Electronic Audio Sequencer with Bluetooth enabled data transfer

Brandon Marcoux, Giani Francis, Miguel Chavez and Alexis San Javier

University of Central Florida, Department of Electrical Engineering and Computer Science
Orlando, Florida, 32816

***Abstract* — This paper provides a detailed overview related to the design methodologies and strategies for developing an electronic audio sequencer with Bluetooth enabled data transfer. The purpose of an audio sequencer is to allow musicians and hobbyists to develop musical patterns quickly. The goal is this project is to provide a more cost effective, portable and simple alternative to other audio sequencers in the market. The paper will first provide an overview of the system components, and then go into more detailed and technical explanations in the sections that follow.**

I. Introduction

An Audio Sequencer is a unit which possesses a specific number of segments (usually 8 or 16) and a specific number of channels (typically between 4 and 16). One can think of this unit as having segments on an x-axis and channels on a y-axis. The x-axis segments are cycled through repeatedly from left to right with respect to time and each segment can trigger a sound sample contained within the channel. Conversely, every channel can contain its own sound sample and be triggered at any cycle. This allows the user to program music, which at it's very essence, is a sequence of sounds organized relative to time. This is accomplished by placing a switch/button on each segment that controls its on and off state.

Physical audio sequencers are typically expensive and aimed at more technically inclined musicians. There are cheaper alternatives that exist purely as a software applications. However, they forfeit the excitement and feeling offered by using a physical unit. Physical units typically require users to follow a manual closely in order to take advantage of all the functionalities available on the unit. Additionally, in order to load new sounds onto the unit, it typically requires the use of a cable, or a storage medium such as an SD card. The typical setup involves a computer with a wired connection, which can be cumbersome.

All this being said, it would be ideal if a physical unit could be of low cost while maintaining a certain level of portability. The main goal of this project is to provide exactly this. The unit is low cost from a user's perspective, and it is lightweight and portable, with only a few buttons and controls aside from the sequencer array. It is also accompanied by a mobile application that allows the user to control what sounds are assigned to which sequencer channel. These channels are illuminated with LEDs when active and triggered with a physical mechanism. The unit is also versatile in its output possibilities (multiple output formats – for example speaker, line-level). Additionally, there is a substantial sound bank accessible to users, along with the ability to contribute to this sound bank with sets of audio samples. This allows the user to really take ownership of the sounds they create.

II. System components

The sequencer as a whole consists of many hardware and software components. The following section will give a brief overview of the main components of the system. These will then be discussed in further detail in the sections that follow.

A. Microcontroller

The microcontroller is the core component of the hardware. It provides functionality and processes for all incoming and outgoing information. The overall design of the sequencer is built around the capabilities of the microcontroller. The device will have a large amount of code to receive and interpret, and then push out to dependent components

B. Multiplexed Components

Both the LEDs and buttons will require multiplexing given the limited input/output pins of the microcontroller. These components are the main connection between the system and the outside world, as such, these features need to be implemented in a smooth and efficient manner. The setups are also complex, and they require detailed schematic documentation.

C. Bluetooth hardware

The Bluefruit UART Friend, Bluetooth 4.0 BLE module is required for our wireless design. The Bluetooth module will allow us to send sound files, and sound sets wirelessly. Using the BLE module, we will send serial data when pressing our upload button on the mobile application. The raspberry pi will receive the data via serial communication from the Bluetooth module as ASCII data. The code that is written for the board will check the data received and store it in the correct sound files. Once the raspberry pi has the sound file, the rest of the

processing is handles by the main PCB which will communicate with the raspberry pi via i2c communication. Also, despite data transfer speed not being a requirement of our design, we decided to optimize the speed of communication. Currently sending 8 sound files takes about 15 seconds. This is how we will interface our mobile application with the microcontroller and the physical sequencer.

D. Raspberry Pi

The purpose of the raspberry pi is to give our project processing power. The ATmega328 could not handle playing 8 sound files simultaneously, and therefore we needed a microcontroller that could. The raspberry pi is the receiver of serial data via Bluetooth communication. The raspberry pi also functions as our sound player. When the main PCB triggers sounds to be played via i2c communication, the pi will handle it.

E. Database

Both the mobile application and the website will be sending and retrieving data from the same database. The database will be built using Google's web platform known as Firebase. Firebase provides a real-time database that allows users to sync and store data among users. This real-time database is a cloud hosted database where data is stored in JSON format. This means that the database can be updated in real-time across various platforms including iOS and web applications, which makes it suitable for this project

F. Mobile Application

The mobile application is responsible for downloading the sound information from the database and sending it to the sound board via Bluetooth. In order for a user to download the sound files to the app, they will first have to create a user profile.

When a user creates a profile, their preferences, sound files, and user information will be stored into a database. When the user's account is approved by the admin, their login credentials are encrypted and saved to the database. Every time the user logs in, their credentials will be compared with those in the database, and they will be granted or denied access. If they login successfully, they will then be granted access to the rest of the mobile application. This is helpful because a user can store private information.

The user will also be able to modify their own information, such as emails and passwords. Additionally, they will be able to create projects inside the application, and the project will the sent to the database to be stored. Whenever a user thinks a project is ready to be tested/played, the user will send the project to the sound board via Bluetooth.

G. Website

A key feature of this project is a web application. The web application is a necessary part of this project because it handles the role of storing information to the database. This stored information are sound files, that will be later used to create music on the sound board by the user. Like the mobile application, the user will also have the ability to login and register, and they will be able to modify their own information, such as emails and passwords.

III. System

Block diagrams are useful when representing a high-level abstracted view of a system.

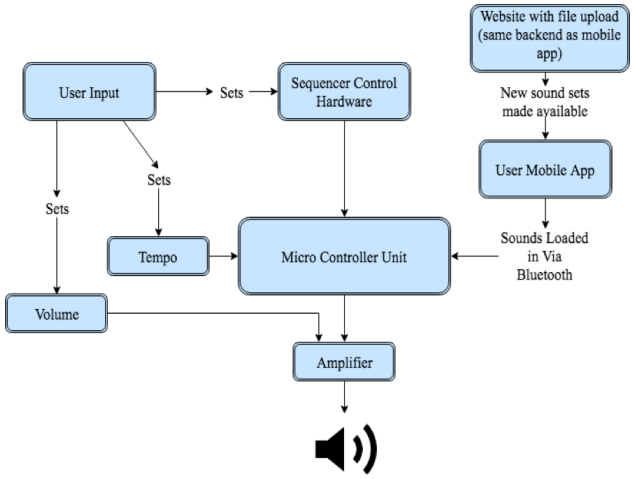


Fig. 1. System Block Diagram

. This diagram explains the interconnection of the different parts of the system. The three main parts of the system are the mobile application, the physical sequencer, and the website. The website and the mobile application will share the same database and will function as the software segment of the project. The hardware segment will be built separately and designed to communicate with the app and the website

The Sequencer Control Hardware encompasses the LED drivers and multiplexing hardware as well as the button matrix and its respective multiplexing hardware. The tempo segment is a single potentiometer which will be used to select a discrete tempo from a determined range. The user input may come in various forms to influence each part of the system. The triggering of the sound samples in the sets is controlled by the user interaction (button push) via the Sequencer Hardware Control, while the tempo or speed with which the samples are

triggered is controlled by the user turning a potentiometer (as is the volume).

This project is a small board approximately 12 by 15 inches which contains solely LED buttons or switches, potentiometers for volume and tempo, a button for start and stop, and sound outputs. This is all controlled by a microcontroller and a Raspberry Pi which will communicate via some Bluetooth to an application on a mobile device. This connection will allow the transfer of various audio files and the ability to assign them to individual channels.

IV. Hardware Details

The hardware components outlined in section II, will now be explained in technical detail.

A. Secondary LED array

The design of the secondary LED array is mentioned here before the first because it is a simpler design, the main array builds upon the functionality herein. The main array controls one color with the ability to multiplex all 128 LEDs. The secondary array controls the second color which is illuminated column by column showing the state of the sequence, or which column is being triggered, with respect to time. Whichever buttons have been selected in a certain column in a certain row, will be illuminated with the primary color to show that they are ready to be triggered, when the secondary color arrives at this column at a specific time *t*, the sound/sounds associated with the primary color illuminated row/rows will be illuminated.

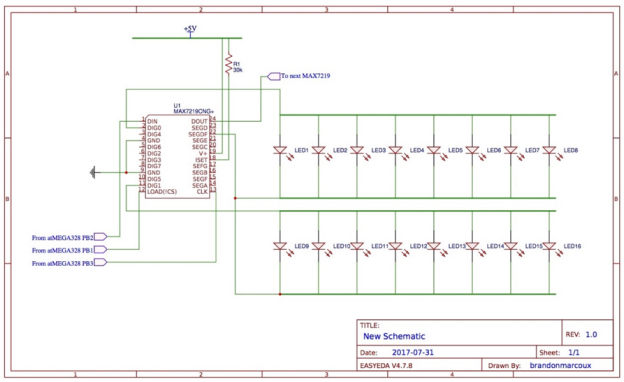


Fig. 2. Secondary LED array schematic

The secondary LED array can be seen as sweeping across the matrix. Since bi-color LEDs are being used, the primary and secondary color arrays will seem as one, however the respective anodes of the LEDs. This means that in viewing the schematic diagrams it is important to note that they will be technically "on top" of each other. The one value which is necessary to calculate in order to use the MAX7219 is the Iset value. This is a value for a resistor which is connected between an Iset pin and the positive voltage pin on the MAX7219. The value of this resistor determines the amount of current that will pass through the LEDs. This value is calculated from the Forward Voltage and Forward Current of the LEDs.

There is a table within the datasheet for the MAX7219 which Forward Voltage/Current pairs and the associated resistor value that should be used. The red-green LEDs we will be using have a Forward Current of 20mA for each color, a 2-2.2 V Forward Voltage for the red side, and 3-3.3 V Forward Voltage for the green side. This means a 28k resistor for the red, and a 25k resistor for the green. A higher resistor value is safe, so we can choose a value close to these that is found in common pairs of resistors. This provides a value higher than necessary which provides a 20% margin above the max current (forward current value). Limiting the current further can have the effect of dimming the LEDs, however, so if in testing they seem to be too dim, this is a value we may adjust (lowering the resistor to exactly the specified value of 25k) in order to try to achieve brighter lights.

C. Main LED Array

The main LED array is capable of controlling all 128 LEDs individually and in any combination. For this we need more than one MAX7219. The units support cascading, so they take up no more than three pins including the secondary array. All of the LEDs are controlled by 3 MAX7219s, one for the secondary array, and two for the main array. Setting this up is as straight forward as setting up one 8x8 matrix with one MAX7219 and then repeating the process. To then make them one the DOUT pin of the first is connected to the DIN pin of the second and the clock and load signals are connected to each from the same source.

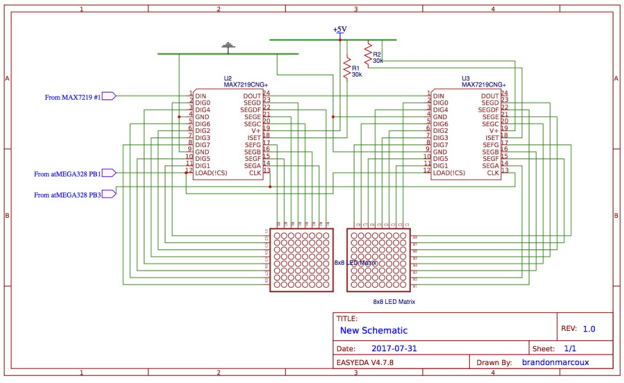


Fig. 3. Main LED array schematic

Here we use the same resistor value calculated previously for the Iset value. Since the values are very similar, we can use the same values and change them if we notice a big difference in testing. The 5V supply comes from the main voltage regulator in the atMEGA328. TheDIN of the first MAX7219 is coming from the DOUT of the secondary LED driver

C. Button Multiplexing

The 74LS148 units are used to decode a row of 8 LEDs. This means that all 16 rows are decoded with a single 74LS148 unit, the first 8 columns with another, and the second 8 columns with a third. Each of these decoders has 3 outputs for a total of 9. The two column decoders have their output combined with a multi NAND gate (LS08 unit) to form 4 output lines, instead of 6 for a total of 7 output lines. This is accomplished by detecting rows together and columns together in a daisy chained configuration. This triggers each entire row and entire column, which is enough information to determine which individual LED is signaling.

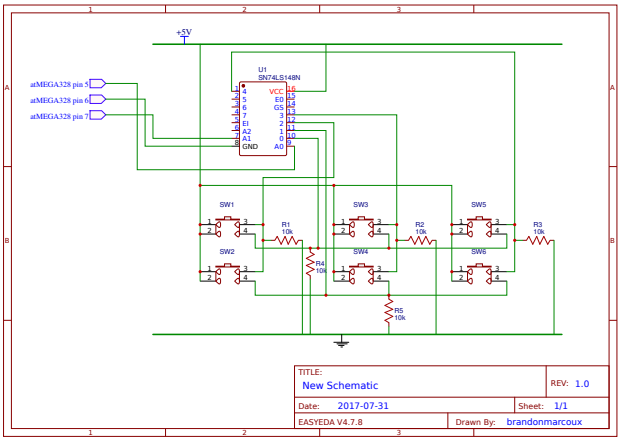


Fig. 4. Button Array Daisy Chain Configuration

D. Bluetooth hardware

Serial communication is the process of sending data bit-by-bit, sequentially, over a communication channel or computer bus. The communication channel in this case is the Bluefruit UART friend Bluetooth module. Serial communication is very similar to using a telephone. The speaker on the phone that is held to your ear would be the receiver (RX) and the microphone would be the transmitter (TX). The transmitter on your end of the phone is connected to the receiver of the caller on the other end of the phone. The receiver on your end of the phone is connected to the transmitter of the person on the other side of the phone. Serial communication allows for bidirectional communication where either person, or in this case, component/application can send or receive information.

A serial communications interface (SCI) is a device that allows the serial exchange, of one bit at a time, of data between the microcontroller and peripherals such as computers, mobile applications, websites, etc. In our case our serial communication interface is the Bluefruit UART friend Bluetooth module.

We can show the steps involved behind our design for data transfer. Below is the steps taken to control the sequencer from the mobile application.

1. Mobile app connects with the Bluetooth module peripheral.
2. Mobile Application sends sounds associated with each row in the LED matrix.
3. Bluetooth module receives serial data.
4. Microcontroller begins dumping the data into the appropriate sound files
5. The main PCB connects with the microcontroller via i2c communication
6. The main PCB tells the microcontroller how and when to play the sounds sent to it via the mobile application.

The serial port on the microcontroller and board communicates with the TTL voltage levels which are 0-5V. This type of serial communication requires three connections to function. The three connections include a transmitter (TX), receiver (RX), and a common ground connection. The Bluefruit UART friend Bluetooth module was designed for serial communication. The transmitter line on the Bluetooth module connects to the microcontroller's receiver port. Conversely, the Receiver line on the Bluetooth module connects to the transmitter port on the microcontroller.

By default, the Raspberry Pi which utilizes GPIO pins 14(TX) and 15(RX) for serial communication. In order to communicate with the Bluefruit Bluetooth module, the raspberry pi's TX and RX lines are connected to module's RX and TX pins, respectively. This connection protects against any data collision when the mobile application is uploading sound sets to the microcontroller.

V. Audio Playback

The initial idea for the system was to combine multiple audio files into one in the mobile application and send and play the combined sound on the atmega328. This would have simulated multiple sounds while using the single thread available in the single core atmega328. This became infeasible when we realized that there were some edge cases which required the sound files to change instantaneously while others persisted. To solve this problem, we implemented

sound playback using the Raspberry pi. While the atmega328 keeps track of the states of each button and their corresponding LED signals, the state is also send via I2c to the Raspberry pi.

When the atmega328 receives an interrupt, it sends a unique signal to the Raspberry pi prompting the pi that some change has occurred. The signal that follows this signal is a unique signal which is interpreted as the button state which replaces the current button state in the Raspberry pi's memory.

At each change of state the state is sent to the Raspberry pi, and at each step execution the Raspberry pi is instructed to play the sounds for the given state. This means that if a given row has three sounds selected, when the step for this row is executed, these three sounds will be played. A Java application loads and plays all of the sounds and handles the multithreading while the audio files are updated through a bash-script whenever new sounds come in from the mobile device over Bluetooth. This configuration allows a sound files of any length to play and at least 16 sounds to be playing at once. The Java application is controlled by a python application which is monitoring the I2c bus and handling the incoming signals.

In order for the sounds to be interpreted properly by the program, they are vetted at the point that they are uploaded to the database from the website. The input dialog and associated backend ensures that the sound files being uploaded are of a format which the end java sound player application can interpret.

VI. Software

The main objective of this project is to give users the ability to develop musical sequences rapidly and conveniently. This means giving them the ability to change and edit their sounds sets in real time through both a mobile and web platform. Given these requirements, it is critical then to develop a highly detailed and stable system software architecture.

A design can be thought of as the plan resulting from the process of determining how to implement all of the system requirements. Good design is about selecting, adapting, and integrating several architectural design styles to produce the desired result. In order to ensure that the software architecture meets the specified requirements, it is therefore necessary to develop the system using an architectural model. This not only helps us to understand the system, but it also provides a blueprint for analyzing any and all of the dependencies within the system.

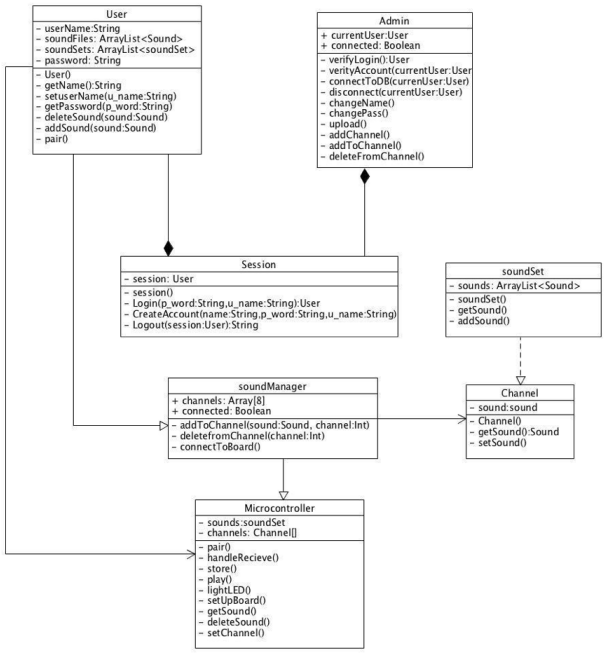


Fig. 5. Class Diagram

From a single diagram, we can see an abstracted view of the functionality and all its dependencies.

A. Database

The database is NoSQL, so unlike a SQL database there are no tables or records. Instead, when data is added, it is added to a JSON tree and becomes a new node with an associated key. When data is fetched at a location in the database, all the corresponding child nodes are also retrieved. Therefore, it is important that the data structure be kept as flat as possible. To accomplish this, we split the data into separate paths, using a process called denormalization. In a SQL database, relational schema is typically represented with tables. However, since Firebase doesn't have a query language, it must be represented through the use of a JSON Structure.

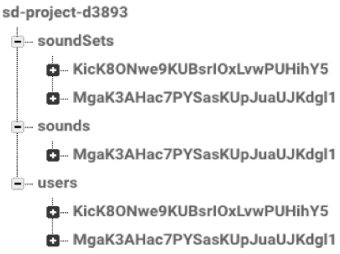


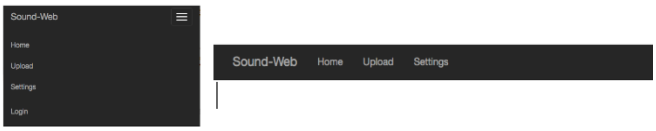
Fig. 6. Database Nodes

In general, the database stores three main nodes, Users, Sound Sets, and the individual Sounds as shown in the table. These nodes follow one similar pattern. When accessing an object from the database the user must provide its ID to get the object. The user ID is private information and even the user themselves won't even know their user ID. This makes all information private and available to the user only. This means any user that is longed on won't have access to any other user's information.

B. Web

The web application has a very user-friendly interface, and it is size adaptive. This is useful because in today's tech-world, users consume web application in a variety of different devices with varying dimensions. So no matter the device that is used to view the website, it will be able to still have all the main functionalities present. This can be achieved through the bootstrap framework. Inside this framework, we have the ability to specify how the user interface is supposed to look at different sizes. This is very noticeable on the navigation bar. On a desktop, the navigation bar will be across the top with different tabs to select from. While, if the website in smaller device such as a phone or tablet the menu will dynamically change to a stair mobile menu instead of the bar across. And this is not limited to just the navigation bar, it's actually very beneficial for all the view in the application.

Fig. 7. Web User Interface



All the pages on the web application have a footer at the bottom and a navigation bar at the top. The footer consists of more information about how the website was made. It also displays the rights associated with the application. This footer and navigation bar are found on all the pages. This gives the user access to all of our different pages. The pages are Home, Upload, Setting, Login/Register.

At home the user will find a welcome page. On this page, we have information about our product and who we are. Here we have information on how to use the sound board as well. The next page is Upload. The user will only have access to this page if they are logged in. If the user is not logged in they will be routed to the login page. Once the user is logged in they will be able to access the upload page. From here they'll be able to edit these sound sets, or edit their profile settings. If they choose to edit a sound set, they will be taken to another page, that will display the current sounds in that set and they will be given the option to delete sounds from that set or upload new ones. The Setting page works similar to the Upload page. It has restricted access unless the user is logged in. Once logged in the user will have the ability to change different settings of their profile.

The final pages are the Login/Register page. The login page consists of two text boxes where the user is able to input their information to get verified. If the user does not already have an account the user can choose the link below the login to be able to create an account on the register page. This will be the only way to access the register page, as we want the user to try to sign on first. This gives us better user account control. In the registration page the user can enter all the information to create an account. This will be sent to the database so that after the account is created the user will be able to sign on.

B. Mobile

The mobile application also has a very user-friendly interface. The interface is mostly done without the use of storyboard found in Xcode. This decision was made because user experience is very important to us. To deliver the user experience that we want we need to make our user interface customizable to the need of each page.

The user will be greeted to a login page. This page will appear if the user is either new to the application or is not login in. If the user is new they will be able to select register that will bring them to another page to register. The registration will provide an area where the user can input information about themselves. They will also be able to include a picture for the user profile image.

After the user is logged in they will have access to the app. Here they will be brought to the Project page. In this page, there will be a list of projects the user has made. All this information is getting pulled from the database in real time. From here the user will have a few options: they can either create a new project, go into an existing project, or go to the menu.

To create a new, project the user will select a '+' sign that is found on the top right of the navigation bar. This appends another item to the top of the list. To customize the project the user then selects the newly created project found on the top of the list. To go into an existing project the process is the same as long as the user selects the one they want to modify. From here the user is sent to a new page. In this page, the user can do modifications to the project. Including changing different sounds for each channel, editing the name of the project, and any other modification we need to implement for the projects. The information will not be saved automatically as we want to be sure that all project changes are verify by the user before saved. The action to save will actually consist of pressing the

save button and confirming to save in a pop up. After the user saves the information for the project the application returns to the project list. From here the user will see the menu button. If the button is selected the side menu would appear giving the option to switch to the other parts of the project; Upload and Settings

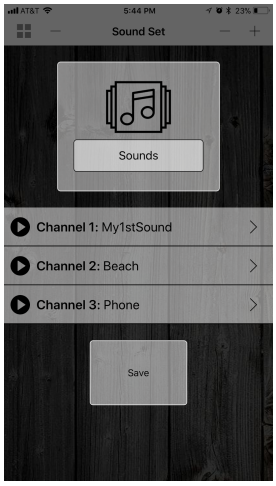


Fig. 8. Home Screen

If the user selects settings than the setting menu list will appear on the screen. Similar to the project list the settings list will be able to be selected. If the user selects one than the user will be taken to the modification list. Here the user will be able to modify the selected setting, i.e. name, username, phone number, etc. Once the user is done modifying the settings they will return to the setting list screen. Here the user will still have access to the menu on the top left corner as well as a button to press to log out on the right corner. The menu can take the user to any other page including upload. If upload is select the user will encounter an upload page. Here the user will be able to select from a dropdown with all the names of the project they have stored on the database. After selecting one the user can then press send and the project will be sent to the sound board.

C. Firmware

There are many levels of firmware used to implement the design. The lowest level firmware is the code on the atmega328 chip. This firmware is running in a 'check for interrupts' fashion. The main loop of the chip is constantly checking the input registers which include the chips reading button presses and the volume and tempo control. The SN74LS148 decoder sends an output detected signal whenever a button is pressed. The volume and tempo controls read analog signals from potentiometers. The interrupt check looks to see if there is an output detected signal from the decoders or if the tempo or volume values have changed significantly (small fluctuations are expected, so we need to check if there has been significant enough change to indicate a change in potentiometer position). If any of these interrupts are detected, associated actions are taken. This method is able to detect changes even while processing other code due to the fact that the potentiometer changes are persistent, and the button state changes are of a longer duration than the number of clock cycles used in any intermittent operations.

At the press of a button the overall button state is updated and the led array is updated to reflect this. The state of the LEDs is controlled by a register inside the MAX7219s, therefore the atmega328 is not required to constantly update the LEDs. Aside from the button interrupts, the current step is kept track and incremented based on the set value of the tempo potentiometer. The press of a button and the increment of the current step are the two values the Arduino sends to the Raspberry pi in order to play the corresponding sounds.

The Raspberry pi has firmware in place to interpret the state change signals from the atmega328, play sound files, and read and store new sound files received from the Bluetooth chip (from the mobile application). The Raspberry pi also has the ability to write a signal back to the atmega328 when new sounds are being loaded so that the atmega328 may notify the user (via the LEDs) that songs are currently being loaded

VII. PCB Design

The PCBs for our project were manufactured by our senior design team using the Toner method. This method involves printing our PCB layout onto paper and etching the design onto copper boards using heat and an acid mixture. The steps for this process are detailed below.

1. Laser print the PCB sketch onto a glossy paper
2. Cut out the PCB and fit to the dimensions of the copper boards
3. Sand off the oxidized layer of the copper board and clean off any extra dust with acetone (nail polish remover)
4. Prepare a bowl of cold water, acid mixture (2 parts muriatic acid, 1 part hydrogen peroxide), a hot clothing iron, an even surface to iron on, a fine point sharpie, a dry towel, and another bowl of water.
5. Melt the toner onto the copper board by placing the hot iron onto the non-glossy side of the paper. Be sure that the glossy side of the paper in face down on the copper board. Equally distribute heat and pressure onto the paper on board for about 10-20 minutes.
6. When the toner is fully transferred to the board, quickly dump the board and paper into a bowl of ice cold water. This step ensures that the paper will easily slide off of the board.

7. Once all of the paper is removed from the board, examine your sketch. If there are any broken or faded lines, fill them in will a permanent marker.
8. Now the board is ready to be dumped into the acid mixture. The time that the acid takes to remove the copper from the board depends on the strength of the acid. Be sure to mix and agitate every 5 minutes are so.
9. When the board has no visible copper left, place the board into water to neutralize the effects of the acid.
10. Finally, lightly scrub of the remaining toner with steel wool, and now the board is ready to be tested.

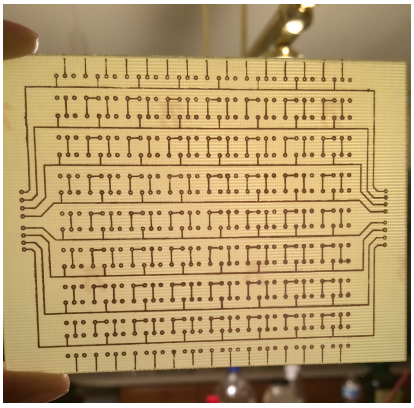


Fig. 9. PCB for Button Array

Our main PCB housing the atmega328, all of the chips to control the LEDs, and all of the chips to read in the button states, are implemented as a two-layer PCB. There were complications in ordering our PCB so we went about taking the 2-layer design we send out for manufacturing and printing each layer on a separate single layer PCB. We then attached these back to back and manually routed the traces.

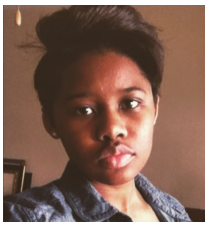
CONCLUSION

This project has given our senior design team many challenges and we've had to adjust and make tough design choices as a result. Through all of the challenges and hurdles, we have not only managed to create a product that we envisioned, but we've also learned valuable lessons on how to manage a team.

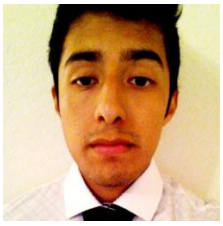
THE ENGINEERS

Brandon Marcoux is a 23-year-old computer engineer student. Brandon hopes to secure a Software Engineering job out of school and start his own company someday. He also hopes to never stop making new things at home with the skills and experience gained from school and projects.

Miguel Chavez is a 23 year -old Computer Engineering student. Miguel currently works as intern at the Walt Disney World Company. He develops sections of a .NET web application that ensures accurate time and pay recordings. As a hobby, Miguel develops iOS applications and he hopes to publish an app to the Apple App Store by the end of next year.



Giani Francis is a 23-year-old Computer Engineering student and entrepreneur. Giani has worked in the modeling and simulation field as a Software Engineer for over a year and hopes to continue after graduation. She plans create a company that engineers business solutions in the near future.



Alexis San Javier is a 23-year-old Computer Engineering student. Alexis is passionate about learning new programs and working with cutting-edge technologies. Alexis plans to get his masters while working full-time as a software engineer after graduation.

REFERENCES

[1] ARDUINO Access the Online IDE (2017) Retrieved 6/30/2017 from https://www.arduino.cc/en/Main/Software

[2] Firebase Firebase Guides (2017) Retrieved 7/5/2017 from https://firebase.google.com/docs/guides/

[3] "How it works | Bluetooth Technology Website." Bluetooth. (n.d.) 7/5/2017 from <https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works/le-p2p/>