Self-Playing Guitar

Pedro Contipelli
Department of Computer Science
University of Central Florida
Orlando, USA

Ethan Partidas
Department of ECE
University of Central Florida
Orlando, USA

Blake Cannoe
Department of ECE
University of Central Florida
Orlando, USA

Jon Catala
Department of ECE
University of Central Florida
Orlando, USA

Kyle Walker
Department of ECE
University of Central Florida
Orlando, USA

***Abstract*—This paper details the planning, prototyping, and final construction of Group 30's Senior Design Project titled "Self-Playing Guitar" for UCF term Spring 2023. The goal of the project is to take an acoustic guitar, outfit it with appropriate hardware and program said hardware so as to allow the performance of music with the use of input MIDI files. Before designing the project, a background investigation on existing technologies was performed in order to narrow the scope of the design to something that could be clearly written to and visualized. Once the initial component selection was determined, prototyping began to develop a minimum viable product. Finally, once the proof of concept was complete, proving that the product could be designed, construction of the final design began. The final project is to be completed by the end of Spring 2023, presented between the days of April 17th and April 19th.**

Index Terms — Microcontrollers, Microprocessors, Music, Servomechanisms, Synchronous Signals

I. Introduction

This document outlines the ideas, motivations, design considerations, and implementation details of ECE Group 30 CS Group 22's Senior Design project. The end goal was ultimately to build a fully functional self-playing guitar using a combination of both hardware and software implementation principles. Due to the nature of the project being that it requires both mechanical and computer science principles, the project - and by extension, this group - is interdisciplinary. Our main objective is to be able to take any valid MIDI file and play those notes by fretting and strumming all 6 strings independently on an acoustic guitar (without self-interference). Specifications include being able to play all 29 valid notes between E2 and G#4, giving us a range of about 2.33 octaves.

We will accomplish this by using a custom algorithm written in Python to take MIDI data and convert it in realtime to playable notes on the guitar the exact start timing and duration of each note. This code will then run on a microprocessor that is connected to both the strumming motor assembly and fret motor assembly. The strumming assembly will consist of 6 servo motors positioned to pluck at each string. And the fret motor assembly will consist of servo motors with linear actuators positioned at fret locations 1-4 on each string. In total, giving us 30 unique playable combinations = 6 strings * (4 fret positions + 1 open string) and totaling 29 unique notes (B3 is repeated), as demonstrated in the below calculation.

$$6 * (4 + 1) - 1 = 29 \qquad (1)$$

As this is a student-led and self-sponsored project, our budget was mostly limited to what we as students can afford: We initially estimated that it would cost around \$200 across the 5 of us. There are little to no consumer products available at the moment which do this - However, there are a few hobbyists who have built similar machines as personal projects and posted videos of it on YouTube. Essentially, all projects of this nature are simply built for fun, entertainment, and as a personal challenge rather than for mass consumption. Our initial rough timeline of the project as of the end of Senior Design 1 is as follows: The final design document report would be completed by the end of Senior Design 1 on December 5th 2022, our minimum viable product ready by halfway through Senior Design 2 on February 24th 2023, and we aimed to have the final product ready and presentable by one month before the end of Senior Design 2 on March 24th 2023. The project additionally is to be presented on March 31st 2023 as part of STEM day.

II. Project Description

The below section serves as the detailing of our project. It provides an overview of the project and personal motivations before going into detail about its goals and the objectives we have laid out in order to achieve them. In order, the section provides an overview of the project, which then leads into the specific goals that we wish to achieve to realize our design. Following that are the specific objectives we have laid out in pursuit of those goals before going into the specific requirement specifications that we will adhere to throughout our time in Senior Design in order to ensure that the project works as intended and is fully reproducible and testable. The section ends off with supplemental material in order to better elucidate the broad strokes in what the project will ultimately look like.

A. Overview

The goal of this project is to create an autonomous self-playing guitar that is able to produce its own music. The system would be able to take in MIDI files and play the notes on the guitar using separate mechanisms for strumming and pressing select strings against frets. It would be lightweight and maintain the general form factor of the guitar (i.e, fits closely to the body). The design should ideally be portable, and thus it would be powered by portable batteries. It should be responsive enough to accurately replicate the provided MIDI file compositions, comparable to - if not exceeding - the abilities of the average learnt guitar player. Not only should this design be lightweight and portable, an issue with similar concepts is the price and size. They are typically not an attachment for a guitar and are more commonly an entire unit within the guitar. They are also extremely expensive with some models going for up to $1,100. Our goal for this project is to bring this idea to reality for significantly cheaper.

B. Design Goals

Our ultimate goal for this project is to modify a guitar with electronics to be able to play itself. In pursuit of this we want our project to be:

- Able to reliably play digital audio data. This is the ultimate factor that will determine the functionality of our product. It should be able to take data in, interpret that data as musical notes, and then play those notes using the guitar that the design is built on.
- Portable. We want our design to fit the general form factor of a guitar, not departing from how a guitar looks; we don't want to create a device for the machine floor, but rather still want it to resemble a guitar once all is said and done.
- Affordable. As we are a collection of undergraduate university students, we want to minimize the cost investment into the product - especially for prototyping and moving from the minimum viable product to final presentation.
- Lightweight. We do not want the design to weigh too much. We still want the product to be held like a guitar, and too much weight would take away from that.
- Reliable. Ideally, we would want our design to require as little maintenance as possible in order to present to the Senior Design board. A high enough fault rate would consume too much time to fix which would be better spent prototyping and refining the final product.

C. Objectives

In order to realize the above goals, we laid out the following objectives to guide our design. The project must:

- Be built on an acoustic guitar
- Be battery powered
- Take MIDI file input over USB
- Process MIDI files using a Microcontroller
- Play a wide range of notes
- Utilize servos for mechanical action (strumming and fretting)
- Be able to strum all 6 strings either at once or independently
- Be able to press frets to play individual notes
- Use 3D Printed assembly parts

III. Technology Investigation

The following section follows the thought process behind our selection for parts. It describes the types of technologies we've looked into and the parts that seemed most favorable for our project as a result. This is largely dictated by our goals (i.e, low weight, cost), further criteria are elaborated upon case-by-case.

A. Power Source

1) TalentCell Rechargeable 12V 6000mAh: Battery Packs refer to batteries that are often used to charge mobile devices. Specifically, the battery pack mentioned here features a 12 volt output with 6000mAh, which should yield 42 minutes worth of use at max current draw. Realistically this current draw will be much lower (say, 25%), so we can estimate it to be around 3 hours worth of operation. The issue arises with the observation that this, and all other mobile device battery packs that we could find, only output around 2A - 3A of current. Placing them in parallel would alleviate this but buying 3 of this brand would start to weigh heavily on our budget. This would not work for our project with the current demands of our motors and the scope of our budget, so we decided to pass over this option.

2) Amazon Basics 9 Volt Everyday Alkaline Batteries: 9 Volt batteries see use for applications that require high voltage and power, which accurately describes the power demands of our project. However, the cost-to-capacity ratio of these batteries are notable; alkaline 9 volts typically have a capacity of 550mAh, and assuming our average current draw to be approx. 25% of our max we have a demand of 2.15A. This yields 15 minutes of powering our device, which is extremely low - not to mention the excessive current could damage the batteries. Both of these problems could be solved by placing the batteries in parallel - although the max current characteristics of this battery were not able to be found, 8 9V batteries in parallel should be enough to maintain our average current demand for a reasonable amount of time.

3) Lead Acid - 12V 8A ExpertPower: The 12V 8A Emergency Light Battery from Mighty Max Batteries boasts an impressive 12V output voltage and 8A output current. This falls within our reasonably expected tolerances for current - while it would be nice to break 8.6A and be 100% certain, we can safely assume that the current will be well below this value. Of note is the 8Ah capacity, which falls within our expectations for power supply longevity in a single convenient package. This would yield us a little less than an hour's worth of operation at max current draw, and if we were to approximate the average current at 25% of the max as we have done for the 9 Volts this would yield around 4 hours of operation for the device. The price is also affordable, being $29.05 when purchased from Amazon. - The profile is small, being a rectangular design that is 3.94x5.94in with a depth of 2.56in. Ultimately, this is the most promising power supply that we will be considering.

B. Power Supply - Regulation

1) LP2985: The LP2985 is a linear voltage regulator made by TI that is characterized explicitly by its low-dropout voltage. It comes in 8 fixed output voltages - the ones we will be interested in are the 3.3V, 5V and 10V options. The low dropout characteristics (typically 0.27-0.28V at its max current of 150mA) means that we would be able to comfortably limit the output of our power supply to 10.5V minimum, with the extra .2V as additional tolerance. In addition, the pricing is remarkably affordable - From TI, the LP2985 is available by the reel in quantities of 1 to 99 for $0.359. A discrepancy exists in our power output, however - The device is recommended to be run such that it outputs a 150mA continuous load current, and cursory research into the most popular servo on the market (SG90) indicates an approximate running input current of 270mA. Thus, this device would fail to power even one motor before failure. This could be circumvented if we use multiple regulators for each component or implement current amplifier circuits - both of these implementations would be impractical due to space, cost and time restraints, so the LP2985 was not considered.

2) TPS62992-Q1: The TPS62992-Q1 is a newer model of Switching Regulator from TI. It is described by their datasheet as a "Highly efficient, small, and flexible step-down DC-DC converter that is easy to use." Ease of use and implementation will be important for our project, as the limited time and cost scope means that utilizing components which take less time to accommodate is preferable. Step-down (or, "buck") converters will likely be preferable for our motor assembly, as the 8.1A current draw is a considerable hurdle that can be solved by converting excess voltage to current. The device features above-average safety characteristics, with overcurrent and over-temperature protection. This further cemented the part as a good candidate for our microcontroller, as any failures that could damage the microcontroller would be catastrophic for our project and lead to significant time and cost penalties. However, we run into an issue when we analyze the input characteristics. The input voltage range of the device is typically recommended to be between 3-10V. This is an issue since our supply voltage is most likely going to be 12V. It is not guaranteed that this will cause an issue with our parts, as the device is still able to be ran at 12V - however, it would inject some uncertainty as to the maintainability of the part itself. Ultimately, we decided against the use of this part due to this.

3) LM2576: The LM2576 Regulator comes in multiple configurations, but the one we considered is the adjustable version which utilizes a feedback voltage divider. This allows us the flexibility to introduce new parts to the project in the pursuit of stretch goals. The biggest motivating factor for the selection of this regulator is the fact that we've used it before; the LM2576 is the exact model of regulator used in Electronics 1 and 2 labs. This gives us a wealth of material to look back on in the event of circuit failure. The regulators can supply up to 3 amps, yet are cheap enough to allow for the utilization of multiple ICs. The LM2576 also comes in multiple packages, one of which can be utilized for breadboards. Thus, this regulator looked the most promising for our project.

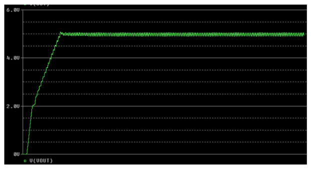


Fig. 1. Output from the regulator.

C. Micro Controller

1) ESP32 Development Board: The ESP 32 Dev Board is a through hole board in that helps us utilize some of the functions from the ESP 32 micro controller, our goal for this board is to use the i2c pins on it paired with the servo drivers. We went with this specific micro controller initially due to the high number of output pins as well as the controllers ability to process micro-python, a version of python that is used on micro controllers. Now we will be just using a few of the pins as our circuit design has become simpler. Our ESP 32 will be able to store midi files up to 4 MB. Another goal we have is to have a user interface communicate with the ESP 32 via WiFi in order to have some basic operations performed on the computer. We will be using the ESP32S 38-pin version of the ESP32.

2) ESP32 Benchmark: From earlier prototyping, we saw the need to benchmark the performance of the ESP32 processor operating under Micropython code versus Arduino C code. We wrote some simple code that does 1 million multiplications and additions and measures the time taken per operation. For the Micropython code, it took 19,373 nanoseconds per operation, which is roughly 50,000 operations per second. For the Arduino C code, it took 8.4 nanoseconds per operation, which is roughly 120 million operations per second. The Arduino C speed more closely matches the expected speed of the processor, and is 2,300 times faster than Micropython. There was some difficulty in getting the Arduino C result, since C compilers like to make optimizations to simple tests that pretty much make them instant.

D. Motors

1) Solenoid actuator: Solenoids essentially act as a push/pull arm, in this manner this is potentially the best choice for pressing down on frets. Solenoids would be more efficient than the servos due to the linear path they take, there is less room for error or chance it hits a neighboring fret. Some challenges that come with the solenoid is the cost and availability of them, they can vary from 5-20 dollars per solenoid which gets pricey with 29 needed in order to play frets. Additionally, solenoids typically have just 2 terminals, meaning we will need to use transistors or motor drivers like the L293 to isolate the control signals from the power.

2) Servos: Servos are the easiest motors to work with. They have separate connections for power and control, so it's trivial to isolate the two. The power is simply 5V - likely, this will make up the bulk of our power demand. The control signal is a PWM (pulse-width modulation) signal with a period of 20ms and an on-time of roughly 0.6 to 2.3 ms. The duty cycle determines the angle that the servo will attempt to move towards. Servos are also very cheap, about 2 dollars each.

Servo	Solenoid
less than 2 dollars	more than 9 dollars
requires 4-6V	requires 12V
Has to be adapted for linear actuation	Already a linear actuator
Actuation is slower than a solenoid	fast actuation

E. Servo Drivers

1) 74HC595 Shift Register: For our first attempt at building the Servo driver circuit we were using the 74HC595N shift register to control the signals for the servos, we went with this originally because it was a popular choice for Serial In Parallel Out communication which had 8 output pins and connected to the micro controller with 3 wires with the possibility of wiring them together in a daisy chain configuration since we would need four of them to fit all thirty of our servos onto them. After some success with testing the shift registers they would fail or the signal it received would be inaccurate we decided to switch to a more reliable method of servo control.

2) PCA9685 Servo Shield: This board uses an I2C connection to the micro controller to control 16 servos per board the communication with this board is much easier to manage compared to the shift registers and our research shown that similar devices to our self playing guitar also make use of servo shields. This board can get the information for the turning angle of the servo from the micro controller and the board will generate the PWM signal automatically. We would need two of them for our project and they are more expensive than the shift registers are.

IV. Relevant Constraints & Standards

The following section outlines the constraints and standards that we will have to work with and around. They consist not only of strict requirements to follow from Senior Design but also industry practices that will have to be designed to. Additionally, some standards are simply best practices that can be used to guide our design and ensure that everything works as intended. The specifics of each standard will not be elaborated upon in depth, but rather a general overview of the standard as a whole and how it will be factored into our project will be detailed in the sections below.

A. PCB Standards

The standards that govern the entirety of general PCB design are outlined within the IPC-2220 Family. The IPC (or, Institute of Printed Circuits) Is a Standards Development Organization (SDO) located in Bannockburn, Illinois that seeks to standardize the design and production of electronic equipment and assemblies. It is recognized worldwide, and thus whatever PCBs we order are likely to follow these standards closely - while our PCB will likely be routed by a third party company, the onus is on us to design our boards in such a way as to conform to these standards. The IPC-2221 standard specifically outlines the generic standards for a PCB design, including specific parameters for component mounting and interconnections. For our project, this standard is also supplemented by the IPC-2222 standard which specifically refers to Rigid PCBs (as opposed to Flex, MCM-L or HDI boards). Clearance is a major consideration for PCB design. Characteristics of schematic design such as parasitic capacitance and inductance mean that we will have to take into the account the gap between individual parts and copper runs - these considerations are well documented, and as a result have been included in the IPC-2221 standard. The different clearances within a PCB are categorized largely by what part is under consideration, with each component having different tolerances for trace clearance.

B. Soldering Standards

The inclusion of PCBs into our project naturally leads us to the need for Soldering. The term "solder" refers to both the action of creating permanent electrical junctions by use of melting metal onto connections as well as the metal itself. For hobbyist applications it is most utilized for connecting PCBs to other components across a project - in order for our components to be powered and communicate with one another they need to be connected electrically, and by far soldering is the most efficient way to accomplish this as individuals. The standards that govern soldering components are largely defined under the IPC J-STD-001 Standard, and define what types of solder can be used for industrial applications as well as specify what an exemplary solder connection should look like. These standards are designed in order to ensure that connections made throughout a project last as long as possible with as little failure as possible, maximizing reliability of electrical components. Under the J-STD-001 standard, a number of general rules of thumb are outlined. These are more or less for industry practices, but apply to our project as well as good industry practices yield high quality results no matter who complies.

C. WiFi Standards

The Self-Playing Guitar uses WiFi to connect to a server for transferring files, selecting songs and initiating playing the guitar. WiFi is covered by IEEE 802.11, The ESP32 model that we are using is not compatible with IEEE 802.11ax or any other WiFi standard that is 5GHz it does support 802.11b/g/n standards of WiFi that are 2.4GHz this has been something that has needed to be taken into consideration when testing and using the Self-Playing Guitar.

D. *Financial Constraints*

When approaching this project the goal was to keep costs low, this decision led to the implementation of the servo motors and using linear actuation. The use of the servo motors has led to some issues with power consumption as well as reliability. The servos draw a high current and as we are using a lot of servo motors this leads to high current for the circuit overall. On average we find the current draw for a single servo to run to be around 400 mA. This is something we had to decide to live with as the cost of a servo is around $2 per servo, and around $60 total for all servos. This project could be done implemented easier using solenoids as they draw significantly less current but in doing that we run into an issue with our budget, it is simply not affordable for us to obtain solenoids as they are about $10 per which would drive our cost up way more than we are comfortable with. The servos are not the only way we are constrained, the materials we use are made out of wood which aesthetically is not very attractive. We have ran into issues with our material as originally we had wood for our linear actuation. Quickly after this we were forced to move to a more robust material, acrylic. It would be fantastic to make our project entirely out of acrylic however it is not entirely necessarily essential.

V. Final Hardware & Software Design Details

The following section outlines our final project design. This will be the target configuration for our Final Presentation.

A. *Mechanical Overview*

For this project, custom mounting hardware was required to attach the 30 servo motors to the guitar, and enable them to fret and strum the guitar strings.

1) Fretting Mount: The first hurdle for the fretting mount was fitting 6 servos across the width of the guitar neck. Not even 3 servos could be placed side by side without being too far apart to reach every other string. Thus, we decided to mount the fretting servos in 3 layers, with 2 servos each. Each successive layer would use long fretting 'sticks' to extend the actuation of the servo all the way down to the strings. The bottom of the mount feature a panel with holes that guide the sticks into the correct positions.

2) Strumming Mount: The strumming mount was much easier to design and assemble. It consists of a simply bracket that positions the servos above the strings. Near the sound hole, the strings are further apart, allowing us to place 3 servos side by side. We attached small plastic picks to the end of each arm to get a consistent sound out of the guitar.

B. *Electrical Overview*

1) Power Supply: The total load that the project demands from its power supply consists of the power demands of our 30 servos, 4 servo shields and ESP32. The servos each pull an average of around 400-500mA when moving during testing. The power supply will consist of 5 LM2576 Switching Regulators configured to step down 12V from the power source to 5V. As the LM2576 is not rated for back current, the output node of each regulator will be isolated from one another. 4 regulators are dedicated to powering 30 servos, 3 of which power 8 of our 24 fretting servos and 1 which powers our 6 strumming servos. Each regulator is rated to supply 3 Amps, and each fretting regulator will see a theoretical max demand of 4 Amps. While this is suboptimal, there would be no situation in which all 8 servos are moving at once for any regulator. Thus, we can safely assume that our power demand will be within tolerance for the regulator. The strumming regulator is able to handle our 6 strumming servos at max load. The last regulator will be dedicated to powering the ESP32 and Vcc for the Servo Shields, the current load of which is negligible.

2) PCB: The PCB contains the regulator circuits that will power the project, as well as the ESP32 and I2C outputs. Note that the ESP32 is integrated with its Devboard, and the Devboard interfaces with the PCB using through-hole connections. The PCB consists of two layers with ground planes surrounding signal and power traces. The PCB takes in power from our SLA battery through a Molex Mini-Fit Jr. two-pin header and regulates it internally. 4 of the regulators will be output through similar 2-pin molex headers, while the last regulator output voltage powers the ESP32 through the Devboard. The Devboard then outputs the I2C data from the ESP32 to pins 21 (SCL) and 22 (SDA) which is then output from the board using two pins of a 4-pin Molex header. Alongside this, the voltage from the 5th regulator is output alongside ground in order to power the control logic for the 4 Servo Shields.

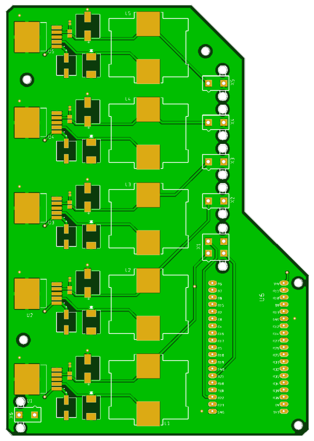


Fig. 2. PCB board layout, note that this does not include parts

C. *Programming & Code*

1) MIDI Preprocessing: Before we can play a MIDI track on the self playing guitar we need to first use a MIDI preprocessing algorithm to make the song more guitar friendly. The steps to doing this includes

- Inputting the MIDI file into the algorithm.
- Muting or Removing unplayable tracks such as drums or any synth sounds and consolidate everything into one track.
- Shift the range to accommodate the playable range of notes since we are not using every fret.
- Loop through the track and ignore unplayable sounds and compress the notes into the playable range and force the overlapping notes occurring on the same string to end.
- On the 2nd pass through the loop it will remove notes that happen to quickly to accommodate the hardware if the movement speed is impossible.

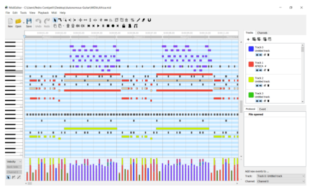


Fig. 3. MIDI File before Preprocessing Algorithm

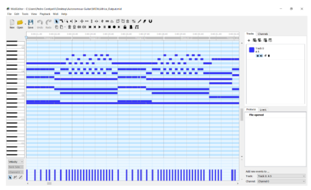
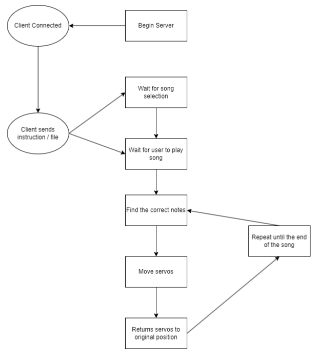


Fig. 4. MIDI File After Preprocessing algorithm

2) Micropython: For programming the micro controller we are using Micropython which is a version of python that is optimized for use on a micro controller. We will use this to parse MIDI files for the micro controller to send signals to the Self-Playing Guitar.


Client Connected
Begin Server
Wait for song selection
Client sends instruction / file
Wait for user to play song
Find the correct notes
Repeat until the end of the song
Move servos
Returns servos to original position
Fig. 5. Basic operation of the self playing guitar

VI. Prototype Construction & Coding

The following section describes the considerations required to prototype our project. The purpose of our prototype is to demonstrate a minimum viable product and proof-of-concept so as to demonstrate both to ourselves and our advisors that we are on the right track.

A. Circuit Prototyping

In order to prototype our project, we needed to have our circuitry up and running before the final design takes shape, consisting of our regulators alongside the datalines to our Servo Shields. However, the monetary cost of PCB production combined with the time investment in getting the board means that creating PCBs is no small investment and impractical for our prototype. Rather than having the PCBs made, we instead elected to create the circuits using breadboards. Breadboarding allows us to rapidly prototype our circuits as well as changing them as needed, saving us a considerable amount of time. On top of this, because of the availability of breadboards in both the senior design lab and other labs across UCF, the cost of prototyping becomes essentially free. This, however, did not account for the cost of the parts needed to create the circuit. Our components had to arrive before we were able to properly prototype, thus it was paramount that components be ordered ahead of time. However, we had to make considerations for the type of components we will be ordering. Because the parts we have considered are surface mount, replicating the circuitry would be difficult on a breadboard. To account for this, we only ordered and utilized a package for the LM2576 that was able to be used on a breadboard. We supplemented this with through-hole parts with the same values as those that would be present in our final circuit in order to get our power supply up and running. This would not have the same stability as our final design, but would be good enough for basic actions from the guitar.

VII. Conclusion

This project has had it's share of difficulties through the large amount of mechanical design we have had to incorporate. However through these struggles we were able to gain a lot of experience in disciplines not typically introduced to us in our traditional course work. Throughout the duration of working on this project we encountered plenty of setbacks. Our initial planning we found potential solutions to both our strumming mechanism as well as circuit design. Our first solution for the strumming was ideally to use solenoids which we found to be too expensive so we moved to wooden mounts which were laser cut. We found this to be a sufficient solution to our project. Our circuit design was where we struggled the most, in doing initial research we though our best solution to control our servo motors would be through shift registers. This solution allows us to use the least amount of pins but also have enough control signals to send to all the servo motors. In basic testing we had some success in being able to control the servos however we experienced several issues such as the shift registers totaling failing or would send incorrect signals. This is due to the current running through our circuit being high relative to what the shift registers can handle as well as the latency of the shift registers were not being able to control the signals as precise as we are looking for. Unfortunately for us we had a lot of time invested into this design before deciding we couldn't move forward with this reliably. Doing some research from there we were able to find inspiration from some people on YouTube who were using servos in a similar way to us where we found servo shields which are able to process 16 servos on one board with 1 voltage input. Through using two of these we were able to simplify our circuit significantly since the shift registers were using a lot of wires making it messy. Our current design using the ESP 32 dev board as well as servo driver boards is one we are finding a lot of success early on as we were able to demonstrate the ability to play some simple songs using this circuit design on a breadboard. Due to the simplicity of the circuit design we believe our PCB design should work no problem on the first try and from there the only thing for us to do is fine tune our code and run some additional test trials. Through all our setbacks and research done the project has gained us a lot of much needed real world experience and team work skills that will help us moving forward in the industry.

VIII. Biography

A. Pedro Contipelli

Pedro Contipelli is a senior at the University of Central Florida majoring in Computer Science. He has many years of programming experience in several languages such as Java, Python, and C. His main academic interests include space exploration, artificial intelligence, robotics, and philosophy.

B. Ethan Partidas

Ethan Partidas is a senior at the University of Central Florida, pursuing a Bachelor's degree in Computer Engineering. He has a wide range of experience, having competed in mathematics and programming competitions, designed circuits and 3D models for laboratory experiments, and researched novel CAD algorithms for flow-based in-memory computing. He continues to pursue learning a wide variety of skills and topics.

C. Blake Cannoe

Blake Cannoe is a senior at the University of Central Florida and is scheduled to graduate with a degree in Computer Engineering on May 4th 2023. He is currently seeking employment. His interest mostly lies in embedded systems.

D. Jonathan Catala

Jonathan Catala is currently a senior at the University of Central Florida and will receive his Bachelor's of Science in Electrical Engineering in May of 2023. He has attended the University of Central Florida for four years now and plans to continue his Masters here. He is currently working as an intern at LaserStar Technologies. His primary interests lie in control systems, and signal processing.

E. Kyle Walker

Kyle Walker is a senior at the University of Central Florida pursuing a Bachelor’s Degree with a major in Electrical Engineering along the Comprehensive track. He is slated to graduate May 4th 2023. Currently, he is contracted out to Lockheed Martin on behalf of UCF as part of their College Work Experience Program. While his interests were largely undecided throughout university, he has developed an interest in Power systems due both to this project and his time in Lockheed.