Group 20: James Braza, Henry Russell, Quentin Torgerson 4/29/2016

Mechatronics Final Lab Documentation

Automated Custom Slide Whistle

Report Requirements:

- A *single* document that describes the system so that it can be reasonably reproduced by a classmate. The document must include
    - i) brief description of the device and its operation (enough to enable someone to reproduce the system)
        - drawing(s)
        - sketch(es)
        - labeled photo(s) of the device (a CAD model is not necessary, but is helpful)
    - ii) drawings of developed circuitry (hand-drawn is sufficient) with enough detail for the reader to recreate the circuit
    - iii) bill of materials, including part numbers and vendors of acquired parts.
- Code files (e.g. .c or .asm file(s))

Part 1: Description of the device and its operation

Figure 1. The Slide Whistle

The musical instrument that was played was a slide whistle. The slide whistle behaved similar to store-bought cheap slide whistles, featuring a moveable air stop and a homemade fipple for creating sound. A brushless DC ducted fan was used to blow air into the slide whistle to make noise. The fipple is the slotted portion of the flute that creates the notes. A stepper motor used a

T5 belt and pulley system to move the slide in and out of the whistle, changing the pitch produced. The slide was mounted to a steel guide rod and constrained by linear roller bushings. A brushed DC motor with a 3D printed lever on the end covered the fipple hole to control sound production. An 8x8 LED matrix display was used to signal when notes were being played, as well as to display an FFT representation of the ambient noise. The FFT representation took the form of a graphic equalizer. As an added bonus, a brushed DC gearmotor powered a dynamic shark mechanism to dance when notes were played.

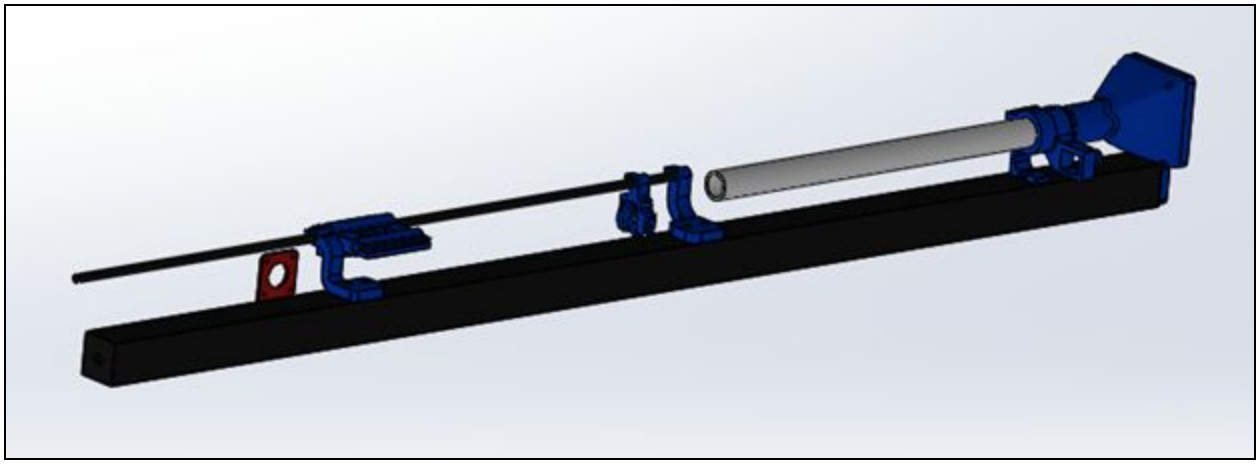


Figure 2. Overall Assembly of the Slide Whistle in CAD

A quick note on Figure 2 above. The black parts came completely fabricated. The blue parts were 3D printed using PLA on standard Makerbot 5th Generation settings. The red parts were laser cut out of ¼" acrylic. The white part is the PVC tube used as the whistle body.

Frame

The frame for the project was a single piece of 40mm wide aluminum extrusion. This made design extremely flexible, allowing quick iterations. All components were mounted into the slots of the extrusion using standard fasteners. The aluminum extrusion was also very rugged, allowing the design to be very robust and repeatable, removing variance between tests that could occur if the frame was less stable and allowed unwanted motion between components.

Whistle

The whistle was made out of an 18" length of ¾" PVC pipe. It was constructed based on the Flutopedia tutorial for the Roura flute (link). Cutting the fipple consisted of using an end mill to properly cut slots in the end of the PVC pipe based on the specified dimensions, then using a small very sharp chisel to finish shaping the openings as shown in Figure 3

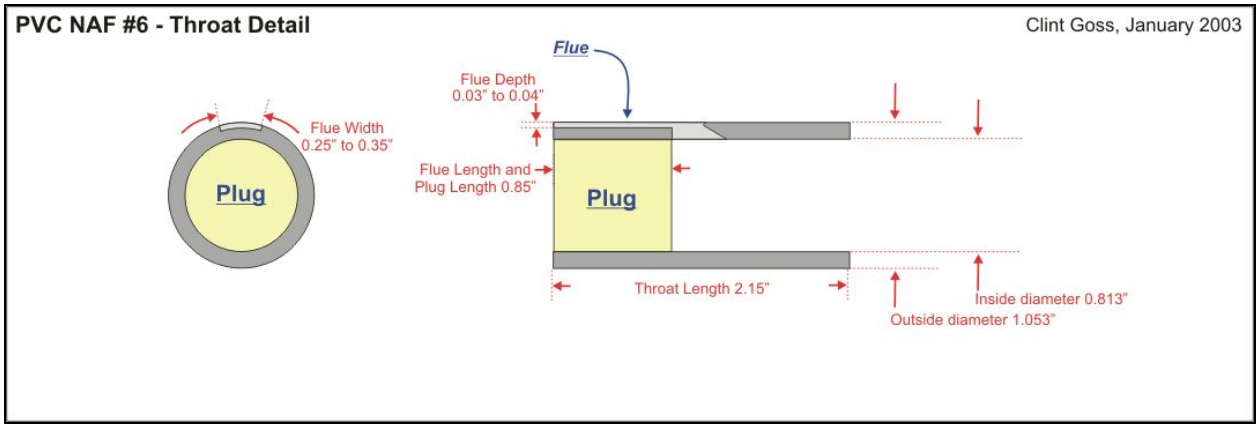


Figure 3. Dimensions used to cut fipple in PVC slide whistle.

The slide consisted of a ⅜" diameter plastic tube with a cork glued to the end. The cork would slide up and down inside of the whistle body in order to change the pitch being played. To further increase the quality of the seal between the cork and the PVC, we used some spray grease and hot glue to better seal the cork. This plastic rod was attached to the drive belt and powered by the stepper that will be discussed later.

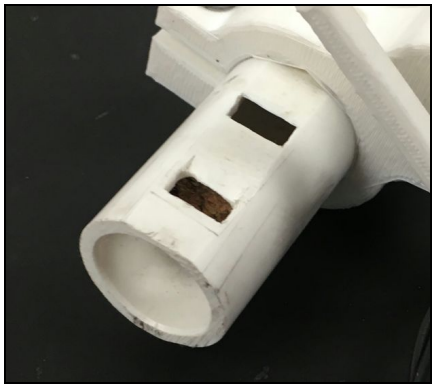


Figure 4. The Fipple

DC Brushless Fan

The DC brushless fan was much larger in diameter than the whistle that it was meant to play, so some sort of reducer needed to be used to efficiently funnel the airflow into the whistle. A reducer was designed in SolidWorks and then 3D printed that exactly matched the dimensions of the flute and the fan, allowing an efficient conversion as shown in Figure 5. The reducer attached to the end of the aluminum extrude mount. The fan mounted onto the back of the reducer using 8-32 screws. The extrude highlighted in the black box was crucial to the fipple, as it constrained the air to maximize sound generation.

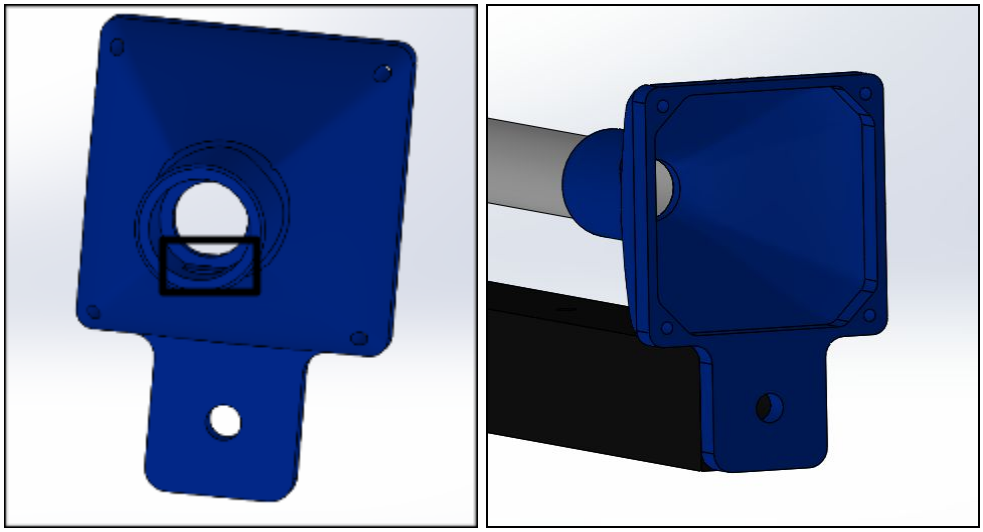


Figure 5. Fan reducer funnelled airflow into flute.

The fan was chosen so that it was capable of playing the flute. A paper (link) written about the pressure required to play different wind instruments was referenced to determine the amount of pressure that would be required by the DC fan to properly play the flute. The paper stated that 2000 Pascals was the maximum pressure required to play a flute. To determine the flow rate required to play the slide whistle, we completely filled our lungs, then played a well sustained note for as long as we could, which turned out to be about 5 seconds. We combined this information, with research data about average human lung capacity, and determined that the flow rate was about 1 liter/second. We then used this flow rate and pressure data, and started searching Digikey for a fan that was capable of sustaining our desired flow rate and pressure difference. Figure 6 shows the pressure vs. flow rate diagram of the fan that we chose.

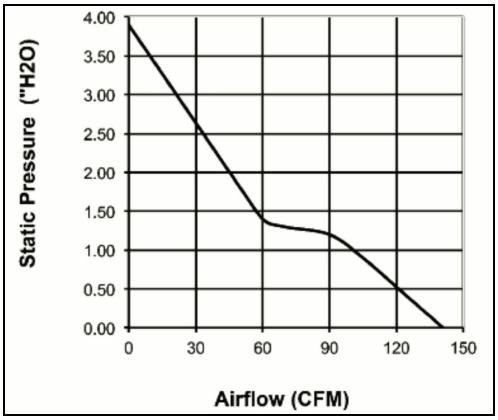


Figure 6. Pressure vs. Airflow diagram for 1053-1622-ND fan

Linear Slide

The linear slide clamped onto a carriage that was guided by two LM8UU linear bearings that slid along a hardened steel shaft. The shaft was held onto the aluminum extrusion using 3D printed brackets. As the carriage moved along the shaft, it the slide and cork stopper changed the acoustic length of the flute, adjusting its pitch. The slide is pictured below in Figure 7.

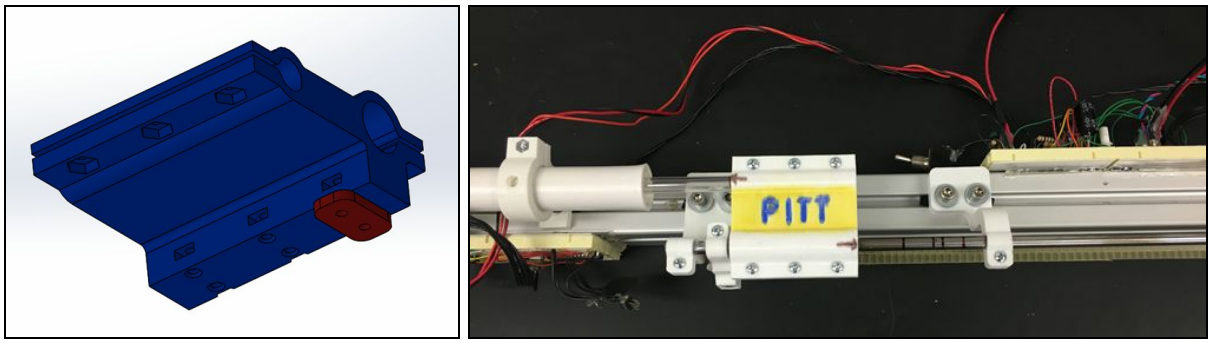


Figure 7. Linear slide used to change whistle pitch.

The slide was pushed back and forth using a T5 timing belt powered by a pulley attached to a NEMA 17 high torque stepper motor. The stepper motor is further explained below.

Stepper Motor

The stepper motor was used to move the slide back and forth. Choosing the stepper motor required several assumptions about the design, such as friction and mass, because we had to choose the motor before beginning construction. To choose the motor, we made an educated guess that our slide would be about 0.2kg, and have a friction force of 15N that will be required to move it. Based on the part we were assigned, we decided that we should be able to switch between notes at 0.1 second intervals. Based on our maximum distance of travel, and the standard kinematic motion equations, we determined that we would need to accelerate at 30m/s/s in order to achieve these specifications. The total force required was 21N, which translates to 30N-cm of torque required at our top speed based on our pulley diameter. The top speed of 3m/s for the slide translates to 2728 steps/s from our stepper, which means the coils are only active for 3.66ms at a time. The equations for reduction in torque with velocity for a stepper motor are based on step duration, coil resistance, and coil inductance (link). The motor will only have the coils active for two time constants, therefore only 86% of the rated holding torque will be available at the top velocity. After searching on Sparkfun.com, a stepper with 48N-cm of holding torque and 2.2N-cm of detent torque was discovered. After subtracting the detent torque from the holding torque, and then multiplying by 0.86, it was found that the motor would still have 41.3N-cm of torque available at the top speed. This is higher than the required torque of 30N-cm, making it a suitable motor. The next smallest motor available at Sparkfun.com was far

too small, with a holding torque of only 23N-cm. Adafruit.com did not have steppers that met the torque specifications, and Digikey's steppers were more expensive, and therefore ignored.

A DRV8834 stepper motor driver was used to drive the stepper motors because it was capable of driving the chosen stepper at the specified current and voltage. The driver can be seen below in Figure 8. A stepper driver was used instead of a standard H-bridge because the stepper driver was capable of adjusting output current, and therefore holding torque. The stepper driver also makes it very easy to incorporate microstepping by simply pulling certain setup pins high or low.

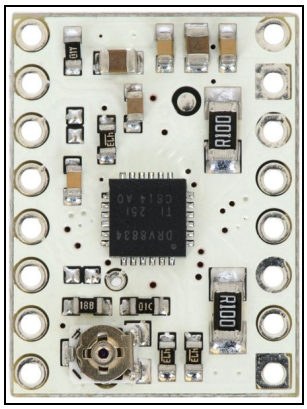


Figure 8. Top View of the Stepper Driver

Note Stopping Lever

In order to play music well, the instrument must start and stop notes quickly. At first, stopping the airflow was a strategy that was looked at, but proved to be difficult and unreliable because the fan took a long time to start spinning, and any indirect method of turning a valve or pinching a tube containing the airflow was over complicated and less reliable. In the end, the airflow remained continuous, and the sound was interrupted by covering the fipple, or sound producing opening of the flute, using a small plastic lever. As this lever covered the fipple, it stopped the oscillations in the air, therefore stopping the noise. This lever was attached to a very small DC motor that was ungeared and allowed to run at stall conditions. The maximum current through the motor did not exceed 500mA at stall, so our stacked H-bridge motor drivers were capable of driving them at stall conditions for sustained durations. To ensure that our H-bridge did not overheat and shut off during operation, two H-bridge chips were stacked and soldered to each other. This is shown below in Figure 9. By stacking them, each bridge takes about half of the current and thus proportionally heats up about half as much.

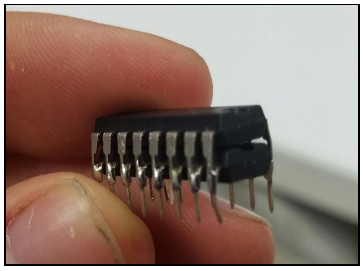


Figure 9. Stacked H-bridges

The added stress of running the DC motor at stall condition did not shorten the life of the motor enough to affect the project, so it was also deemed acceptable. Having a lightweight lever directly attached to the motor shaft allowed it to be turned on and off nearly instantly, making timing of notes much easier. Figure 10 shows an image of the flipper in place and then lifted, allowing the notes to be played.

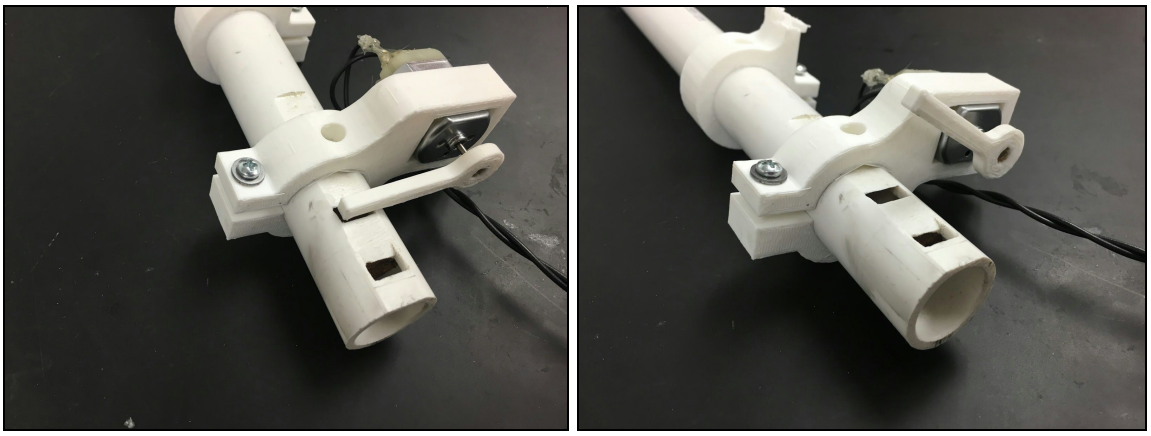


Figure 10. Note stopping lever stopping sound in the left image and enabling it in the right

Microphone

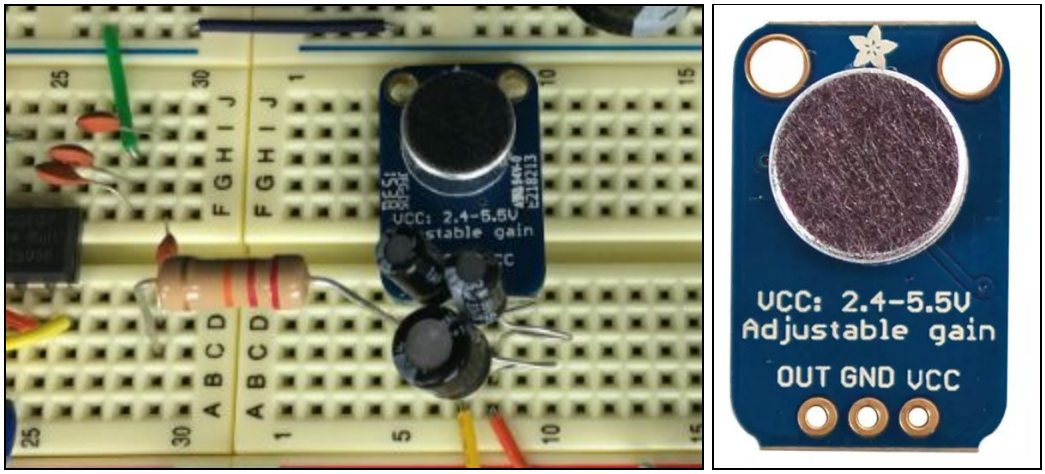


Figure 11. The microphone with filtering capacitors

The microphone shown above was used to interpret the sounds from the surrounding environment. Shown above in Figure 11, the microphone's pinout is simple. It needed to be connected to Vcc and ground, with the output line containing the sound signal. An assortment of capacitors were placed between the Vcc and ground lines going to the microphone to ensure that it received a power signal that would be smooth at all frequencies. The output signal took the form of a variable voltage, which we fed into the FFT chip as described below. The microphone had a trimmer pot on the back which enabled us to adjust its gain. The expected value was tuned based on trial and error once the graphic equalizer was functioning, as described below.

LED Matrix Display

An 8x8 LED display was used to visually indicate when notes were being started. The display also showed a real-time FFT analysis of the ambient noise based on the data collected with a microphone. The FFT analysis was done using a MSGEQ7 chip, which communicated the levels of seven different frequency ranges to the microcontroller.

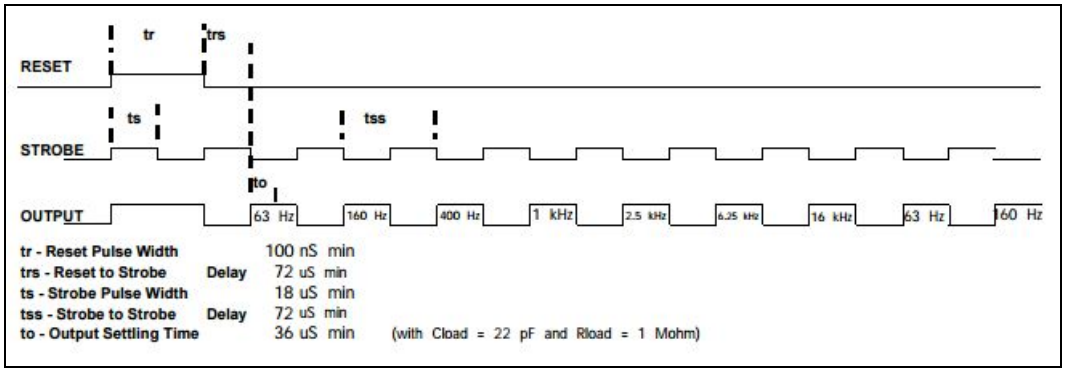


Figure 12. The communication protocol used by the MSGEQ7 chip

The microcontroller interpreted this information following the guidelines specified by Figure 12 above. This interpretation is further explained in our pseudocode in part 4 below. The microcontroller then decided how it should be displayed on the LED matrix. It then communicated this information to the MAX7219CNG which multiplexes the power signals to the LEDs in the display. The eighth row of LED's on the display was either fully on or off, indicating when a note was supposed to be played. Figure 13 below shows the LED matrix functioning. The row in the blue box is the signal LED's indicating the whistle is producing sound.

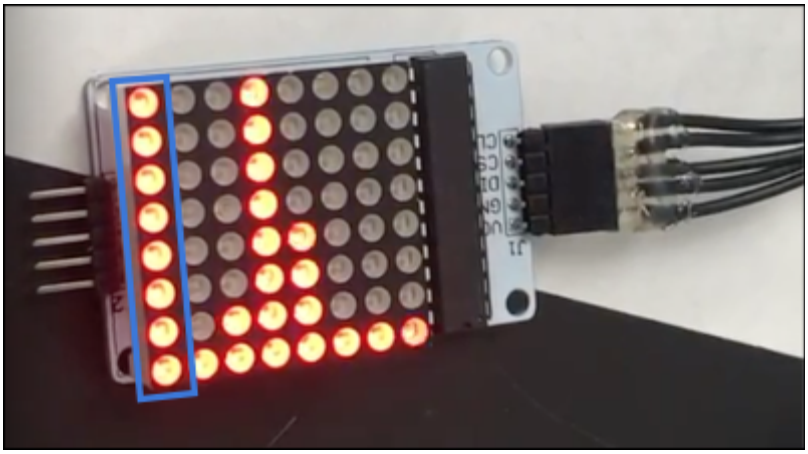


Figure 13. Real time FFT analysis displayed on the 8x8 LED matrix.

Dancing Shark

The slide whistle featured a mobile component that visually represented when the slide whistle was producing sound. The mobile component was in honor of left shark during Katy Perry's 2015 super bowl halftime performance (link). The component works using a simulated crankshaft, shown below in Figure 14. The crankshaft was hooked up directly inline with a DC brushed motor, controlled by the opposite side of the H-bridge. When the motor spun, it rotated the crankshaft, which moved the flags up and down vertically.

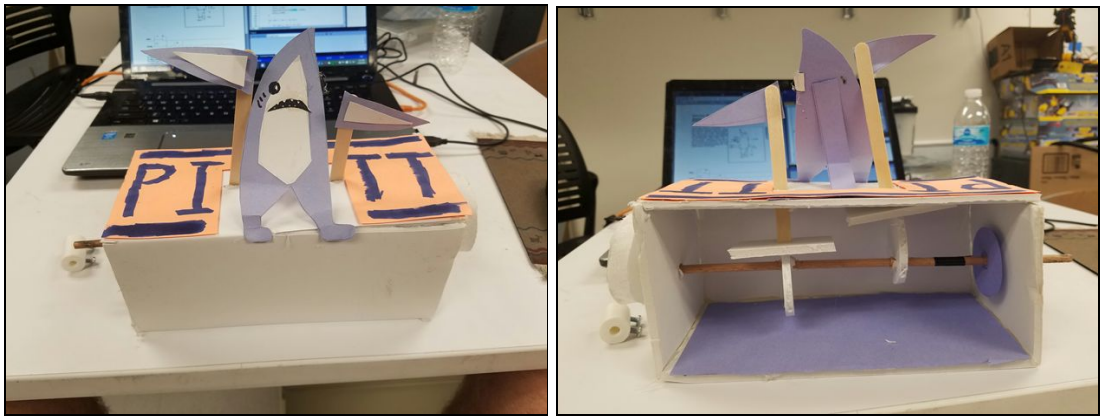


Figure 14. Dancing Shark Mechanism

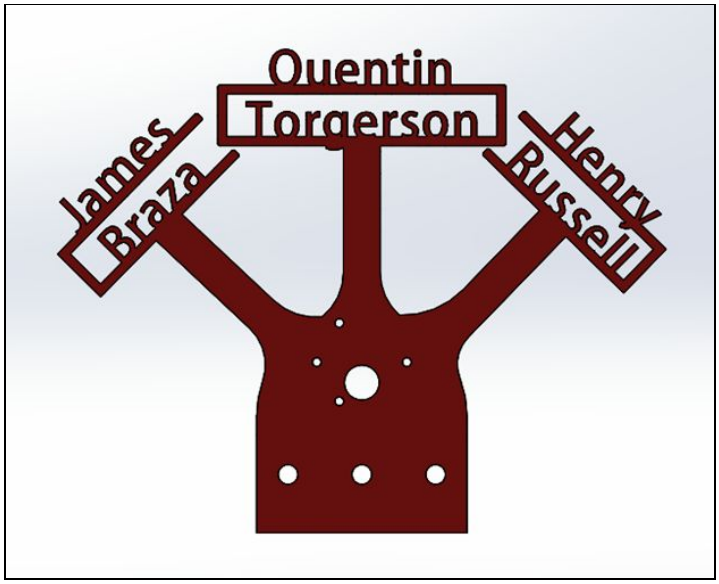
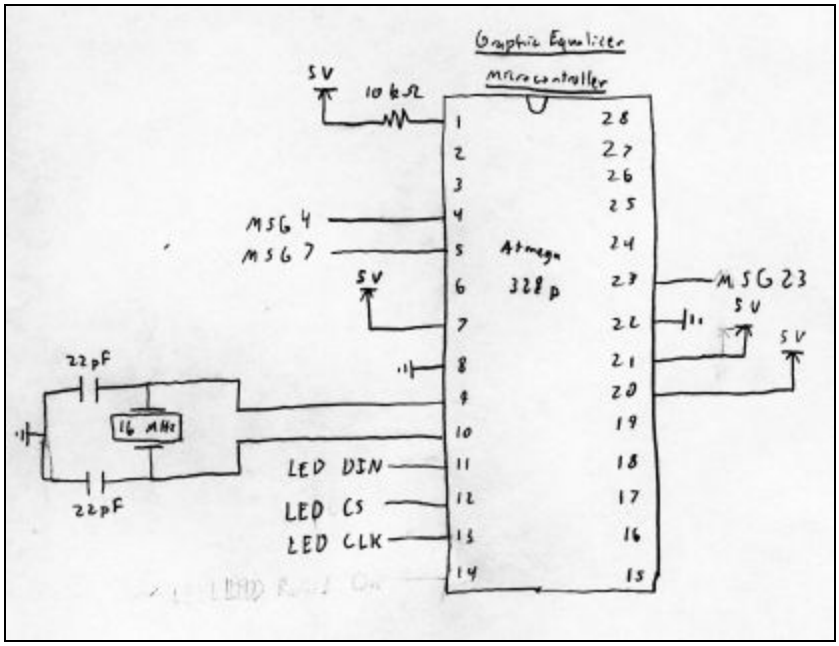


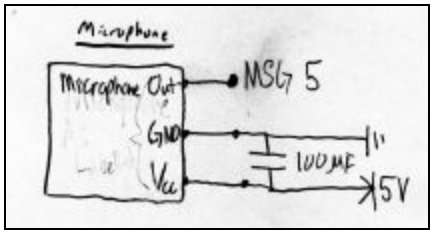
Figure 15. Dancing mechanism motor bracket

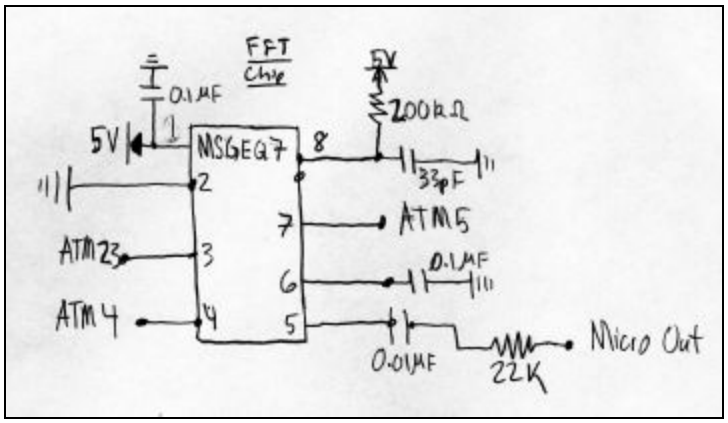
Figure 15 above is of the motor bracket that was laser cut to hold the shark motor. It was one last finishing touch on the project. This was a great project to end the year on, and it was a ton of fun learning how to connect everything up and get it working.

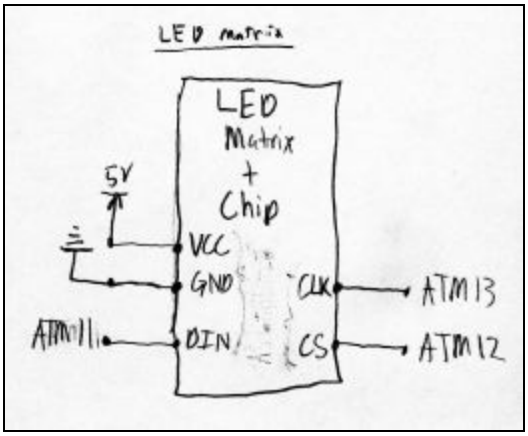
Part 2: Circuit Diagram

Microphone Circuit

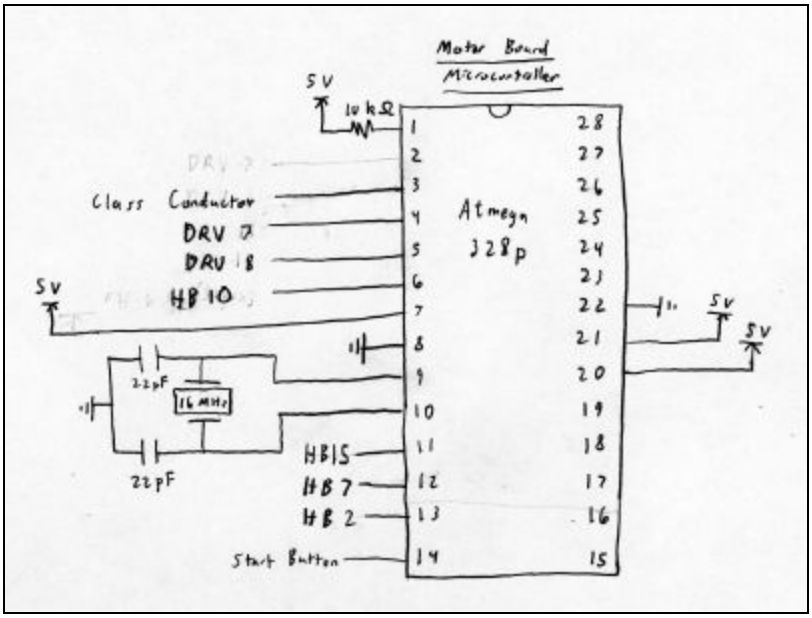

Graphic Equalizer
Microcontroller
5V
10 kΩ
MSG 4
MSG 7
Atmega
328 p
MSG 23
5V
22pF
16 MHz
22pF
LED DIN
LED CS
LED CLK
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

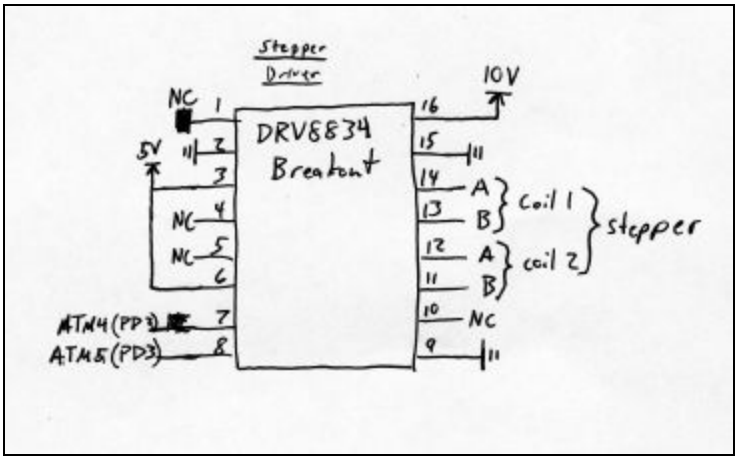

Microphone
Microphone Out
MSG 5
GND
Vcc
100μF
5V

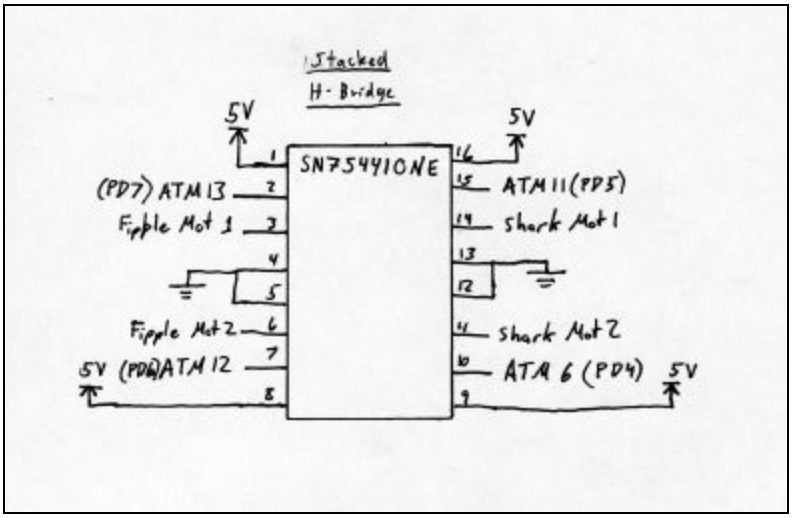

FFT
Chip
0.1μF
5V
200kΩ
MSGEQ7
33pF
ATM5
0.1μF
ATM23
ATM4
0.01μF
22K
Micro Out
1
2
3
4
5
6
7
8



Motor Circuit







Part 3: Bill of Materials

Item Name	Item Description	Vendor	Vendor PN	Quantity	Unit Price
Microcontroller	Atmel Atmega328P Microcontroller	Class Materials	ATMEGA 328P	2	
Crystal	16 MHz Crystal Oscillator	Class Materials	FOXSDLF/160R-20	2	
H Bridge	SN754410 Quadruple Half-H Driver (Rev. C)	Class Materials	SN754410	2	
Breadboards	Bread Board 400 Point Solderless Breadboard	Class Materials		4	
Chip	Graphic Equalizer Display Filter - FFT Chip	Sparkfun	MSGEQ7	1	4.95
Fan	FAN AXIAL 80X38MM 12VDC WIRE	Digikey	1053-1622-ND	1	21.78
Microphone	AMP MICROPHONE ADJUSTABLE	Digikey	1528-1013-ND	2	6.95

LED Display	LED MATRIX 8X8 BICL I2C BACKPACK	Digikey	1528-1167-ND	1	15.95
Stepper Driver	DRV8834 Low-Voltage Stepper Motor Driver Carrier	Pololu	2134	1	
Stepper Motor	Stepper Motor - 68 oz.in (400 steps/rev)	Sparkfun	ROB-10846	1	16.95
Cork	Trade-Sized Tapered Round Plug Compressible Cork, Size 7, Fits 13/16" Large End ID	McMaster Carr	9566K21	1	13.02
Whistle Tube	3/4" PVC Pipe	Robotics Club		1	
Resistors	Various resistors	Robotics Club		-	
Capacitors	Various ceramic capacitors	Robotics Club		-	
Wires	Various length breadboarding wires and custom made power supply wires	Robotics Club		-	
Buttons	Various buttons	Robotics Club		2	
3D Printed Parts	Various 3D printed structural components	Robotics Club		-	
Acrylic Parts	Various acrylic laser cut structural components	Robotics Club		-	
Set Screw	6-32 set screw T5 timing pulley	Robotics Club		1	
Motor	7.2VDC 175RPM 99.04oz-in GHM-04 Spur Gear Head Motor (w/ Rear Shaft)	Robotics Club		1	

Motor	Small DC motor 6/9V	Robotics Club	TFK-280SA-22125	1	
Plastic Tube	Small plastic tube that the stepper motor moved to actuate the cork in the whistle	Robotics Club		1	
Dancing Shark	Paper shark with moving flags that work via a camshaft	Robotics Club		1	
Power Supply	Benchtop Power Supply for Stepper: 10V at 1.7A	Robotics Club		1	
Power Supply	Benchtop Power Supply for Various Electronics: 5V at 5A	Robotics Club		1	
Power Supply	Benchtop Power Supply for Fan: 12V at 5.8A	SERC		1	
Timing Belt	T5 timing belt and pulley	Quentin Torgerson		1	
Steel Rod	Linear rod for support slide	Quentin Torgerson		1	
Aluminum Extrude	40 mm x 40 mm Aluminum Extrude	Quentin Torgerson		1	
Fasteners	Standard fasteners: washers, nuts, and bolts for the 40 mm Aluminum extrude	Quentin Torgerson		-	
Linear Bearings	Linear bearings used to reduce friction in the timing belt's slave pulley	Quentin Torgerson		2	
Linear Bearings	Linear bearings used to reduce friction when sliding along linear rod	Quentin Torgerson	LM8UU Linear Bearing	2	

Part 4: Code

All code was done in Atmel Studio v7.0 in a C/C++ Executable Project. The code was a combination of code snippets from past labs and original code. The code files were uploaded into two separate ATMEGA328P chips that controlled their two respective breadboards on a mostly independent basis. The code follows the below general structure:

General Main Loop: Plays our part referencing an array of functions

Takes place in the first Atmega chip

1. A time-killing while loop that waits for a pin high. This signifies to start playing the music segment. The signal can come from an:
    a. An external controller
    b. A button press
2. Delay 2 seconds
3. Drive the dancing shark motor
    a. The shark will start dancing
4. Move slide to position corresponding to specified note
    a. Further explained in 'Slide to Position' code below
5. Enable sound production
    a. Move the fipple blocker tab low
6. Send a digital high to other controller to enable 8th row of indicator LED's to turn on
    a. Further explained in 'Graphic Equalizer' code below
7. Delay the amount of time the note will be sustained
8. Send a digital low to other controller to enable 8th row of indicator LED's to turn off
9. Disable sound production
    a. Move the fipple blocker tab high
10. Repeat steps 4 - 9 for each note in music segment
11. Stop driving the dancing shark motor
    a. The shark will stop dancing
12. Delay a half second to ensure the note has stopped
13. Return stepper to home position
    a. Home position is automatically assigned to the slide's initial position
    b. Home position is specified to be the position of the slide at the far left
        i. The slide needed to be manually reset to this position before the start signal

Slide to Position: Brings the slide to the position corresponding to a given note

Takes place in the first Atmega chip

1. Each note has a corresponding character. Send the function noteLookup a character corresponding to desired note
    a. The below table corresponds to the lookup table used

Desired Note	Name in Code	Position (mm)
Low A#	w	32.37
Low D	d	50.02
F	F	0
G	G	14.5
A	A	26.84
High A#	a	32.37
B	B	37.12
D	D	50.02

2. Scale the linear position to the amount of steps the stepper motor needs to turn
    a. Return this value
3. Send this value to the updateStepper function
    a. Decide if based on the current step count if the stepper needs to rotate clockwise or counterclockwise
4. Accelerate the stepper to the desired step count
5. Exit the function

Graphic Equalizer: Analyzes sound production and displays it on the LED Matrix

Takes place in the second Atmega chip

1. Toggles the set/reset pin of the FFT chip high then low
    a. This tells it to begin interpreting the mono sound signal coming in from the microphone
2. Delays 0.1ms according to the reset pulse width
3. Send strobe pin low
    a. This tells the FFT chip to output one frequency range
4. Delay .05ms waiting for output
5. Atmega performs a digitalRead in C

6. Scale the data from a range of 256 to a range of 8
    a. This configures it for the LED matrix
7. Delay .05ms
    a. Delays are given by TSS, time delay from strobe - strobe and TS, strobe pulse width
8. Send strobe pin high
    a. This tells the FFT chip to stop outputting the given frequency
9. Delay .04ms
10. Repeat steps 3 - 9 seven times, corresponding to each frequency range analyzed
    a. Bandpass ranges: 63 Hz, 160 Hz, 400 Hz, 1 kHz, 2.5 kHz, 6.25 kHz, 16 kHz
11. Use nested for loops to send FFT data from Atmega to the MAX7219CNG multiplexor
    a. The data is transmitted serially
    b. We utilized a C library pre-made for sending serial data to the LED matrix
12. Loop steps 1 - 11 indefinitely, continually outputting new sound information onto the graphic equalizer

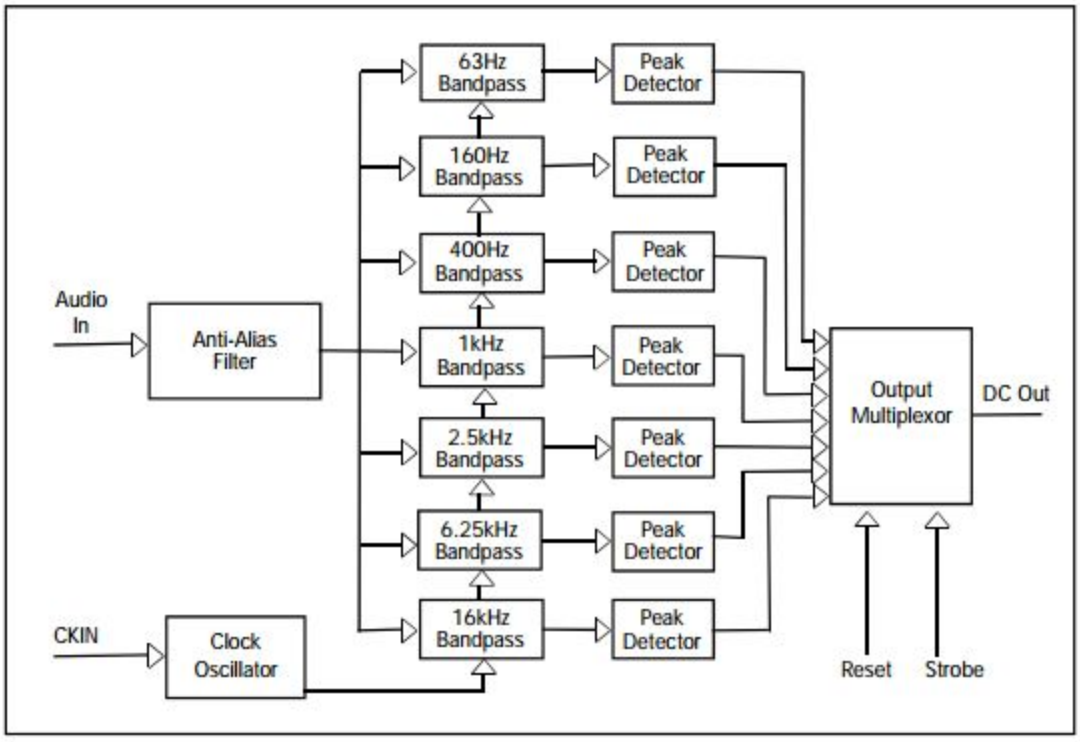


Figure 16. An graphic from the MSGQE7's datasheet on how to interpret the FFT output