ECE 445: Auto-Played Guitar

Jiyu Hu (jiyuhu2), Peilin Rao (peilinr2), Qianlu Chen (qianluc2)

Final Report for ECE 445, Senior Design, Fall 2020

TA: Yifan Chen

9 Dec 2020

Team #: 12

Abstract

In this project, we designed, implemented, and tested the Auto-Played Guitar, a device that can be installed on any kind of guitar to play music. The device consists of two main components of fulfilling the basic functionality: 36 actuators and six servo motors. The 36-actuator component mimics the left hand of a guitar player to press the frets on the guitar. The 6-servo motor component mimics the right hand of a guitar player to strike the guitar strings. Combined, they produce the desired sound. Besides, the device can produce drum beats to add flavor to the music played. We have successfully encoded several classical music pieces into the device. They can all be played at an adjustable tempo.

Table of Contents

1. Design

1.1 Introduction

We built an auto-played guitar for those people or places that need it. Some people love the original sound of guitar music but lack the time and energy to practice it; Just like the existence and common use of auto-played piano in the lobby of grand hotels [1], some bars may need a playing guitar for entertainment or creating atmosphere; Guitar stores want to show the good sound quality of the guitars they are selling but it is usually very expensive to hire someone to play for a long period of time; Music creators want to hear the sound of their customized guitar for testing or for remix; Some people want to hear the authentic sound of guitar music as their wake up alarm; Some new guitar learners wish to hear a demonstration of the musical piece that they are practicing on while watching the notes they should play. Despite that people can listen to guitar music by simply using electronic music players, the beautiful original sounds from different real guitars are irreplaceable. Our design solves our customer's problems by presenting the auto-played guitar. Although the idea of the auto-played guitar is not an invention, all the existing auto-played guitar devices are expensive to manufacture, too ponderous, or lack compatibility to be reinstalled on a variety of guitars. Therefore, the pivot of our design is to build a piece of automatic guitar playing unit that is affordable, portable, and compatible with any type of guitar.

1.2 Solution Overview

In order to build a device that can be installed on any type of guitar for automatic music playing, we break the design down into six subsystems. They would work together to mimic the left hand and the right-hand motion of a real guitar musician while also providing extra functionalities such as alarm and metronome. In order to mimic the left-hand motion, our device can press any string swiftly from the third fret to the eighth fret using 36 linear actuators, which is a wide enough range for most of the guitar music pieces. It can activate at most 8 linear actuators concurrently, which means it can produce any chords in this range. Also, to mimic the right-hand motion, our device can strum the string to play clear sounds with six servo motors. All the fret pressing and string strumming motions are controlled by the signal pins from our PCB and can respond quickly to our multitasking software. The alarm is embedded into the PCB as well, which provides several buttons for users to set the alarm time and plays the pre-figured music when the time is up. The metronome produces a periodic drumbeat sound by hitting the guitar surface using a servo motor. It acts as a practicing aid and also adds flavor to the guitar music produced. All those parts are powered up by the design idea of separating power and control signals to ensure safety and stability.

1.3 High-Level Requirements

To verify the overall system works, we require our device must:

A. Be able to play the correct notes on the guitar in tune and loudly.

B. Be able to deliver the drumbeat and guitar note at the correct rhythm and adjustable tempo.

C. Be able to perform an alarm function that is synchronized with real-world time.

1.4 Block Diagram

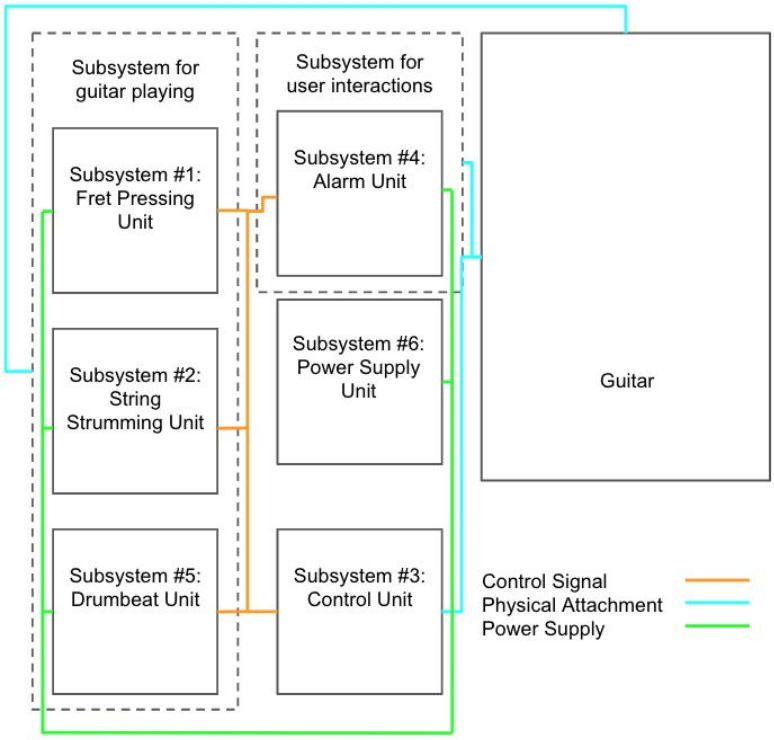

Figure 1. Block Diagram of overall design

Our design of the auto-played guitar contains six submodules: Fret Pressing Unit, String Strumming Unit, Control Unit, Alarm Unit, Drumbeat Unit, and Power Supply Unit. According to the block diagram, the Fret Pressing Unit and the String Strumming Unit work together to produce guitar music while the Drumbeat Unit handles tempos and adds flavor to the performance. The Alarm Unit allows user interactions with our device to set the time of the alarm. The Control Unit can process the code to create control signals for the three guitar-playing units and the Alarm Units. The Power Supply Unit figures out the power requirement for others and provides the corresponding voltage. The Power Supply Unit has its own PCBs with transistors matrices for driving the linear actuators in the Fret Pressing Unit, which requires a larger current than others.

2. Implementation

2.1 Visual Aid

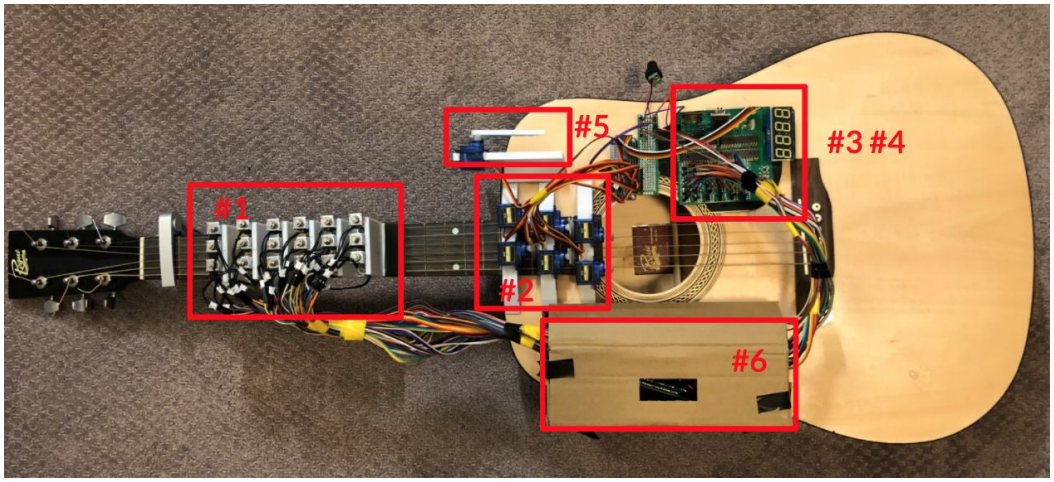

Figure 2. Auto-played Guitar overview

The Fret Pressing Unit is attached to the fretboard of the guitar while the String Strumming Unit is placed on the guitar body. The Control Unit and Alarm Unit are both placed on the main PCB and attached to the guitar body. The Drumbeat Unit is placed on the edge of the guitar surface. The Power Supply Unit contains the power jack, all the power wires, and the transistor matrices placed in a box.

2.2 Fret Pressing Unit

The Fret Pressing Unit mimics the left-hand motion of a guitar player by pressing and holding the strings on the frets according to the control signal from the Control Unit. It consists of two parts: 36 actuators and the 3D printed parts that fix them on the fretboard. The linear actuators of our choice are Sparkfun Push Actuators ROB11015 [9], which are powered by the transistors matrices in the Power Supply Unit.

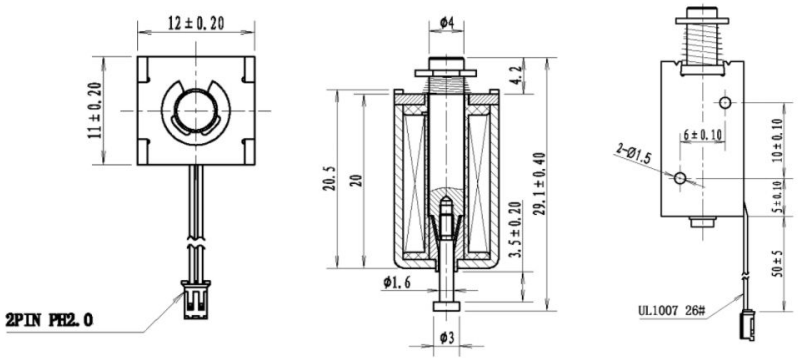

Figure 3. Sparkfun Push Linear Actuator ROB11015

The actuators can generate up to 8 gram-force when working at 5V and 1.1A, which is verified to be strong enough to hold the strings tightly on the fret bar. To ensure that strumming the pressed string does not prevent the actuators from holding it, we designed a 3D printed holder for 36 actuators.

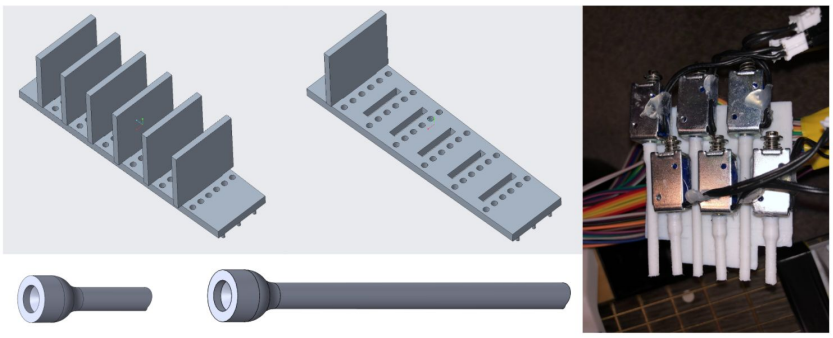

Figure 4. 3D printed holder for linear actuators

The holder consists of a base with 36 holes based on the direct measurement of the guitar frets, six square boards that can be inserted perpendicularly to the base, 18 short extension rods, and 18 long extension rods. To fit the design of this plastic holder, six actuators are attached on each square board with a zig-zag arrangement and alternatively using long and short extension rods that touch the strings.

2.3 String Strumming Unit

The String Strumming Unit mimics the right-hand motion of a guitar player to hit the guitar through plastic guitar picks attached to the six servo motors. The servo motors of our choice are SG90 [10]. In order to generate the PWM wave that delivers the analog signal of how fast and how many degrees the motors should rotate, we are using a PCA9685 servo driver chip [12], which is controlled by the Control Unit. To make our device as compact as possible, we 3D printed three bridge-shaped holders and placed two servo motors on each holder. Each servo motor is in charge of strumming one specific string and the guitar pick attached to servo motors can barely touch that specific string without interfering with each other.

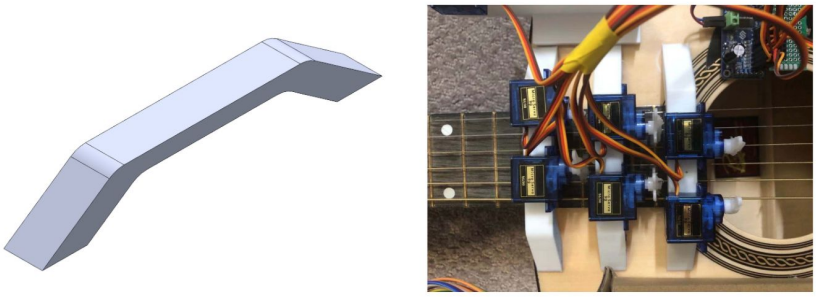

Figure 5. 3D printed holder for servo motors

2.4 Control Unit and PCB design

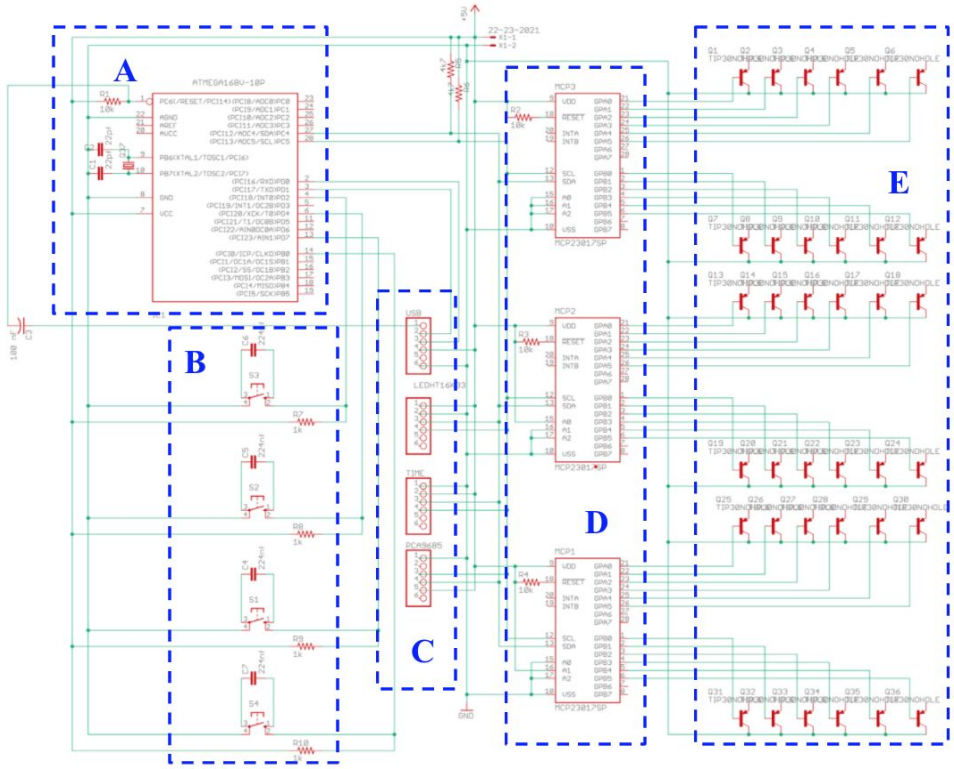

Figure 6. PCB schematics divided into different sections

The aim of the Control Unit is to generate control signals for other units based on the code imported into the microcontroller. The PCB schematics can be divided into several sections, which serve the purposes of driving different units. Section A from Figure 6 is the microcontroller chip ATMega328 chip [13], which is the central processor of the Control Unit that determines what other units should do.

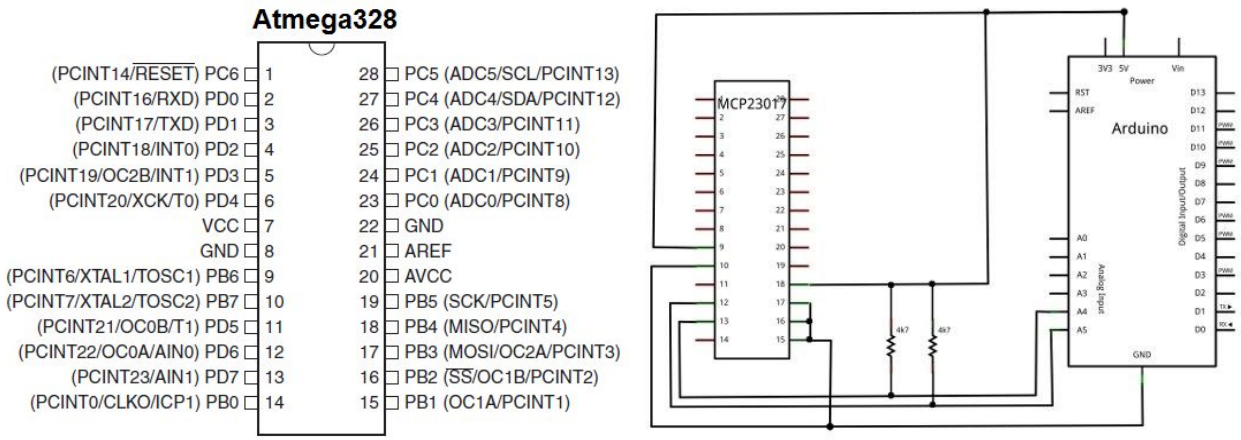

Figure 7. ATMega328 Pinout (left), MCP23017 configuration (right)

However, ATMega328 itself does not have enough digital pins to drive 36 actuators. Therefore, we connected three I2C pin extenders chips MCP23017 [11] in Section D in the circuit schematics. The detailed connection configuration is shown by the right diagram in Figure 7, where A4 and A5 pins of an Arduino board correspond to the SDA and SCL pins on ATMega328. When different address pin configurations on the pin 15, 16, 17 of the MCP23017, ATMega328 allows software to access all 16 digitals pins on each MCP23017. Section B in the circuit schematics is for four active-low push buttons that can be used in the Alarm Unit. Section C contains four six-pin ports for USB chip, LED chip, time chip and servo driver chip PCA9685. LED, time and servo drivers chips require connection to SDA and SCL pins to ATMega328 while the USB chip needs RX and TX pins for Serial Peripheral Interface Communication. Section E provides simply 36 control pins for all the actuators. Those control pins are delivered to the transistors matrices in the Power Unit, which then provide sufficient voltage for actuators to work.

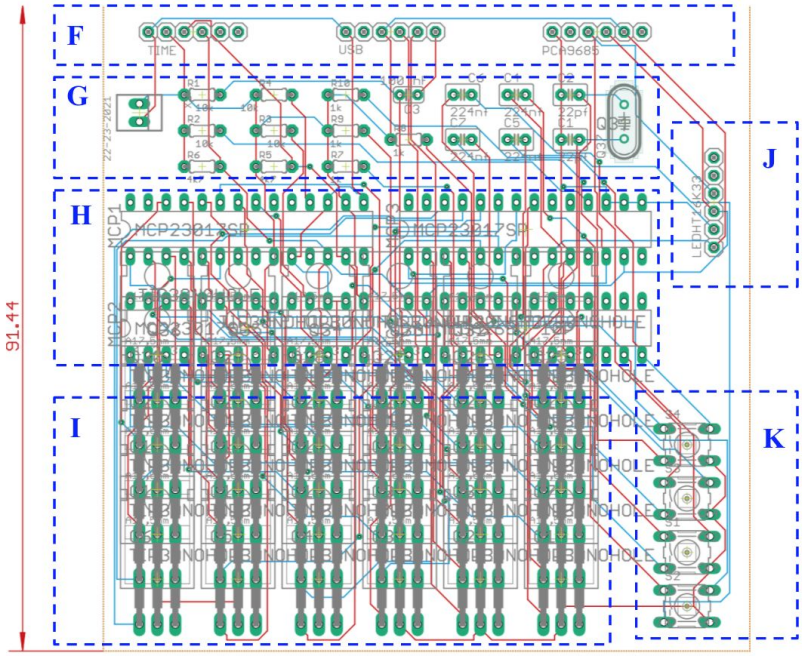

Figure 8. PCB components arrangement

The PCB arrangement of all the components shown in Figure 8 is deliberately designed with the principle of clustering. The ports of time chip, USB chip and servo driver chip in section F and the LED chip in section J are placed on the edge of the board to leave enough room for those chips to be inserted. Section G contains all small parts including resistors, capacitors, and an oscillator. They are all clustered in one block to ease the soldering process. Three MCP23017 chips and the ATMega328 chip are placed in Section H, four buttons are placed in Section K and the 36 control signal pins are placed in Section I.

2.5 Alarm Unit

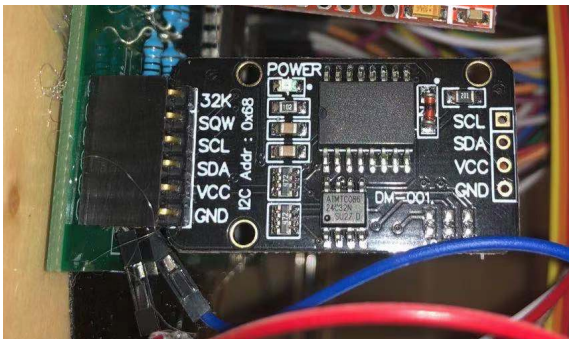

Figure 9 . Real Time Chip DS3231

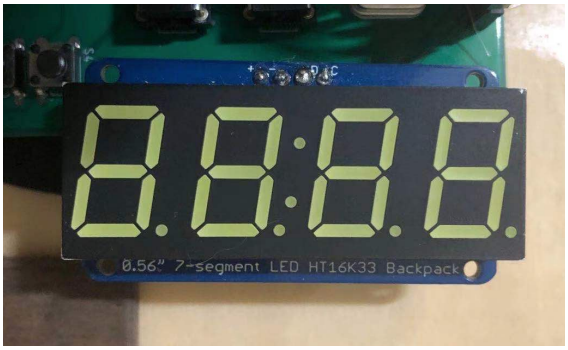

Figure 10. Adafruit 7-segment LED chip

The Alarm Unit is designed to make the guitar function as a wake up alarm. The goal is to play a certain piece of music on the guitar when the preset time is reached because some people find it very enjoyable to be woken up by the real guitar sound. The main components of the Alarm Unit are a time chip [5] and an LED chip [7]. The ports of both chips are placed on the main PCB board with the Control Unit. In order for this unit to function normally, they need to be connected to the ATMega328 along with three control buttons as shown in Figure 11. They both use standard SDA/SC pins to communicate with the ATMega328 and both work at 5V DC. The time chip itself has a small battery on its back and is individual of the power supply. After its time has been set, it can keep the inner clock on its own even after the power is cut off.

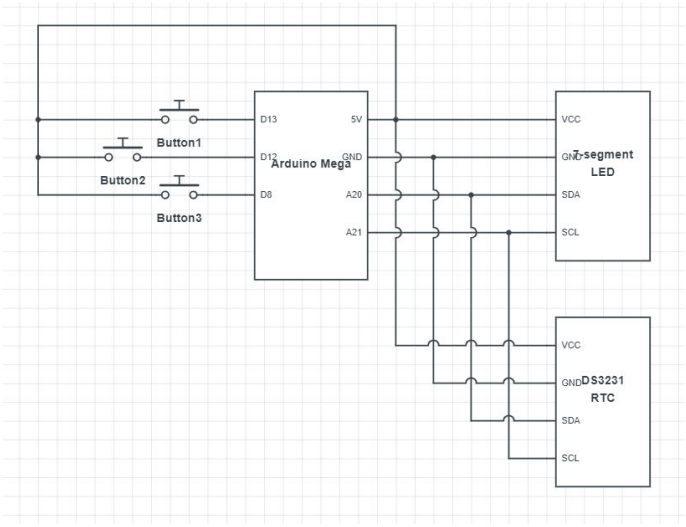

Figure 11. Alarm Unit schematics

The Alarm Unit has two main functions: time telling and alarm. For time telling, the ATMega 2560 will get the data from the time chip and then command the LED to show the numbers, also asking the colon in the middle to blink every 500 ms. For the alarm function, the time chip has inner built alarm registers, and can store two alarms[8]. However, for simplicity we are only using one. In order to set the alarm, one way is to serial transfer it to a string of data in predetermined format. Yet, we can not do that for common user operation. There are four buttons on our main PCB. While one of them is for direct music playing tests, the other are for the Alarm Unit. These three buttons are embedded to make it function like some of the buttons on a watch and help us set the alarm. Button 1 is the mode button. There are three modes in the inner software state machine for the alarm. Mode 0 is the time telling state where the system is usually in. After pressing Button 1 once, the system will switch to Mode 2 where the time stops refreshing on the LED and we can use Button 2 (Add) and Button 3 (Minus) to adjust the hour number for the alarm. Pressing again Button 2 and the system will switch to Mode 3 where we can adjust the minute number for the alarm in a similar manner. Finally when Button 1 is pressed the third time, we will switch from Mode 2 back to Mode 0. Then the time resumes on the LED and the previously set hour and minute will be transformed into an instruction and sent to the time chip to keep the alarm.

2.6 Drumbeat Unit

The Drumbeat acts as a metronome with adjustable tempo. It contains a servo motor SG90, a holder, and an L-shaped arm. When placed on the edge of the guitar body surface, two tips of the L-shaped arm can hit the guitar body and the guitar edge. Since the guitar body and the guitar edge are made of different materials, the sounds produced by the hits are different. The Drumbeat Unit uses those two different drumbeat sounds to add flavor and rhythm to the music produced.

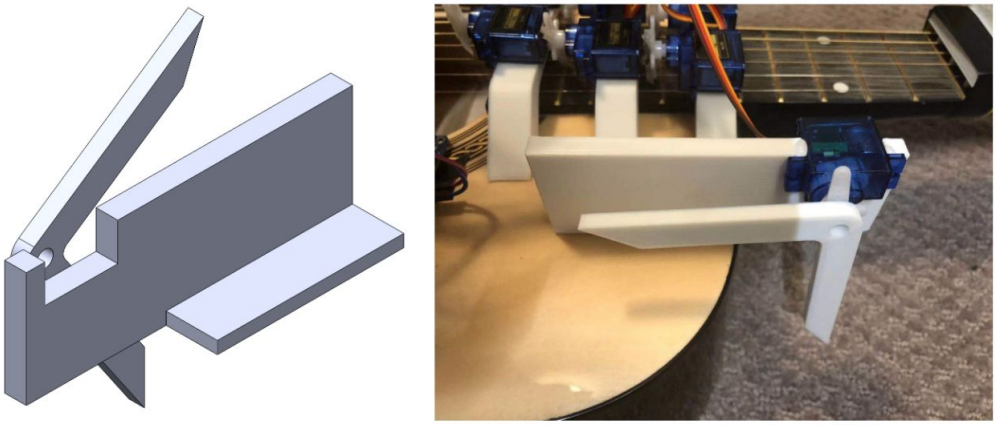

Figure 12. 3D printed holder for servo motor and the L-shaped arm

2.7 Power Supply Unit

2.7.1 Power Source

Device	Working Current
Actuator ROB 11015	1.1A
Servo Motor SG90	360mA
LED	160mA
RTC	160mA
ATMega328 microcontroller	10mA

Table 1. Power requirements of all parts

We first provide a brief calculation of the total power supply required for our system to function properly. As indicated in Table 1, the component that consumes most of the power is the actuator. Compared to it, the power consumption of other components is almost negligible. By design, the Auto-played Guitar needs at most six actuators to work at the same time, since at most each string needs one actuator to press. All of the components run under *5 V* DC. Therefore there is no need in our design to perform voltage transformation in the circuit. According to the electrical power formula:

$$P = V \times I \tag{2.7.1.1}$$

Each actuator functions with a power supply of 5.5 *W*. Therefore, six actuators working in parallel requires *33.0 W* of power supply in total. Following the same formula (2.7.1.1), each SG90 Servo Motor runs with *1.8 W*. Seven servos (six for string strumming and one for drum beating) in total consumes *12.6 W*. The LED unit, RTC and ATMega328 microcontroller together occupy *1.65 W* power distribution, which can be ignored. We choose a *5 V 75 W* power source that is easily accessible through websites such as Amazon [3]. *75 W* is more than enough to guarantee an abundant power supply to our system.

Figure 13. Power Adaptor [3]

2.7.2 Separation of Power and Logic Circuits

Two problems arise with the high power requirements of the actuators: 1) the ATMega328 microcontroller digital I/O pins only supports a maximum current of *50 mA*, which is not enough to drive the actuator directly; 2) the large current for driving the actuators can burn the ground wires on our PCB board because the wires are too thin to support large currents.

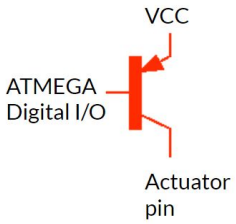

Figure 14. Control circuit for actuators

To address the first problem, we use the output signal from ATMega328 as a logic signal that triggers transistors that are directly connected to the power source. As indicated in Figure 14 The transistors function as logical switches controlled by the output digital signals to drive the actuators.

The second problem is more difficult to solve. We decided to completely separate the control signals on the PCB board from the power supply circuits that drive the actuators and servos. In this way, the PCB only provides logic control that runs with very small current, safe from burning the PCB circuit.

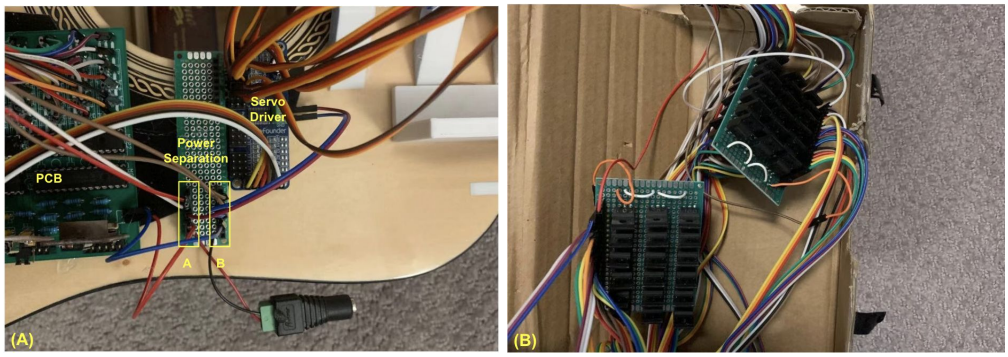

Figure 15. Separation of control and power circuit

As Figure 15 (A) shows, the power jack is connected to the Power Separation board. On the Power Separation Board, power supply is divided into section A and B. Section A supports the function of the PCB control logic, while section B powers the Actuator Driver Unit in Figure 15 (B) and Servo Driver. The Actuator Driver Unit are two 36-transistors matrices connected to the power source and controlled by digital signal from the PCB.

2.8 Software

2.8.1 Workflow

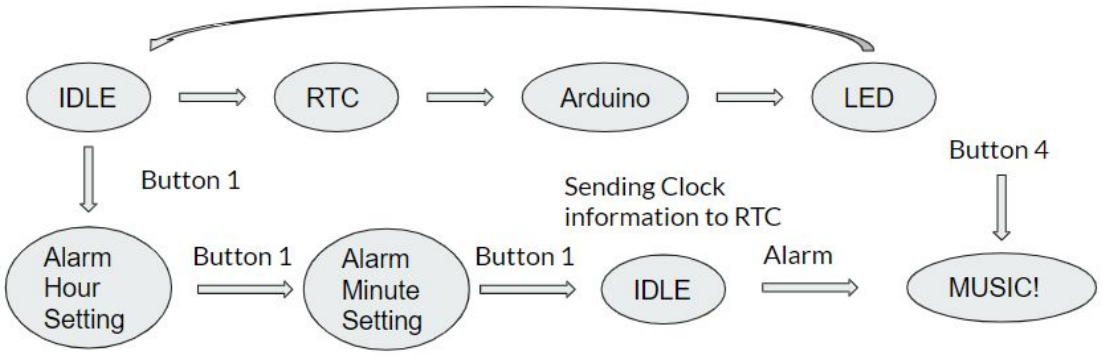

Figure 16. workflow diagram

The system's software has three modes, two of which have been briefly described in the **2.5 Alarm Unit** section. Usually the software is only running a loop to get data from RTC to LED for showing the correct time. This is called the IDLE mode.

Only when Button 1 is pressed the system jumps to a different loop of setting the alarm: ALARM mode. Here time is still kept by the RTC but data is no longer requested. The numbers on the LED start acting as a temporary register for the hour and minute we want to set the clock at. The code constantly checks only for signals from the buttons.

The most important part of software is the MUSIC mode. This can be triggered by either pressing Button 4 manually or when the RTC sends out a signal that the alarm has gone off. However, unlike the other two modes, MUSIC mode needs to command the Fret Pressing Unit, the Drumbeat Unit and the Alarm Unit. To achieve this we implemented a simulation of multithreading and will be discussed in later sections.

2.8.2 Sheet Music Data Structure

```
0: do nothing
1: release actuator
2: Servo strike
3 - 8: Press corresponding fret on guitar
```

Figure 17. Data Structure Rules

```
const uint8_t scoreSheet[][6]
PROGMEM =    {{3, 0, 0, 0, 0, 0},
              {4, 0, 0, 0, 0, 0},
              {5, 0, 0, 0, 0, 0},
              {6, 0, 0, 0, 0, 0},
              {7, 0, 0, 0, 0, 0},
              {8, 0, 0, 0, 0, 0},
              {1, 3, 0, 0, 0, 0},
```

Figure 18. Data Structure Example

We designed a special data structure for storing the music piece being played in the software. Due to the way we designed our hardware, we only need the information of when to press which fret and when to

strike which string in order to play the music. Since at each minimum time interval we have at most six strings to press or strike, we decided to use a 6-input vector to store the information. Each input would represent one string. Usually when we press a string we would also need to strike it in order for the sound to be heard. However, our actuators push down fast and hard enough so that the simple "striking" of the fret can produce the same note as pressing and striking simultaneously. As our hardware covers from the third to the eighth fret, we use number 3-8 to represent which fret to press. If the same note is requested after that note has been played by Fret Pressing, we will need to strike that string, which is indicated by number 2 input. If we want all actuators to be released on a certain string we will give number 1 and if we want the actuators and servos to remain in their current position we would use number 0.

2.8.3 Multithreading

When playing the music, we need the program to control the movement of the actuators, string strumming servos and the drum beating servos. At the same time, we also require the time to be undated and the colon between the hour and minute number to blink. The actuators and string strumming servos can be triggered in a sequential order, as we always first press the fret and then strike the string. However, in the case of producing the drum beats and updating the time, these behaviors are not necessarily synchronized with the fret pressing and string strumming. As a consequence, we need to keep a different tempo for these three instructions, thus requiring some sort of multithreading in the control flow. This is illustrated in the multithreading workflow in Figure 19.

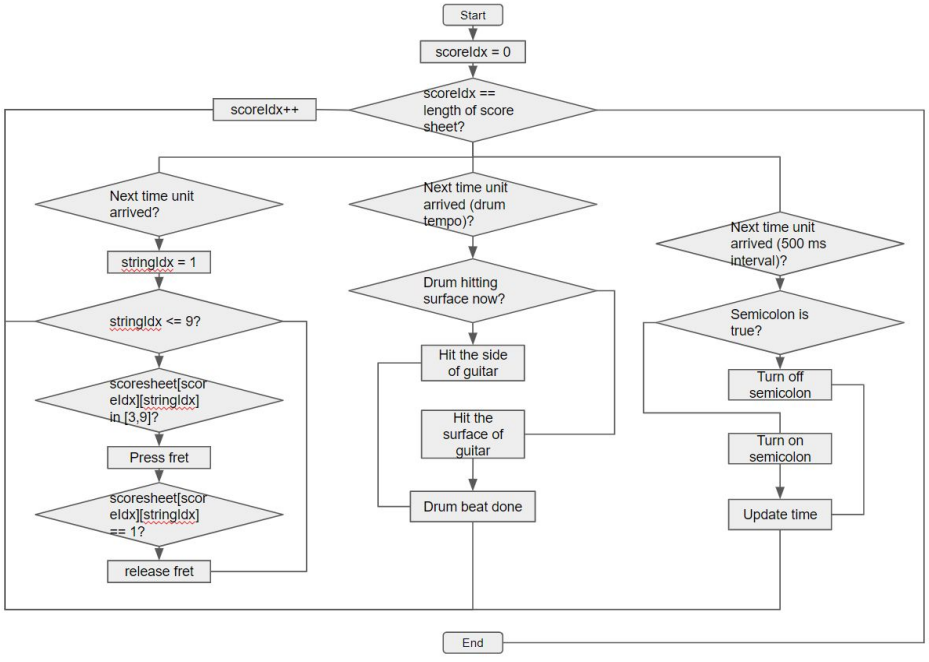

Figure 19. Multithreading workflow

As it is known to all, ATMega328 is a basic microcontroller that does not support multithreading by itself. Therefore, we need a pseudo-multithreading mechanism, i.e. a task switching mechanism, implemented in the software. We utilize Protothreading as the solution [4]. Protothreading quickly switches between different tasks and checks for conditions for execution. If the condition is met, then the corresponding function is executed. Usually the condition is the some time interval between the two consecutive times the function needs to be called. This is shown by Figure 20, the code example.

```
565  static int actuatorThread(struct pt *pt, int interval, int * counter, bool * clockWise) {
566    // static unsigned long timestamp = 0;
567    PT_BEGIN(pt);
568    while(1) {
569      PT_WAIT_UNTIL(pt, millis() - timestamp > interval * (*counter));
570      (*counter)++; // take a new timestamp
571  //    Serial.print("counter");
572  //    Serial.println(*counter);
573  //    delay(1000);
574      for (int i = 0; i < 6; i++){
575        uint8_t temp = pgm_read_byte_near(&scoreSheet[scoreIdx][i]);
576        if (temp == 1){
577          releaseActuator(i);
578        } else if (temp == 2) {
579          heavenlyStrike(i, clockWise);
580        } else if (temp > 0 ){
581          releaseActuator(i);
582          pushActuator(i, temp);
583        }
584      }
585      done = true;
586      scoreIdx++;
587    }
588    PT_END(pt);
589  }
```

Figure 20. Code example for the actuator thread

Line 569 is the condition statement. Here *millis()* is a function to retrieve real world time in milliseconds, and timestamp is the first time the function is executed, so the difference is the time passed since last execution, this should be multiples of the designated interval.

3. Verifications

3.1 Overall Verification

For the Fret Pressing Unit, we need to verify that the fret pressing unit can actually play the correct sound. The verification process is very straightforward: to play each note by order and check the quality of the sound with the help of a tuner. We have developed a test case in the control program to activate each actuator in order. The attempt of the general test failed to pass due to some of the actuators not installed in the right place. We smoothly fixed this problem by reinstalling them at the correct position. After that the test was easily passed. For the String Strumming Unit, we also included a small unit test for this unit in our control program to drive each of the servos by order, each note turns out to have at most 2% discrepancy in frequency with its correct value, which is completely acceptable. The overall verification is performed after the detailed verification. Therefore, we passed this verification on the first attempt - all servos can strike to produce the sound correctly. The Control Unit is the most important part of our design because it coordinates the behavior of all other units, but the verification is in contrast the simplest. This is because the ATMega microcontroller is easily programmable and debuggable on the Arduino platform. We do not need to specifically verify the correctness of the program, as the functionality is reflected by all other units.The verification of the Drum Beat Unit shows that there is at most 1.5% discrepancy in the time intervals provided under the same tempo, which is acceptable. For the Power Supply Unit, we met and solved a major problem that is discussed below.

3.2 Major Verification Problem

When verifying the functionality of the PCB, we burned several of them during the process. In order to pinpoint the problem on the PCB, we used a multimeter to measure the current and voltage between all pins on the PCB. We found that the current between the transistor and ground hit 5.6 A for a very short time, and then the PCB burned. Discovering this, we immediately realized that the over-large current was burning the board.

To solve this problem, we separated the power circuit and the control logic circuit, as described above in **section 2.7.2**. We further verified the correctness of the new circuit with the multimeter.When driving six actuators and seven motors at the same time, the current in the power circuit could go up to 8 A instantaneously. However, the current in the PCB is always smaller than 400 mA. We measured the current on the PCB over the span of playing “Canon in C”, our most power demanding piece of music. The average current was 206.3 mA, with a variance of ±85.4 mA. This current is absolutely within the safety range of the PCB board.

4. Cost

The total cost of our project is:

$$Total\ Cost = Purchasing\ Cost\ +\ Labor\ Cost\ =\ \$320 + \$15000\ =\ \$15320 \quad (4.0.1)$$

4.1 Purchasing Cost

Items	Number	Cost (US dollars)
PCA9685 servo driver chip	1	7
MCP23017 I2C pin expander chip	3	3 × 3 = 9
3D printing material	-	30
ATMega 2560 microcontroller chip	1	10
servo motor SG90	7	3 × 7 = 21
linear actuator ROB-11015	36	4 × 36 = 144
transistor 2SK3703	36	1.25 × 36 = 45
wires, resistors, capacitors, oscillators	-	20
USB chip	1	6
time record chip	1	4
Hex LED	1	4
PCB manufacture	2	2 × 10 = 20
Sum	-	320

Table 2. Purchasing Cost

4.2 Labor Cost

We assume our hourly payment is 25 US dollars. Each of us spent about 80 hours on various tasks including designing, building, testing, soldering, and coding. Therefore, the total labor cost should be:

$$Labor\ Cost = 3 \times 25 \times 80 \times 2.5\ =\ \$15000 \quad (4.2.1)$$

5. Schedule

Date	Peilin Rao	Jiyu Hu	Qianlu Chen
10/7/2020	Finish the 3D printing part of the fret pressing unit. Start to design the 3D printing part of the string strimming unit.	Finish the prototype circuit of the Drumbeat unit. Finishing purchase of all the requirements parts.	Finish Testing the LED and RTC chips on the breadboard. Achieve function of showing time, and setting alarm with button.Draw the PCB routing for LED, RTC, and button.
10/14/2020	Draw the PCB part for the fret pressing unit.	Draw the PCB part for the string strumming unit.	Integrate PCB design for all units.
10/21/2020	Test the prototype of the fret pressing unit circuit.	Test the prototype of the fret pressing unit circuit.	Integrate all circuits on breadboard for testing and await for PCB arrival.
10/28/2020	Start to write code in center control part, including possible finite state machines and control logic	Take tests on all individual parts except center control, fix bugs if any.	Soldering and verification for first round PCB. Make necessary changes and improvements, submit the final version of PCB.
11/4/2020	Program music into the control control music.	Test the code and testing.	Integration of code and testing.
11/11/2020	Edge tests on basic functional units (#1 #2 #3)	Edge tests on advanced units (#4 #5 #6)	Integration of code and testing
11/18/2020	Demonstration	Demonstration	Demonstration
After that	Write final paper	Write final paper	Write final paper

Table 3. Schedule

6. Conclusion

6.1 Accomplishments

In conclusion, we successfully implemented a Auto-Played Guitar that is capable of playing any music and can be installed on any guitars based according to the high level requirements. We demonstrated a few music pieces including “The Imperial March (Darth Vader's Theme)”, “Autumn Leaves”, and “Canon in C”, which are very challenging tasks for the stability of our mechanical parts and the functionality of our control logic. The decent performance for those complex music pieces proved the completeness of design as well as the correctness of implementation. Our design of the music sheet storage data structure provides a concise interface for customizing the music played. Also, with the power of our multithreading control software, the system can perform add-on features such as alarm and metronome.

6.2 Uncertainties and Future Improvements

Due to the shortage in budget and time, the Auto-Played Guitar has some limitations. We used a cheap 3D printer for creating the mechanical structures in the Fret Pressing Unit and the String Strumming Unit. The position of the extension rods on linear actuators are not very accurate because of 3D printing uncertainty of ± 0.2 mm. In order for those structures to function properly on different guitars, some manual adjustments on the position of the actuators are required. This problem can be solved by using a better 3D printer.

In addition, given the current project as a prototype, we can optimise the connections between each unit. By applying better packaging, we can turn all models in our system to a unified part, which would improve the portability and stability of the Auto-Played Guitar as a commercial product.

6.3 Ethnic Issues

We anticipate concerns about our work going against #2.6 in *ACM Code of Ethics*[2] since some people think that machines should never replace human beings in terms of creating art and music. However, in our opinion, the project’s purpose is never to replace the human efforts in creating and performing fine music. In contrast, it is rather a means for many more people to enjoy music and get more access to the beautiful and original sound of guitars.

7. References

[1] Yamaha Disklavier ENSPIRE Product Demo | Piano Gallery Utah: (Apr.9.) Retrieved on October 01, 2020 from https://www.youtube.com/watch?v=sxI0PIPZjBQ
[2] The Code affirms an obligation of computing professionals to use their skills for the benefit of society. (n.d.). Retrieved October 01, 2020, from https://www.acm.org/code-of-ethics
[3] 5V 15A 75W Power Supply 100V-240V or 110V - 220V AC to DC Adapter 5V 15 amp Switching Converter 5.5x2.1mm Plug for WS2811 WS2812B WS2813 2801 LED Strip Pixel Lights. Retrieved October 25, 2020, from https://www.amazon.com/gp/product/B07K9Q4DV1/ref=ppx_yo_dt_b_asin_image_o04_s00?ie=UTF8&psc=1
[4] How to "Multithread" an Arduino (Protothreading Tutorial). Retrieved December 8, 2020, from https://create.arduino.cc/projecthub/reanimationxp/how-to-multithread-an-arduino-protothreading-tutorial-dd2c37
[5] DS3231 Datasheet(2015), Retrieved November 2nd, 2020 from https://datasheets.maximintegrated.com/en/ds/DS3231.pdf
[6] Transistor 2SK3703 Datasheet, Retrieved December 08, 2020 from https://alltransistors.com/pdfview.php?doc=2sk3703.pdf&dire=_sanyo
[7] Adafruit LED backpack Datasheet (2020, June 19) Retrieved November 02, 2020: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-led-backpack.pdf?timestamp=1604359487
[8] Earl, B. (n.d.). All About Arduino Libraries. Retrieved November 03, 2020, from https://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use
[9] ZHO-0420S-05A4.5 SPECIFICATION.pdf (2015, 05) Retrieved October 01, 2020: https://cdn.sparkfun.com/datasheets/Robotics/ZHO-0420S-05A4.5%20SPECIFICATION.pdf
[10] Servo motor SG90 Data Sheet(2016, 06) Retrieved October 01, 2020: http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf
[11] MCP23017/MCP23S17 Data Sheet (2016, 07) Retrieved October 01, 2020: https://ww1.microchip.com/downloads/en/DeviceDoc/20001952C.pdf
[12] PCA9685 16-channel, 12-bit PWM Fm+ I2C-bus LED controller(2015, 04) Retrieved October 01, 2020 from https://cdn-shop.adafruit.com/datasheets/PCA9685.pdf
[13] "ATmega328," ATmega328 - 8-bit AVR Microcontrollers. [Online]. Available: https://www.microchip.com/wwwproducts/en/ATmega328. [Accessed: 10-Dec-2020].

8. Appendix

8.1 Requirements and Verification Tables

Requirements	Verification
The actuators should be hanged precisely and stably fixed in the correct place above the guitar neck. The distance between adjacent actuators is **6.64mm ± 0.2 mm** (the distance of adjacent strings on a guitar). The unit is fixed **8.0mm ± 0.2mm** above the guitar neck. The unit should not move after functioning.	A. After actuators are assembled onto the frame box, use a Vernier caliper to measure the distance between each adjacent actuator is **6.64mm ± 0.2 mm.** B. Fix the frame box onto the guitar neck. Measure the distance between the unit and guitar neck (**8.0mm ± 0.2mm**). C. Activate each actuator sequentially 100 times, perform A. and B. again to verify that the unit has not moved.
Most people can easily tell the discrepancy between theoretical tone and real tone when they are different by a semitone. So we are limiting this discrepancy to be less than half of a semitone to ensure every tone sounds correct. Typically, the range of a semitone is about **16Hz**. Therefore, our tone discrepancy should be within **4Hz**.	A. Activate one actuator and pluck at the corresponding string. Measure the produced sound with a tuner. Verify that the sound is within **±4Hz** of the standard note. B. Perform A. for all 36 actuators to verify that the unit plays notes in tune.
The 36 actuators should be able to work individually without breaking the consistency of music. It is reasonable to require the function period on the same actuators should take less than **0.5s±0.05s.**	A. Use a slow-motion camera to film several function cycles of the actuators and calculate the mean of time taken for push and release actions of the actuator. B. Perform A. for all 35 actuators to verify that actuators and press the string on time.

Table 4. R&V for the Fret Pressing Unit

Requirements	Verification

The servos should be working and attached firmly above the strings on the guitar body. The displacement of each actuator after a strike is less than **0.2mm**. The tips of the guitar picks should be **0.5mm ± 0.2 mm** below the string when strumming.	A. Fix the servos on the guitar. Verify that the guitar pick tips can touch the string. B. Activate each servo to pluck at the string sequentially for 100 times, perform A. again to verify that the unit has not moved.
Each servo should respond fast and correctly (less than **0.2s ± 0.05s**) to the signal of rotation. Up to six servos can be triggered by the same signal at the same time (less than **0.1s ± 0.02s** time discrepancy)	C. Use a slow-motion camera to film several function cycles of the servos and verify that the servo can respond within **0.2s** after the signal is given. D. Use a slow-motion camera to film six servos triggered with the same signal. Verify that the time lapse between the servos is less than **0.1s.**
The sound played should be loud enough (greater than **60 dB**)	Let the servos pluck at the strings, measure the volume of the sound with a sound meter. Verify that the sound is greater than **60 dB**.

Table 5. R&V for the String Strumming Unit

Requirements	Verification
When the fret pressing unit, string strumming unit, drumbeat unit, and alarm unit is correctly figured, the produced music should be correct and consistent. And the ATMega should operate at **16 MHz.** To ensure the consistency of music pieces, there should be less than **0.1s** delay between signals when activating the same piece, such as actuators and servos.	The correct behavior of this unit is verified by all the other units performing correctly. Not separate verification needed for the Control Unit.

Table 6. R&V for the Control Unit

Requirements	Verification
After pressing Button 1, the LED should respond in 500ms and change mode.	Connect to Arduino and print timestamp to the serial port every time it receives and responds to a push signal. Press the button for 10 seconds, calculate the time interval through the time stamp.

Button 2 and 3 should change nothing in Mode 0. In Mode 1 and 2, Button 2 and 3 should add or subtract the number by 1 every 500ms.	Connect to Arduino and print time stamp every time to the serial port every time it receives and responds to a push signal. Press the button for 10 seconds, calculate the time interval through the time stamp, and the number should add by 20 or minus by 20.
The alarm will go off and date once it reaches the set hour and minute. The difference should be no bigger than one second.	A. Set the time for one minute later by date of the month and wait for it to start the guitar at the exact time. B. Set the time for one minute later by day of the week and wait for it to start the guitar at the exact time. C. Set the time for 10 hours later by date of the month and wait for it to start the guitar at the exact time.

Table 7. R&V for the Alarm Unit

Requirements	Verification
The tempo of the drum beats should be correct (tempo discrepancy **± 5%** of time gap).	Record the tempo as an audio file and analyze the time difference between each peak. The drum beats should be at the correct frequency.
Should be able to provide two kinds of drumbeat (hit on the surface of the guitar & hit on the side of the guitar). The two kinds of drumbeat should be clear enough to tell from each other. Both sounds should reach at least **70 dB**.	A. Hit the guitar with this unit on both guitar body and guitar side. Verify that the produced sound is different B. Measure several drum beats with a sound meter to verify that the volume of the sounds are greater than **70 dB**.

Table 8. R&V for the Drumbeat Unit

Requirements	Verification
The Power Supply Unit must be able to supply a voltage of **5V±0.2V** for a current load up to **10A**.	A. Link Fret Pressing Unit to PCB and use test points to confirm that **5V±0.2V** voltage and sufficient current is supplied (**1.1A±0.1A** more for every actuator activated).

	B. Link String Pressing Unit to PCB and use test points to confirm that **5V±0.2V** voltage is supplied and can play at least 70dB when playing. C. Link Arduino Mega to PCB and confirm that 5V output pin can output **5V±0.2V** voltage D. Link Alarm Unit to PCB and confirm that **5V±0.2V** voltage is supplied and **1mA±0.1mA** current is supplied E. Link Drum Beat Unit to PCB and confirm that **5V±0.2V** voltage is supplied and **360mA±5mA** is supplied.
The Power Supply Unit must be able to convert 120AV voltage to 5V DC.	Link to PCB and use test point to confirm the output voltage is **5V±0.2V**

Table 9. R&V for the Power Supply Unit