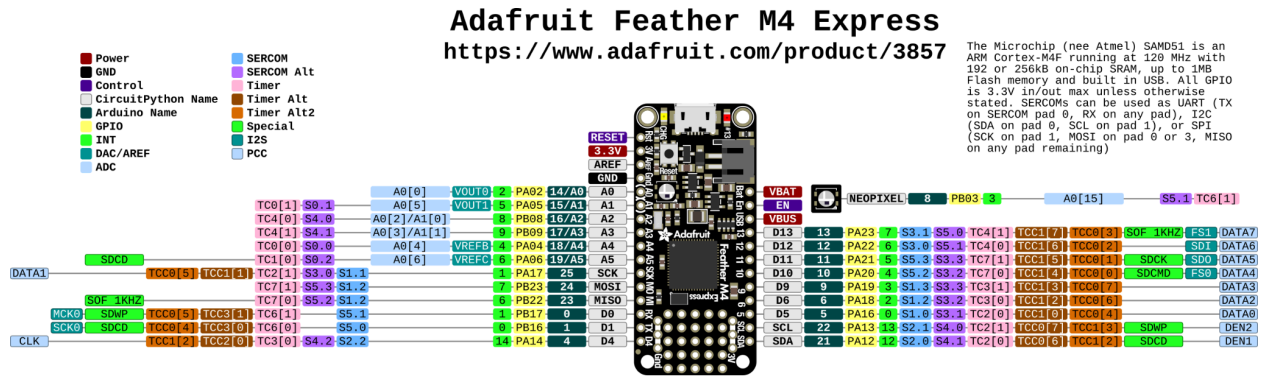


Fig 1: Adafruit Feather M4 Express Pinout Diagram

The diagram in Fig 1 shows the pinout capabilities of the Feather M4 Express. The key in the top left designates what type of input or output a pin can be.

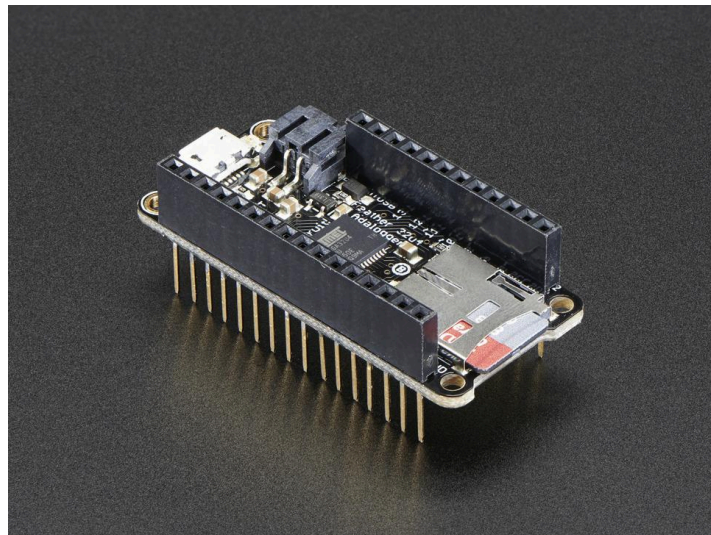


Fig 2: Adafruit Feather M4 Express Stacking Headers

The diagram in Fig 2 shows how stacking headers can be mounted onto the Feather M4 Express. The male headers below the controller are how the microcontroller will connect to the PCB in our design.

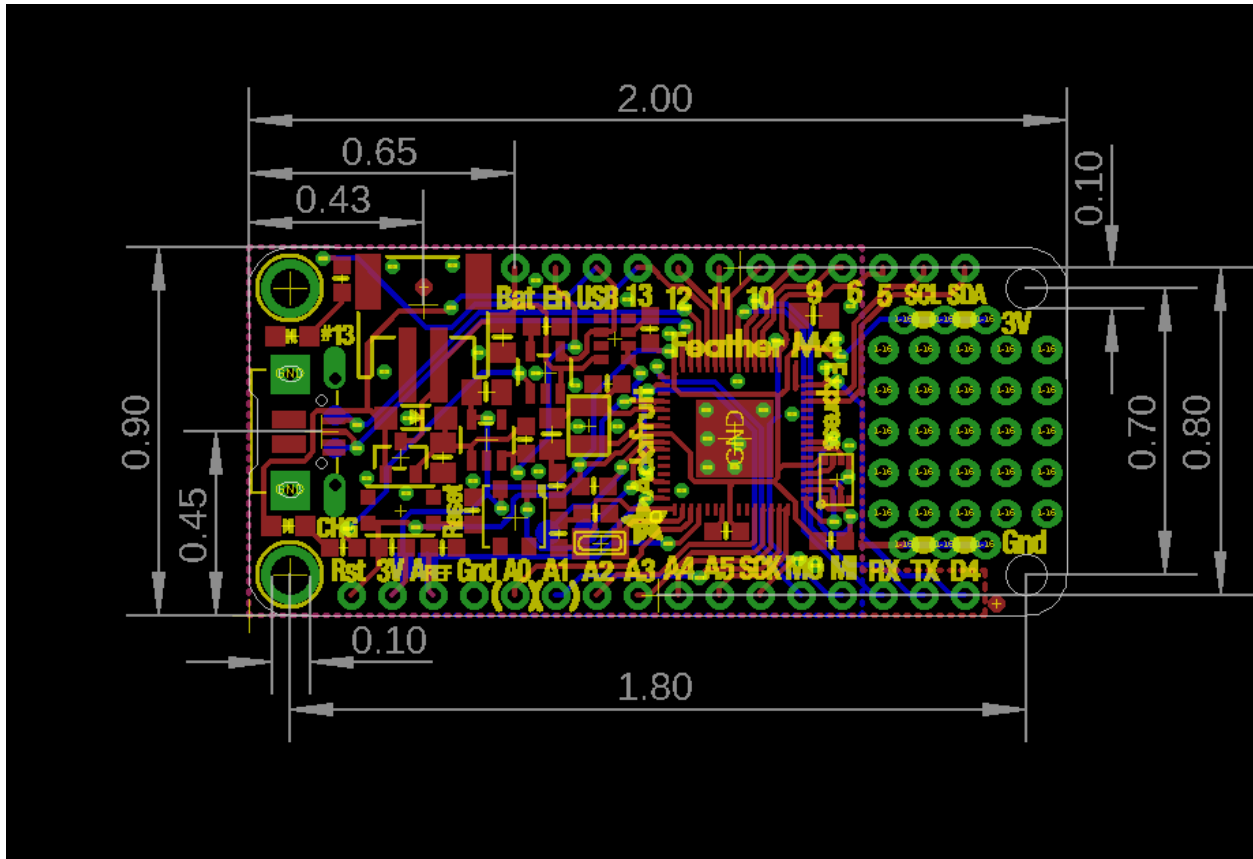


Fig 3: Adafruit Feather M4 Express Dimensions

The diagram in Fig 3 shows the dimensions of the Feather M4 necessary for ensuring that the female header holes in the PCB are lined up properly.

#### 4.4.3 General Validation

The main reason that this microcontroller works for this project is that the ARM Cortex M4 chip has native USB support [8]. This allows for use of the USB MIDI library and for a computer to recognize the device as a MIDI USB device. It also has Arduino IDE support, which was a request from our Project's customer. It also has enough I/O pins to support all of the connectivity required for the distance sensors and mallet drivers.



#### 4.4.4. Interface Validation

The tables below validate all interfaces to and from the microcontroller within the system.

<b>Interface Property</b>	<b>Why is this interface this value?</b>	<b>Why do you know that your design details for this block above meet or exceed each property?</b>
---------------------------	--	--

TABLE I:  
Fault Detection Output Interface Property Validation

**mlt\_drvr\_mrcntrlr\_dsig : Output**

Logic-Level: 3.3V (active LOW)	Microcontroller requires 3.3V logic [7]; TPIC46L01 grounds fault on detection [1]	Pull-up resistors (Fig 2, 3) hold TPIC46L01 fault lines at 5V; SNx4HCT08 outputs 0V when any fault line falls to 0V [2]; TXB0108 translates 5V logic to 3.3V logic [???
Vmax: No fault: 3.3V	Microcontroller requires 3.3V logic [7]	SNx4HCT08 has 3.3V max HIGH level output voltage [2]
Vmax: Fault: 0.8V	Microcontroller requires 3.3V logic [7]	SNx4HCT08 has 0.8V max LOW level output voltage [2]
Vmin: No fault: 2V	Microcontroller requires 3.3V logic [7]	SNx4HCT08 has 2V min HIGH level output voltage [2]

TABLE II:  
SPI SCLK Input Interface Property Validation

**mrcntrlr\_mlt\_drvr\_dsig : Input**

Logic-Level: 3.3V (active HIGH SCLK)	Microcontroller outputs 3.3V logic [7]	TXB0108 translates 3.3V logic to 5V logic [3] required by TPIC46L01 [1]
Max Frequency: 10MHz (max clock frequency)	TPIC46L01 has 10MHz max max SCLK frequency [1]	TPIC46L01 has 10MHz max max SCLK frequency [1]
Vmax: HIGH: 5V	Microcontroller outputs 3.3V logic [7]	TXB0108 has unbounded max HIGH level input voltage [3]

Vmax: LOW: 0.8V	Microcontroller outputs 3.3V logic [7]	TXB0108 has 0.8V max LOW level input voltage [3]
Vmin: HIGH: 2V	Microcontroller outputs 3.3V logic [7]	TXB0108 has 2V min HIGH level input voltage [3]

TABLE III:  
SPI MOSI Input Interface Property Validation

**mcrntrlr\_mllt\_drvr\_dsig : Input**

Logic-Level: 3.3V (active HIGH MOSI/SDI)	Microcontroller outputs 3.3V logic [7]	TXB0108 translates 3.3V logic to 5V logic [3] required by TPIC46L01 [1]
Vmax: LOW: 0.8V	Microcontroller outputs 3.3V logic [7]	TXB0108 has 0.8V max LOW level input voltage [3]
Vmax: HIGH: 5V	Microcontroller outputs 3.3V logic [7]	TXB0108 has unbounded max HIGH level input voltage [3]
Vmin: HIGH: 2V	Microcontroller outputs 3.3V logic [7]	TXB0108 has 2V min HIGH level input voltage [3]

Table IV:  
SPI CS Input Interface Property Validation

**mcrntrlr\_mllt\_drvr\_dsig : Input**

Logic-Level: 3.3V (active LOW chip select)	Microcontroller outputs 3.3V logic [7]	TXB0108 translates 3.3V logic to 5V logic [3] required by TPIC46L01 [1]
Vmax: HIGH: 5V	Microcontroller outputs 3.3V logic [7]	TXB0108 has unbounded max HIGH level input voltage [3]
Vmax: LOW: 0.8V	Microcontroller outputs 3.3V logic [7]	TXB0108 has 0.8V max LOW level input voltage [3]
Vmin: HIGH: 2V	Microcontroller outputs 3.3V logic [7]	TXB0108 has 2V min HIGH level input voltage [3]

TABLE V:  
 msc\_prfrmnc\_cd\_mrcntrlr\_code Interface Validation table

**msc\_prfrmnc\_cd\_mrcntrlr\_code : Output**

Other: Uses USB Micro B to USB A cable for uploading	These are the physical ports that connect the microcontroller to the PC for uploading the code.	The pseudocode above is designed to be implemented on an Adafruit Feather M4 Express microcontroller, which has a USB Micro B connector [8]
Other: Source code file types include .ino and .h	These are supported file types for the Arduino IDE, which crucially support the libraries used for this system [9, 12, 15].	The pseudocode is designed to be implemented in a .ino file by using principles and libraries available to arduino [9, 12, 15]
Other: Upload Baud Rate: 921600	This is the default upload baud rate as shown in the verbose upload output window [14].	When uploading the code onto the microcontroller, you can verify that the microcontroller is connected for uploading at 921600 baud

TABLE VI:  
 Micro USB Connection to Microcontroller Interface Validation

**otsd\_mrcntrlr\_comm : Input**

Messages: Note Value (0-127)	This value indicates which note is being sent over the MIDI USB interface [13]	The microcontroller can receive these messages over the micro USB interface via the USB MIDI library
Messages: Velocity Value (0-127)	This value indicates which dynamic/velocity level is being sent over the MIDI USB interface for a given note [13]	The microcontroller can receive these messages over the micro USB interface via the USB MIDI library
Protocol: USB MIDI	This is the protocol as defined by the USB MIDI library which allows the microcontroller to act as a MIDI device [9]	The SAMD architecture of the Cortex M4 supports native USB which is the key requirement for this protocol

Vnominal: 5V	This is the nominal voltage level for power over USB [16]	The Feather M4 Microcontroller uses USB protocol for all power input needs
--------------	---	--

TABLE VII:  
Autonomous Mode Switch Interface Validation

**otsd\_mcrctrllr\_usrin : Input**

Other: Label indicating autonomous mode is next to switch	This indicates to the user what the switch is for	There is a silkscreen label on the switch on the PCB which indicates this
Timing: Takes <10 seconds to flip switch	When the system is fully assembled, flipping the switch on the PCB requires reaching under the drum with a long object which is why it takes up to 10 seconds	A user should be able to flip the switch in this time period
Type: Mode Switch	The switch will switch back and forth from USB MIDI input to autonomous mode	The code uploaded onto the microcontroller configures the pin this switch is mapped to to have it switch modes when the voltage is high versus low

4.4.5. Verification Plan

1. Set up a computer with MIDI software.
2. Set up the system with one note's 3 solenoids disconnected from the PCB.
3. Ensure that the power supply is connected, and only turned on **after** the system is fully assembled.
4. Plug in the microcontroller to the computer with the autonomous mode switch OFF.
5. Upload the code with verbose mode on and verify that the baud rate, cable, and file types match the interface validation table.
6. Configure MIDI software to play notes on the drum with their solenoids connected.
7. Play MIDI note sequence. Ensure that the appropriate solenoids are actuating.
8. Probe SPI and CS pins to verify those interfaces.
9. Now play MIDI note which has its solenoids disconnected.
10. Ensure the system responds to fault signal appropriately by disabling input to the mallet drivers.
11. Probe fault signal to verify that interface.

12. Reset system by switching power supply off and unplugging microcontroller. Plug microcontroller back in and power on the power supply again.
13. Close MIDI software.
14. Flip the autonomous switch and verify that it can be flipped in less than 10 seconds.
15. Ensure that the system is playing notes autonomously.

#### 4.5.6. References and File Links

##### 4.5.6.1. References

- [1] Texas Instruments, “6-Channel Serial And Parallel Low-Side Pre-FET Driver,” TPIC46L01 datasheet, Nov. 1996 [Revised Aug. 2001], [https://www.ti.com/lit/ds/symlink/tpic46l01.pdf?ts=1701936974753&ref\\_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTPIC46L01](https://www.ti.com/lit/ds/symlink/tpic46l01.pdf?ts=1701936974753&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTPIC46L01) (accessed Dec. 7 2023)
- [2] Texas Instruments, “SNx4HCT08 Quadruple 2-Input Positive-AND Gates,” SNx4HCT08 datasheet, Nov. 1988 [Revised Oct. 2022], <https://www.ti.com/lit/ds/scls063f/scls063f.pdf?ts=1706149182318> (accessed Jan. 24 2024)
- [3] Texas Instruments, “TXB0108 8-Bit Bidirectional Voltage-Level Translator with Auto-Direction Sensing and  $\pm 15$ -kV ESD Protection,” TXB0108 datasheet, Nov. 2006 [Revised Aug. 2020], [https://www.ti.com/lit/ds/symlink/txb0108.pdf?ts=1709917608320&ref\\_url=https%253A%252F%252Fwww.mouser.com%252F](https://www.ti.com/lit/ds/symlink/txb0108.pdf?ts=1709917608320&ref_url=https%253A%252F%252Fwww.mouser.com%252F) (accessed Mar. 8 2024)
- [4] Texas Instruments, “4-Channel Serial And Parallel Low-Side Pre-FET Driver,” TPIC44L01 datasheet, Nov. 1996 [Revised Aug. 2001], <https://www.ti.com/lit/ds/symlink/tpic44l01.pdf?ts=1709943806716> (accessed Mar 8. 2024)
- [5] Texas Instruments, “SNx4HC595 8-Bit Shift Registers With 3-State Output Registers,” SNx4HC595datasheet, Nov. 1988 [Revised Oct. 2022], <https://www.ti.com/lit/ds/symlink/sn74hct595.pdf?ts=1709902865074> (accessed Mar. 8 2024)
- [6] Texas Instruments, “CSD88539ND Dual 60 V N-Channel NexFET Power MOSFETs,” CSD88539ND datasheet, Feb. 2014 [Revised Dec. 2023], [https://www.ti.com/lit/ds/symlink/csd88539nd.pdf?ts=1706170848347&ref\\_url=https%253A%252F%252Fwww.mouser.kr%252F](https://www.ti.com/lit/ds/symlink/csd88539nd.pdf?ts=1706170848347&ref_url=https%253A%252F%252Fwww.mouser.kr%252F) (accessed Dec. 7 2023)
- [7] Adafruit Industries, “Adafruit Feather M4 Express”, Adafruit Feather M4 Express datasheet, Nov. 2023, <https://www.adafruit.com/product/3857#technical-details> (accessed Dec. 7 2023)

- [8] “Adafruit Feather M4 Express,” Adafruit Learning System.  
<https://learn.adafruit.com/adafruit-feather-m4-express-atsamd51/overview>
- [9] Arduino-Libraries, “Arduino-libraries/MIDIUSB: A Midi Library over USB, based on PluggableUSB,” GitHub, <https://github.com/arduino-libraries/MIDIUSB> (accessed Jan. 25, 2024).
- [10] Texas Instruments, “6-CHANNEL SERIAL AND PARALLEL LOW-SIDE PRE-FET DRIVER”, SLIS055B, 2001.  
<https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/1103/TPIC46L01%2C02%2C03.pdf> (accessed Jan. 25, 2024).
- [11] H. Jung, “Making music: When simple probabilities outperform deep learning,” Medium, <https://towardsdatascience.com/making-music-when-simple-probabilities-outperform-deep-learning-75f4ee1b8e69> (accessed Jan. 25, 2024).
- [12] “SPI,” SPI - Arduino Reference,  
<https://www.arduino.cc/reference/en/language/functions/communication/spi/> (accessed Jan. 28, 2024).
- [13] “Midi Basics,” Acoustica, <https://acoustica.com/mixcraft-10-manual/midi-basics> (accessed Jan. 28, 2024).
- [14] “Use verbose output in the Arduino IDE ,” Arduino, Jan. 29, 2024.  
<https://support.arduino.cc/hc/en-us/articles/4407705216274-Use-verbose-output-in-the-Arduino-IDE> (accessed Mar. 07, 2024).
- [15] “Prandom - Arduino Reference,” [www.arduino.cc](http://www.arduino.cc).  
<https://www.arduino.cc/reference/en/libraries/prandom/> (accessed Mar. 08, 2024).
- [16] “USB,” Wikipedia, Mar. 19, 2022. <https://en.wikipedia.org/wiki/USB#Power>

#### 4.5.7. Revision Table

03/13/2024	Liam Warner: Created section and filled in necessary information to match the rest of the blocks in the project.
------------	--

## 4.6 Power Supply

### 4.6.1 Description

The 12V 30A DC Universal Regulated Switching Power Supply is an important part of the electrical system mentioned. It can accept both 110V and 220V AC inputs to meet world power standards. This trait is very important for systems that are used across borders because it lets them be flexible without the need for extra adapters. The power source changes the AC input to a steady 12V DC output, which effectively powers electronic parts and solenoids.

By incorporating an ON/OFF control, users can deactivate the system when not in use, thereby preventing electrical hazards and conserving energy.

Using a DC Buck Converter connection, the system effectively decreases the 12V DC output to 5V DC, thereby ensuring that low-voltage electronics function within acceptable voltage thresholds. Additionally, the system provides 12V DC power directly to 24 solenoids, enabling the functioning of mechanisms that require precise regulation, such as mechanical movements and valve actuation.

This electrical setup is essential for providing the system with the precise voltages and currents it needs, guaranteeing that each component receives the right amount of power for optimum performance.

### 4.6.2 Design

The black box diagram in Figure 1 defines the inputs and outputs of the Power Supply. Figure 2 and Figure 3 shows that the connection and process of the input and output of power supply . Figure4 is the over view of the 12V 30A DC Universal Regulated Switching Power Supply.

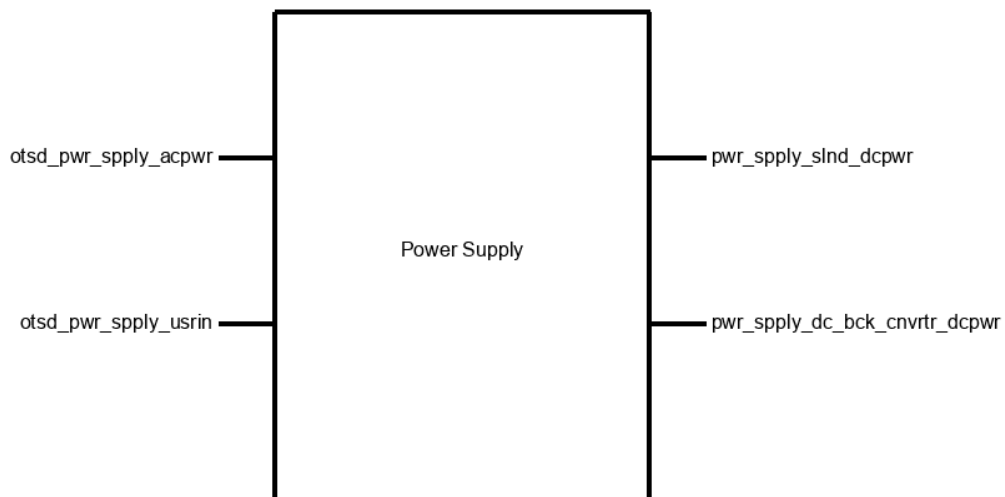


Fig 1. Black Box Diagram

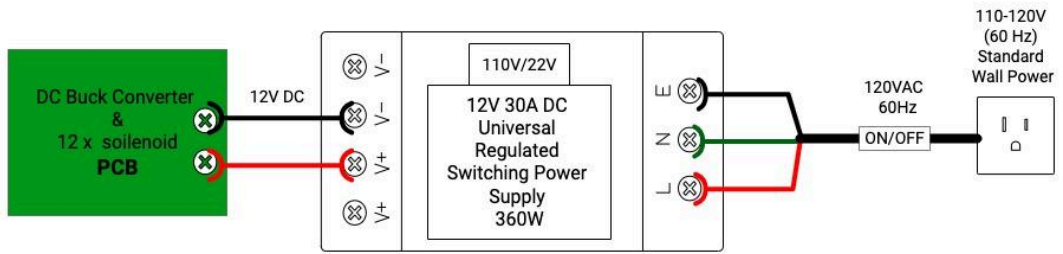


Fig 2. Power supply Connection

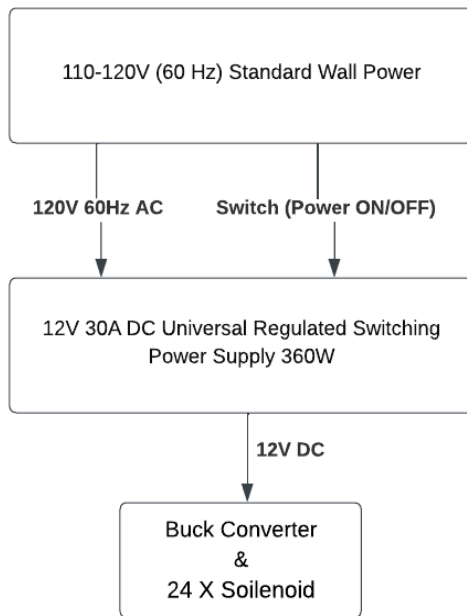
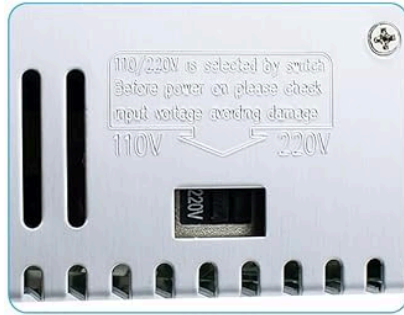


Fig 3. Process Diagram





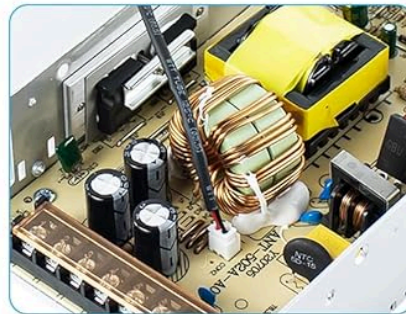
110V/220V must be selected by switch using to avoid damaging. Please set the switch to 110V for USA.



Three sets of output channel  
Can power many electronics.



Built in temperature detector, cooling fan will not run until the inner temperature reaches 113°F (45°C), which makes heat dissipation more efficient.



Safety protection system. Automatic overload cut-off, over Voltage cut-off, automatic thermal cut-off, short circuit protection.

Fig 4. 12V 30A DC Universal Regulated Switching Power Supply

Note:

otsd\_pwr\_spply\_acpwr: Input

Otsd\_pwr\_spply\_usrin: Input

Pwr\_spply\_slnd\_dcpwr: Output

Pwr\_spply\_dc\_bck\_cnvtr: Output

#### 4.6.3 General Validation

The 12V 30A DC Universal Regulated Switching Power Supply[1] was selected due to its performance-to-cost ratio. This component is economically accessible, resulting in cost reductions attributed to economies of scale. The widespread availability of such a power supply additionally streamlines the process of procuring replacement components, guaranteeing that any such needs will be met with minimal disruption. This aspect substantially enhances the system's longevity and maintenance.

When choosing the connection from the wall outlet to the power supply, opt for a 6ft 18 Gauge 3 Prong AC Power Cord Cable[4] with an on-off switch, pigtail (open-end wiring), and a NEMA 5-15 Plug suitable for US SJTW 18AWG standards. This cord can handle an input current of 10

Amps. The setup has been carefully selected to match the power supply's need for an AC input of 100-120VAC at 6.3A, guaranteeing a safe and dependable energy source for the system. Connecting to the power supply's screw terminal is made easier and more reliable by using spade connectors[2]. NEMA 5-15 plugs are the industry standard for both residential and commercial applications. Ensure that your system performs well in a range of conditions. Choose 18AWG wires to effectively control the required current without overheating and meet power standards. Installing an on/off switch directly on the cable allows users to conveniently adjust power flow without unplugging devices. This not only increases user safety but also conserves electricity. The pigtail end features spade connections for simple connection to the power supply's screw terminals. This streamlines the installation procedure. This solution avoids the need for additional cables or complex arrangements, making it suitable for people with diverse technical skills.

When connecting the power supply to the combined buck converter and solenoid PCB, the system's high current needs were addressed by utilizing a 10AWG extension cable[3] with spade connectors on both ends. Considering various factors such as compatibility with current screw terminals on the power supply and PCB, the ability to handle high currents up to 50 amps, and simplicity of connection and upkeep. The chosen 10AWG cable is specifically designed to carry large currents of up to 50 amps, meeting system requirements and ensuring safe and efficient power transfer. This capacity is critical for system current satisfaction, particularly during full-load operation or peak loads. The spade connector allows for simple, secure connections to the power supply and screw terminals on the PCB, reducing the danger of bad connections or power loss. The connectors' compatibility with existing screw terminals allows for easy integration and lowers the chance of errors during installation and operation.

Consider creating a specialized power cord with a built-in circuit breaker or surge protection for added safety from overcurrent and electrical surges. This solution would enhance the protection of the power supply and connected components against potential electrical hazards.

#### 4.6.4 Interface Validation

<b>Interface Property</b>	<b>Why is this interface this value?</b>	<b>Why do you know that your design details <u>for this block</u> above meet or exceed each property?</b>
---------------------------	--	---

TABLE I:

AC Wall power Power supply Validation

**Otsd\_pwr\_spply\_acpwr : input**

Inominal: 2.25A	This comes from the average amount of current that the power source is thought to draw	Based on the calculation of the Inomial of Solenoid and DC buck converter which is 80% efficiency of the power supply. $P=UI$ , $VAC=120V$
--------------------	--	--

	<p>when it's not under a lot of stress. It is found by dividing the system's total power needed by its operating voltage (120V). This makes sure that the power source is the right size for the system and not too big or too small.</p>	
<p>I<sub>peak</sub>: 3.013A</p>	<p>The peak current is the most current that is drawn when the system is first starting up or when a lot of high-demand parts are turned on at the same time. It gives the power source and wiring a safety margin above the nominal current to make sure they can handle short surges without damage or performance loss.</p>	<p>Based on the calculation of the I<sub>peak</sub> of Solenoid and DC buck converter which is 80% efficiency of the power supply. <math>P=UI</math>, <math>VAC=120V</math></p>
<p>V<sub>nominal</sub>: 120V</p>	<p>These numbers come from standard AC power specs in North America. These are made to safely handle the operating load of home and business electronics.[7]</p>	<p>Since the power supply is made to work with 120V AC, it will work properly with the standard wall outlets in the places where it is meant to be used. Design specifications and compliance with important electrical standards show that these two things can work together directly.</p>

TABLE II:  
User Input for Power Supply Validation

Otsd\_pwr\_spply\_usrin

Timing: Takes <1 second to use	This value is based on how quickly and easily users can connect with the system. When you flip the ON/OFF switch on the power supply, you get immediate feedback that makes the system feel more responsive and easier to manage.	The design includes a high-quality switch that works well mechanically and has been proven to work reliably within this time frame. This switch was chosen because it has a good track record and good feedback from users, so it meets the requirement for a quick answer.
Type: Power ON/OFF	User contact with the power source requires a simple power ON/OFF control. It simplifies system management by letting the user manually control power flow.	Since the power supply's enable or power control circuit is directly wired to the user input, commands are immediately executed.
Usability: 10/10 Usability	Power control user interfaces must be intuitive, accessible, and reliable for all skill levels.	To ensure usability, its placement, size, and feedback

TABLE III:  
Power supply to Solenoids Validation

**Pwr\_spply\_slnd\_dcpwr : output**

Inominal: ON: 750mA per solenoid, ~18A	To ensure the best performance, the nominal current for each solenoid is set to 750mA per the manufacturer's standards. Given that all solenoids may be operating at the same time, the aggregate nominal current is around 18A.	The chosen power supply has a capacity of 30A, which is enough to sustain the total nominal current draw of all 24 solenoids. This has been proven by testing with a load equivalent to 24 solenoids functioning under normal conditions. [5]
---	--	--

<p>I<sub>peak</sub>: ON: 1A per solenoid, ~24A</p>	<p>The peak current compensates for the initial inrush current or startup surge associated with solenoid activation. It is adjusted higher than the nominal value to ensure dependability during these temporary conditions.</p>	<p>The power supply and cabling are designed to handle peak currents of up to 30A, which exceeds the combined peak current requirement of 24 A. This is based on both component ratings and empirical data from peak current testing.[5]</p>
<p>V<sub>max</sub>: 13V</p>	<p>The maximum voltage of 13V offers a buffer over the nominal to account for possible tolerances in the power supply's output, ensuring that minor changes do not significantly impact solenoid performance.</p>	<p>The power supply is controlled to keep the output from exceeding the maximum voltage, even when the input voltage or load changes. [1]</p>
<p>V<sub>min</sub>: 11V</p>	<p>The minimum voltage ensures that, even with permissible decreases in supply voltage, the solenoids receive enough voltage to function successfully without reaching their drop-out or under-performance thresholds.</p>	<p>The power supply's lower voltage limit is set to keep at least 11V under all working settings, as proved in testing with diverse loads and input conditions.[1]</p>
<p>Nominal: 12V</p>	<p>The voltage range lets the power source change while still making sure the solenoids work within their recommended voltage range.</p>	<p>The output of the 12V DC power supply is centered on this nominal voltage and is designed to remain steady even when the load varies. This consistency is achieved via a mixture of regulating circuitry within the power supply, which is validated by continuous operational testing.[1]</p>

TABLE IV:  
Power supply to DC Buck Converter Validation

**Pwr\_sply\_dc\_bck\_cnvtrr\_dcpwr : output**

<p>Inominal: 19mA</p>	<p>The nominal current of 19mA is based on the typical operating current required by the buck converter to efficiently step down the voltage to its output level. The value is decided by the converter's internal circuitry and the load demands of the downstream components it powers. [6]</p>	<p>The power supply is rated for substantially higher output than 19mA, thus it can provide adequate current even when numerous buck converters or other loads are connected. This capacity is a built-in feature of the power supply and has been verified through testing, which involved measuring the current draw under normal working settings. [6]</p>
<p>Ipeak: 100mA</p>	<p>The peak current is the maximum instantaneous current that the buck converter may draw during startup or a sudden increase in load. The value provides a buffer to absorb these transient spikes without jeopardizing the converter's or power supply's integrity.</p>	<p>The design contains cabling and connectors that can handle currents significantly greater than 100mA, providing a buffer to accommodate transient peak demands. These components were chosen based on their ratings and confirmed during stress testing at peak loads.[6]</p>
<p>Vmax: 13V</p>	<p>A maximum voltage restriction of 13V protects the buck converter from overvoltage caused by power supply variations.</p>	<p>The power supply is designed with a voltage regulation system to keep the output from exceeding this maximum limit. The 13V ceiling is part of the design specification and is validated through testing that simulates high input voltage conditions for the power supply.[1]</p>
<p>Vmin: 11V</p>	<p>The minimum voltage of 11V is the point at which the buck converter can still perform properly without significantly reducing functionality. This compensates for the lowest allowable</p>	<p>The power supply's control ensures that the output does not go below 11V even when the input voltage drops or when the load causes undervoltage. This specification has been validated using load regulation experiments.[1]</p>

	voltage loss while maintaining efficient down-conversion.	
Nominal: 12V	Most buck converters require a nominal voltage of 12V to function well, providing for best performance and compatibility with a wide range of electrical components that operate at this voltage.	The power supply's fundamental design focuses on producing a consistent 12V output under normal working conditions. This has been tested and validated by continuously measuring the voltage output while the buck converter is operating at a nominal load.[1]

4.6.5 Verification Plan

1. Verify Wall Power Input to Power Supply

- 1.1 Connect the power supply to the wall outlet.
- 1.2 Connect an AC meter to the input of the power supply.
- 1.3 Verify that the AC meter reads approximately 125V AC power input. This is within the acceptable range for the power supply's operational input voltage, noting that the nominal input voltage is 120V with an acceptable variance.

2. Setup for Solenoid and Buck Converter Testing

- 2.1 Connect the output of the power supply to an Electrical DC load machine, which will simulate the load of both the solenoids and the buck converter.
- 2.2 Ensure that the DC load machine is capable of setting specific load conditions, including maximum power, peak current ( $I_{peak}$ ), and nominal current ( $I_{nominal}$ ).

3. Testing Solenoid Load Conditions

- 3.1 Set the Electrical DC load machine to simulate the solenoid's operational conditions, specifying a maximum power setting of 250W (to ensure the machine's load limit does not interfere with testing).
- 3.2 Set the DC load machine to draw the nominal current for solenoids (750mA per solenoid, 18A for 24 solenoids) and observe the voltage output from the power supply.
- 3.3 Increase the load to simulate the peak current draw (1A per solenoid, 24A for 24 solenoids), and again observe the voltage output.
- 3.4 Record the voltage readings to ensure they fall within the specified range ( $V_{min}$ : 11V,  $V_{nominal}$ : 12V,  $V_{max}$ : 13V).

4. Testing Buck Converter Load Conditions

4.1 Adjust the Electrical DC load machine to reflect the buck converter's operational conditions with specified Inominal and Ipeak values (19mA nominal, 100mA peak).

4.2 Observe and record the voltage output from the power supply to ensure it remains stable and within the required voltage range for the buck converter.

## 5. Analyze

5.1 Compare the recorded voltage outputs under both solenoid and buck converter load conditions to the specified properties (Vmin, Vnominal, Vmax).

5.2 Confirm that under nominal and peak loads, the voltage output remains within the acceptable range, ensuring reliable operation of the solenoids and buck converter.

## 6. Documentation

6.1 Document all settings, measurements, and deviations from intended findings.

6.2 If any measurements fall outside the stated properties, look into possible causes and make changes to the power supply or load parameters before retesting.

### 4.6.6 References and File Links

#### 4.6.6.1. References

[1] MOUO 12V 30A DC Universal Regulated Switching Power Supply 360W  
[www.amazon.com/BMOUO-Universal-Regulated-Switching-Computer/dp/B01EWG6YT8/ref=sr\\_1\\_8?dib=eyJ2IjoiMSJ9.OgJ1EgHrvSiTXppmGrXaIY6eeEf6eOMiS9CrkOy3Xy4Fmv2FT33COjnl nuE8zIXChNB4-Km6pW5ZBXnATiDOtlvxqrRgD4R3I8cjhsx9DY0C lzkg0ZKsvbsmRAxwIUUGk N7eYj9VvrTezrUtv2f6g.jYvG6DjHGd30mlHJib0d37qrrwLWs5yCt38R4eY8IT4&dib\\_tag=se&keyw ords=12v+power+supply&sr=8-8](http://www.amazon.com/BMOUO-Universal-Regulated-Switching-Computer/dp/B01EWG6YT8/ref=sr_1_8?dib=eyJ2IjoiMSJ9.OgJ1EgHrvSiTXppmGrXaIY6eeEf6eOMiS9CrkOy3Xy4Fmv2FT33COjnl nuE8zIXChNB4-Km6pW5ZBXnATiDOtlvxqrRgD4R3I8cjhsx9DY0C lzkg0ZKsvbsmRAxwIUUGk N7eYj9VvrTezrUtv2f6g.jYvG6DjHGd30mlHJib0d37qrrwLWs5yCt38R4eY8IT4&dib_tag=se&keyw ords=12v+power+supply&sr=8-8) . Accessed 25 Jan. 2024.

[2] Spade Wire Connector, (N.d.). Retrieved from  
[https://www.amazon.com/dp/B0BB25LSF2?ref=ppx\\_yo2ov\\_dt\\_b\\_product\\_details&th=1](https://www.amazon.com/dp/B0BB25LSF2?ref=ppx_yo2ov_dt_b_product_details&th=1).  
Accessed 15 Feb. 2024

[3] Terminal Connector Extension Cable 10AWG Battery Adapter Cable.(N.d.). Retrieved from  
[https://www.amazon.com/dp/B07WP5CKM7?psc=1&ref=ppx\\_yo2ov\\_dt\\_b\\_product\\_details](https://www.amazon.com/dp/B07WP5CKM7?psc=1&ref=ppx_yo2ov_dt_b_product_details).  
Accessed 30 Feb. 2024

[4] 6ft 18 Gauge 3 Prong AC Power Cord Cable with on Off Switch, Pigtail (Open End Wiring). (N.d.). Retrieved from  
[https://www.amazon.com/dp/B08BRF11T7?psc=1&ref=ppx\\_yo2ov\\_dt\\_b\\_product\\_details](https://www.amazon.com/dp/B08BRF11T7?psc=1&ref=ppx_yo2ov_dt_b_product_details).  
Accessed 30 Feb. 2024



[5] Industries, A. (n.d.). Large push-pull solenoid. Retrieved from <https://www.adafruit.com/product/413>. Accessed March 8, 2024

[6] TPS54232. (n.d.). Retrieved from <https://www.ti.com/product/TPS54232>. Accessed March 8, 2024

[7] North American Voltage Ranges. (n.d.). Retrieved from <https://quick220.com/pages/north-american-voltage-ranges>. Accessed March 8, 2024

#### 4.6.7. Revision Table

03/08/2024	Kexin Liu: Created the section and filled in the necessary details to match the rest of the project's blocks
------------	--

## 4.7 DC Buck Converter Validation

### 4.7.1 Description

The DC Buck Converter block, which revolves around the TPS54232D microprocessor, is an advanced circuitry specifically engineered to effectively reduce a 12V input to a 5V output. The output is essential for the functioning of four-mallet drivers that, under normal circumstances, consume a total current of approximately 12mA, which may reach its highest point at around 18mA during maximum load.

The primary component of the block's operational framework is the TPS54232D microcontroller, which employs pulse-width modulation (PWM) as a means to regulate the power supplied to the load. To prevent potential fluctuations or noise, a 10uF capacitor is employed to stabilize the input voltage. The initiation of the converter is regulated by an enable pin, which, by use of a particular configuration of resistors R1 and R2, permits the microcontroller to initiate its functioning upon the application of power.

The block operates by implementing a feedback network consisting of resistors R5 and R6, as well as capacitors C3 and C4, and resistor R4. This network collectively contributes to maintaining stability and precision in voltage control. The integration of a soft-start mechanism serves the purpose of regulating the initial surge of power, ensuring the protection of the converter, and extending the lifespan of its components.

In the realm of energy storage and conversion, the inductor L1 and Schottky diode D1 play a crucial role. The inductor facilitates the transmission of energy, while the diode, renowned for its minimal voltage drop and rapid response, reduces power dissipation, hence improving the overall efficiency of the converter. Capacitor C6 functions as the output filter, effectively mitigating voltage fluctuations to produce a pristine and consistent direct current (DC) output.

### 4.7.2 Design

The black box diagram in Figure 1 defines the inputs and outputs of the the DC Buck Converter. Figure 2 shows the design of the TPS54232D chip layout. Figure 3 is the PCB design schematic used of Kicad and Figure 4 shows the integrated PCB design layout. Figure 5. is the TPS54232D Functional Block Diagram and Figure 6 is the photo of the TPS54232D chip.

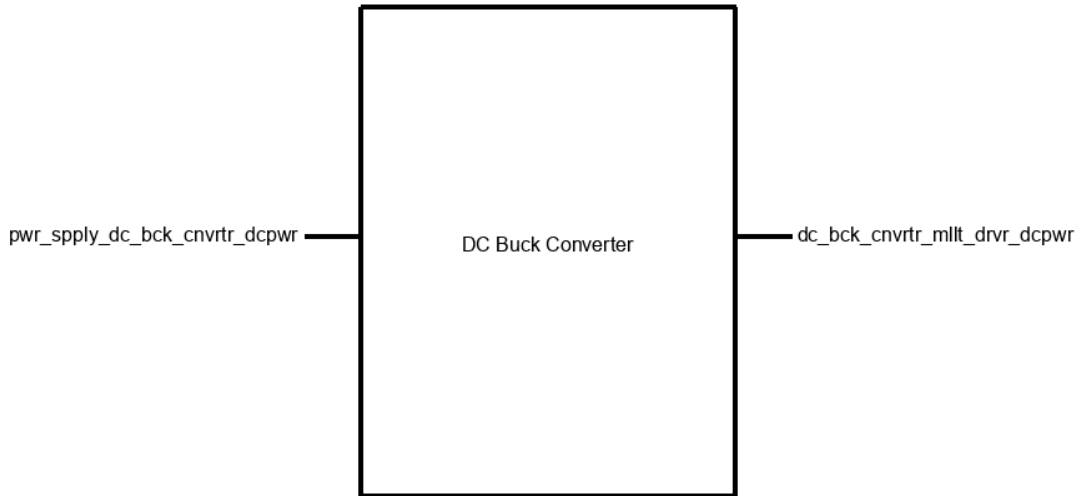


Fig 1. Black Box Diagram

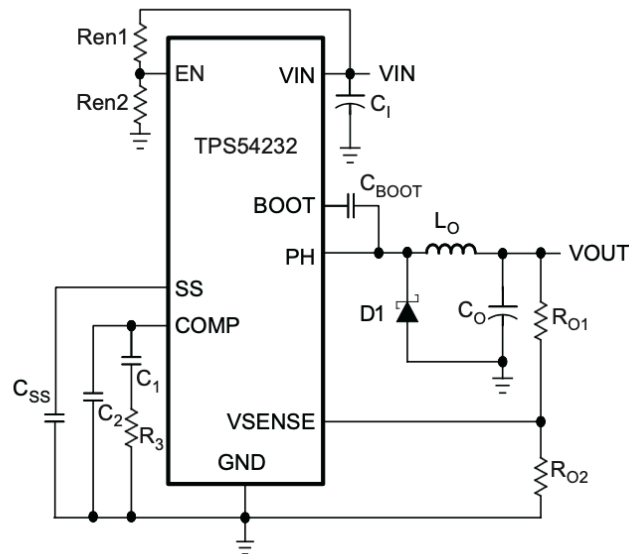


Fig 2. TPS54232D Simplified Schematic

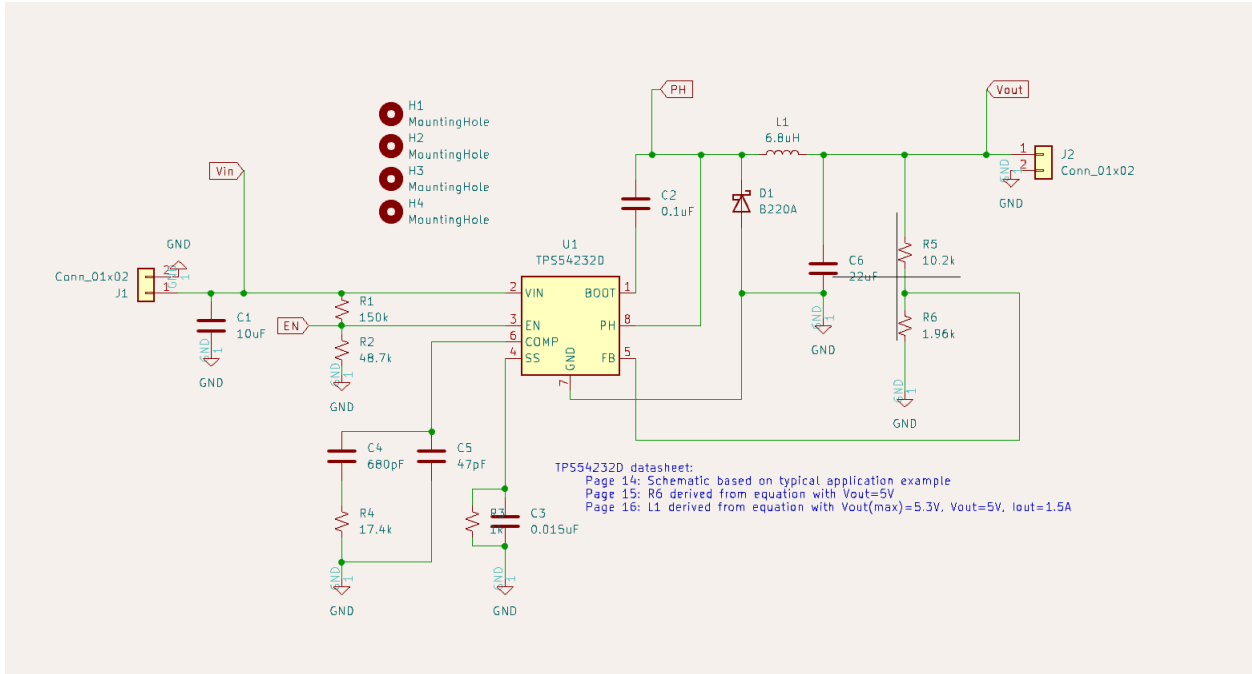


Fig 3. PCB Schematic Design

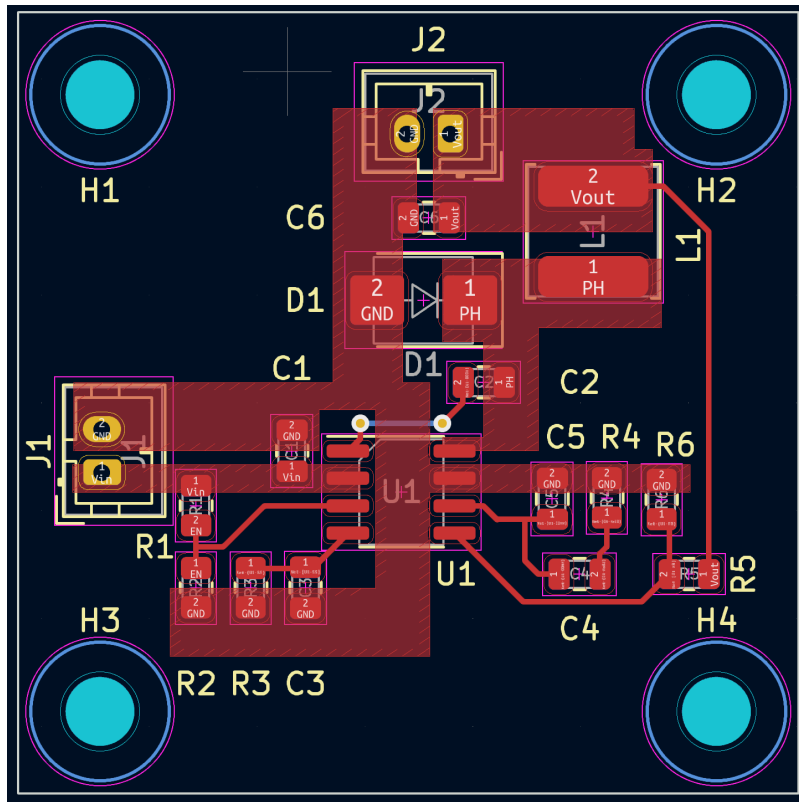


Fig 4. DC Buck Converter PCB Layout

### 4.7.3 General Validation

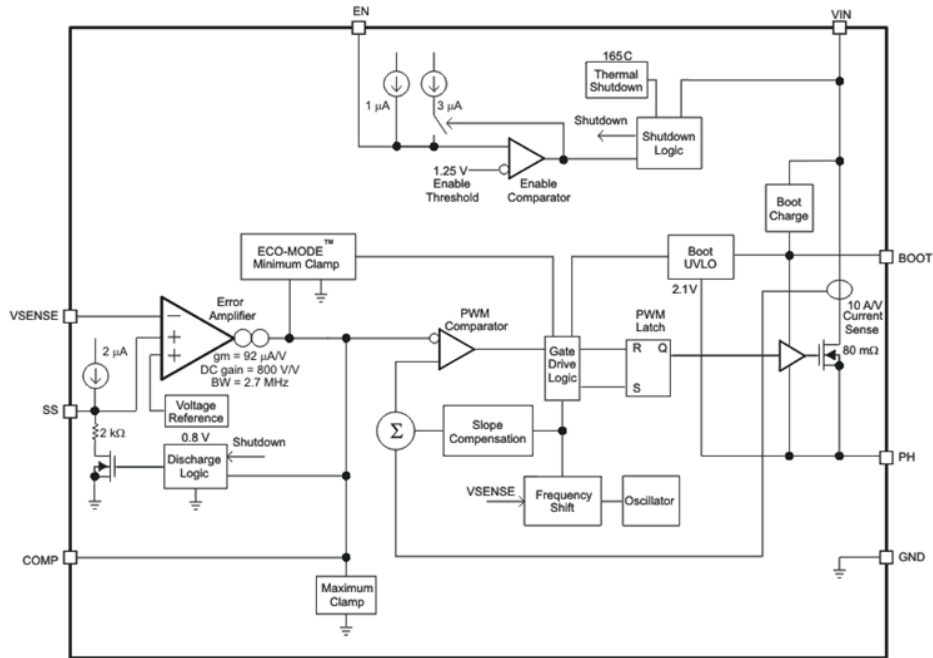


Fig 5. TPS54232D Functional Block Diagram

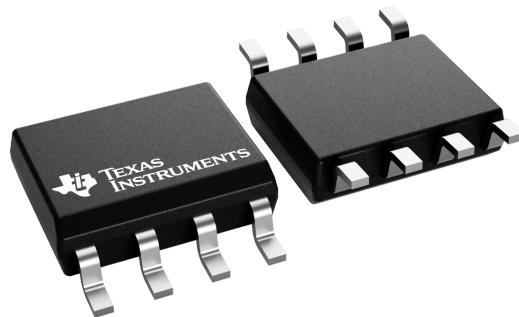


Fig 6. TPS54232D Chip

The DC Buck Converter PCB combines cost-effectiveness, part availability, technical performance, system compatibility, and compliance with partner and system requirements.

The TPS54232 can handle input voltages ranging from 3.5 to 28V and output voltages as low as 0.8V. The TPS54232 is the ideal solution for the DC Buck Converter. High-side MOSFETs can take up to 2A of continuous output power, making them suitable for a variety of applications. Because Eco-Mode™ performs better with light loads, the system is more efficient, with an output voltage as low as 0.8V. The TPS54232 is the ideal solution for the DC Buck Converter. Because Eco-Mode™ works better with light loads, the system is more efficient, and the constant 1-MHz switching frequency allows the PCB to be compact. The device's low shutdown

current also allows portable gadgets to have longer battery life. Overvoltage protection, current limitation, and thermal shutdown are only a few of the many safety features that ensure the power system's reliability. The system is safe and reliable.

The component supply was adequate to ensure that the project's timetable was met and that significant component shortages did not cause delays. In terms of performance, it meets the power supply's input voltage range as well as the mallet driver's nominal and peak current requirements. The buck converter is designed to be easily integrated with other system components and provides a standardized 5V output, simplifying the system's power architecture and allowing the block to be used in a variety of applications.

The converter's tiny form factor was purposely chosen to accommodate the requirements of a dense PCB layout in a restricted space. Additionally, it guarantees that the design's dimensions are met.

PCB layout follows established practices, with jumpers and connectors strategically placed for easy wiring to other components. Trace thickness and component footprints adhere to established standards and are directed by component datasheets to guarantee that the PCB can manage the predicted electrical loads without issue and fit into the specified space.

Alternatives: A linear regulator is another option; its simpler design may be advantageous in applications where the efficiency provided by a buck converter is less important. At the same time, consider the increased power loss and possible thermal management difficulties.

#### 4.7.4 Interface Validation

<b>Interface Property</b>	<b>Why is this interface this value?</b>	<b>Why do you know that your design details <u>for this block</u> above meet or exceed each property?</b>
-------------------------------	--	---

TABLE I:  
Power Supply to DC Buck Converter Validation

**Pwr\_sply\_dc\_bck\_cnvtr\_dcpwr : input**

Inominal: 19mA	The nominal current of 19mA is set to match the expected steady-state consumption of the downstream circuitry powered by the buck converter.	The TPS54232 is capable of delivering a continuous output current of 2A, which is well above the nominal current requirement. This provides confidence that our design can supply the necessary current without strain on the system.[1]
-------------------	--	--

<p>I<sub>peak</sub>: 100mA</p>	<p>The peak current of 100mA accounts for possible transient surges in the system that may occur during power-intensive operations or at startup, ensuring the power supply can handle short bursts of higher current.</p>	<p>With an integrated high-side MOSFET that supports up to 2A, the design ensures that transient currents up to 100mA are well within the TPS54232's capabilities. This has been further corroborated by referencing the datasheet specifications that indicate the buck converter can manage surge currents significantly higher than 100mA.[1]</p>
<p>V<sub>max</sub>: 13V</p>	<p>A maximum voltage restriction of 13V protects the buck converter from overvoltage caused by power supply variations.</p>	<p>The power supply is designed with a voltage regulation system to keep the output from exceeding this maximum limit. The 13V ceiling is part of the design specification and is validated through testing that simulates high input voltage conditions for the power supply.[2]</p>
<p>V<sub>min</sub>: 11V</p>	<p>The minimum voltage of 11V is the point at which the buck converter can still perform properly without significantly reducing functionality. This compensates for the lowest allowable voltage loss while maintaining efficient down-conversion.</p>	<p>The power supply's control ensures that the output does not go below 11V even when the input voltage drops or when the load causes undervoltage. This specification has been validated using load regulation experiments.[2]</p>
<p>Nominal: 12V</p>	<p>Most buck converters require a nominal voltage of 12V to function well, providing for best performance and compatibility with a wide range of electrical components that operate at this voltage.</p>	<p>The power supply's fundamental design focuses on producing a consistent 12V output under normal working conditions. This has been tested and validated by continuously measuring the voltage output while the buck converter is operating at a nominal load.[2]</p>

TABLE II:  
DC Buck Converter to Mallet Driver as DC Power Validation

#### Dc\_bck\_cnvtrr\_mllt\_drvr\_dcpwr : output

Inominal: 2.6mA per driver, ~12mA	Based on the operational current required by each mallet driver.	The TPS54232's current output capacity far exceeds the nominal combined current requirement of the mallet drivers, ensuring ample power delivery.[1]
Ipeak: 4.2mA per driver, ~18mA	This shows how much current each driver can draw at its peak under the toughest conditions.	The TPS54232 supports a continuous output current of up to 2 A, which is much higher than the 18mA combined peak current of the mallet drivers, providing a high safety margin.[1]
Vmax: 5.5V	The highest output voltage that the drivers are safe to manage.	The TPS54232 makes sure that the output never goes beyond 5.5V by having the capacity to give a controlled output and overvoltage protection.[1]
Vmin: 4.5V	The minimal voltage is needed for the drivers to work properly.	The TPS54232's architecture ensures a consistent output even with input fluctuations, keeping the minimum voltage above 4.5V.[1]
Vnomial: 5V	The average voltage required to run the mallet drivers efficiently.	The TPS54232 is designed to give a constant 5V output, which fully matches the nominal voltage requirement of the mallet drivers.[1]

#### 4.7.5 Verification Plan

1. Set up a DC power supply, a DC Buck Converter PCB, and a DC electronic load machine for testing.
2. Set the DC power supply to generate a 12V output, which matches the system's nominal voltage need.
3. Connect the 12V output to the input terminals on the DC Buck Converter PCB.
4. Connect the output of the Buck Converter PCB to the DC electronic load machine.
5. Configure the electronic load to draw a nominal 18mA current, which is the operating current for the mallet drivers.
6. Activate the system and monitor the output voltage of the electronic load. Check that the voltage stabilizes between 4.5V and 5.5V.
7. Increase the load to achieve a peak current of 100mA, representing increased demand from the mallet drivers.

8. Ensure that the voltage remains between 4.5V and 5.5V during this high load condition.
9. Record all voltage values at nominal and peak currents.
10. Repeat the test to ensure consistent results over numerous trials.

#### 4.7.6 References and File Links

##### 4.7.6.1. References

[1] TPS54232. (n.d.). Retrieved from <https://www.ti.com/product/TPS54232>. Accessed 25 Jan. 2024.

[2] MOUO 12V 30A DC Universal Regulated Switching Power Supply 360W  
[www.amazon.com/BMOUO-Universal-Regulated-Switching-Computer/dp/B01EWG6YT8/ref=sr\\_1\\_8?dib=eyJ2IjoiaMSJ9.OgJ1EgHrvSiTXppmGrXaIY6eeEf6eOMiS9CrkOy3Xy4Fmv2FT33COjnl nuE8zIXChNB4-Km6pW5ZBXnATiDOtlvxgrRgD4R3l8cjhxsx9DY0C\\_lzkg0ZKsvbsmRAxwIUUGk N7eYj9VvrTezrUtv2f6g.jYvG6DjHGd30mIHJib0d37qrrwLWs5yCt38R4eY8IT4&dib\\_tag=se&keywords=12v+power+supply&sr=8-8](https://www.amazon.com/BMOUO-Universal-Regulated-Switching-Computer/dp/B01EWG6YT8/ref=sr_1_8?dib=eyJ2IjoiaMSJ9.OgJ1EgHrvSiTXppmGrXaIY6eeEf6eOMiS9CrkOy3Xy4Fmv2FT33COjnl nuE8zIXChNB4-Km6pW5ZBXnATiDOtlvxgrRgD4R3l8cjhxsx9DY0C_lzkg0ZKsvbsmRAxwIUUGk N7eYj9VvrTezrUtv2f6g.jYvG6DjHGd30mIHJib0d37qrrwLWs5yCt38R4eY8IT4&dib_tag=se&keywords=12v+power+supply&sr=8-8) . Accessed 25 Jan. 2024.

##### 4.7.7 Revision Table

03/08/2024	Kexin Liu: Created the section and filled in the necessary details to match the rest of the project's blocks
------------	--

#### 4.8. Distance Sensors Block Validation

##### 4.8.1. Description

The SHARP distance sensor enhances the performance by using IR reflection to detect audience proximity within 20cm to 150cm. It dynamically adjusts the drumming speed based on the spectators' proximity, creating an interactive experience.



## 4.8.2. Design

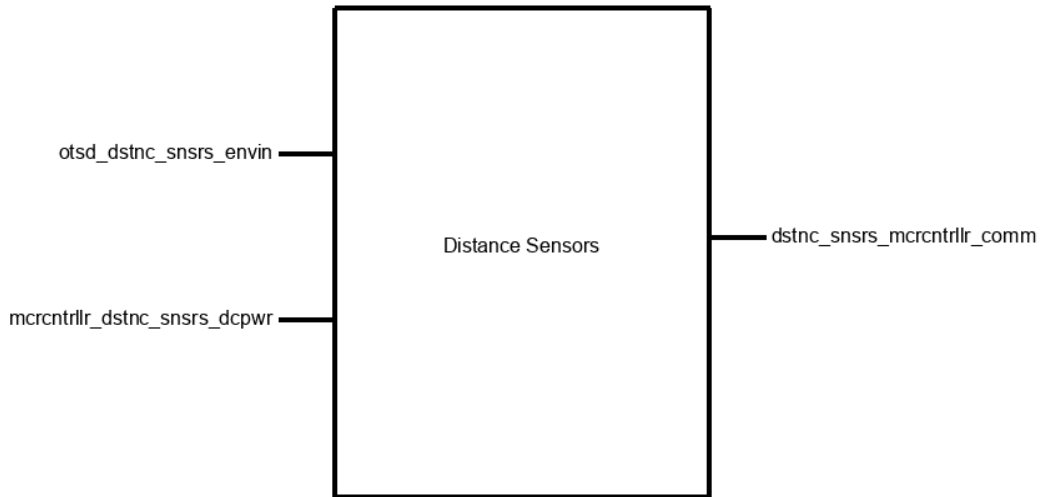


Fig 1. Black Box Diagram

## 4.8.3. General Validation

Note: The Distance Sensors block is new. General Validation will be updated soon.

An internal block diagram image is shown below in Figure 2. This shows that the block is student designed, meaning that it was constructed or “built” in some way.

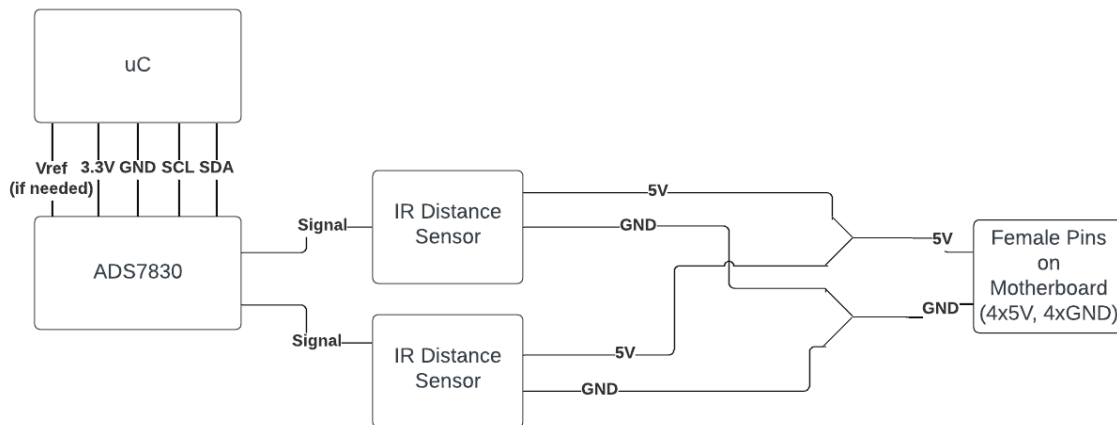


Fig 2. Distance Sensor Internal Block Diagram

## 4.8.4. Interface Validation

Note: The Distance Sensors block is new. Interface Validation Table will be updated soon.

**Interface Property****Why is this interface this value?****Why do you know that your design details for this block above meet or exceed each property?****otsd\_dstnc\_snsrs\_envin : Input**

Other: Distance Range: 20cm to 150cm		
Temperature (Absolute): Temperatures from -10°C to +60°C		

**mrcntrlr\_dstnc\_snsrs\_dcpwr : Input**

Inominal: 33mA		
Ipeak: 50mA		
Vmax: 7V DC		
Vmin: 4.5V DC		
Vnominal: 5V DC		

**dstnc\_snsrs\_mrcntrlr\_comm : Output**

Datarate: At 20cm, 3V; At 150cm, 0.4V		
Vmax: 3V		
Vmin: 0.4V		
Vnominal: between 0.4 V and 3.0 V		

**4.8.5. Verification Plan**

Note: The Distance Sensors block is new. Verification Plan will be updated soon.

**4.8.6. References and File Links**

Note: The Distance Sensors block is new. References and File Links will be updated soon.

#### 4.8.7. Revision Table

4/26/2024	Nicholas Kim: Created section
-----------	-------------------------------

## 5. System Verification Evidence

### 5.1. Universal Constraints

#### 5.1.1. The system may not include a breadboard

All circuits in the system are either on bought PCBs, such as the microcontroller, or on PCBs designed by us. Images of the PCBs used in the system are shown below. The small breakout PCB in the top left is the ADC, and in the center of the image is the large student designed PCB and the microcontroller.

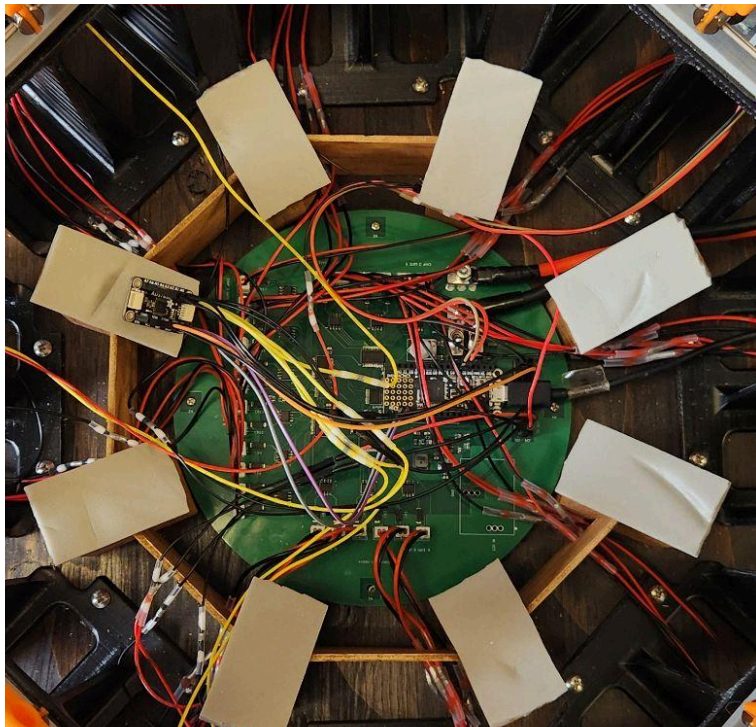


Fig 1. System PCBs

#### 5.1.2. The final system must contain a student designed PCB.

The mallet driver blocks and buck converter will be contained on one large PCB that sits under the drum in the enclosure. An image of the student designed PCB is below. The microcontroller is plugged into it via stacking headers.

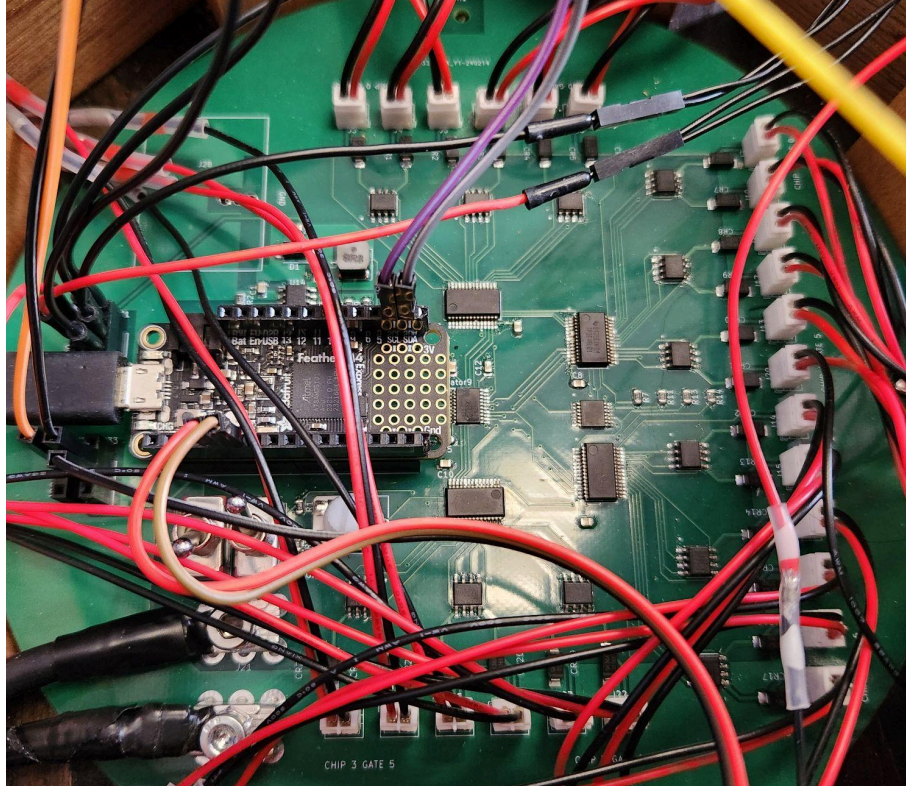


Fig 2. Student Designed PCB

### 5.1.3. All connections to PCBs must use connectors.

The main PCB uses JST connectors to connect to all the 24 solenoids. The power supply connects to the PCB via screw terminals. The microcontroller connects to the PCB via stacking header connections. The ADC is connected to the PCB and microcontroller via female header pins and standard jumper cables. These connectors can be seen in Figure 2 above.



5.1.4. All power supplies in the system must be at least 65% efficient.

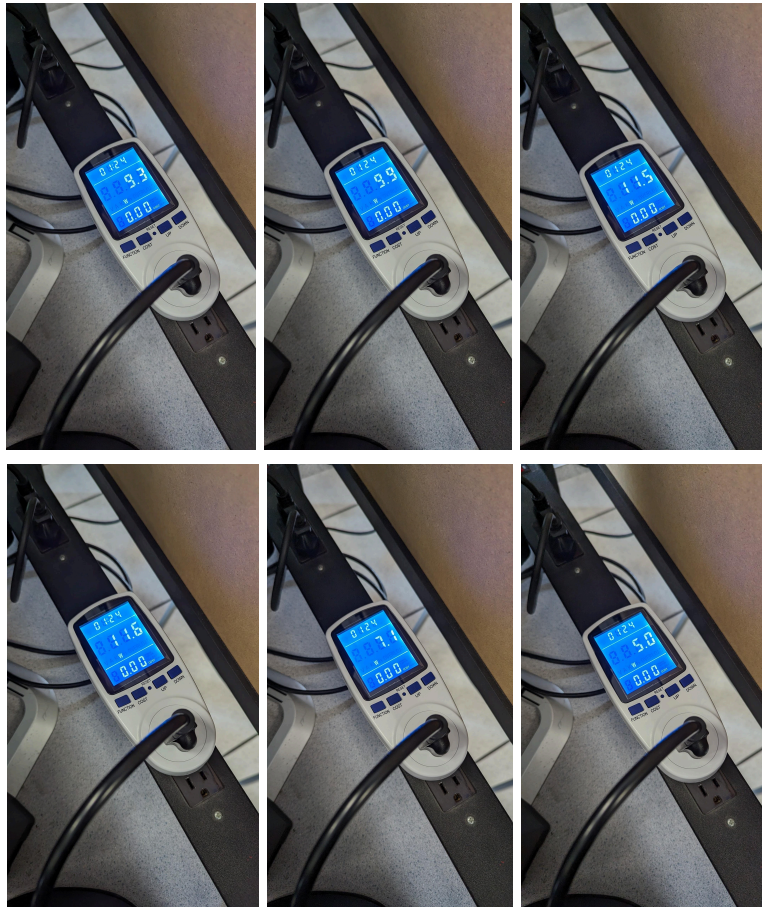


Fig 3. Sample images of a watt meter at the power supply input during autonomous mode. The average power consumption in autonomous mode is 9.07W.



Fig 4. Power supply DC voltage in autonomous mode

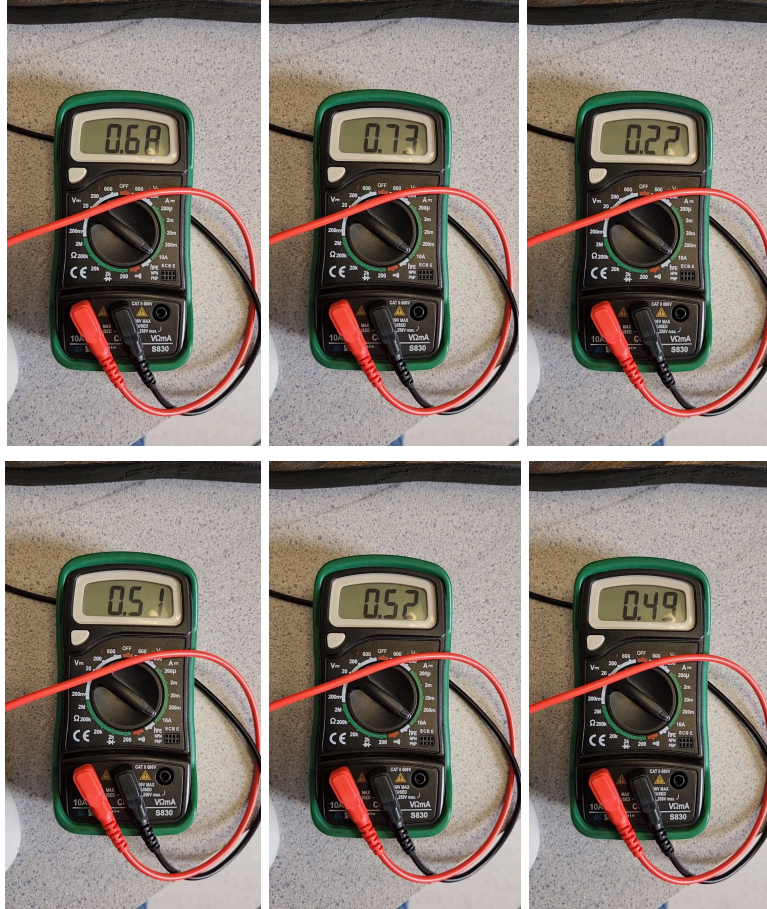


Fig 5. Sample images of an ammeter at the power supply output during autonomous mode. The average current output in autonomous mode is 0.525A.

Using the measured voltage and current shown in Figures 4 and 5, the average power usage in autonomous mode is 6.34W. The efficiency of the power supply is  $6.34W / 9.07W = 69.9\%$ .

5.1.5. The system may be no more than 50% built from purchased 'modules.'

The two code blocks, mallet driver block, distance sensor block, and buck converter block are all student designed. These are shown in sections 4.2, 4.3, 4.4, 4.7, and 4.8. Please see those sections for visual evidence. This makes up more than half of the eight total blocks in the system.

## 5.2 Requirements

### 5.2.1. Autonomous Performance

**5.2.1.1. Project Partner Requirement:** The system should be able to perform autonomously without real-time MIDI input.

**5.2.1.2. Engineering Requirement:** The system will have an autonomous performance mode in which 9 out of 10 audience members will agree that "the system plays interesting musical ideas".

**5.2.1.3. Testing Method:** Demonstration

**5.2.1.4. Verification Process:**

1. Autonomous mode will be given parameters values for energy level, sequence length, and available note values.
2. Autonomous mode will be initiated by pressing a button or by MIDI input.
3. A new MIDI sequence will be generated by code on the microcontroller based on parameters values specified in step 1.
4. The sequence will be performed by the mallet arms.
5. Ten audience members will be given a poll which indicates whether they agree with the statement "the system plays interesting musical ideas".

**Pass Condition:** 9 out of the 10 audience members agree with the statement in step 5 of the testing process.

**5.2.1.5. Testing Evidence:**



**OctoDrum Survey Signatures**  
ECE 44x Group 33

By signing this document, you are agreeing/disagreeing with the following statements about the OctoDrum:

1. In autonomous performance mode, the system plays interesting musical ideas.
2. The system's autonomous performance is responsive to human interaction.

Audience Member Name	I agree with statement 1	I agree with statement 2	Audience Member Signature
Johannes Heurman	✓	✓	Johannes Heurman
Lucas Edwards	✓	✓	Lucas Edwards
Griffin LeBlanc	✓	✓	Griffin LeBlanc
Genelle	✓	✓	Genelle
Nathan Fy	✓	✓	Nathan Fy
BEN REINBERG	✓	✓	Ben Reinberg
JASTIN GILSON	✓	✓	Jastin Gilson
LAWRENCE HEIGES	✓	✓	Lawrence Heiges
Cherminé Chong	✓	✓	Cherminé Chong
Zander Pong	✓	✓	Zander Pong
Lily Wang	✓	✓	Lily Wang
Linnea Lochner	✓	✓	Linnea Lochner
Ylva Malk	✓	✓	Ylva Malk
Kristin Rorrer	✓	✓	Kristin Rorrer
Samatha Ramirez	✓	✓	Samatha Ramirez

Fig 1. OctoDrum Survey Signatures

Figure 1 shows 15 audience member responses agreeing with the statement “In autonomous performance mode, the system plays interesting musical ideas.” This verifies the pass condition for the Autonomous Performance requirement.

5.2.2. Dynamic Range

**5.2.2.1. Project Partner Requirement:** The system should be capable of different dynamics.

**5.2.2.2. Engineering Requirement:** The system will vary the strike speed of the actuator to have at least 3 different strike speeds.

**5.2.2.3. Testing Method:** Test

**5.2.2.4. Verification Process:**

1. Set up one sample actuating device
2. Set up slow-motion video recorder
3. Perform 3 independent actuator strikes in 3 different dynamic levels using dynamic control on the USB MIDI controlling device.
4. Playback slow motion video
  - a) Align all three videos such that actuation begins on same frame
5. Confirm that end of actuation (mallet contacting drum) occurs at different times

**Pass Condition:** End of actuation (mallet contacting drum) occurs at different times

**5.2.2.5. Testing Evidence:**

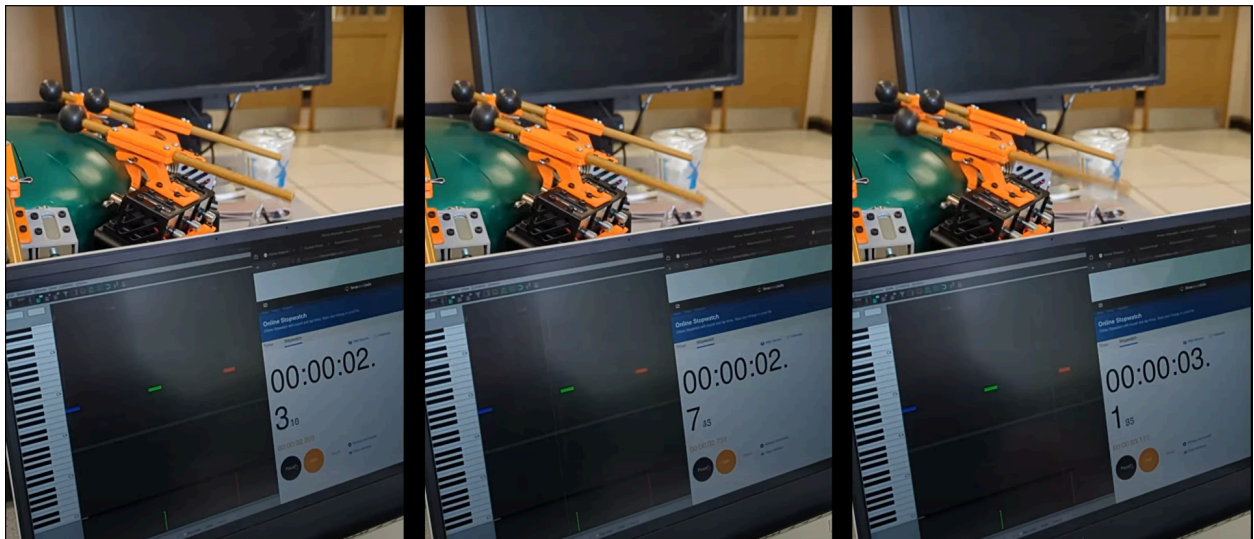


Fig 2. Screenshot from [OctoDrum System Verification: Dynamic Range](#)

The video in File Links section 5.3.2 [1] shows three side-by-side slow motion captures of the same mallet arm strike under 3 different dynamics using MIDI control. The side-by-side videos are aligned such that the actuation begins at roughly the same frame. It can be observed that each arm strikes the drum at a different time: 5 frames for high velocity, 7 for medium velocity, and 10 for low velocity. Thus, the strike velocity is different for each dynamic strike. This verifies the pass condition for the Dynamic Range requirement.

### 5.2.3. Human Interactivity

**5.2.3.1. Project Partner Requirement:** The system's autonomous performance should be responsive to human interaction.

**5.2.3.2. Engineering Requirement:** 9 out of 10 audience members will agree that the system's autonomous performance is responsive to human interaction.

**5.2.3.3. Testing Method:** Demonstration

**5.2.3.4. Verification Process:**

1. Run system in autonomous performance mode
2. Demonstrate human interaction
3. Request feedback from survey takers

**Pass Condition:** At least 9 out of 10 agree that the system's autonomous performance is responsive to human interaction.

**5.2.3.5. Testing Evidence:**

**OctoDrum Survey Signatures**  
ECE 44x Group 33

By signing this document, you are agreeing/disagreeing with the following statements about the OctoDrum:

1. In autonomous performance mode, the system plays interesting musical ideas.
2. The system's autonomous performance is responsive to human interaction.

Audience Member Name	I agree with statement 1	I agree with statement 2	Audience Member Signature
Johannes Heurman	✓	✓	<i>Johannes Heurman</i>
Lucas Edwards	✓	✓	<i>Lucas Edwards</i>
Griffin LeBlanc	✓	✓	<i>Griffin LeBlanc</i>
Genelle	✓	✓	<i>Genelle</i>
Nathan Fy	✓	✓	<i>Nathan Fy</i>
BEN RENEWERRY	✓	✓	<i>Ben Renewerry</i>
JUSTIN GILSON	✓	✓	<i>Justin Gilson</i>
LAWRENCE HEIGES	✓	✓	<i>Lawrence Heiges</i>
Cherine Chong	✓	✓	<i>Cherine Chong</i>
Zander Pong	✓	✓	<i>Zander Pong</i>
Lily Wang	✓	✓	<i>Lily Wang</i>
Linnea Lochner	✓	✓	<i>Linnea Lochner</i>
Yves Malk	✓	✓	<i>Yves Malk</i>
Kristin Rorrer	✓	✓	<i>Kristin Rorrer</i>
Samatha Ramirez	✓	✓	<i>Samatha Ramirez</i>

Fig 3. OctoDrum Survey Signatures

Figure 3 shows 15 audience member responses agreeing with the statement “The system’s autonomous performance is responsive to human interaction.” This verifies the pass condition for the Human Interactivity requirement.

5.2.4. Latency

**5.2.4.1. Project Partner Requirement:** The system should be playable by MIDI input in near real-time.

**5.2.4.2. Engineering Requirement:** The system will perform an actuator strike on the steel tongues within 50ms of the live MIDI input.

**5.2.4.3. Testing Method:** Analysis

**5.2.4.4. Verification Process:**

1. Set up sample actuator arm with MIDI software control
2. Set up slow motion camera with stopwatch in frame
3. Take slow motion video of actuation
4. Observe time delay between MIDI software note click and drum strike

**Pass Condition:** Drum strike is within 50ms of the MIDI software note click

**5.2.4.5. Testing Evidence:**

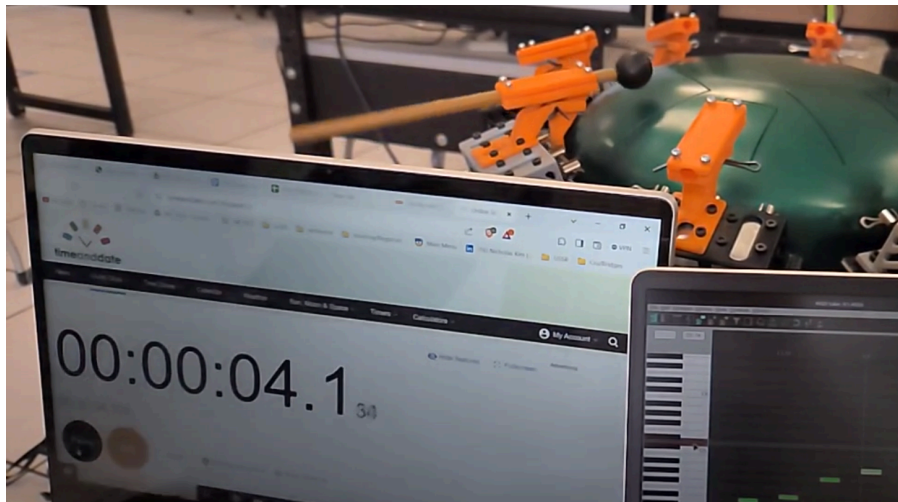


Fig 4. Screenshot from [OctoDrum System Verification: Latency](#)

The video in File Links section 5.3.2 [2] shows a slow motion video of a mallet arm actuation with a stopwatch. The MIDI software is shown to determine the time difference from clicking the note in software to the drum strike. The time difference is less than 50ms. This verifies the pass condition for the Latency requirement.

**5.2.5. MIDI Input**

**5.2.5.1. Project Partner Requirement:** The system should perform using USB MIDI input.

**5.2.5.2. Engineering Requirement:** The system will actuate each of the 8 mallet arms independently using USB MIDI data input.

**5.2.5.3. Testing Method:** Demonstration

#### 5.2.5.4. Verification Process:

1. Plug in Computer with Digital Audio Workstation (DAW) software into the microcontroller.
2. Observe 8 different notes are represented in a MIDI sequence in a DAW on an external computer.
3. Play the MIDI sequence specified in step 2.
4. Observe that each of the 8 mallet arms are being actuated independently, each mallet arm corresponding to one note in the DAW

**Pass Condition:** Observe each mallet is being actuated by a different MIDI note.

#### 5.2.5.5. Testing Evidence:



Fig 5. Screenshot from [ECE 44x - Team 33 - MIDI Input Requirement](#)

The video in File Links section 5.3.2 [3] shows that there is a MIDI sequence programmed which includes 8 different notes. As the MIDI notes are played in the program, the corresponding mallet arm is being actuated for a total of all 8 mallet arms. This verifies the pass condition for the MIDI Input requirement.

#### 5.2.6. Power Supply

**5.2.6.1. Project Partner Requirement:** The system's actuators should be fully-powered by one US electrical outlet.

**5.2.6.2. Engineering Requirement:** The system will draw power from one US electrical outlet and at most one USB input to the microcontroller.



### 5.2.6.3. Testing Method: Inspection

### 5.2.6.4. Verification Process:

1. Observe that the USB cable from a computer (which is on) is connected to the Adafruit Feather M4 Express. Confirm that the microcontroller is on via the on-board LEDs.
2. Observe that the power supply 120VAC input is connected to a US outlet and that 12VDC output from the power supply is connected to the PCB.
3. Turn power supply on.
4. Observe that the system functions.

**Pass Condition:** Observe that the only possible power sources connected to the system is one US electrical outlet and one micro USB cable.

### 5.2.6.5. Testing Evidence:

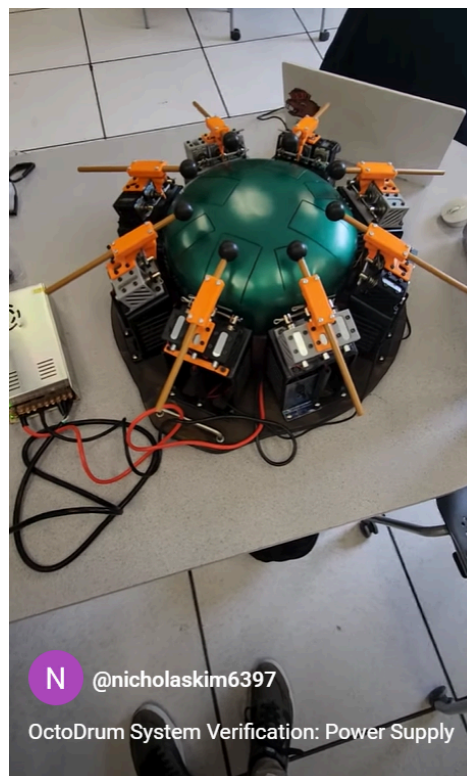


Fig 6. Screenshot from [OctoDrum System Verification: Power Supply](#)

The video in File Links section 5.3.2 [4] shows that the system is functioning and the only possible sources of power are the micro USB cable connected to the microcontroller and the Power supply cable connected to the Power Supply. This verifies the pass condition for the Power Supply requirement.

## 5.2.7. Tempo

**5.2.7.1. Project Partner Requirement:** Each mallet arm should play at a high beats-per-minute.

**5.2.7.2. Engineering Requirement:** The system will actuate each independent mallet arm at a rate of at least 8 hits per second.

**5.2.7.3 Testing Method:** Test

**5.2.7.4. Verification Process:**

1. Prepare MIDI software test for high-frequency actuation for all arms
2. Set up running stopwatch
3. Record video of mallet arm and stopwatch while executing high-frequency actuation test
4. Analyze video to determine number of strikes within a 1 second period.

**Pass Condition:** Each mallet arm performs at least 8 strikes within a 1 second period.

**5.2.7.5. Testing Evidence:**

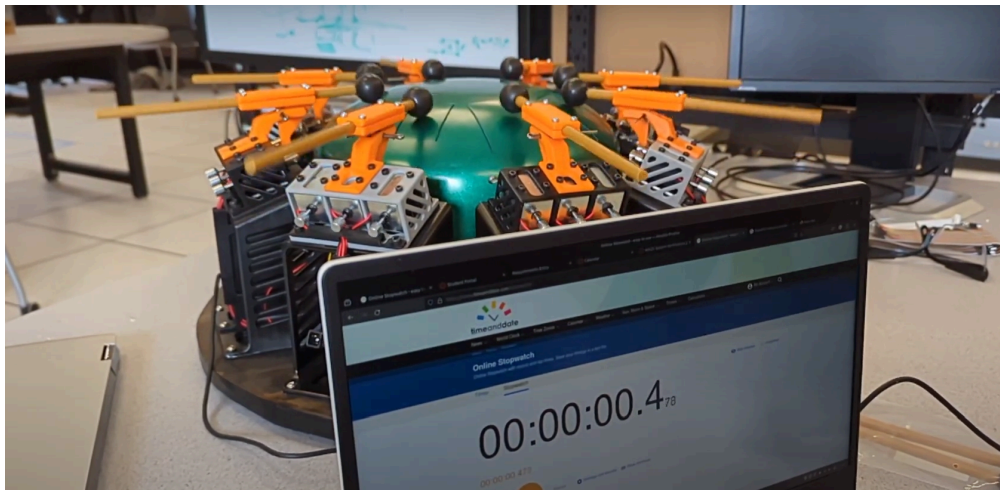


Fig 7. Screenshot from [ECE 44x - Team 33 - Tempo Requirement](#)

The video linked in File Links section 5.3.2 [5] shows that the system is actuating all solenoids (each independent mallet arm) at 130 beats per minute and is playing 4 notes per beat (sixteenth notes). Thus, there are 520 notes per minute or  $520/60 = 8.66$  notes/actuations per second. This is supported by counting the number of strokes within 1 second as shown on the stopwatch. All mallet arms are being actuated at this rate. This verifies the pass condition for the Tempo requirement.



## 5.2.8. Thermal Management

**5.2.8.1. Project Partner Requirement:** The system should not break during operation due to overheating.

**5.2.8.2. Engineering Requirement:** The system will withstand repeated actuation of all solenoids at 8Hz (or highest achieved frequency) for at least 1 minute.

**5.2.8.3. Testing Method:** Test

**5.2.8.4. Verification Process:**

1. Prepare MIDI software test for high-frequency actuation
2. Set up running stopwatch beside system
3. Record video of mallet arm and stopwatch while executing high-frequency actuation test for 1 minute
4. Analyze video to determine average strike frequency within 1 minute.
5. Confirm average strike frequency of at least 8Hz (or highest achieved frequency)

**Pass Condition:** All system actuators function at 8Hz (or highest achieved frequency) for a 1 minute duration.

**5.2.8.5. Testing Evidence:**

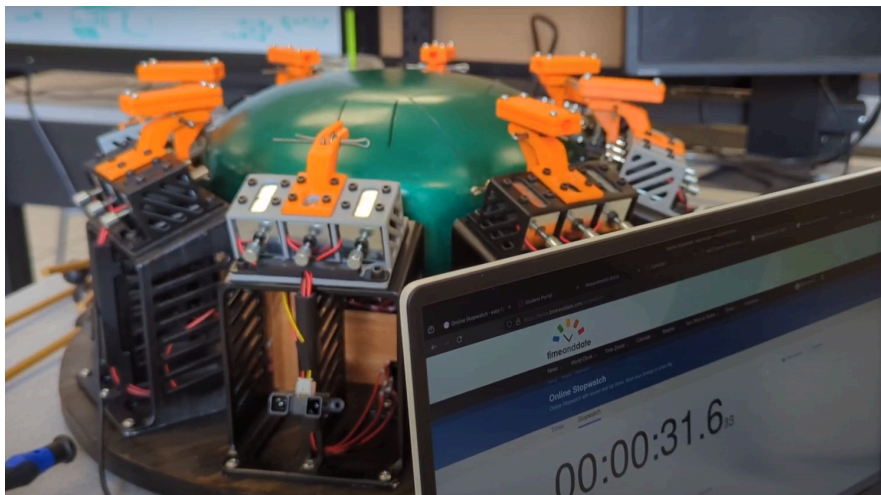


Fig 8. Screenshot from [ECE 44x - Team 33 - Thermal Management Requirement](#)

The video linked in File Links section 5.3.2 [6] shows that the system is actuating all solenoids at 130 beats per minute and is playing 4 notes per beat (sixteenth notes). Thus, there are 520 notes per minute or  $520/60 = 8.66$  notes or actuations per second. This is supported by counting the number of strokes within 1 second as shown on the stopwatch. The stopwatch in the video shows that this occurs for a duration of more than 1 minute, which verifies the Thermal management requirement pass condition.

## 5.3 References and File Links

### 5.3.1. References

[1] Industries, A. (no date) *Adafruit Feather M4 Express - Featuring ATSAMD51*, *adafruit industries blog RSS*. Available at: <https://www.adafruit.com/product/3857> (Accessed: 3 December 2023)

### 5.3.2. File Links

- [1] Dynamic Range Requirement: The system will vary the strike speed of the actuator to have at least 3 different strike speeds. <https://youtu.be/oNPGQo1a8Wo>
- [2] Latency Requirement: The system will perform an actuator strike on the steel tongues within 50ms of the live MIDI input. <https://youtu.be/n9jC9OdPiR8>
- [3] *MIDI Input Requirement*: each mallet arm actuates using live MIDI input. <https://youtube.com/shorts/yfjY1ftXZaQ>
- [4] *Power Supply Requirement*: the system is powered by one US electrical outlet and one USB while operating. <https://youtube.com/shorts/WLPE29YcqY8>
- [5] *Tempo Requirement*: each arm actuates faster than 8Hz; 16th notes at 130 beats per minute is equal to 520 beats per minute; 520 beats per minute is equal to 8.67 Hz. <https://youtu.be/Gk1IBZeggqU>
- [6] *Thermal Requirement*: the system is able to operate at above 8Hz for over 1 minute without heat failure. <https://youtu.be/EOaNxTHniBY>

## 5.4 Revision Table

12/1/2024	Liam and Nick: copied over requirement details from project portal
3/13/2024	Liam: updated requirements from student portal
3/14/2024	Liam, Nick, Liukee: filmed videos [1-3] as shown in the File Links section and wrote testing evidence for Tempo, Thermal Management, Power Supply, and MIDI Input.
05/04/2024	Liam and Nick: updated testing evidence and file links to reflect current system, changed the power supply testing process
05/05/2025	Liam/Nick/Liukee: updated power supply constraint evidence, added images to support evidence in standalone document

## 6. Project Closing

### 6.1. Future Recommendations

#### 6.1.1. Technical Recommendations

##### PCB Revision

The current version of the mallet driver uses the TPIC46L01 pre-FET driver. Unfortunately, the TPIC46L01 is approaching End-of-Life and will not be available for small-scale orders. There are no other single components that provide the same functionality as the TPIC46L01. An alternative mallet driver system will need to be designed using different components. The simplest design is to implement a series of shift registers to achieve 24 parallel outputs, each driving a respective gate driver module which drives the CSD88539ND MOSFETs.

##### Modular System

The current OctoDrum system is dependent on one model of a steel tongue drum. However, the mechanical actuator design, mallet driver PCB, and operating principles can be modified and expanded upon to create a system that can play various percussion instruments such as drums, cymbals, marimbas, xylophones, etc. This would require modifications to the mallet arm stand design, and would require redesigning the PCB to be split into multiple smaller PCBs that make more sense for the modular system. This also creates intense cable management concerns, so wireless communication between the microcontroller and mallet arms may be an area of further exploration.

##### More Robust User Interface

The user interface we implemented consisted of simple switches to switch between the MIDI, autonomous, and sensor-only modes. Incorporating an OLED screen with a simple 5-way switch may be very convenient for modifying parameters and accommodating a more module design per the previous recommendation. This user interface would need to be mounted securely and be attached close to the microcontroller for easy cable management.

##### More Frequent Code Backups

The code was managed mostly locally, with a GitHub repository created for pushing significant updates. Code was pushed to this repository only about 4 times over the course of the entire project. This turned out fine, but making more frequent pushes to GitHub would be a good idea in case a local machine crashes, files get corrupted, etc.

#### 6.1.2. Global Impact Recommendations

##### Complementary educational materials

One aspect of this project that we did not capitalize on was its educational value. The project has huge potential to be used as a method to teach kids about engineering and music concepts. For example, the solenoids could be used as an example to teach about electric and magnetic fields, the TPIC ICs and distance sensors could be used to teach about SPI versus I2C interfaces, and more. Our project is great for this because it is very interactive, encouraging

audience members to play and have fun while learning. So, it may be useful to create some educational materials which can both serve as aid for presenting the system and as project documentation. This could be a slide deck, pamphlets, or some other physical/digital medium.

#### More Sustainably Sourced Components

One aspect of engineering that the team could have focused on more is sustainability, especially with sourcing components. Many of our components, apart from the PCB components, came from Adafruit or Amazon. There was no way to verify the parts' sustainability or manufacturing processes, so one recommendation would be to spend more time making sure that components are manufactured in a place with good working conditions, and materials that are sustainably sourced, i.e. low carbon emissions. Also, 3D-printed components could be printed out of recycled filament. This would ensure that the project follows the DEI statement better and is more ethical in general.

#### 6.1.3. Teamwork Recommendations

##### Allocation of assignment responsibilities

One recommendation is to ensure that, early on, each individual knows what their roles/tasks are for assignments, week-to-week tasks, portions of the project etc. This would make it easier for each team member to be productive with their time. Coordination between team members would be more streamlined and upfront.

##### Team bonding

Building strong connections with your team members helps team morale and trust. Meeting regularly as a team outside of official team capstone meetings can help to make everyone feel more comfortable around each other to avoid bitter conflicts and to help motivate each individual team member to do their part.

##### Timely communication

It's important for each member to find their own voice and provide clear feedback. Timely communication allows everyone to stay informed and up-to-date with the latest developments in a constantly changing environment.

##### Start date for assignments

We as humans (and engineering students) are very deadline driven, so often our team would start assignments very close to the deadline. Having a start date for assignments, especially those that include the project document, would inspire the team to work on and finish assignments earlier so that stress levels are decreased. This would also help the team manage our time better, as we would be more aware of upcoming deadlines and assignments.



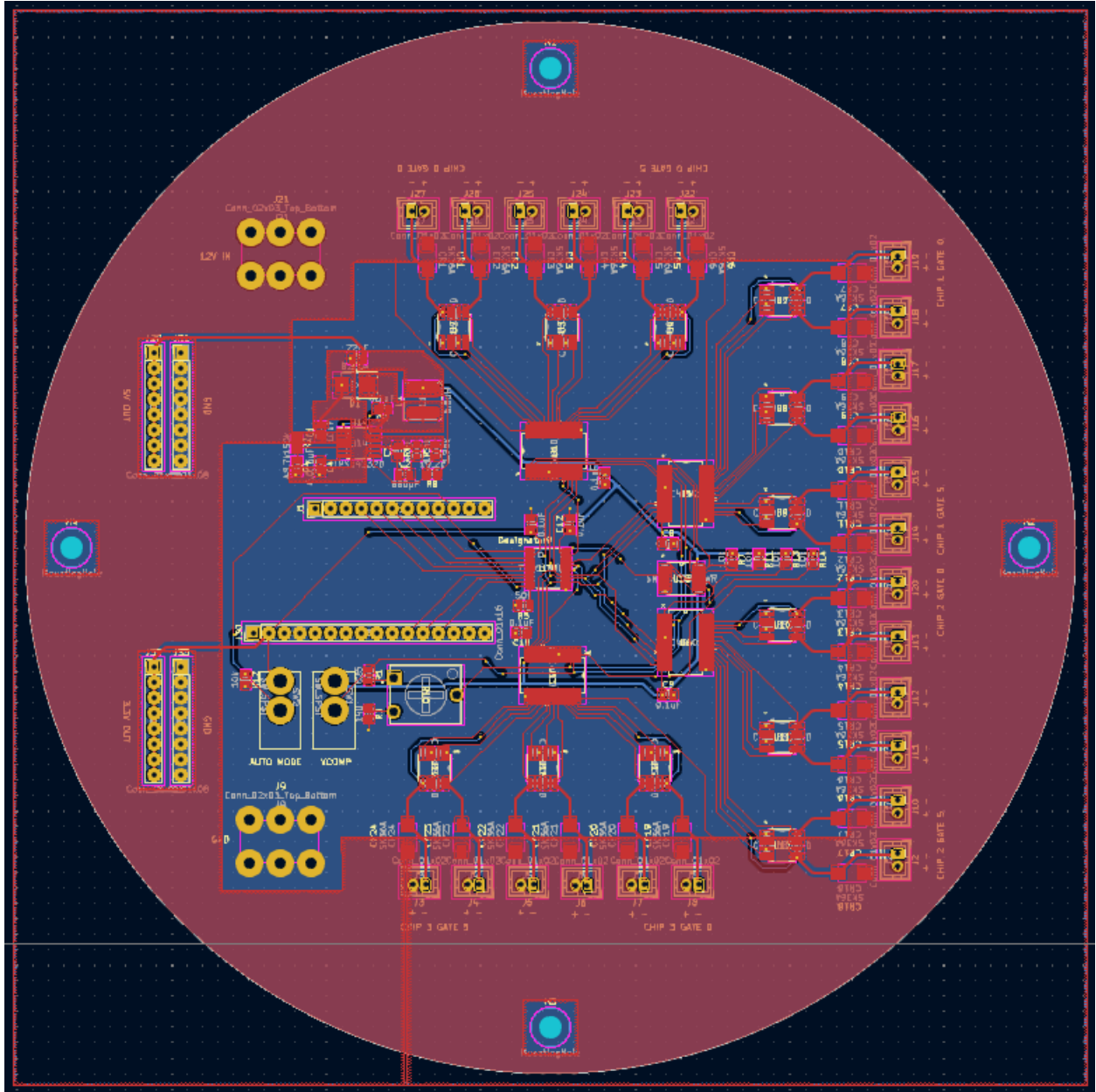
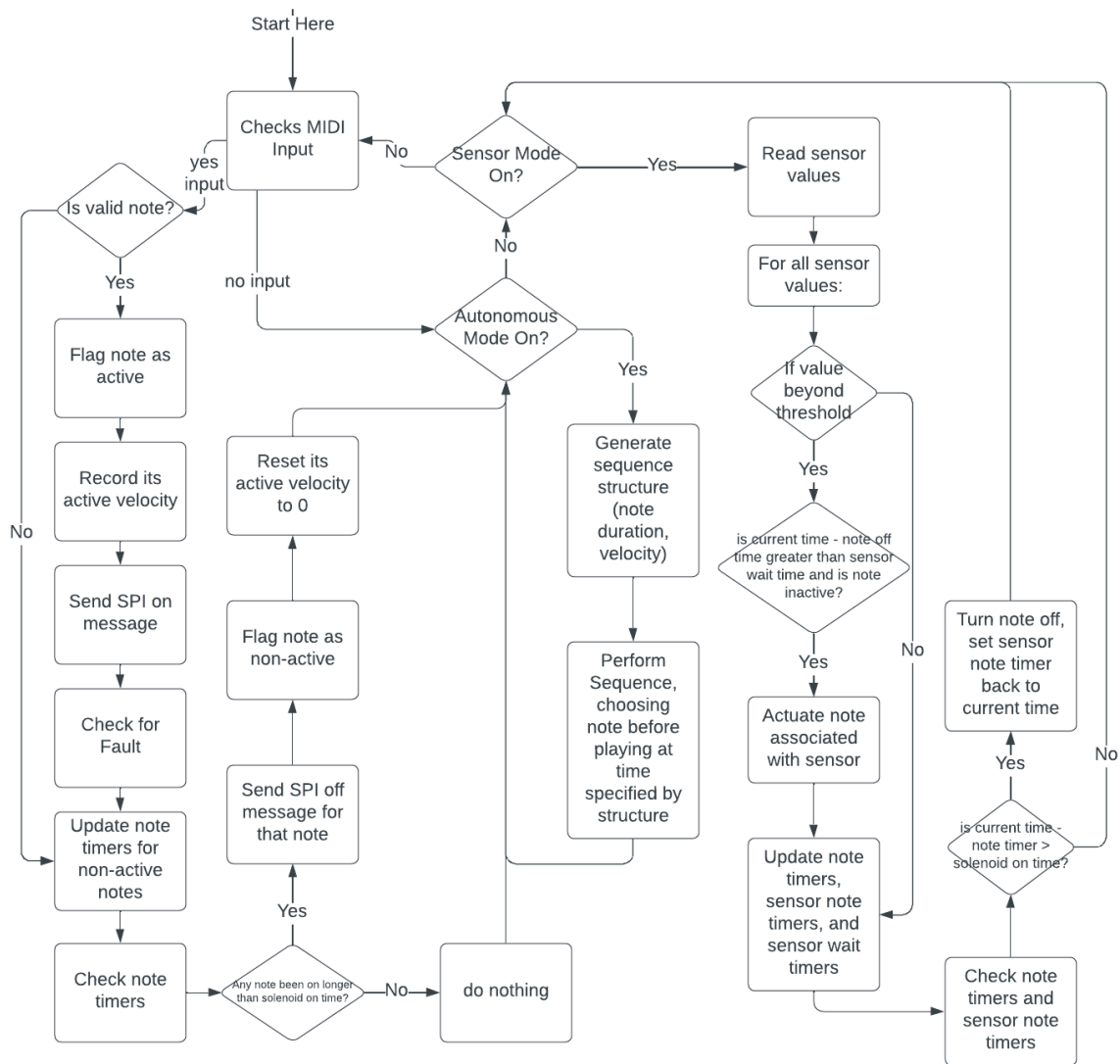


Figure 1: PCB Layout [1]

## 6.2.2. Code Materials

### Code Flow Diagram



Code Files on GitHub:

<https://github.com/liamgwerner/orchestrion>

### 6.2.3. Power Supply, Sensors, and Miscellaneous Electrical Materials

#### Non-PCB Component BOM:

Description	Source	Units	Part number	Price/unit	Total
Feather M4 Express	<a href="https://www.adafruit.com/product/3857">https://www.adafruit.com/product/3857</a>	1	3857	\$22.95	\$22.95
Solenoids	<a href="https://www.adafruit.com/product/413">https://www.adafruit.com/product/413</a>	24	413	\$13.46	\$323.04
PETG Filament	<a href="https://www.amazon.com/OVERTURE-Filament-Consumables-Dimensional-Accuracy/dp/B07PFS4J97?th=1">https://www.amazon.com/OVERTURE-Filament-Consumables-Dimensional-Accuracy/dp/B07PFS4J97?th=1</a>	1	OVPETG 175	\$21.99	\$21.99
Stacking headers	<a href="https://www.adafruit.com/product/2830">https://www.adafruit.com/product/2830</a>	1	2830	\$1.25	\$1.25
Mallets, 2 pcs set	<a href="https://www.amazon.com/Luomorgo-Glockenspiel-Xylophone-Percussion-Drumsticks/dp/B0C8J586XB/ref=sr_1_5?crid=3C9MJB0G9HGPO&amp;dib=eyJ2ljojMSJ9.08XdJeS5zFaknQeU2iqCZXgg3l6rn0lGa-Zj9scpx0M4011XkjMi_NkhCnmu2JdGB60lYLZzRErQQMgTYVf5RqX9U6jNtDPhzPthCINwZ-Dkct7WmhpYkyfsqC4OlwL2m-wpDyPuBFNeLDOKuAU3X5RzBSknSLEx9OtXBQsz85yCTCSN3PzDg16ZLuWoYx4iw7BqMZONY0RzIPkXmQu-KL0ry9OciHzyQNTfpNw6dhO-X36YCR5XqxlpIngC0NxdQK2NUlo70FeH7lo3BFJ8oolrcYMMom2GUQyl8RQQKl.RxFNDYTHd-GHkZddezQAkkAk9NP9kyrLar-dm5qTHvA&amp;dib_tag=se&amp;keywords=luomorgo+2+pcs+glockenspiel&amp;qid=1708453871&amp;srefix=luomorgo+2+pcs%2Caps%2C122&amp;sr=8-5">https://www.amazon.com/Luomorgo-Glockenspiel-Xylophone-Percussion-Drumsticks/dp/B0C8J586XB/ref=sr_1_5?crid=3C9MJB0G9HGPO&amp;dib=eyJ2ljojMSJ9.08XdJeS5zFaknQeU2iqCZXgg3l6rn0lGa-Zj9scpx0M4011XkjMi_NkhCnmu2JdGB60lYLZzRErQQMgTYVf5RqX9U6jNtDPhzPthCINwZ-Dkct7WmhpYkyfsqC4OlwL2m-wpDyPuBFNeLDOKuAU3X5RzBSknSLEx9OtXBQsz85yCTCSN3PzDg16ZLuWoYx4iw7BqMZONY0RzIPkXmQu-KL0ry9OciHzyQNTfpNw6dhO-X36YCR5XqxlpIngC0NxdQK2NUlo70FeH7lo3BFJ8oolrcYMMom2GUQyl8RQQKl.RxFNDYTHd-GHkZddezQAkkAk9NP9kyrLar-dm5qTHvA&amp;dib_tag=se&amp;keywords=luomorgo+2+pcs+glockenspiel&amp;qid=1708453871&amp;srefix=luomorgo+2+pcs%2Caps%2C122&amp;sr=8-5</a>	5	B0C8J586XB	\$6.99	\$34.95
PLA, White	<a href="https://www.dynamism.com/material/material-brands/ultimaker/ultimaker-nfc-pla-white.html?adgroupid=1225955739684353&amp;campaignid=388582310&amp;device=c&amp;keyword=&amp;matchtype=e&amp;msclkid=db7d7e9961e71c09501e1dfbdda0b734&amp;network=o&amp;position=&amp;utm_campaign=Shopping_Ultimaker&amp;utm_content=Shopping_Ultimaker&amp;utm_medium=cpc&amp;utm_source=bing&amp;utm_term=4580221856672496">https://www.dynamism.com/material/material-brands/ultimaker/ultimaker-nfc-pla-white.html?adgroupid=1225955739684353&amp;campaignid=388582310&amp;device=c&amp;keyword=&amp;matchtype=e&amp;msclkid=db7d7e9961e71c09501e1dfbdda0b734&amp;network=o&amp;position=&amp;utm_campaign=Shopping_Ultimaker&amp;utm_content=Shopping_Ultimaker&amp;utm_medium=cpc&amp;utm_source=bing&amp;utm_term=4580221856672496</a>	2	1613	\$49.95	\$99.90
Panel-mount Micro USB	<a href="https://www.adafruit.com/product/4217">https://www.adafruit.com/product/4217</a>	1	4217	\$4.95	\$4.95
IR Distance	<a href="https://www.adafruit.com/product/1031">https://www.adafruit.com/product/1031</a>	8	GP2Y0A0	\$15.95	\$127.60

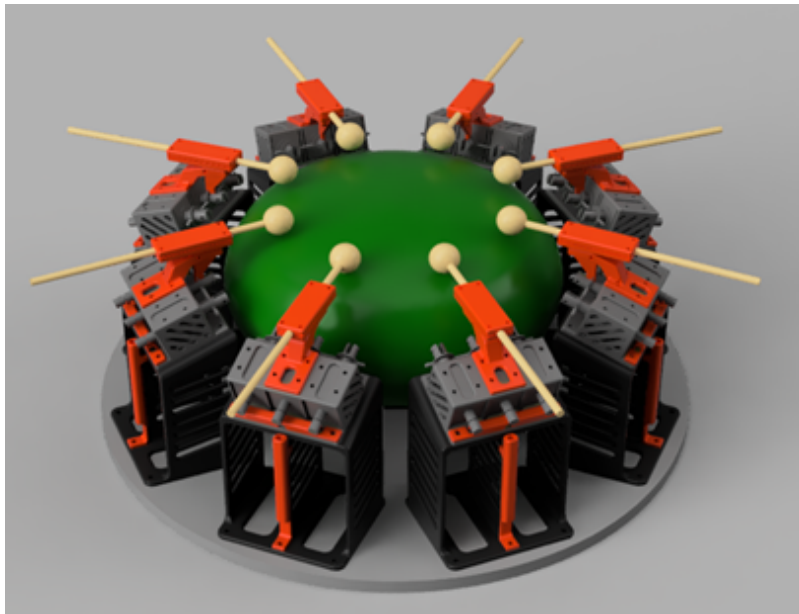


Sensor			2YK		
Jumper Splitters	<a href="https://www.amazon.com/dp/B09P52PLK4?psc=1&amp;ref=ppx_yo2ov_dt_b_product_details">https://www.amazon.com/dp/B09P52PLK4?psc=1&amp;ref=ppx_yo2ov_dt_b_product_details</a>	1	B09P52P LK4	\$7.98	\$7.98
ADC Breakout Board	<a href="https://www.adafruit.com/product/5836">https://www.adafruit.com/product/5836</a>	1	5836	\$5.95	\$5.95
Capacitor	<a href="https://www.yageo.com/upload/media/product/productsearch/datasheet/mlcc/UPY-GPHC_X7R_6.3V-to-250V_23.pdf">https://www.yageo.com/upload/media/product/productsearch/datasheet/mlcc/UPY-GPHC_X7R_6.3V-to-250V_23.pdf</a>	1	311-1126- 1-ND	\$0.10	\$0.10
MOSFET	<a href="https://www.ti.com/lit/ds/symlink/csd88539nd.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-ww&amp;ts=1702164823643&amp;ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Fcsd88539nd">https://www.ti.com/lit/ds/symlink/csd88539nd.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-ww&amp;ts=1702164823643&amp;ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Fcsd88539nd</a>	12	296-3730 4-1-ND	\$0.91	\$10.92
DC Buck	<a href="https://www.ti.com/general/docs/suppproductinfo.tsp?distId=10&amp;gotoUrl=https%3A%2F%2Fwww.ti.com%2Flit%2Fgpn%2Ftps54232">https://www.ti.com/general/docs/suppproductinfo.tsp?distId=10&amp;gotoUrl=https%3A%2F%2Fwww.ti.com%2Flit%2Fgpn%2Ftps54232</a>	1	296-2403 7-5-ND	\$2.03	\$2.03
Diode	<a href="https://www.smc-diodes.com/propdf/SK36A%20N0104%20REV.A.pdf">https://www.smc-diodes.com/propdf/SK36A%20N0104%20REV.A.pdf</a>	24	1655-SK3 6ACT-ND	\$0.48	\$11.52
Resistor	<a href="https://www.seielect.com/catalog/sei-rmcf_rmcp.pdf">https://www.seielect.com/catalog/sei-rmcf_rmcp.pdf</a>	1	738-RMC F0805FT 1K96CT- ND	\$0.10	\$0.10
Resistor	<a href="https://www.seielect.com/catalog/sei-rmcf_rmcp.pdf">https://www.seielect.com/catalog/sei-rmcf_rmcp.pdf</a>	1	RMCF08 05FT10K 2CT-ND	\$0.10	\$0.10
Resistor	<a href="https://www.seielect.com/catalog/sei-rmcf_rmcp.pdf">https://www.seielect.com/catalog/sei-rmcf_rmcp.pdf</a>	1	RMCF08 05FT17K 4CT-ND	\$0.10	\$0.10
Resistor	<a href="https://www.seielect.com/catalog/sei-rmcf_rmcp.pdf">https://www.seielect.com/catalog/sei-rmcf_rmcp.pdf</a>	1	738-RMC F0805FT 48K7CT- ND	\$0.10	\$0.10
Resistor	<a href="https://www.seielect.com/catalog/sei-rmcf_rmcp.pdf">https://www.seielect.com/catalog/sei-rmcf_rmcp.pdf</a>	1	RMCF08 05JT150 KCT-ND	\$0.10	\$0.10
Capacitor	<a href="https://www.samsungsem.com/error500.html">https://www.samsungsem.com/error500.html</a>	1	1276-110 0-1-ND	0.13	\$0.13

Diode	<a href="https://www.diodes.com/assets/Datasheets/B220_A-B260_A.pdf">https://www.diodes.com/assets/Datasheets/B220_A-B260_A.pdf</a>	1	B220A-F DICT-ND	0.47	\$0.47
Capacitor	<a href="https://search.murata.co.jp/Ceramy/image/img/A01X/G101/ENG/GRM21BR61C106KE15-01.pdf">https://search.murata.co.jp/Ceramy/image/img/A01X/G101/ENG/GRM21BR61C106KE15-01.pdf</a>	1	490-6473 -1-ND	0.16	\$0.16
Inductor	<a href="https://product.tdk.com/en/system/files?file=dam/doc/product/inductor/inductor/smd/catalog/inductor_commercial_power_vls6045ex_en.pdf">https://product.tdk.com/en/system/files?file=dam/doc/product/inductor/inductor/smd/catalog/inductor_commercial_power_vls6045ex_en.pdf</a>	1	445-1730 57-1-ND	0.39	\$0.39
Pre-FET Driver	<a href="https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/1103/TPIC46L01_02_03.pdf">https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/1103/TPIC46L01_02_03.pdf</a>	4	TPIC46L 01DB	3.66	\$14.64
Capacitor	<a href="https://connect.kemet.com:7667/gateway/IntelliData-ComponentDocumentation/1.0/download/datasheet/C0805C470J5GAC7210.pdf">https://connect.kemet.com:7667/gateway/IntelliData-ComponentDocumentation/1.0/download/datasheet/C0805C470J5GAC7210.pdf</a>	1	399-C080 5C470J5 GAC7210 CT-ND	0.1	\$0.10
Capacitor	<a href="https://connect.kemet.com:7667/gateway/IntelliData-ComponentDocumentation/1.0/download/datasheet/C0805C153K5RACTU">https://connect.kemet.com:7667/gateway/IntelliData-ComponentDocumentation/1.0/download/datasheet/C0805C153K5RACTU</a>	1	399-C080 5C153K5 RAC7800 CT-ND	0.1	\$0.10
				<b>TOTAL</b>	\$691.62

6.2.4. Mechanical and Enclosure Materials

Mechanical CAD Model Full Assembly



Individual CAD/STL Files can be found in File Links section 6.4.2. [2]

## 6.3. Presentation Materials

### Project Poster Image

**COLLEGE OF ENGINEERING**
**Electrical Engineering and Computer Science**
**ECE 33**

**SUMMARY**

The purpose of this interdisciplinary project was to explore the intersection between engineering and the arts and to create a steel tongue drum robot capable of innovative performances, autonomous sequence generation, and human interactivity.





Figure 1: Full System

**ENGINEERING REQUIREMENTS**

- Tempo (Actuation Frequency):**  
All mallets will be able to strike faster than 8Hz
- Dynamic Range:**  
There will be at least 3 distinct dynamic levels
- Latency:**  
Less than 50ms between MIDI input and actuator strike
- Thermal Management:**  
System withstands 1 minute of repeated actuation at highest achieved frequency
- MIDI Input:**  
Standard USB MIDI protocol controls each mallet arm independently
- Autonomous Performance:**  
Audience agrees that the OctoDrum generates interesting music
- Human Interactivity:**  
Autonomous performance is noticeably responsive to interaction via distance sensors
- Power Supply:**  
Only requires power from one US outlet and one USB cable



# OCTODRUM

An electromechanical orchestration device for performance and interaction on a steel tongue drum

SYSTEM-LEVEL BLOCK DIAGRAM

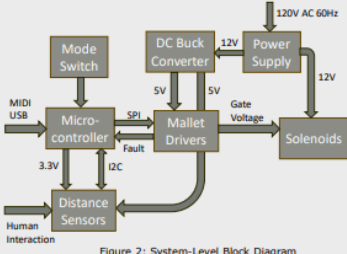


Figure 2: System-Level Block Diagram

HARDWARE

**Power**

- Power supply provides stable 12V DC to the solenoids from standard electrical outlet with ON/OFF switch, 24 amps peak current
- DC buck converter regulates internal 5V supply for distance sensors and mallet driver circuitry

**PCB**

- Serial Peripheral Interface on pre-FET drivers controls 24 MOSFET devices which actuate solenoids with flyback protection
- Open-load and shorted-load fault protection

**Sensors**

- SHARP infrared distance sensors detect linear proximity from 20cm to 150cm on each arm
- Dynamically adjusts drumming speed based on performer and/or spectators for an interactive musical experience once system is in sensor mode

SOFTWARE

**MIDI Input**

- Adafruit Feather M4 Express receives real-time USB MIDI data and sends appropriate SPI messages to mallet driver ICs

**Autonomous Performance**

- Microcontroller generates music sequences using Markov matrices and probability distributions

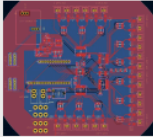


Figure 3: Fully-Integrated PCB Layout




Figure 4: Dynamic Levels for 3-Solenoid Design





Figure 5: Markov Chain example for one note, C4

**PROJECT CUSTOMER**


Dr. Chet Udell is an Associate Professor of Biological & Ecological Engineering at OSU, director of OPENS lab, and designer of electronic music instruments.



The OctoDrum is the first of Dr. Udell's lineup of orchestrations — interactive electromechanical musical instrument systems. Go to [chetudell.com](http://chetudell.com) to check out his other innovative musical projects.


**ENGINEERING TEAM**


The OctoDrum was designed by a multidisciplinary team of electrical and mechanical engineering students.





Left to Right: Nicholas Kim, Kexin Liu, Gianluca Rianda, Liam Warner, Liam Hodge, and Luke Lutnesky


**ECE TEAM MEMBERS**




**Nicholas Kim**  
kimnich@oregonstate.edu  
PCB design/layout/assembly, solenoids  
LinkedIn Profile: 



**Liam Warner**  
warnerli@oregonstate.edu  
Programming, microcontroller interface validation, autonomous mode design  
LinkedIn Profile: 



**Kexin Liu**  
liukex@oregonstate.edu  
Power supply, DC buck converter, IR sensor design  
LinkedIn Profile: 

[Link to Project Showcase Webpage:](https://ecs.engineering.oregonstate.edu/project-showcase/projects/?id=86E19S1gOmNz1X7N)

<https://ecs.engineering.oregonstate.edu/project-showcase/projects/?id=86E19S1gOmNz1X7N>

## 6.4. References and File Links

### 6.4.1. References (IEEE)

### 6.4.2. File Links

[1] KiCad Project Files (contact Chet Udell for access to Orchestrations shared drive):  
<https://drive.google.com/file/d/1w8usB-FgqWetEHRBm7RtfughDvBwxGcK/view?usp=sharing>

[2] OctoDrum CAD Files:  
<https://drive.google.com/drive/folders/1ajvJDmSWHbPcZiUGvtIjD6yU9zsJaD5o?usp=sharing>

## 6.5. Revision Table

04/25/2024	Liam: created section and filled in some details
04/26/2024	Liukee: Added and filled in some details
4/26/2024	Nicholas Kim: Added KiCad materials
5/16/2024	All: Fixed table formatting, revised materials list and total cost, updated expo poster image, and revised file links order