Designing an Accessible MIDI Controller: Enhancing Usability and Reducing the Educational Barriers in Music Production for Beginners

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Haizhou Yu

Spring 2025

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Advisor

Adam Barnes, Charles L. Brown Department of Electrical and Computer Engineering

To: Be Continued Smart MIDI Controller for Novice Music Producers

Statement of Work

Jennifer Shern: At the start of the semester, I concentrated on writing the proposal and creating the poster. Following this, Wanghaotian and I researched and investigated suitable PCB structures. My responsibilities included drawing the schematic, laying the PCB, and placing the order. While awaiting the PCB's arrival, I assisted in debugging and improving the user interface. Wanghaotian and I collaboratively completed the hardware integration of the Smart MIDI Controller for the final integration. In this final report, I wrote the Abstract, Background, Project Description (performance objectives and specifications, most of the How It Works section, and the PCB Design of Technical Details), Social Impact, External Standards, and Engineering Insights.

Haizhou Yu: As the team's project leader, I am responsible for various parts of the project. First, as a leader, I planned the features and investigated the materials and hardware to construct the device. Then, I researched the software and coded the MIDI section of the device. I am also responsible for the CAD construction, the structural planning, and design of the device. Besides, I worked between various groups and assisted in many project parts, such as PCB design, UI, 3D printing, and more. I also supervised the work split in each of the team members. I oversaw most of the parts and possess the know-how in the deliverables. In this report, I am responsible for the MIDI technical details and the CAD and structural parts.

Wanghaotian Zhang: I wrote half of the project proposal for the first month of the semester. I also made the poster for the poster session. After the poster session, I helped generate ideas on the PCB design and figured out the structure. After the arrival of the PCB parts, I helped to solder the button PCBs and main PCBs and did basic testing on the connectivity of PCBs. For the final assembly part, I helped to correspond buttons and GPIO ports, tried to assemble the PCBs into final parts, and did the final testing to verify functionality. I wrote the testing plan, physical constraints, and costs for the final report.

Tong Zhou: In the first half of the semester, I focused primarily on UI design. I spent the first two weeks researching and exploring potential libraries and IDEs, followed by implementing and displaying the design on the screen. Additionally, I assisted in designing, soldering, and testing the PCBs. During the final assembly, I tested the calibration of buttons linked to the GPIO ports and collaborated with all group members to test functionality. For the final report, I handled the technical details of the UI design.

Jinhong Zhao: At the beginning of the semester, I focused on learning the necessary libraries for UI design and familiarizing myself with CircuitPython. I actively participated in discussions on the appearance of the user interface and ensured that the functionality of the 5-way button was effectively integrated with the code. Subsequently, I transitioned to working on the software side, contributing to implementing chord and key mapping for the MIDI controller. In the meantime, I spent time developing my 3D printing skills, which allowed me to access the 3D printing facilities at our school and ensure the timely printing of our project components. During the integration phase, where the UI design was combined with the software, I tested edge cases and debugged potential errors to ensure seamless functionality. For the final report, my primary contributions included drafting sections on intellectual property issues, project timelines, and proposed future work.

Table of Contents

Abstract

The evolution of technology has transformed the landscape of musical composition and performance beyond traditional physical instrumentation. The Musical Instrument Digital Interface (MIDI) controller represents a significant advancement in this domain, facilitating the generation of digital music based on the input MIDI data. [1] To be more specific, MIDI data triggers the virtual instruments or synthesizers within the Digital Audio Workstations (DAWs), which allows the computer to generate audio output. This project focuses on developing an innovative MIDI controller that encompasses the comprehensive functionality of existing commercial models while introducing novel features designed to enhance the learning curve for beginning music producers, thereby optimizing the accessibility of this technology.

Background

The MIDI controller plays a critical role in modern music production. This technology enables a more straightforward and cost-efficient approach to music production for all genres. Users can create a wide variety of samples without purchasing the actual instrument. Moreover, using MIDI over the instruments recording provides multiple advantages in terms of adjusting specific aspects of music, including volume, panning, and

modulation [2]. As many of our team members had previous experiences of music production, we decided on this Smart MIDI Creator project, which benefits novice music producers.

Despite the variety of models available, most of them need to be more user-friendly, especially for novice users. Music production demands a foundational understanding of music theory; harmonious music requires specific notes arranged into appropriate chord progressions. As each key or pad on current models only corresponds to one note, music producers must memorize the corresponding notes of the chord. In addition, the relationship between notes and buttons often needs to be more intuitive due to the limited instructions. For MIDI controllers with a drum pad structure, users must familiarize themselves with the layout to learn the location of each note, which can harm the learning curve for beginners. Admittedly, the keyboard structure MIDI controllers do not have this issue, as the keys closely resemble those of a piano. However, these models typically provide a left-to-right layout with limited vertical options. Non- keyboard MIDI designs, such as drum-pad controllers, allow for more keys within a compact area, which enhances portability without sacrificing user experience.

Several MIDI controller models are addressing these market deficiencies. Here, I will discuss two such systems. The Launchpad Pro [MK3] by Novation features a drum pad structure with various functionalities that are manipulated by its corresponding software [3]. Users can switch to chord mode through the software, enabling each pad to correspond to a chord instead of a note. This alleviates the need to memorize note-chord correspondence. However, the complexity of the software is not suitable for initial beginners, which can pose a steep learning curve. Another downside is that users must manage both the MIDI controller software and the DAW, leading to overcrowded screens and constant disruption switching.

Furthermore, while the pads light up when pressed, identifying the corresponding notes or chords depends on the MIDI controller software. This design may be counterintuitive, as users are likely to focus on the pad instead of the screen. Although several MIDI controller software options, such as Dexed, include a chord mode, compatibility issues with operating systems can limit usability [4]. Moreover, based on our investigation into musicians' preferences, they favor the experience of physical pads or keys over virtual ones on the screen. In conclusion, none of the current models sufficiently addresses the challenges mentioned above.

Our Smart MIDI Controller distinguishes itself from the competition with various innovative features that enhance both usability and creativity. Featuring an embedded screen, the controller visually displays the corresponding notes or chords, enabling users to swiftly identify which notes or chords each button represents. Furthermore, our Chord Mode and Strum Mode provide novice users with a smooth experience by removing the

challenges associated with chord progressions. These features greatly lessen the learning curve, allowing new users to concentrate more on their music and less on technical aspects, which promotes a more user-friendly production process.

Our coursework has established a solid foundation for the success of this project. The project is divided into three main components: hardware, software, and CAD modeling.

1.! Hardware: For the hardware aspect, we utilized PCBs to transmit the input signals from the buttons to the microcontroller. Our understanding of circuit design and layout was acquired in the Fundamentals of Electrical Engineering course series.

2.! Software: In terms of the software component, our experience with embedded systems and various computer science courses has provided us with the skills necessary to work with microcontrollers. This experience enables us to effectively handle the user interface (UI) while generating and sending MIDI data.

3.! CAD Modeling: Although none of us have taken a specific course in CAD modeling, one of our team members has prior experience in this field and was responsible for designing the exterior casing of the project.

Project Description

!"#$%#&'()"*+,-")./0"1*'(2*34")/$)'./%(1!

Our MIDI controller combines the functions of traditional MIDI controllers with innovative features tailored for learners, who are our intended users.

•! Screen Feature

The proposed Smart MIDI controller sets itself apart with an advanced interactive screen interface. Unlike conventional MIDI controllers that are limited to keys or drum pads, our design incorporates a 5.62” x 9.25” display screen. The upper section of the screen provides comprehensive operational instructions and mode selection features. Users will navigate mode selection using a five-way button located next to the first row of buttons. More details about the modes will be elaborated in the upcoming paragraphs below.

The lower screen area, located beneath transparent buttons, dynamically displays contextual text that adjusts based on the selected operational mode. This smart screen design promotes intuitive mode selection and offers guidance on the placement of certain chords, effectively reducing the traditional learning curve linked with complex MIDI controller software interfaces while also improving the overall user experience.

- Traditional Mode

  The MIDI controller operates like current products in markets, where each button corresponds to a musical note. The design covers 12 chromatic notes within a single octave, including C, C#/Db, D, D#/Eb, E, F, F#/Gb, G, G#/Ab, A, A#/Bb, and B.

- Chord Mode

  This represents one of the two distinctive modes unique to our Smart MIDI Controller. Upon mode selection, the user interface dynamically presents an array of musical keys for selection. Once the user selects the key, the screen beneath the buttons populates the corresponding chords. This innovative approach eliminates potential disharmonies in chord selection, providing substantial benefits for novice users with limited music theory comprehension.

- Strum Mode

  The second unique feature is an advanced strum mode that simulates a guitar performance. Users first select a musical key and then choose three preferred chords from a key-specific chord inventory. In this mode, each button on a single row accurately represents an individual guitar string. When users sweep in a single row, the Smart MIDI Controller generates MIDI data that closely mimics authentic guitar strumming techniques.

5%6*7.*8%#91*

From a user's viewpoint, manipulating our Smart MIDI Controller shown in Figure 1 is akin to operating current MIDI controllers. To begin, the user needs to connect the USB cable located at the top of the device to a computer. Pressing a button sends MIDI data for a note or chord to the computer, which produces the audio output.

!"#$%&'()*+&%",%'-"&.',/'+0&'123%+'4565'7,8+%,99&%''

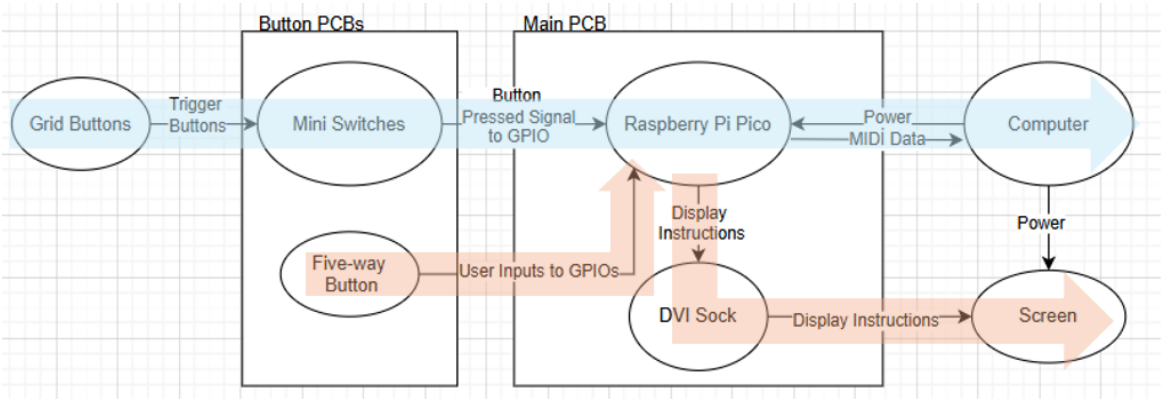
Figure 2 shows the vertical cross-section view of the Smart MIDI Controller. When a Grid Button is pressed, it activates the corresponding switch on the Button PCB, sending an input signal to the Main PCB, where the Raspberry Pi Pico (referred to as Pico in the following paragraphs) is located. Pico then generates the appropriate NoteOn MIDI message and transmits it to the computer. Conversely, when a button is released, a NoteOff MIDI message is sent.

!"#$%&':'-&%+";39'-"&.',/'4565'7,8+%,99&%'

The internal operation steps are outlined in the block diagram shown in Figure 3. The computer supplies power to both the screen and the Main PCB. As mentioned earlier, pressing a Grid Button activates the corresponding mini switch on the Button PCB, which sends a signal that is converted into MIDI data and transmitted to the computer. This signal path is highlighted in light blue in Figure 3.

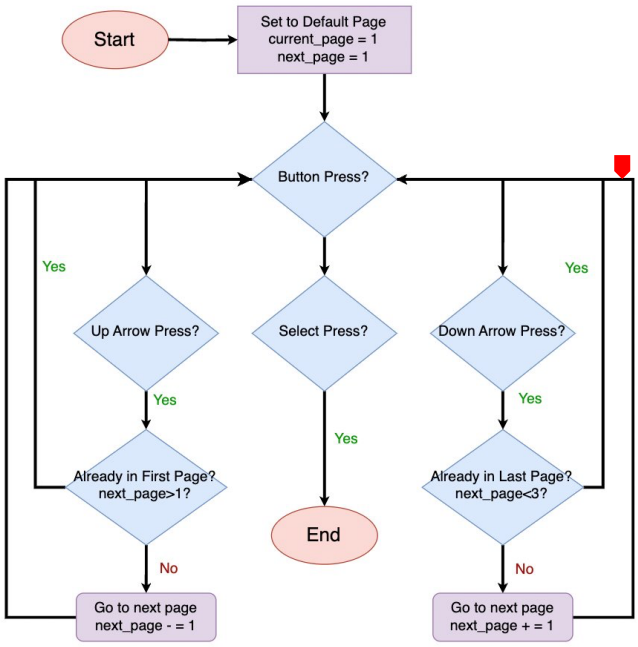
For the five-way button, the up, down, left, and center directions connect to different GPIOs. The user directly manipulates the five-way button, which is soldered onto the upper Button PCB. Based on the current display, the Pico generates the

appropriate instructions and outputs them via the DVI socket to the screen. This signal path is highlighted in orange.



!"#$%&'<'=9,;>'6"3#%32',/'+0&'123%+'4565'7,8+%,99&%'

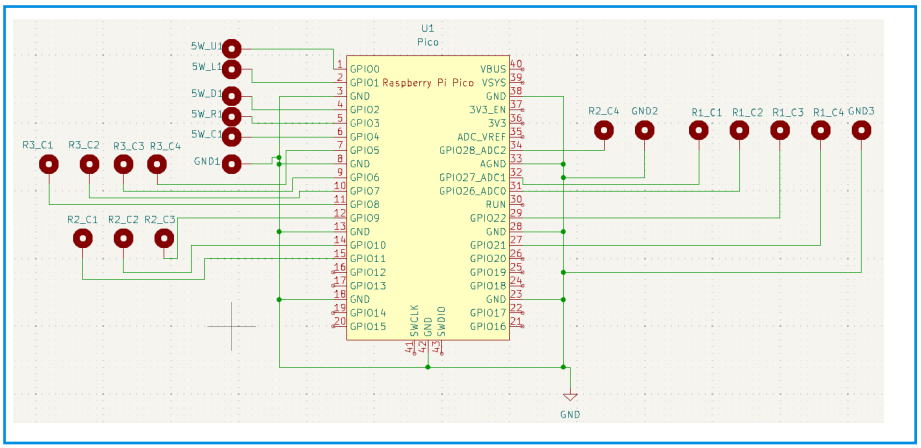
Figure 4 illustrates the block diagram for designing the User Interface. The two constructed variables are current_page, which represents the page currently displayed on the screen, and next_page, which denotes the intended page that the user selects. The basic logic adheres to a Finite State Machine (FSM) algorithm, where current_page indicates the current state, while next_page signifies the intended target state. Initially, the system sets both current_page and next_page to 1- the default page. The FSM accepts three possible inputs based on user actions: Up Arrow Press, indicating the user's desire to move to a previous page; Down Arrow Press, signaling the user's desire to progress to the next page; and Select Press, which confirms the selected page. When the user presses the buttons, corresponding transitions are triggered. For the Up Arrow Press, the system navigates the user to a previous page if they are not already on the first page; for the Down Arrow Press, it advances the user to the next page unless they are already on the last page; and for select press, it concludes the FSM by confirming the current selection page.



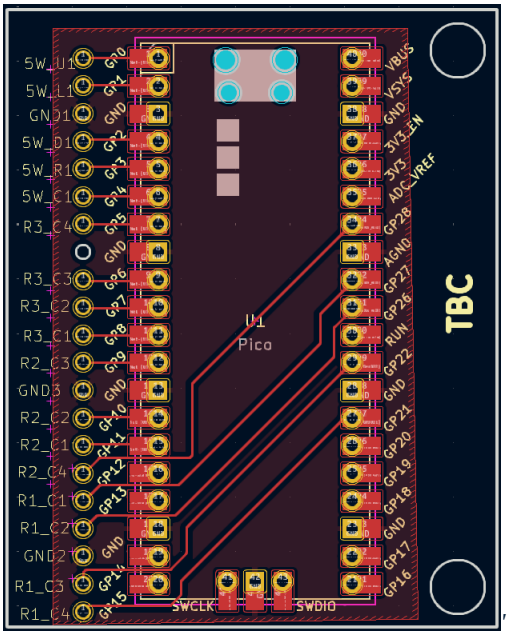
!"#$%&'?'=9,;>'6"3#%32',/'+0&'@A&%'58+&%/3;&'

The circuit schematic of the Main PCB is illustrated in Figure 5. Although it is not shown in the schematic, the DVI socket is soldered to the Pico and occupies GPIO12 through GPIO19. The functions of the other GPIOs are listed below, and they connect to the through holes on the left edge of the Main Button PCB, as shown in the layout in Figure 6.

- GPIO0 through GPIO4 connects to the five-way button, which corresponds to up, left, down, right, and center, respectively.
- GPIO5 to GPIO8: they are designated for the four mini switches of the Button PCB on the top row.
- GPIO9 to PGIO11 and GPIO28: they are designated for the four mini switches of the Button PCB on the middle row.
- GPIO21, 22, 26, 27: they are designated for the four mini switches of the Button PCB on the bottom row.
- GND: there are a total of three GND connections through holes, and they are connected to each Button PCBs.

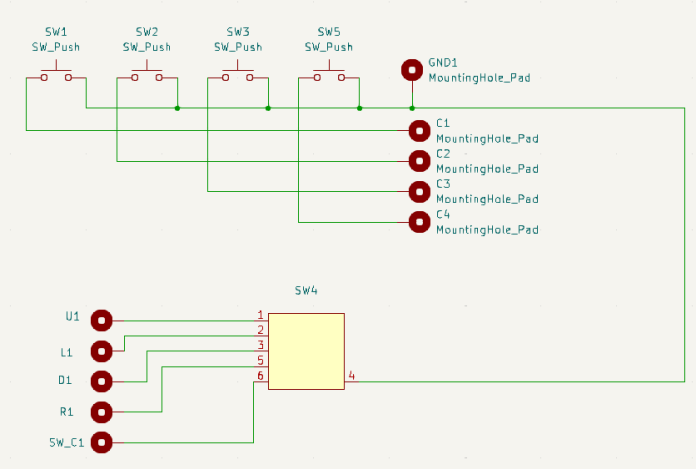


!"#$%&'B'43"8'C7='7'%;$"+'1;0&23+";'

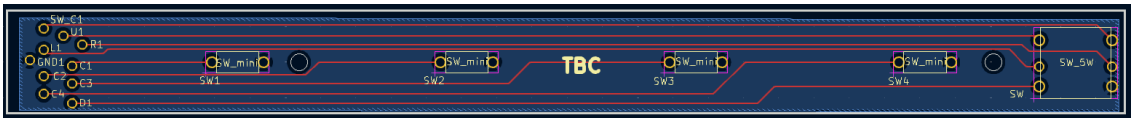


!"#$%&'D'43"8'C7='=,3%E'F3G,$+'

The circuit schematic of the Button PCBs is presented in Figure 7. Each mini switch on the Button PCB corresponds to a 3D printed Grid Button, which users can press. The five-way button is located on the far right of the Button PCB. All pins from the mini switches and the five-way button connect to the through holes on the left, as illustrated in Figure 8. We connected the through holes on the Main PCB and Button PCBs using jumper wires to transmit the user input signals to Pico.



!"#$%&'H'=$++,8'C7='7'%;$"+'1;0&23+";'



!"#$%&'I'=$++,8'C7='=,3%E'F3G,$+'

:") ; (/) '<*="."/<1*

"#$!%&'()!*

- *#+, -+*&*.!/&0&1.(+*!!*

The first step in PCB design was to gather component information. The main PCB only required the Pico manufactured by Adafruit [5]. For the Button PCBs, we decided to use the Thru-hole Navigation Switch, referred to as the five-way button in other sections, to control the UI display. This five-way button was selected instead of other options because it is also produced by Adafruit. The Button PCBs were situated below the frames between two rows of Grid Buttons. Therefore, we selected the 3x6x5mm Mini PCB Push Button Switch, which is the smallest switch we found [6].

- PCB Design Tool Selection

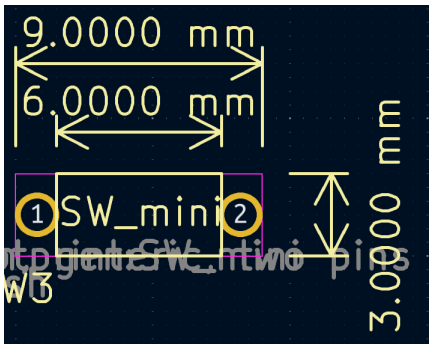
  We chose KiCad as our PCB design software. It is an open-source tool, which offers more information online compared to Multisim. Additionally, we found the schematic and layout library already created for Pico, which can be directly imported into KiCad.
- Footprint
  - Through-holes

    Initially, we intended to solder a 6-position right-angle receptacle connector onto the Main PCB [7]. This connector was meant to replace direct wire soldering to the through-holes, enabling us to disassemble the

three-layer case of the Smart MIDI Controller. Given our mission to minimize the case height, this type of connector is the flattest one we discovered. The through-hole footprint was designed according to the datasheet, featuring a hole diameter of 1.1mm and a 0.4mm annular ring. Unfortunately, we accidentally left insufficient space between the through holes, which prevented us from using the connectors. Consequently, we modified the middle case to allow for device disassembly but opted to solder wires directly to connect the Main and Button PCBs instead.

- Mini Switch

  There was no existing footprint for the Mini Switch, so we created one based on our measurements of the switch. As shown in Figure 9, the Mini Switch measures 3 x 9 mm, with a hole size of 1.2 mm and a 0.4 mm annular opening ring.



!"#$%&'J'4"8"'1."+;0'!,,+K%"8+'

- Five-way button

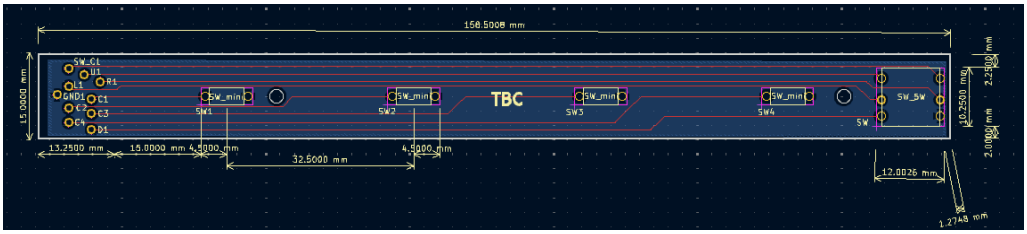
  We drew the footprint of the five-way button based on its datasheet [8]. All the holes have an annular ring of 0.4 mm.

- Layout
  - Main PCB

    The through-holes on the Main PCB are aligned vertically along the left edge. Initially, the design included two holes on the right for screws to secure the PCB to the back of the display screen. However, we ended up placing the PCB inside the case without using the screw holes. To reduce wiring inductance, a ground plane was implemented on the copper underside of the board [9]. The dimensions of the Main PCB are 45 mm by 57 mm.
  - Button PCB

    Each Mini Switch is placed at the middle length of the 3D printed button. Each Mini Switch is positioned at the midpoint of the 3D printed button. The length of the Button PCB is designed to be slightly longer than the width of the display screen. The footprint of the five-way button is

located at the far right of each Button PCB. However, the five-way button is only soldered onto the Button PCB that is nearest to the user interface display. A ground plane is also created at the bottom of the Button PCBs. The five-way button is located at the very right of every Button PCB. Yet, the five-way button was only soldered on to the Button PCB that is located closest to the user interface display. The ground plane is also created at the button of the Button PCBs. The detailed dimensions can be found in Figure 10.



!"#$%&'(L'=$++,8'C7='F3G,$+'

•! Fabrication

We chose JLCPCB as the manufacturer of our PCBs due to budget constraints.

2 (13+1+*.3+00&3!"3+)34 , , (*)!

Raspberry Pi Pico was selected to be our microcontroller. Contrary to Raspberry Pi, which is a minicomputer, Raspberry Pi Pico is a programmable board without operating system [10]. Raspberry Pi Pico supports both MicroPython and CircuitPython. Considering that we had no prior experience in Raspberry Pi Pico, we chose to program with CircuitPython, which provides a more straightforward interface than MicroPython [11]. Additionally, CircuitPython includes a built-in MIDI library, making it easier to find well-documented MIDI resources [12]. STM32 is also one of the options we were considering since there's an existing MIDI library found [13]. Yet, Raspberry Pi Pico stands out due to the simplicity of its coding language and its cost efficiency. We would prefer Raspberry Pi Pico 1 rather than 2 for two reasons. First, version 2 is reported to have hardware flaws where the pin latches at 2.15V. In addition, version 2 is only available in backstock. Considering our time constraints, we only order parts that are not in backstock.

As this is our first experience coding with the Raspberry Pi Pico, our capstone team assessed Thonny and Mu as Integrated Development Environments (IDEs) for CircuitPython, focusing on usability, features, and compatibility. Thonny distinguished itself with its user-friendly interface, making it ideal for team members who were unfamiliar with CircuitPython programming. It includes a built-in debugger that allows for step-by-step program execution, simplifying the identification and resolution of issues

during development. Moreover, Thonny integrates smoothly with CircuitPython, offering features such as syntax highlighting and an interactive commands [14]. While Mu offers a solid alternative, Thonny's simple interface and effective debugging tools make it the preferred choice for our project. Additionally, there are many online tutorials available for Thonny, which facilitate easier learning.

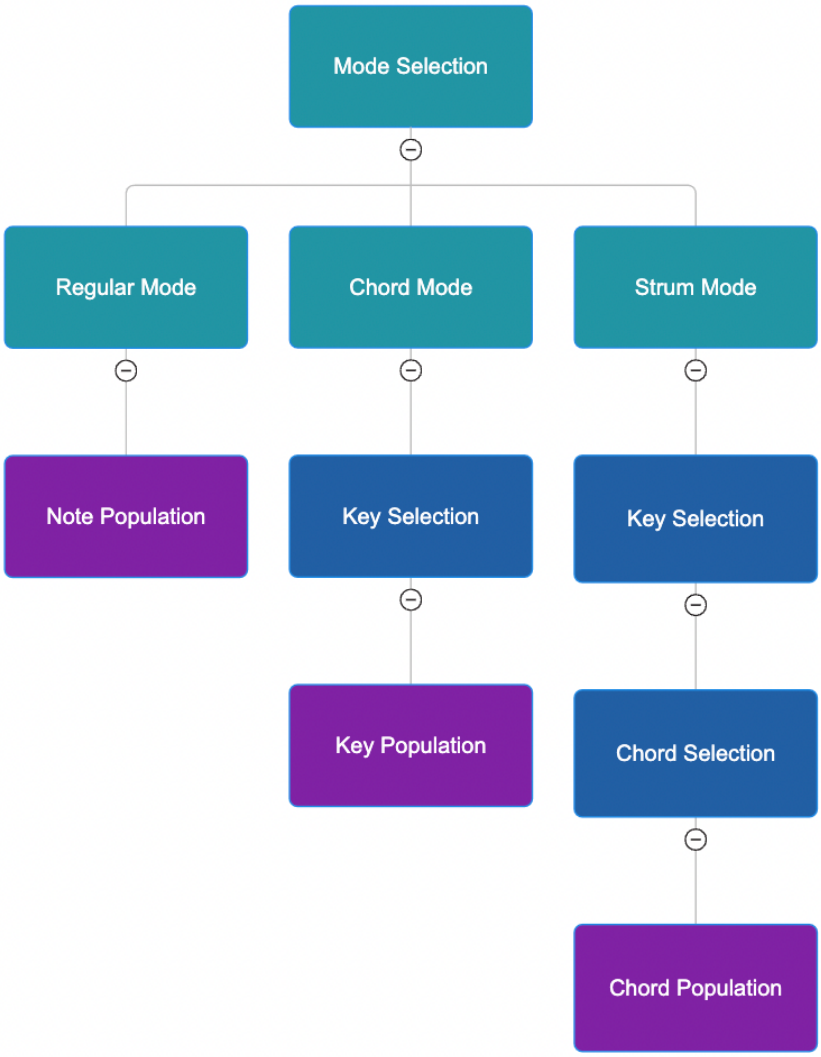
!

"#$%!& '($%)+$!, "&-! . $#/0 '!*

The UI implementation visually provides instructions for users and guides them to the intended functions. The design primarily consists of two components: user inputs via buttons and graphic output. It is built on a Raspberry Pi Pico using CircuitPython and manages multiple modes, such as traditional, chord, and strum, while navigating between pages.

- ! Display layout

According to the Tree Diagram shown in Figure 11, the display consists of three main sections: the top section, which features the current mode at the top center of the screen; the middle section, which contains interactive elements like chord and key selections; and the bottom section, which shows the chosen chord or key screen.



!"#$%&'(('@5'6&A"#8'M%&&'6"3#%32'

•! Expectation

The key requirements for UI design are, firstly, dynamic updates that provide immediate feedback to the user based on their input, and secondly, intuitive usability that ensures instructions are clear and straightforward, minimizing the effort needed from the user to understand.

•! Library/Packages Selection

When evaluating the libraries in use, hardware support is essential. The selected libraries must facilitate low-level communications such as I2C or SPI while also providing high-level abstractions to simplify the implementation process. Consequently, the primary libraries we utilize include Displayio [15], which manages visual components

through “groups,” eliminating the need to redraw everything during dynamic updates and supporting a wide variety of displays and protocols. We also consider Framebufferio [16] as an alternative; however, its lower-level implementation requires more manual oversight. Thus, we prefer Displayio over Framebufferio. The next significant library is Adafruit_display [17], which simplifies the creation and positioning of elements like text and shapes. While exploring potential packages, we also evaluate Terminalio [18], which is suitable for text handling but lacks advanced layout features, complicating its use for dynamic user interfaces. In contrast, Adafruit_display allows for dynamic content updates without a full-screen refresh and offers numerous useful functions, such as front scaling, alignment, and anchored positioning. With these factors in mind, we opt for Adafruit_display. Lastly, the key library, Digitalio [19], is employed to configure and read GPIO pins for managing button inputs. It facilitates interaction with physical buttons and provides a straightforward API for setting pins as inputs or outputs and configuring pull-up resistors, ensuring stable functionality for buttons signals.

- Setup

To start, we configured the screen using the implemented display library, setting the resolution to 320 x 240. This resolution keeps a balance between visual quality and reduced memory usage and processing demands. Next, we created a default text label that displays "Regular mode” to indicate the current mode. For the shape setup, we established a root group to contain all the graphical elements that will be initially displayed. This includes a rectangle frame, triangles that serve as navigation pointers, and grids that present lists of keys or chords. The position of each grid is determined by the following calculations:

$$\text{Horizontal spacing} = \text{Starting x-position} + (\text{Column index} \times \text{grid width})$$

$$\text{Vertical spacing} = \text{Starting y-position} + (\text{Row index} \times \text{grid Height})$$

- User interaction handling

To manage user interactions, we configured each button to input mode with a pull-up resistor, ensuring stable readings. This setup allowed each button to trigger a specific action as explained earlier. Next, we implemented button press detection using an infinite while loop that continuously checks if a button was pressed. Since the buttons are designed as active low, we determined their state by reading the signal: a low value indicates the button is pressed, while a high value indicates it is unpressed. Finally, a brief delay is implemented after detecting a button press to prevent multiple actions from being triggered by a single press. This delay is typically short, about 300 milliseconds, ensuring that it does not create a noticeable wait for users.

- Display Update

To provide dynamic and immediate feedback, we first updated the text label to reflect the new selection while clearing the previous content. Next, we adjusted the elements based on the user’s actions. This adjustment occurs during transitions between modes, such as when a user selects a mode or moves from one selection to another,

including from the last selection back to the previous one. We removed the upper arrow for the first selection to indicate the start and eliminated the down arrow for the last selection to signal the end. Finally, garbage collection is performed to free memory after elements are removed, as they continue to consume memory even when they are no longer visible.

•! Challenges faced and solutions

We faced two significant challenges during the design phase. The first was delayed updates; after a button is pressed, the response time for updates is quite slow, which could negatively affect user experience. To address this, we optimized updates by changing only the elements that vary based on input, rather than refreshing the entire screen. The second challenge pertains to display clarity; the low resolution (320 x 240) limits the amount of information that can be shown. Consequently, we decided to implement a minimalist design featuring large text and shapes to avoid memory overuse while improving clear instructions.

1 & . &!2%'#3/##/4'!5%40%*3 3/'0!*

•! Introduction and Background

As discussed above, one reason we chose the Raspberry Pi Pico is that we can rely on the versatile CircuitPython libraries to achieve our functionalities. Adafruit Industries has conveniently created a MIDI library that enables the Pico to communicate with MIDI-supported devices via a serial connection. The documentation offers many examples for self-study and understanding, making it easy to learn how to use them.

•! Libraries

The primary library we used for this project is the adafruit_midi library [20]. We utilized the library's functions to configure the MIDI channels and ports and to send MIDI messages with a default velocity of 127 (maximum velocity). Another library we employed is the Digitalio library, which assisted us in configuring the GPIO ports. The system continuously monitors whether a button has been pressed, allowing it to respond to actions accordingly.

•! Functionalities

Our device provides functions beyond the standard MIDI controller mode, requiring workarounds for features like chords and dynamic note splitting in the 12-button design without using excessive resources. The software is in an infinite loop, so we added a section that plays user-selected notes and chords when a button is pressed, sending a “MIDI on” message to the computer with each iteration. Both the detection and callback occur within the same iteration of the loop.

•! Traditional Mode

The normal mode consists of an array of default MIDI notes that we predefined in the code. This array starts from C3, also known as central C, and moves one half-note for

each subsequent button. Since the notes in a MIDI message are represented by two-digit numbers, this array is essentially a collection of integers. When the normal mode is activated, the code selects this array and associates it with the corresponding button. Because both the button array and the note array follow a matching index sequence, they can be assigned accordingly.

•! Chord Mode

Following this logic, we developed the chord mode. First, we gather common chords for each key and code them into arrays of strings. Due to the extensive usage of chords across both the UI and MIDI programming, we kept everything as strings to unify variable names for better integration. Next, we implemented a chord translation function to convert the chord name in string format to the exact note numbers in MIDI. When the chord mode is selected, the software precomputes the chord notes and stores them in a nested array. When loading the chords into buttons, the same logic applies as in the normal mode implementation. However, the function needs to incorporate a loop to retrieve the notes from the nested array so that they can be played at the same time.

•! Strum Mode

The strum mode is based on the chord mode but includes slight alterations. Due to the buttons being arranged in a 3 by 4 grid, we can only include three chords total, with four notes per chord. This means that after the user selects the key, they also need to choose their chord of preference. This is then sent into the chord translation function and returned as an array. Another function adds the root note an octave higher to the array, completing the four notes per chord format. A special loading technique was developed so that it takes the notes in the nested for loop and, instead of loading them all with one button, loads them one note at a time button.

!

#5%! 2 +6&0(*)!4*6!/.371.7340!%&'()*!

•! Introduction and design inspirations

To create a more flexible design and assembly option, we developed CAD designs and 3D printed them to finalize our enclosure. We utilized Autodesk Fusion 360 to construct our CAD design and engaged 3D printing companies to meet our specific requirements for transparent materials and oversized components. One of our main goals during the device's design was to make the buttons transparent, allowing them to display different content based on their functions. We explored designs from various devices, such as stream decks, which typically feature transparent buttons along with an embedded screen, and a gaming arcade called Jubeat, which closely resembles the final design we implemented, including a display area for navigation information and the rest of the screen to present content for buttons.

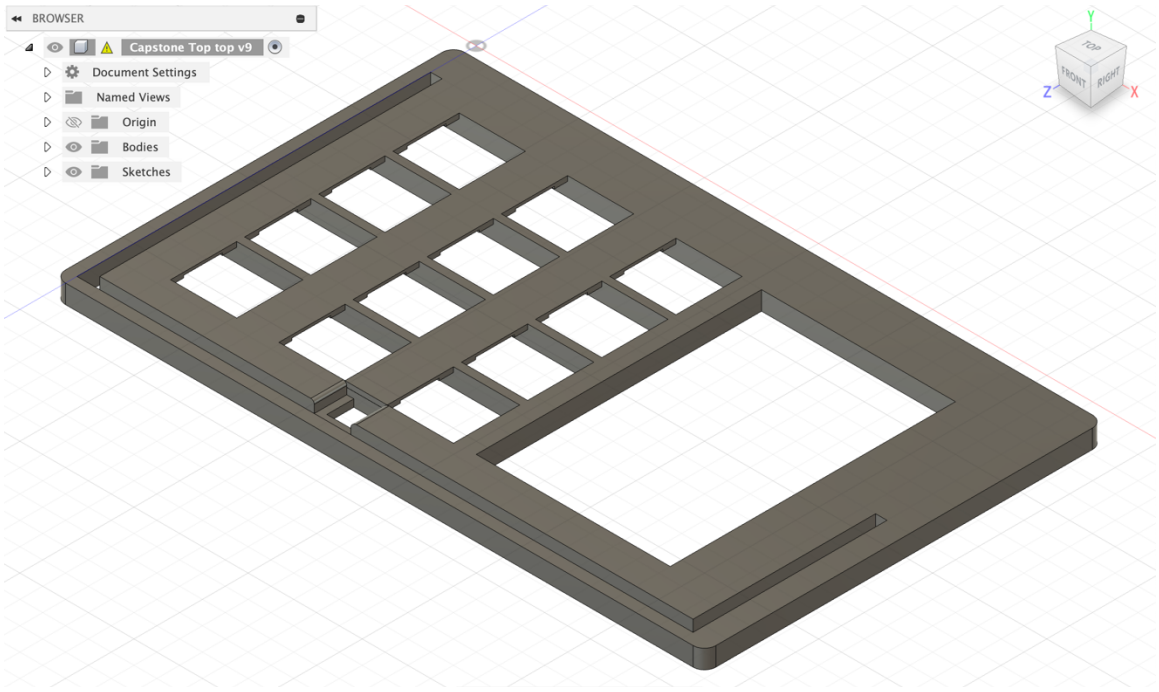
•! Basic design description

To facilitate assembly and manufacturing, we opted for a three-layer sandwich design. The bottom layer contains the main PCB and wiring while also providing structural support for the display screen. It features a cross-shaped beam that supports the screen, raising it to free the screen PCB from an uneven surface. An opening on one side allows cables to pass through the device, enabling USB connectivity and power supply. The middle layer has protrusions on both sides to secure it within the top and bottom layers, accommodating the button PCBs and offering structural support for the buttons. This layer also acts as a physical barrier between the display area and the button area. The top layer holds the buttons in place, preventing them from shifting or falling off, and provides a cover for the PCBs, enhancing the overall aesthetic look.

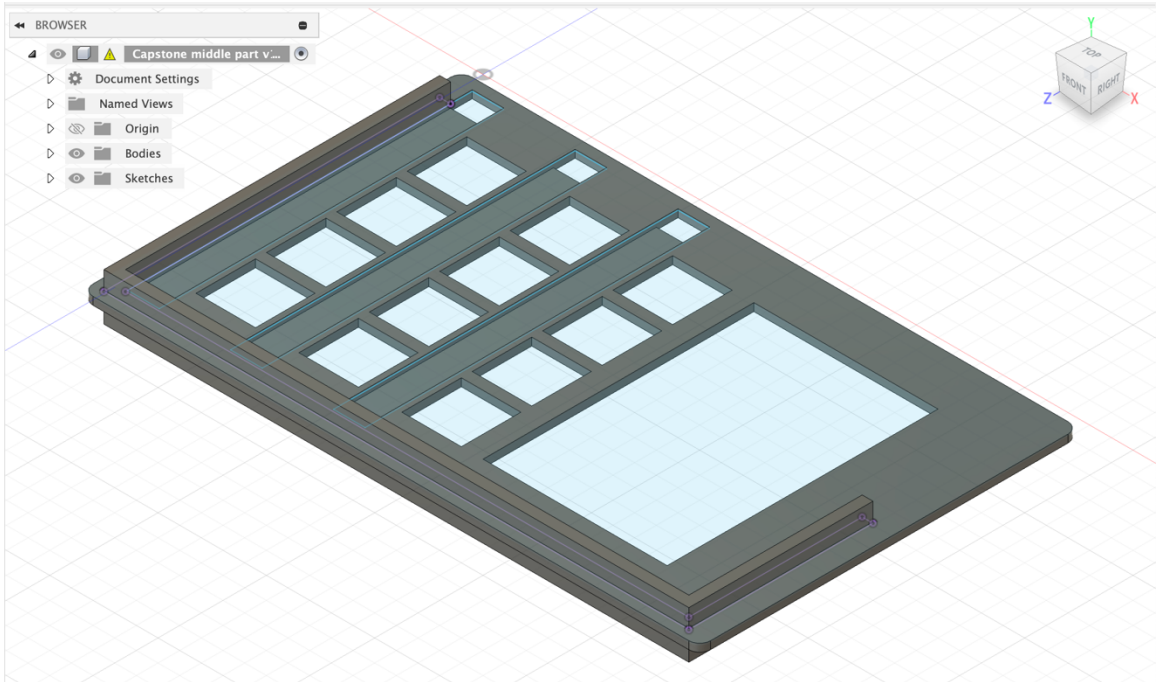
- Button design

The use of transparent buttons prevents us from placing a trigger mechanism directly underneath. As a result, we designed the buttons to have protrusions that extend from the button body to press the actual triggering button on the button PCB, which is also covered by the top layer. Since the triggering buttons are positioned right below the transparent buttons, this means we can only press the lower region to activate them. To enhance the pressing experience while maintaining a flush fit, we designed the button to be slanted, with the side featuring the protrusions angled upwards while the other side keeps the original height. This design provides a pivot point on the button, allowing it to naturally rotate and trigger the mechanism. To ensure a better fit, we added grooves on the protrusions, enabling them to sit snugly onto the triggering surface buttons.

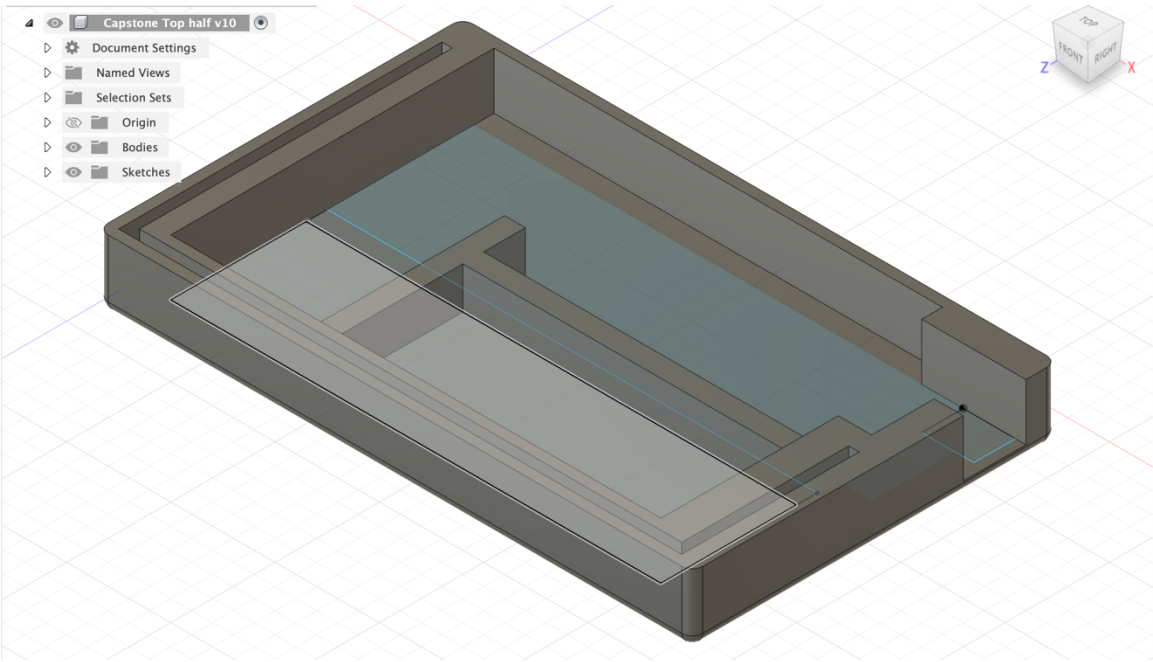
- Design dimensions



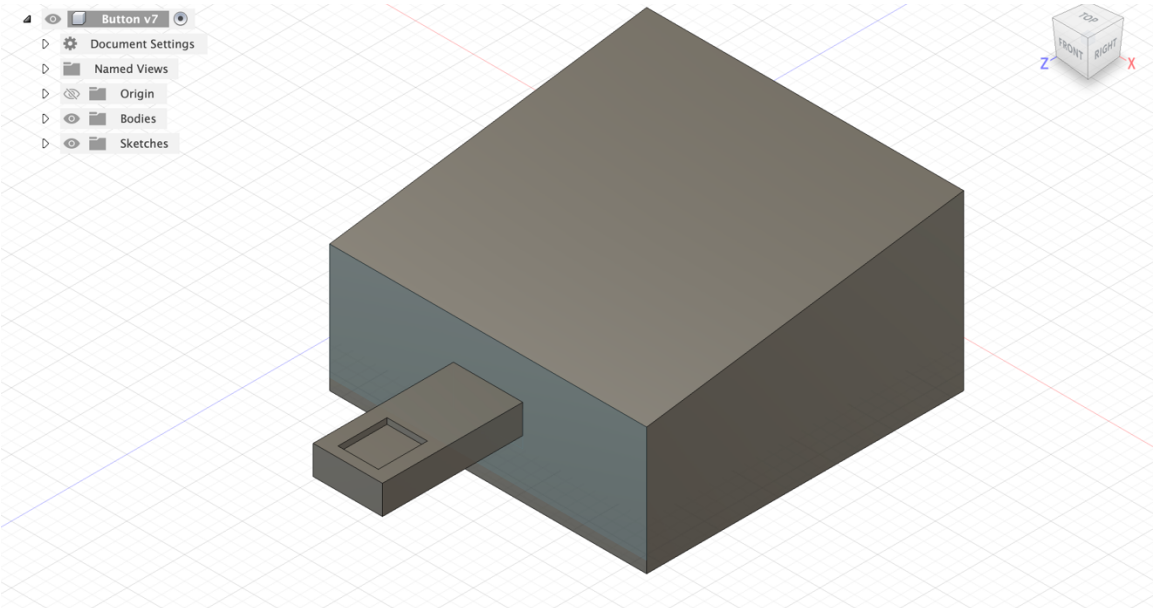
!"#$%&'(:''6"2&8A",8',/'M,K'73A&'N:IB2 20(HB2 20I?:2 2Q'



!"#$%&'(<'6"2&8A",8',/'4"EE9&'73A&'N:IB2 20(HB2 20:(?I2 2Q'

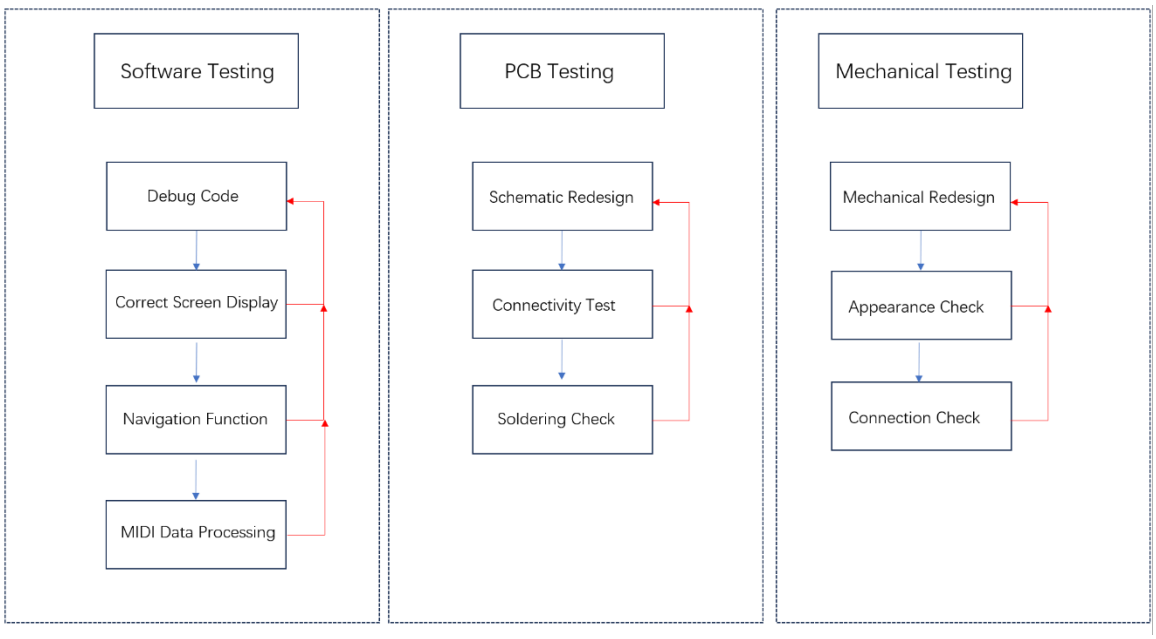


!"#$%&'(?'6"2&8A",8',/'=,++,8'=3A&'N:IB2 20(HB2 20<<P(HJ2 2Q'



!"#$%&'(B'=$++,8'6"2&8A",8'N?(PL:H2 20:HP?B2 20(B2 2Q'

:"1.*4<'(*



!"#$%&'(D'M&A+'K938'R9,;>'

The blue arrows indicate that the previous parts of the test have passed and that the next part will be tested. The red arrows indicate that if the test does not pass, it will return to the initial stage step.

Our test plan can be separated into 3 sections: Software, PCB, and Mechanical.

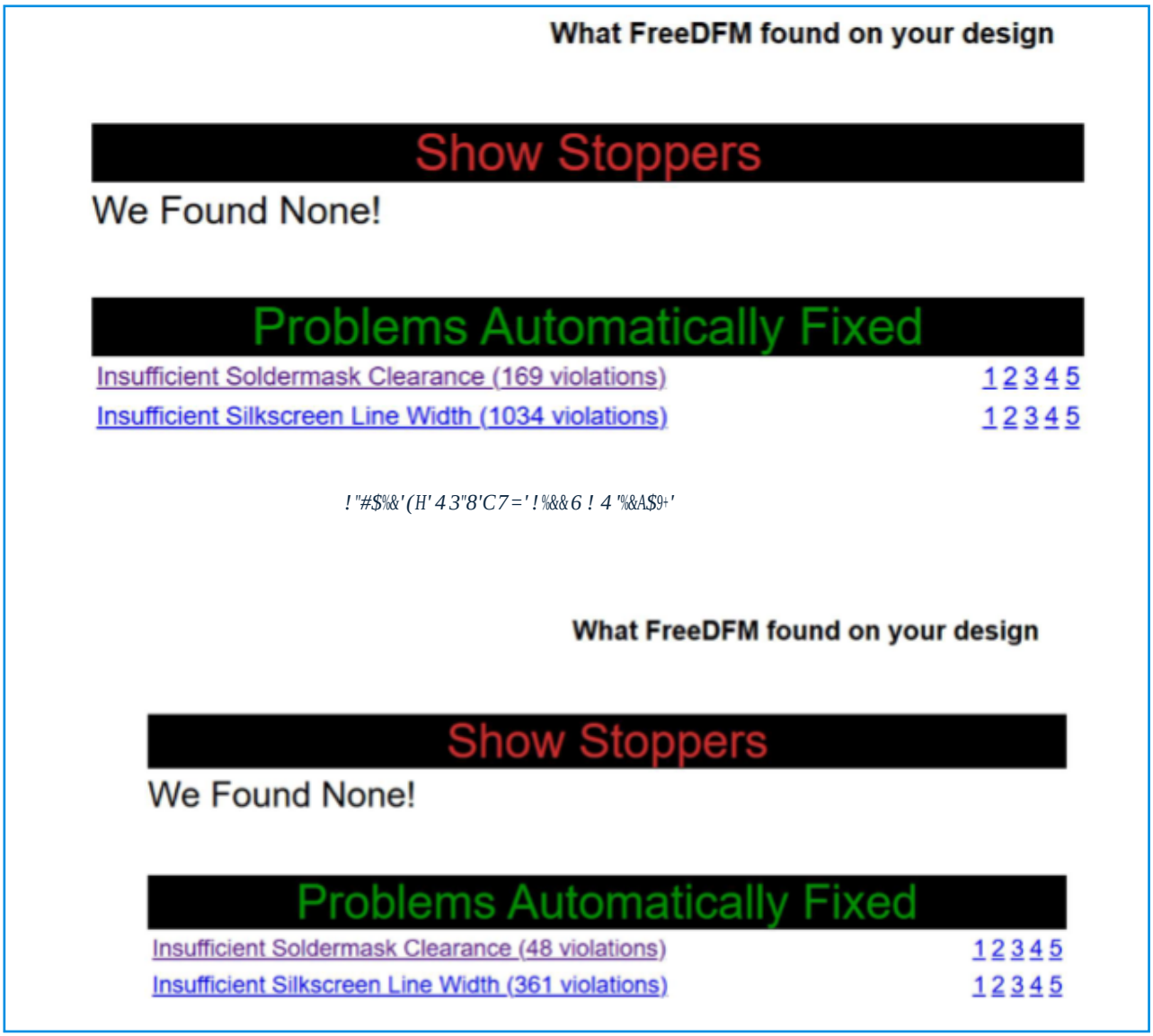
The software testing primarily focuses on software components. First, we will verify whether the screen displays correctly, as this is part of the user interface. Next, we will assess whether the navigation functions properly. This means that if a user wants to access the submenu, they simply need to click the button above that menu. After that, we will check if the MIDI data can be processed successfully, indicating that the microcontroller can effectively transfer and receive data from the computer. If any errors are found during these steps, we will review and debug the system code.

The PCB testing primarily checks for any errors in the PCB design. We begin by verifying that everything is properly connected in Kicad. This software performs DRC and connectivity checks. After the PCB is produced, we will assess whether the PICO, the microcontroller, and the button trigger are properly soldered to ensure successful signal transfer. Currently, we do not design the power supply components for the PCB since the microcontroller is powered by the computer.

The final test involves the mechanical component. First, we will verify whether the 3D printer successfully produces the desired output. For instance, 3D printers sometimes yield products with unstable top layers. Additionally, corners may be too sharp

and pose a risk to users. Next, we need to assemble the outfit, ensuring it won't easily fall apart. Furthermore, when the user presses the button, the structure must remain stable.

For the actual testing, we started by checking our PCB schematic and layout design in KiCad using the DRC check, which went smoothly. Next, we submitted our files to FreeDFM to identify any potential failures. Figure 17 and Figure 18 show the results from FreeDFM for both the main PCB and the button PCB. Afterward, we sent our order to the manufacturer JLCPCB.



For the mechanical part, we had various drafts that were finalized into one result. However, due to the limitations of the school's 3D printer, we had to send the order to a

third-party company to print our structure. The company staff discussed some design flaws with us, and we made adjustments and revisions accordingly.

It took over a week for us to receive the parts we requested, which were longer than we anticipated. We received our PCBs first, followed by the structure. As a result, we decided to solder the PCBs and check whether they were functioning properly. We soldered male-to-male jumper wires on single-button PCBs to five-way connectors and buttons. Next, we soldered both the DVI expansion board and the PICO board to the main PCB. Finally, we attached the jumper wires to the corresponding pins on the main PCB. However, a potential problem emerged. We designed the sockets on the main PCB so closely that our PICO could not be inserted perfectly. We resolved this by gently bending the pins on the PICO. We plugged the PICO board into the computer to power it on and then used a multimeter to check the power supply. However, there was no voltage difference among any pins. We discovered that only the DVI expansion board was soldered to the main PCB, while the PICO board was not. We addressed this by re-soldering the board, and then the power supply appeared normal at 3.29V. We then used the multimeter to verify whether the PCBs were connected properly. Finally, the PCB appeared to be in good condition part.

Next, we connected the DVI socket to the computer. The interface appeared correctly on the screen, and the display test was successful. Later, we checked our software components, which involved different mode functionalities. In Traditional mode, pressing the button produced the correct sound. Both chord mode and strum mode successfully passed. Despite our ongoing use of the features, issues arose, resulting in the screen showing memory allocation error reports. We then researched and discovered that our background and border consumed a lot of memory space. As a result, we chose not to apply a background color and opted for a simple rectangular border to conserve memory. Consequently, all the software was successful in passing.

The 3D-printed shells did not meet our expectations after a week of arrival. The accuracy of their width and height did not align, and even the positioning was not horizontal or vertical. Consequently, it was difficult for us to assemble the parts together. We used a grinder to reduce the width, which allowed the top and bottom parts to fit together smoothly. However, the height of the bottom parts was also insufficient for the DVI socket on the main PCB. The button parts for the upper section were too weak to hold. Fortunately, the buttons fit well, but the heights of the buttons were oversized. We attempted to match everything as closely as possible. Nevertheless, the overall appearance of the first design was not satisfactory. Thus, we developed a second draft with greater tolerance and careful consideration.

The second design was better, but width and position still had mistakes. This time, we created additional width and height for more space to modify. We then soldered

additional button PCBs with jumper wires to the main PCB and attempted to fit it into the shell. Next, we added the screen and buttons. Finally, we assembled the upper part, and everything aligned except for the buttons. This time, the space for the button was printed too large, which caused the buttons to slide. However, this wasn't a major issue for our design.

Finally, we integrated all components and verified that all functions operated correctly. We thoroughly tested each mode multiple times to assess the memory allocation issue. The process was successful, with no problems encountered.

Project Constraints

- Design and manufacturing constraints

While the design functioned well, there are still limitations. First, regarding the UI aspect, memory space poses a significant challenge. The background settings and border shape use a considerable amount of memory. Consequently, we could not further improve the UI design. We could only add a few instructions to the main menu, which may make it difficult for beginners to get started initially. Secondly, the primary PCB layout encountered the issue of the PICO board not fitting snugly in the sockets. Moreover, the soldered main PCB's height is problematic due to its three-board elevation, which may affect the design's portability. Third, the reliability of 3D printing cannot be assured, potentially leading to assembling issues. This could impede the proper fitting of the top cover. Additionally, the buttons might slide during operation, resulting in a poor user experience. Lastly, regarding the cables, the MIDI controller requires two connections from the computer: one for charging the screen and another for charging and transmitting data to the microcontroller. This configuration will occupy two USB ports on the computer. Sometimes, a computer may have only a single USB port available, requiring the use of a USB hub.

- Tools Utilized

The tools utilized are detailed in the Technical Details section under Project Description. In summary, for the PCB design, we used Kicad for designing the schematic and layouts, as Kicad provided models for the Pico board. We learned how to create footprints and design the annular ring size. Additionally, we previously used Ultiboard for layout design, which differed from Kicad, so we had to learn how to design layouts in the new software. We also used a multimeter to check the connectivity of circuits.

For the coding portions, CircuitPython was utilized as the programming language and Thonny as the IDE. Through the implementation process, we not only learned to use a new IDE, but also enhanced our self-learning skills in utilizing new libraries.

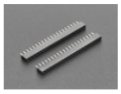
We used Autodesk Fusion 360 for CAD design. Initially, we lacked sufficient knowledge of both design and 3D printing. The first version of the physical parts had many areas for improvement, such as button design and structural strength. Therefore, after learning about the drawbacks of the first design, we created a better version.

•! Cost constraint

The overall cost, including shipping fees and taxes, is shown in Figures 19, 20, and 21. The total cost was $511.99, which slightly exceeded the $500 budget. However, this amount covers our prototypes and includes repeated shipping fees. We nearly have two orders of every part in the final prototype. In mass production, some parts will not be used, while others will require one for each controller. Furthermore, mass production will be more cost-effective. As a result, the cost for each controller in mass production will be significantly lower. However, given varying shipping fees and defective 3D-printed products, the price may fluctuate much.

Number	File name	Price
#1	Adafruit 1.pdf	44.76
#2	Adafruit 2.pdf	34.76
#3	cablecc CYFPV Dual 90 Degree Right-Up Angled HDMI Type A Male to Male HDTV FPC Flat Cable for FPV HDTV Multicopter Aerial Photography 80cm	9.47
#4	Seeed Studio Raspberry Pi Pico Flexible Microcontroller Board Based on The Raspberry Pi RP2040 Dual-core ARM Cortex M0+ Processor for GameCube, 1pc.	7.9
#5	HAMTYSAN 10.1'' Raspberry Pi Screen Touchscreen Monitor 1024 ×600 HDMI Computer Monitor FHD IPS LCD Laptop Monitor for Raspberry Pi 4/3/2/Zero/B/B+ Jetson Nano Win11/10/8/7, Driver Free	59.48
#6	Black Flexible FPV Flat Slim Thin Ribbon FPC Cable Micro USB 90 degree to standard USB A for sync and charging (80CM)	20.85
#7	Adaptermvp USB 2.0 Adapter, Dual USB Female Jack Y Splitter Charger Cable (2 Pack) for Laptop/Tablet/Smartphone Data Transmission/Charging	5.25
#8	Cablecc CYFPV FPV HDMI Male to Up Angled 90D HDMI Male HDTV FPC Flat Cable for FPV HDTV Multicopter Aerial Photography (20cm)	6.73
#9	uxcell 3x6x5mm Panel Mini/Micro/Small PCB Momentary Tactile Tact Push Button Switch DIP 50PCS	7.57
#10	JLCPCB	15.75
#11	JLC3DP first version	154.99
#12	JLC3DP second version	144.48
	Total:	511.99

!"#$%&'(J'M,+39'7,A+'/,%'C%,+,+GK&'

PRODUCTS		INFO
	(1) Adafruit AW9523 GPIO Expander and LED Driver Breakout	STEMMA QT / Qwiic PID: 4886 SID:
	(1) Thru-hole 5-way Navigation switch	PID: 504 SID:
	(1) Arcade Button with LED - 30mm Translucent Clear	PID: 3491 SID:
	(1) Raspberry Pi Pico H - Pico with Headers Soldered	PID: 5525 SID:
	(1) Adafruit DVI Sock for Pico - Works with HDMI Displays	PID: 5957 SID:
	(1) Short Socket Headers for Raspberry Pi Pico - 2 x 20 pin Female	PID: 5585 SID:
	(1) Mini HDMI to HDMI Cable - 5 feet	PID: 2775 SID:
	(1) Adafruit PiCowbell DVI Output for Pico - Works with HDMI Display	PID: 5745 SID:

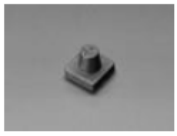
!"#$%&':L'TE3/%$"+'('6&+3"9A'

PRODUCTS

(1) Raspberry Pi Pico 2 - RP2350

(1) Raspberry Pi Pico RP2040

(2) Adafruit DVI Sock for Pico - Works with HDMI Displays



(1) Black Rubber Joystick Nubbin Cap for Navigation Joystick

(3) Thru-hole 5-way Navigation switch

!"#$%&':('TE3/%$"+':'6&+3"9A'

•! Work for a better prototype

Within PCB designs, we only spent $15.75, which included five button PCBs, five main PCBs, and shipping fees. Since one controller requires four button PCBs and one main PCB, mass production will be more cost-effective. We spent most of our money on 3D printing and printed twice. However, it was challenging to maintain the quality of the 3D-printed products, which may have been the most complex part of our spending. The screen was also expensive because we opted for a relatively high-resolution touchscreen. Unfortunately, due to the limitations of the PICO's processing power and the early-stage touchscreen technology, we did not fully utilize the screen. Consequently, there is potential for a more affordable screen replacement. We incurred extra costs for wiring because the height of the original 3D print required flexible and bendable wires. With the enhancements in structure, the second version showed a notable improvement, enabling us to pay less attention to wiring problems. Consequently, this will result in reduced wire costs.

It would also be better if we had a larger memory space, allowing us to enhance the UI design and resulting in a more appealing and intuitive interface for users. For the physical component, a better 3D printing company is anticipated to produce an accurate and precise model. Regarding the PCB aspect, reducing the number of cables used could save a significant amount of space and improve our design portability.

Costs

Part Name	Quantity	Total price
uxcell 3x6x5mm Panel Mini/Micro/Small PCB Momentary Tactile Tact Push Button Switch DIP	12	1.73
Adaptermvp USB 2.0 Adapter, Dual USB Female Jack Y Splitter Charger Cable (2 Pack) for Laptop/Tablet/Smartphone Data	1	4.99
Black Flexible FPV Flat Slim Thin Ribbon FPC Cable Micro USB 90 degree to standard USB A for sync and charging (80CM)	1	19.8
HAMTYSAN 10.1'' Raspberry Pi Screen Touchscreen Monitor 1024×600 HDMI Computer Monitor FHD IPS LCD Laptop Monitor for Raspberry Pi 4/3/2/Zero/B/B+ Jetson Nano Win11/10/8/7, Driver Free	1	56.49
cablecc CYFPV Dual 90 Degree Right-Up Angled HDMI Type A Male to Male HDTV FPC Flat Cable for FPV HDTV Multicopter Aerial Photography 80cm	1	8.99
Raspberry Pi Pico RP2040	1	4
Adafruit DVI Sock for Pico - Works with HDMI Displays	1	2.95
Black Rubber Joystick Nubbin Cap for Navigation Joystick	1	0.5
Thru-hole 5-way Navigation switch	1	1.95
JLCPCB	1	15.75
JLC3DP	1	105.06
	Total	222.21

!"#$%&':::'7,A+A'/,%'C3%+A'$A&E'"8'!"839'C%,E$;+'

Figure 22 illustrates the cost of the parts used in the final product, which amounts to $222.21. Overall, 3D printing is quite expensive, accounting for 47%. However, if companies aim to produce the controllers on a large scale, it would be more economical for us to use mold fabrication for injecting plastic or resin since the shape is fixed. Additionally, in large-scale production, it would lower costs for every listed item or allow for in-house manufacturing, which reduces production expenses. The PCB assembly may be somewhat complex for automated equipment, as the process requires concealing the wires under the shell. However, it would be easy for automated equipment to assemble most of the components, potentially reducing labor costs salaries.

Societal Impact

The stakeholders in this project include the manufacturers and retailers from whom we sourced components, the To: Be Continued team members, and our target customers. Although the project is not directly related to public health, welfare, or safety,

it will significantly impact music production and performance, ultimately contributing to the overall well-being of our audience individuals.

Our primary customers are novice learners of MIDI controllers. As indicated in the Background section, composing melodic music often necessitates an understanding of proper chord progressions. However, after graduating from school, music training becomes less accessible, creating obstacles for those without a strong foundation in music theory. Our project aims to tackle this by providing a smart chord mode and other instructional features, allowing new learners to quickly develop the skills needed to compose music with a MIDI controller and create melodies with ease.

This project also holds significant economic implications. Using our device, beginners can save on costly formal education typically needed to master a MIDI controller. Furthermore, if the device is commercialized, the manufacturing of the Smart MIDI Creator would generate job opportunities and contribute to the expansion of the music industry.

Culturally, since the symbols for music progression are universal, the MIDI controller can be readily adopted worldwide. In conclusion, our MIDI controller enhances the music production and performance experience, fostering increased music creation and positively influencing the well-being of individuals globally.

Regarding environmental impact, the three-layer exterior case and buttons are made from 9600 resin and 8001 resin respectively. While they can be heated and remolded for a new purpose, they cannot be reprinted using the same printer [21]. Since resin in landfills or water sources can contaminate soil and water, it is crucial to dispose of the Smart MIDI Controller [22].

Finally, our project carries a significant ethical responsibility to make music production accessible to individuals with various levels of musical background. By reducing the barriers to entry, we aim to empower those who might otherwise be excluded from the creative process, fostering inclusivity and encouraging broader participation in music-making. Recognizing the significant influence music has on emotional wellness and personal expression, we believe this initiative could greatly enhance lives, positively impacting both individuals and society at large.

External Standards

During the design process, we have carefully considered several important standards. For PCB manufacturing, the manufacturer must adhere to the IPC-2222 and IPC-2221 standards [23] [24]. These ensure that the manufacturer produces PCBs with appropriate tolerances, which is crucial for the proper functionality of the boards.

Given the presence of electrical wiring, we also followed the National Electrical Code (NEC), which sets the minimum safety requirements for electrical installations in the U.S. [25]. This standard guided our wiring practices, ensuring the safety and reliability of our final product.

Regarding connectivity, we have chosen the Pico DVI socket for the interface, as the small size of the I2C and SPI screens was not suitable for our needs. By using the DVI socket, we can select a screen of a more appropriate size, and our coding process will align with the DVI/HDMI interface standard [26].

Additionally, since our project involves generating digital music, we are committed to complying with the OSHA Occupational Noise Exposure Standard [27]. This ensures that the sound produced by the Smart MIDI Controller remains below the allowable audio exposure limits.

Intellectual Property Issues

We have reviewed three similar patents: **CN218471584U**, **US20120031254A1**, and **US7928311B2**, to analyze how their claims overlap with or differ from our project.

US20120031254A1 - Keyboard for musical instrument, and instrument comprising such a keyboard [28]

- •! &'6$7$'6$'(!89*/3#: Covers a device for music creation and education that uses visual and auditory feedback to teach music theory and performance.
- •! .7'6$'(!89*/3#: Include tools for interactive chord progressions and visual aids.
- •! 8437*%/#4': While this patent aligns with our project's educational goals, our Strum Mode (which simulates guitar strumming) and dynamic, beginner-friendly user interface offer functionality not covered in this patent. The focus on simplifying the learning curve for novice musicians is a significant differentiator.

CN218471584U - MIDI controller [29]

- •! &'6$7$'6$'(!89*/3#: Utility Model of a MIDI controller with a small form factor and unique light guide structure. It also has a bottom plate, as well as a surface shell forming an accommodating space where a strip-shaped through-hole is formed on the surface shell. Integration of light guide strip and electric touchpad to improve interaction.
- •! .7'6$'(!89*/3#: Specific features of the structure responsible for stabilization, where the cuboid light guide strip has a wing part, and that the light-emitting

devices arranged on the circuit board, the light-forms on the light guide support for releasing all be equal and to suppress high light mastic and the like.

- *8437*%/#4'*: This patent describes the outer design and structural features that a MIDI controller must possess. In contrast, our project does not include a light guide strip and employs a non-variable structure based on a fixed grid button layout. Our design makes it extremely simple for beginner musicians and is educationally driven, presenting the system in Chord Mode or Strum Mode with an interactive display that guides users through chords in real time. Additionally, we emphasize how to structure music alongside its visual representation, a topic not covered in this patent, as their focus is on the sound's structure and aesthetics rather than the intuitive interaction of the user interface with music education.

US7928311B2 - System and method for forming and rendering 3D MIDI messages [30]

- *&'6$7$'6$'(!8*/3#*: The current patent is the palm-enabled table that allows users to create digital musical notes with touch-sensitive surfaces. It deals with the interaction of an input surface and generating corresponding notes.
- *.7'6$'(!8*/3#:* It recites specific implementations, including touch-surface feedback loops and complex input management systems for responsive sound modification.
- *8437*%#/4'*: Our project also generates digital musical notes, but the grid-based buttons for chord and strum functionalities provide a unique tactile experience suitable for aspiring musicians. Moreover, elements such as our dynamic chord mapping and beginner-friendly interface highlight an educational focus that is not captured within this patent.

None of the patents we reviewed seem to cover the following potential innovations in our project. Chord Mode: Users select a musical key, and the device automatically assigns chords to buttons with a simple touch, eliminating the need to remember complex progressions. Strum Mode: This mode replicates the act of strumming a guitar, permitting users to compose music in a fun and accessible way. A dynamic display provides guidance and feedback that reduces learning time for new users.

These characteristics distinguish our design from previous products, showcasing its freshness and originality. The project addresses a market demand for an accessible MIDI controller that emphasizes ease of use and education. We recommend filing a patent for the unique elements of our project, including the Chord Mode, Strum Mode, and educational interface. While we are not reinventing MIDI controllers, the specific combination of user-friendly, beginner-focused features and unique modes overlaps to warrant investigation of their potential as intellectual property.

Timeline

TASK NAME	START DATE	END DATE	DURATION (WORK DAYS)	TEAM MEMBER
Proof of concept (Prototyping)				
PICO Library Research (Visual GUI/MIDI)	9/2	9/20	15	Haizhou Yu, Jinhong Zhao
Experimentation w/ actual MCU			0	Haizhou Yu, Jinhong Zhao
User Feedback Investigation	9/2	9/27	20	Haizhou Yu, Jinhong Zhao
Product Planning (What Features?)	9/2	9/20	15	
Purchasing	9/2	N/A	N/A	Haizhou Yu
Product Engineering				
Software Engineering	9/16	10/18	25	Haizhou Yu, Jinhong Zhao
Appearance Design (Researching stream design)	10/21	12/1	30	Jennifer Shern, Tong Zhou, Wanghaotian Zhang
User Interface Engineering	9/15	9/27	12	Jinhong Zhao, Tong Zhou
PCB	9/30	12/1	45	Jennifer Shern, Tong Zhou, Wanghaotian Zhang
Final Phase				
Product Integration	11/18	11/30	10	Haizhou Yu, Jennifer Shern, Jinhong Zhao
Testing/Calibration	11/18	11/30	10	Haizhou Yu, Tong Zhou, Jinhong Zhao
Deliverables				
Project Proposal	9/13	9/20	6	Jennifer Shern, Wanghaotian Zhang
Posters	9/20	9/27	6	Jennifer Shern, Wanghaotian Zhang
Initial PCB Design	9/28	10/14	11	Jennifer Shern, Tong Zhou, Wanghaotian Zhang
Final Report/Video	12/1	12/5	4	All

!"#$%&':<'U38++'703%+'/%,2'C%,K,A39'

TASK NAME	START DATE	END DATE	DURATION (WORK DAYS)	TEAM MEMBER	PERCENT COMPLETE
Proof of concept (Prototyping)					12
PICO Library Research (Visual GUI/MIDI)	9/2	9/20	15	Haizhou Yu, Jinhong Zhao	100%
Experimentation w/ actual MCU			0	Haizhou Yu, Jinhong Zhao	100%
User Feedback Investigation	9/2	9/27	20	Haizhou Yu, Jinhong Zhao	100%
Product Planning (What Features?)	9/2	9/20	15		100%
Purchasing	9/2	N/A	N/A	Haizhou Yu	100%
Product Engineering					
Software Engineering	9/16	11/8	40	Haizhou Yu, Jinhong Zhao	100%
Appearance Design (Researching stream design)	10/21	11/29	30	Jennifer Shern, Tong Zhou, Wanghaotian Zhang	100%
User Interface Engineering	9/15	11/1	12	Jinhong Zhao, Tong Zhou	100%
PCB	9/30	10/15	12	Jennifer Shern, Tong Zhou, Wanghaotian Zhang	100%
3D Printing	10/7	11/8	25	Jinhong Zhao	100%
Final Phase					
Product Integration	11/18	11/30	10	Haizhou Yu, Jennifer Shern, Jinhong Zhao	100%
Testing/Calibration	11/18	11/30	10	Haizhou Yu, Tong Zhou, Jinhong Zhao	100%
Deliverables					
Project Proposal	9/13	9/20	6	Jennifer Shern, Wanghaotian Zhang	100%
Posters	9/20	9/27	6	Jennifer Shern, Wanghaotian Zhang	100%
Initial PCB Design	9/28	10/14	11	Jennifer Shern, Tong Zhou, Wanghaotian Zhang	100%
Final Report/Video	12/1	12/5	4	All	80%

!"#$%&':?'!"83'U38++'703%+'

The timeline of the project changed over the semester, which can be seen from the Gantt Chart from Proposal (Figure 23) in comparison with the Final Gantt chart (Figure 24). Each phase is described in detail below, along with the tasks that were undertaken and how timelines adapted for emerging developments and challenges.

1. Proof of Concept (Prototyping Phase)

- **Initial Plan**: This phase was scheduled to take place from September 2!" to September 27#$, with tasks including:

- Pico Library Research (September 2!" to September 20#$)
- Experimental Work with MCU
- User Feedback Investigation (September 2!" to September 27#$)
- Product Planning (September 2!" to September 20#$)
- Purchasing

- **Changes**: The original timeline for this phase was on track. We solved unforeseen technical problem ahead of time and make sure we were adhered to what we have planned.

2. Product Engineering Phase

- **Initial Plan**: This phase involved:
  - Software Development (September 9#$ to October 18#$)
  - Appearance Design (October 21%# to December 1%#)
  - User Interface Engineering (September 15#$ to September 27#$)
  - PCB (September 30#$ to December 1%#)
- **Changes**:
  - The software development tasks were extended slightly due to iterative testing and refinement of the MIDI and chord-mapping features.
  - The appearance design task was marked complete ahead of schedule, indicating efficient collaboration. It was done by November 29#$.
  - The completion of User Interface Engineering was extended beyond the original timeline due to the need for additional time to fully integrate with the hardware.
  - We completed the PCB design ahead of time. Our first version of the PCB works as intended, so the second version was not needed. We finished the design by October 15#$. However, the shipping time is longer than we expected, and thus the actual arrival date of the PCBs was October 25#$.
  - The 3D printing task is added to the Gantt chart to manage progress for 3D printing (October 7#$ to November 8#$)

3. Final Phase

- **Initial Plan**: This phase was allocated from November 18$^{\#\$}$ to November 30$^{\#\$}$ for:
  - Product Integration
  - Testing/Calibration
- **Changes**:
  - The integration timeline was longer than expected. Due to our efficient completion of earlier phases, we had extra time for testing, calibration, and refinement of the project.
  - Some testing tasks overlapped with the Product Engineering phase as the team adopted an agile workflow to parallelize tasks.

4. Deliverables

- **Initial Plan**: Deliverables such as the project proposal, progress report, PCB design, and final report/video were scheduled at regular intervals throughout the semester.
- **Changes**:
  - Every deadline was on track by the end of the semester.

Summary of Changes

The primary shifts in the timeline were due to iterative testing and refinements, as well as re-evaluating what tasks needed to be prioritized in alignment with project milestones. Comparing the proposed Gantt chart to the ongoing Gantt chart shows that we were ahead of schedule in multiple tasks. Our timely completion had resulted in more time for fine testing and early completion.

Final Results

87*1.(+*40(.9!

The Smart MIDI Controller must be connected to a computer to generate audio. It features three modes that users can select from the user interface display using a five-way button. The up and down inputs of the five-way button navigate through the menu. The left input returns to the home page, where different modes can be selected. Press the five-way button to confirm the displayed functions menu.

- Traditional Mode: In this mode, the device operates as a standard MIDI controller, where each button corresponds to a specific note. Twelve buttons are assigned to the notes C, C#/Db, D, D#/Eb, E, F, F#/Gb, G, G#/Ab, A, A#/Bb, and B,

respectively. Pressing a button triggers the corresponding note until it is released. The user can also press multiple buttons simultaneously to generate chords.

- Chord Mode: Once this mode is selected, the user will be prompted to choose a key from the major chords C, D, E, F, G, A, and B. The corresponding chord for the selected major will appear on the display beneath the button. Pressing the buttons will generate the corresponding chords.
- Strum Mode: After selecting a major key, users can choose three guitar chords from the list. These chords will be displayed on the interface. Each button simulates a guitar string, and each row represents a chord. By swiping over the buttons, the device mimics the strumming of a guitar.

#3(.&3(4! , &.

We have successfully met nearly all the expectations outlined in the proposal. The Smart MIDI Controller can generate chords, allowing users to create music even with minimal knowledge of music theory. Although some sacrifices were made regarding the background color due to Pico's limitations, the user interface remains simple and intuitive. Navigating each mode is intuitive, with the five-way button providing clear directional options on the interface. Additionally, the Smart MIDI Controller features an aesthetically pleasing white rectangular case, with a compact and portable size of 285mm x 175mm x 48.179mm. In conclusion, we have effectively achieved all the goals set forth.

Engineering Insights

We have gained valuable insights throughout the process of building this project. In contrast to earlier projects that depended on template files, we created, designed, and organized a functional PCB entirely from the ground up. We also learned to use KiCad, a completely new PCB design tool with which we had no prior experience. Through designing the Pico, we became familiar with a new microcontroller, navigating its challenges without direct guidance or materials from our professors. We explored libraries for screen displays and MIDI data, successfully programming these features while optimizing our code to prevent overloading. In CAD design, due to the complexity of the structure and the need to accommodate various components, we learned to divide and conquer this task by breaking the entire structure into three distinct layers. This allowed us to focus on smaller sections at a time, which we could later combine into the final product. This approach is similar to how large programs are coded in modular sections. Additionally, we learned to estimate tolerances and design with these inevitable variations in mind, ensuring the final assembly would fit together as well as possible intended.

Time management was crucial to the success of our project. We used a Gantt chart to organize our schedule and track progress weekly, which helped us stay on target. Although some tasks were delayed due to shipping times for parts, the Gantt chart clearly showed us what was left to do and how much we needed to catch up on to meet our deadlines.

For resource management, we encountered unexpected challenges with 3D printing errors, which required us to reorder parts, leading to slight budget overruns. This experience taught us the importance of allocating extra budget for unforeseen issues when planning. Additionally, we learned the significance of accurately estimating the number of parts needed, as over-purchasing can result in waste resources.

Teamwork, communication, and morale were important to our success. Each team member took responsibility for their assigned tasks and completed them on time. Regular communication and meetings between sub-teams ensured that we stayed aligned in our progress. Furthermore, we supported each other in overcoming challenges, fostering a caring and considerate environment. With all of the above in mind, we designed a system that functions cohesively after integration.

We would like to offer advice to future Capstone students. First, we recommend starting the brainstorming process early. Selecting the right project is crucial for completing it on time with a manageable workload. We strongly suggest that groups begin generating ideas and discussing feasibility with professors sooner rather than later. Additionally, staying on schedule is vital, especially for a one-semester project. When delays occur, it's essential to catch up and adhere to the Gantt chart. Another key takeaway is the importance of accounting for tolerances in the design phase. For instance, when working with 3D printing, we anticipated errors caused by tolerance issues and addressed them by polishing the parts after manufacturing. Lastly, leaving enough time for testing is critical. Unexpected errors may occur during integration, so it’s wiser to complete early and allow extra time for testing instead of risking late discovery of issues minute.

Future Work

- Improvement for user interface (UI) design

  In the future, memory usage can be optimized further to integrate more interactive and advanced UI features that better guide users through navigation and enhance the aesthetic experience. Some challenges may include increasing the memory load on the Raspberry Pi PICO, which could lead to a memory allocation error that crashes the system program.

- Add more functionality/mode to the current design

Additional modes can also be added, such as MIDI Mapping mode, which allows users to customize button mappings to MIDI notes. This adds more flexibility and personalization to the product. However, adding additional modes requires more memory allocation, which increases the load on the Raspberry Pi PICO. Therefore, one must ensure that the PICO has enough memory to handle the additional demands modes.

- 3D printing/design Adjustment

  The structure of the outer design can be refined to ensure proper fit and assembly. Additionally, the design of the button and the way it is pressed can be improved to create a more comfortable experience. However, it's important to manage your time carefully and ensure you leave enough room for revisions, in case a second edition is needed.

- Integrate wireless MIDI design

  Another possible improvement for future work is to integrate Bluetooth MIDI to enable the controller to connect wirelessly to devices. This expansion enhances the product's flexibility and portability by eliminating the need for USB cables. This requires exploring battery-powered operation to avoid reliance on constant USB connections connectivity.

References

[1] “MIDI Controllers: An Actionable Guide To A Versatile Tool,” Audiomovers. Accessed: Sep. 13, 2024. [Online]. Available: https://www.audiomovers.com/midi-controllers-guide/

[2] T. Romo, “MIDI: A Standard for Music in the Ever Changings Digital Age”.

[3] “Launchpad Pro [MK3] - Refurbished,” Novation. [Online]. Available: https://us.novationmusic.com/products/launchpad-pro-mk3-refurbished. [Accessed: Nov. 17, 2024].

[4] “Dexed Synth,” App Store. Accessed: Oct. 19, 2024. [Online]. Available: https://apps.apple.com/us/app/dexed-synth/id1462551596

[5] "Thru-hole 5-way Navigation switch," Adafruit.com, Nov. 26, 2024. [Online]. Available: https://www.adafruit.com/product/504. [Accessed: Dec. 4, 2024].

[6] “uxcell 3x6x5mm Panel Mini/Micro/Small PCB Momentary Tactile Tact Push Button Switch DIP 50PCS: Amazon.com: Industrial & Scientific.” [Online]. Available: https://www.amazon.com/uxcell-3x6x5mm-Momentary-Tactile-Button/dp/B07H9ZSZD4. [Accessed: Nov. 26, 2024].

[7] Würth Elektronik, “613006143121 Data Sheet,” DigiKey. [Online]. Available: https://www.we-online.com/components/products/datasheet/613006143121.pdf. [Accessed: Nov. 29, 2024].

[8] Adafruit, “SKQUCAA010 Data Sheet,” Adafruit. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/SKQUCAA010-ALPS.pdf. [Accessed: Nov. 29, 2024].

[9] D. Jones, “PCB Design Tutorial” [Online]. Available: https://d1.amobbs.com/bbs_upload782111/files_12/ourdev_422724.pdf. [Accessed: Nov. 29, 2024].

[10] S. Karthikeyan, R. Raj, M. Cruz, et al., “A Systematic Analysis on Raspberry Pi Prototyping: Uses, Challenges, Benefits, and Drawbacks," IEEE Xplore, vol. 10, no. 16, pp. 14397-14417, April, 2023, doi: 10.1109/JIOT.2023.3262942.

[11] L. Clark, “Installing CircuitPython | Raspberry Pi Pico and LED Arcade Button MIDI Controller | Adafruit Learning System.” Accessed: Sep. 18, 2024. [Online]. Available: https://learn.adafruit.com/raspberry-pi-pico-led-arcade-button-midi-controller-fighter/installing-circuitpython

[12] “adafruit_midi — Adafruit MIDI Library 1.0 documentation.” Accessed: Sep. 18, 2024. [Online]. Available: https://docs.circuitpython.org/projects/midi/en/latest/api.html?fbclid=IwY2xjawFYHPJle

HRuA2FlbQIxMAABHX_Dp_maaX79g-bAnDkbGNXOIhBn3EBif-1eevWWhS29h2nMPxPkI4ci7w_aem_TX_TUqLkA0zgDxjBJcEjRA

[13] P. Karlsson, "*#(3 ;<=>#?=3/6/@A!B/(C>?@!*2024. [Online]. Available: https://github.com/ripxorip/stm32_usb_midi. [Accessed: Sep. 18, 2024].

[14] A. Annamaa, “Introducing Thonny, a Python IDE for learning programming,” in *5%4+$$6/'0#!4)!(C$!DE(C!F4%/!8*%%/'0!84')$%$'+$!4'!8437>(/'0!G6>+*(/4'!H$#$*%+C*, Koli Finland: ACM, Nov. 2015, pp. 117–121. doi: 10.1145/2828959.2828969.

[15] “Displayio – High Level, Display Object Compositing System — Adafruit CircuitPython 9.2.1 Documentation” Accessed: Dec. 03, 2024. [Online]. Available: https://docs.circuitpython.org/en/latest/shared-bindings/displayio/

[16] “Framebufferio – Native Framebuffer Display Driving — Adafruit CircuitPython 9.2.1 Documentation.” Accessed: Dec. 03, 2024. [Online]. Available:

https://docs.circuitpython.org/en/latest/shared-bindings/framebufferio/

[17] P. Burgess, “Adafruit GFX Graphics Library,” *16*)%>/(*. Nov. 2024. [Online]. Available: https://learn.adafruit.com/adafruit-gfx-graphics-library. [Accessed: Dec. 03, 2024].

[18] “terminalio – Displays text in a TileGrid — Adafruit CircuitPython 9.2.1 Documentation.” Accessed: Dec. 03, 2024. [Online]. Available: https://docs.circuitpython.org/en/latest/shared-bindings/terminalio/

[19] “digitalio – Basic digital pin support — Adafruit CircuitPython 9.2.1 Documentation.” Accessed: Dec. 03, 2024. [Online]. Available: https://docs.circuitpython.org/en/latest/shared-bindings/digitalio/

[20] Ladyada and K. J. Walters, *16*)%>/(!8/%+>/(5J(C4'*. [Online]. Available: https://docs.circuitpython.org/projects/midi/en/latest/ [Accessed: Nov, 2024]

[21] G. Zhu *$(!*9K*, “Recyclable and reprintable biobased photopolymers for digital light processing 3D printing,” *8C$3K!G'0K!LK*, vol. 452, p. 139401, Jan. 2023, doi: 10.1016/j.cej.2022.139401.

[22] T. Fitzgerald, “Sustainable Resin Disposal: Eco-Friendly Ways To Handle Leftover Resin - Resin Affairs.” [Online]. Available: https://resinaffairs.com/green-ways-to-dispose-of-leftover-resin/. [Accessed: Dec. 01, 2024].

[23] IPC-2221, "Generic Requirements for Designing Printed Boards and Other Forms of Component Mounting or Interconnecting Structures," IPC, 2018.

[24] IPC-2222, "Standard for Design of Rigid Printed Boards," IPC, 2002.

[25] National Fire Protection Association, "National Electrical Code (NEC)," NFPA 70, 2020.

[26] Digital Display Working Group, "DVI Specification," Version 1.0, DDWG, 1999.

[27] Occupational Safety and Health Administration, "Occupational Noise Exposure," 29 CFR 1910.95, U.S. Department of Labor, 2019.

[28] Hotrique, J. (2013, December 24). Keyboard for musical instrument, and instrument comprising such a keyboard.

[29] Wang, L., Huang, P., & Jin, Y. (2023, February 10). Keyboard for musical instrument, and instrument comprising such a keyboard.

[30] Trivi, J.-M., Jot, J.-M., Savell, T. C., & Guzewicz, M. (2011, April 19). System and method for forming and rendering 3D MIDI messages.