

# E155 Final Report: Stepper Motor Music

**Members:** Andrew Q. Pham & Sam Ting

---

## **Abstract**

The team constructed a system that plays music on four stepper motors. Multiple brands of stepper motors were tested for volume and sound quality, and motor mounts were constructed to improve volume and sound quality. The system uses an ATSAM and ESP8266 WiFi module to allow users to request songs through a hosted webpage. The ATSAM parses the request, determines the notes to play from a lookup table, encodes the notes, and sends the encoded notes to the FPGA over SPI. The FPGA decodes the notes and steps the motors at the desired frequency to produce the notes. The final system allows the user to choose from nine distinct songs to play. Additionally, a Python script was developed by the team to help parse MIDI files and generate songs in the team's custom encoding scheme. Future work includes improving the song encoding scheme to produce better sound quality, making the MIDI parsing script more robust and incorporating it into the microcontroller, so a user could input any MIDI file of choice to play.

# Introduction

Inspired by E155 Lab 5 and various online videos [1] [2], the team constructed a system that plays music on four stepper motors using the FPGA, ATSAM, and ESP8266 WiFi module.

The system allows users to select a song to play from a webpage accessed through a computer. The webpage is hosted on the ATSAM, and the computer and the ATSAM communicate through the ESP8266. Song requests are sent via UART from the ESP8266 to the ATSAM, which parses the request and sends the appropriate notes to the FPGA. The FPGA drives I/O pins, which are inputs to the stepper motor h-bridge drivers that drive the motors to play a note.

An overall block diagram of the system and the fully constructed system are shown in Figure 1 and Figure 2 respectively. The schematic for the full system is shown in Appendix B.

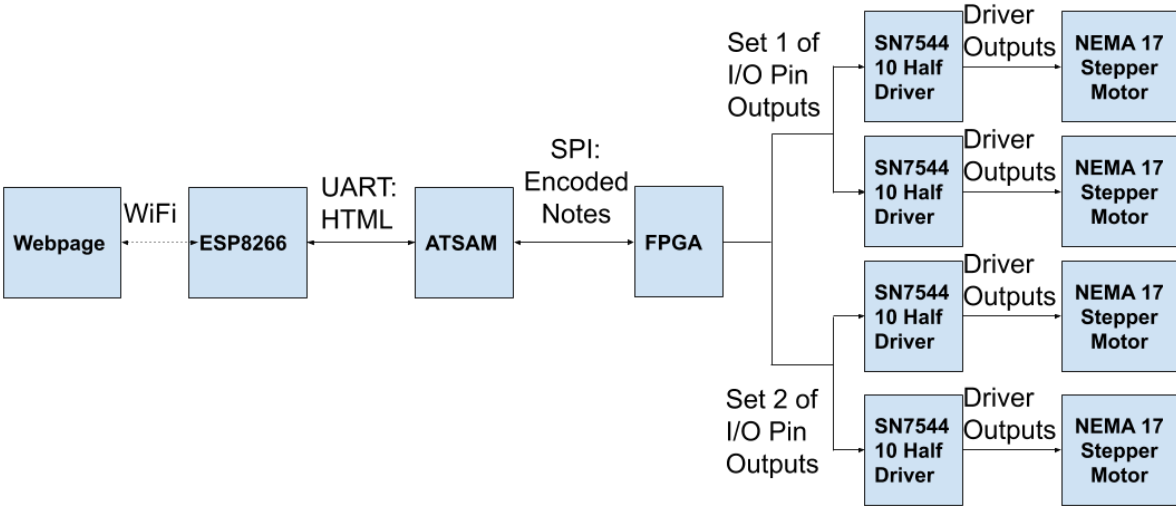


Figure 1. Overall block diagram of full system to play music based on user input from HTML page

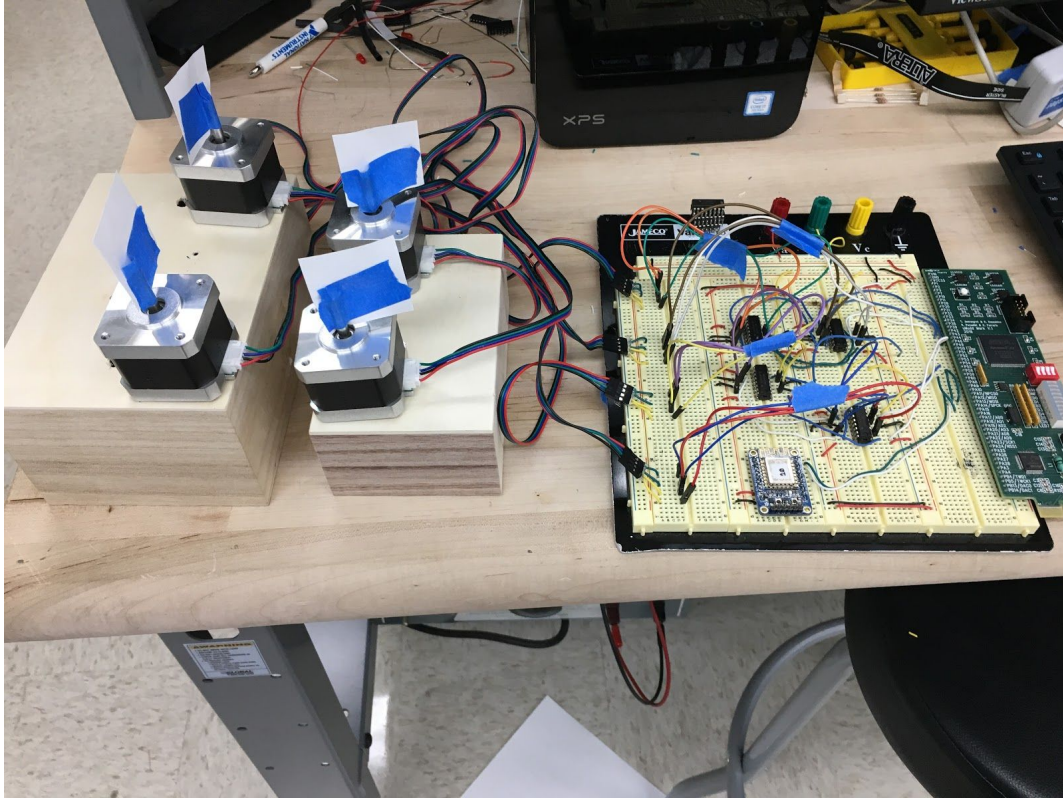


Figure 2. Full system hardware implementation

## Design

### Hardware

#### Motors & Drivers

The team scavenged for stepper motors across Mudd's campus and found five different brands/sizes of motors. The team evaluated each stepper motor for sound quality and required amperage. Two additional types of NEMA17 stepper motors were purchased from Amazon and tested for sound quality. The Usongshine NEMA17 stepper motors [3] were chosen, due to their superior sound quality in comparison to the other motors.

To spin the motors, the team constructed a motor driver circuit for the stepper motors using the SN754410 h-bridge driver. The maximum current rating of the SN754410 is 1 A, while the NEMA17 stepper motors are rated for up to 2 A. To compensate for this, two of the drivers were soldered together in parallel to drive each motor. The circuit schematic of a single motor and driver is shown in Appendix B (the second parallel SN754410 is omitted for diagram clarity).

Given that four motors are being driven, each at a maximum of 1 A continuous current and greater than or equal to 5 V of voltage, a larger power supply was necessary to power the setup. A 50 V, 25 A unregulated power supply was checked out from the stockroom to power the motor drivers. This new power supply is able to output significantly more current than the ones in the Digital Lab. The increase in current increased the output torque of the motors, which increased vibration and volume. The motors were run at ~5 V because at higher voltages the amount of constant current would cause the motor drivers to reach their thermal shutdown. The thermal shutdown would prevent the motors from stepping properly, causing them to vibrate at an undesired frequency. Moreover, the stacked motor drivers could only take a maximum of 2 amps of constant current, so a larger current would cause the drivers to explode.

The stepper motors' frequency ranges were tested by adjusting the slow clock addend value manually in Quartus. The result showed that note volume is significantly lower for notes lower than an E1. For notes higher than a D#4, the stepper motor was unable to step quickly enough to produce the note. Thus, all notes between an E1 and a D#4 were encoded on the ATSAM and were used to create songs. Unfortunately, the motors ordered off Amazon did not have an associated datasheet, so the team was unable to confirm whether this stepping frequency was on par with the expected maximum RPM of the motors.

## Acoustic Hardware Improvements

During the problem presentation, the team presented motor volume as their primary technical problem. This problem was largely absolved by purchasing new motors. However, the motors had very irregular and choppy sound quality, depending on the surface they were fixed to. To solve this problem, two thin wooden boxes were purchased to mount the motors to. This gave the motors a stiff yet thin surface to reverberate against, which produced a much cleaner tone quality. Two boxes were purchased: one 4" x 6" x 3" and one 3" x 5" x 2".

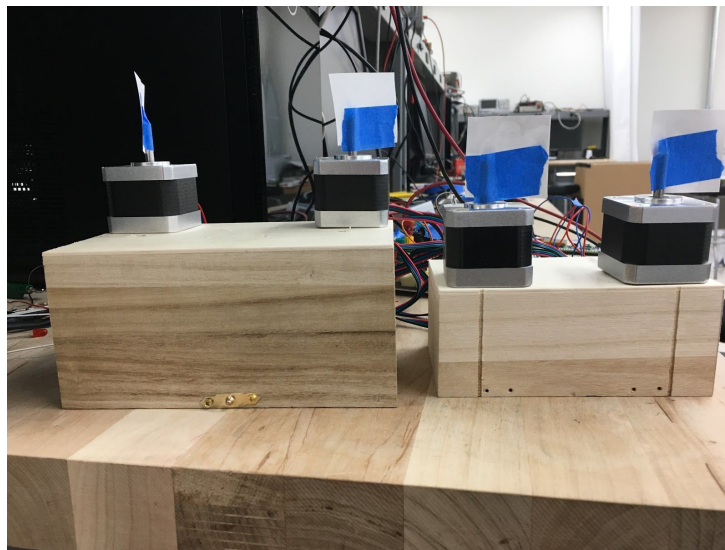


Figure 3. Photo of stepper motors mounted on large and small boxes

The motors were tested/mounted on several different locations across the top of the boxes to test which part of the box would cause the motor to be the loudest. The two box sizes were also tested to see which one produced a louder overall sound

Ultimately, experimental tests showed that the corners of each box caused the loudest sound, so each motor was mounted on a corner of their respective boxes. Moreover, the larger box produced a louder sound. Thus, the motors playing the melody of the song were mounted on the big box, while the background/bass motors were mounted on the smaller box.

## FPGA

The FPGA performs two main tasks: 1) reading in encoded notes to play over SPI from the ATSAM and 2) decoding each encoded note to drive the stepper motors at the correct frequencies. This is depicted in the high-level block diagram in Figure 4.

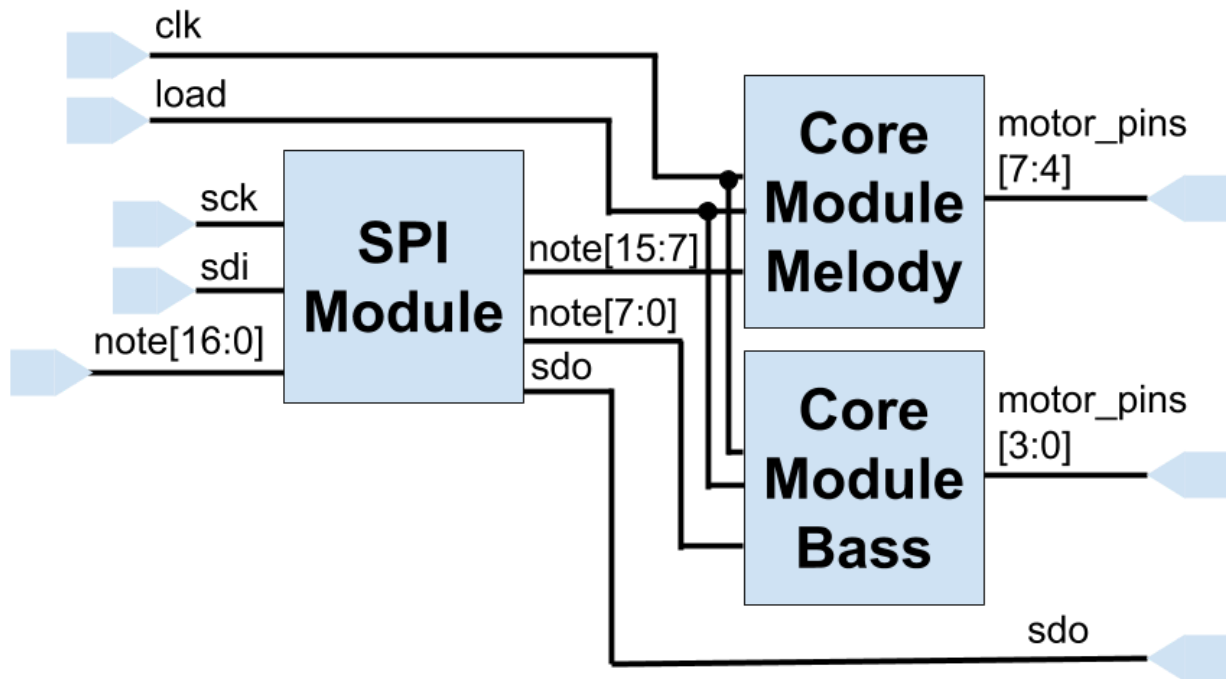


Figure 4. High-level block diagram of the digital hardware on the FPGA

## SPI Communication Note Encoding

For each SPI interaction, the FPGA reads in an 8-bit encoded note. Bits 0-3 encode which note in an octave to play (ex. C), and bits 5-7 encode which octave the note is in. An example SPI interaction is shown in Figure 5. Two SPI interactions occur for each note. The first specifies the note to play on the first and second motors (the melody), and the second specifies the note to

play on the third and fourth motor (the bass part). The octave decoding is discussed further in the description of the decode note module.

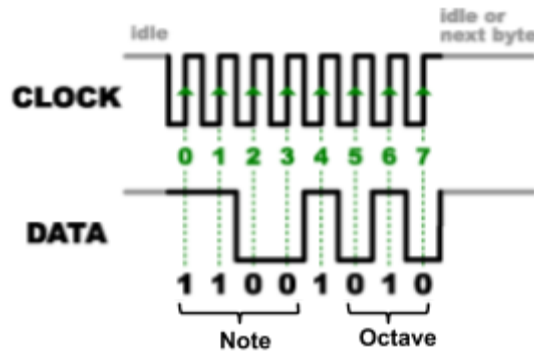


Figure 5. Example SPI Interaction

## Project Core

The “Project Core” module in Verilog decodes the note and steps the motors at the desired frequency to play a note. The digital hardware of the core is shown in Figure 6. It is composed of three main parts: 1) a note decoding module, 2) a  $2^N$  counter to reduce the clock frequency to the desired note frequency, and 3) a state machine to step the motors.

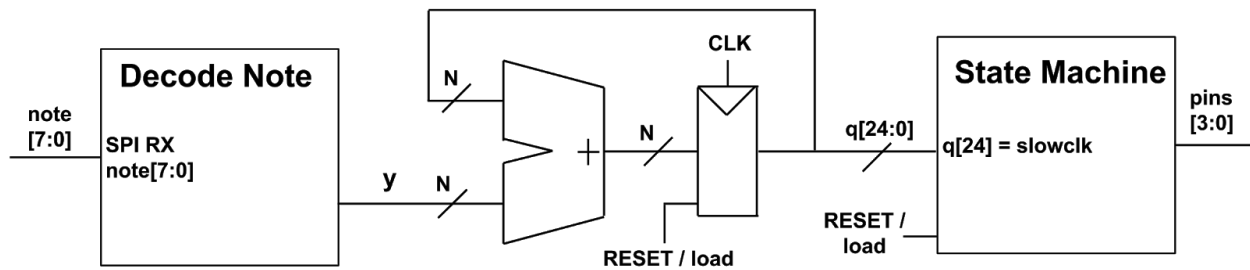


Figure 6. Digital hardware of the core module of the FPGA

## Note Decoding

The decoder module assumes the note is in either octave 2 or 3, based on whether bit 7 of the note input is 0 or 1 respectively. The module then maps it to the number to be added to the  $2^N$  counter to make the MSB of the counter oscillate at the desired note frequency. The spreadsheet of the mapping from note frequency to counter addend is given in Appendix C. The decode note module is implemented using a case statement, so the resulting digital hardware is a large chain of or gates and muxes to account for the 25 different input cases. The RTL viewer of the Decode Note Module is shown in Appendix D.

As described above, bit 7 determines which ‘base octave’ the note will be in; if bit 7 is zero, this indicates octave two -- otherwise the note is in octave three. These two octaves were not sufficient to cover the previously described range of the motor. Thus, bits 5 and 6 were utilized to either drop or raise the note an additional octave. After the decode note module, the addend

for the counter is adjusted based on note[6] and note[5]; the former indicates that the addend should be multiplied by two, while the latter divides the addend by two. As a result, these flags either increase or drop the note by an octave respectively. This additional functionality allows any programmed songs to utilize the entire E1 to D#4 range of the motor.

## Motor Stepping

A finite state machine (FSM) steps the motors. The state machine shown in Figure 7 is used to activate the different coils in the stepper motors in sequence to have the motor step/spin. The output logic, "pins," drives GPIO output pins that facilitate the communication between the FPGA and the stepper motor drivers. The different states of the FSM correspond to the different inputs of the stepper motor drivers being activated/different coils in the motor being activated. The FSM transitions states on the MSB of the  $2^N$  counter.

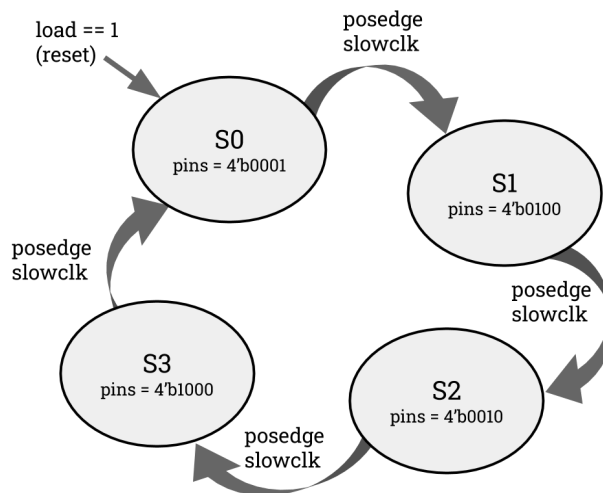


Figure 7. FSM to step the motors

As shown in Figure 7, this FSM only sets one output pin at a time high, based on its state. This is full-step driving, which drives the maximum current of all stepping methods. Half-stepping was also tested, but it was found that the motors were marginally quieter when half-stepping, so the team opted to use full-step operation.

In summary, the FPGA takes in an encoded note over SPI from the ATSAM, decodes the note to set the frequency at which the MSB of the  $2^N$  counter oscillates, and then steps the stepper motors at that frequency using an FSM.

## ATSAM

The ATSAM performs two main tasks: 1) hosting an HTML webpage for song requests and 2) transmitting the notes of the requested song to the FPGA via SPI. The high-level routine of the program is shown in Figure 8. The ATSAM hosts the webpage with an ESP8266, which generates a WiFi hotspot that a user computer can connect to to access the webpage. The

ATSAM parses the HTML song request to determine if a new song has been requested. More details on the HTML webpage and ESP8266 are given in the HTML Page/ESP8266 Section. After determining the song choice, the ATSAM reads the next note for each motor of the chosen song to be played, encodes these notes, sends it over SPI to the FPGA, waits for the note's duration, and then repeats this process until all of the song's notes have been played or a different song is selected.

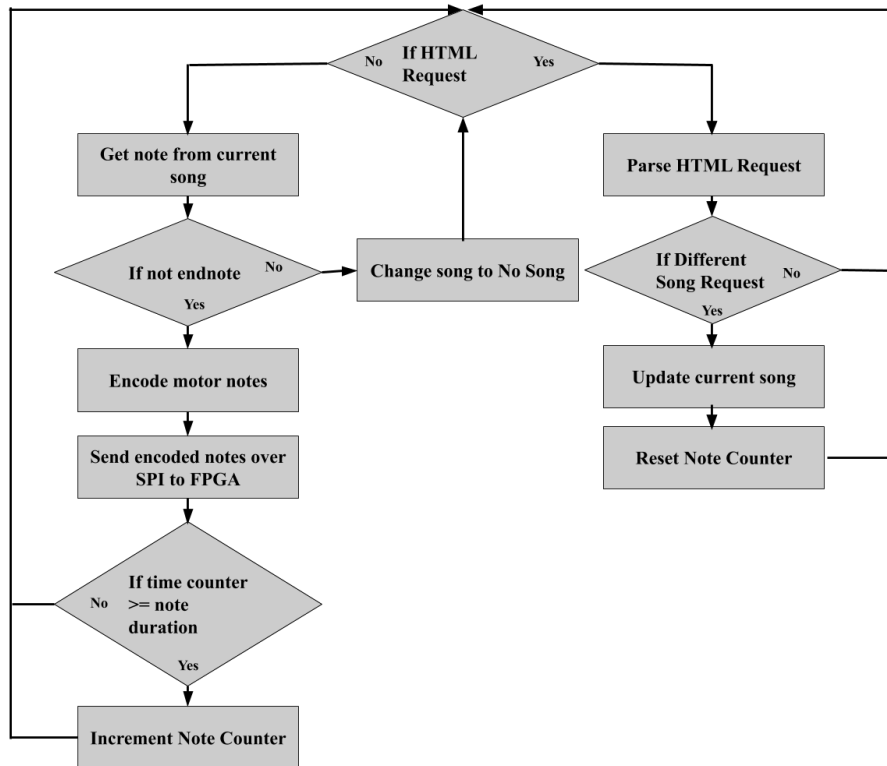


Figure 8. High Level Routines of ATSAM

## Song Encoding

Nine songs were hard-coded in the C code, each of which could be played by pressing a button on the corresponding HTML web page. Songs were encoded with a custom data structure. This structure contains two data types: a float that stores the song's tempo and a two dimensional float array that stores the song's notes to be played on the motors. The array is an Nx2 array, where N is the number of notes in the song, and two is based on the number of distinct parts (melody and bass) of each song. Since the design contains four total motors, each part is played by two motors.

To ensure that the motors were in sync, the notes for each part of each song were divided into sixteenth note increments. For example, each half note in a given song was divided into eight consecutive sixteenth notes of the same frequency. By dividing longer notes into smaller repeated notes, note information for the motors was synchronously sent on every sixteenth note



division. This prevented any timing issues associated with syncing the melody and bass parts of a song.

While this encoding method prevented timing issues, the motors would only play the equivalent number of sixteenth notes instead of a continuous quarter note or longer note. The song encoding could be improved in the future to achieve better sound quality on sustained notes.

## HTML Page/ESP8266

An HTML webpage was designed to allow the user to request one of the encoded songs. An ESP8266 generates a WiFi hotspot that a user's computer can connect to. A user can then press a button on the web page to select a song. A screenshot of the webpage is shown in Figure 9.

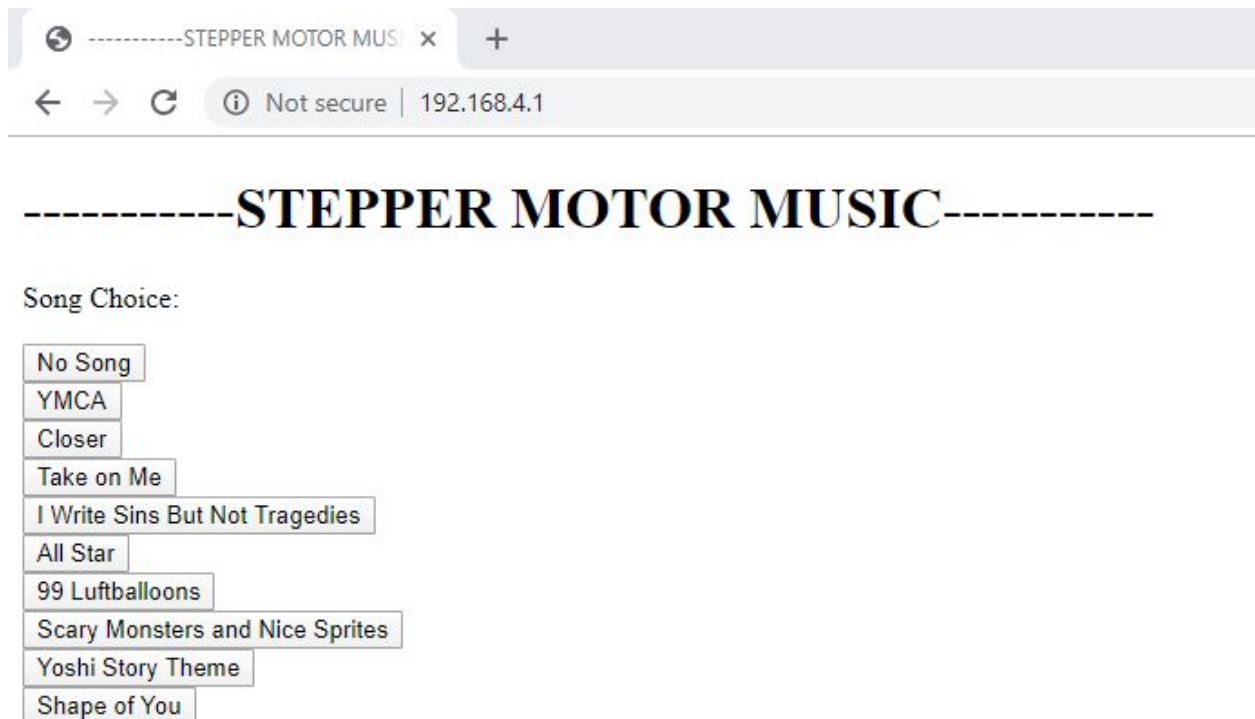


Figure 9. Screenshot of HTML webpage

The ESP8266 sends HTML requests to and from the ATSAM over UART. The implementation details of the UART communication and HTML webpage hosting can be found in the E155 Lab 6 Manual.

The user can choose one of the nine song options or no song to play. The user can also choose a different song to play in the middle of another song. If the user requests the same song again after the song has ended, the song will be repeated.

The HTML request communication had a default error-handling timeout. If the ATSAM was taking too long to respond to the ESP8266, then an error message would appear on the webpage. Due to this, note playing could not be blocking. In other words, the ATSAM could not wait for the duration of a note before parsing another UART response; otherwise, the ATSAM would timeout. The team solved this issue by making note playing non-blocking. The ATSAM samples the UART every one microsecond and counts the number of microseconds passed. If the number of microseconds passed is greater than or equal to the duration of the note, then the ATSAM increments the note counter and resets the microsecond count.

## Results

Upon completion of the project, nine unique songs could be played on the stepper motors, based on user input from the HTML webpage. The songs were largely created by reading sheet music [4] [5] [6] [7] [8] and transcribing it into the previously described encoded format. Additional songs were created using the midi file parsing script in Appendix A [9].

A slideshow containing a sample selection of songs can be viewed at the following link: <https://docs.google.com/presentation/d/1OD58lbyFZtqookncDBI-VVhnwSGIX3ZMpAzS-ybHats/edit?usp=sharing>.

## Conclusion/Future Work

The final system ultimately expanded upon the project proposal -- the final design could play nine distinct songs on four stepper motors instead of the originally proposed three and two, respectively. This arose due to additional time left in the last week to polish the project and create additional songs.

A Python script was developed by the team to help parse MIDI files and extract notes to create the note arrays for the song structures. The script was successful in parsing notes from a file; however, the team found that the MIDI file note layouts did not perfectly translate to a playable song on the system. MIDI files separate song melodies and basslines on different tracks within the file, so the parser had to be ran multiple times to extract the melody and bassline. From there, the two outputted files had to be manually edited and merged together to be playable. Future work could include making this script more robust and incorporating it to work on the microcontroller, so a user could input any midi file of choice to play.

Other future work includes making the hardware set up more robust. Alternative motor drivers could be purchased that could handle more current -- this would eliminate the need to solder multiple h-bridges together. Moreover, additional motors could be purchased to play additional harmonies in the programmed songs and increase the overall volume of the musical setup. Better wooden housing enclosures could also be built/tested to increase the volume of the motors.

# References

- [1] "Super Mario Theme - stepper motor music", *Youtube.com*, 2019. [Online]. Available: <https://www.youtube.com/watch?v=0DBnGYMPaf4>. [Accessed: 13- Dec- 2019].
- [2] J. Kayne, "'Fireflies" (Owl City) - Played on 32 Stepper Motors", *Youtube.com*, 2019. [Online]. Available: <https://www.youtube.com/watch?v=5YFajoHbdtM>. [Accessed: 13- Dec- 2019].
- [3] "Usongshine Nema 17 Stepper Motor", *Amazon.com*, 2019. [Online]. Available: <https://www.amazon.com/Usongshine-nema-17-Stepper-Motor/dp/B07T8G9QQQ>. [Accessed: 13- Dec- 2019].
- [4] "All Star Sheet Music," *Musescore.com*. [Online]. Available: <https://musescore.com/user/12898551/scores/2755271>. [Accessed: 12-Dec-2019].
- [5] "I Write Sins Not Tragedies String Quartet," *Musescore.com*. [Online]. Available: <https://musescore.com/user/178992/scores/171148>. [Accessed: 12-Dec-2019].
- [6] "Yoshi's Story Medley," *Musescore.com*, 02-Apr-2018. [Online]. Available: <https://musescore.com/cleverusername/medude/a-yoshis-story-medley>. [Accessed: 12-Dec-2019].
- [7] "Shape of You Sheet Music," *Musescore.com*, 16-Jul-2018. [Online]. Available: <https://musescore.com/user/3435661/scores/3798956>. [Accessed: 12-Dec-2019].
- [8] "Take On Me Sheet Music," *Sheetmusic-Free.com*. [Online]. Available: <https://sheetmusic-free.com/take-on-me-sheet-music-a-ha-piano-sheet/>. [Accessed: 12-Dec-2019].
- [9] "Scary Monsters And Nice Sprites Lead.mid," *BitMidi*, 2018. [Online]. Available: <https://bitmidi.com/skrillex-scary-monsters-and-nice-sprites-lead-mid>. [Accessed: 12-Dec-2019].

## **Appendix A. Code**

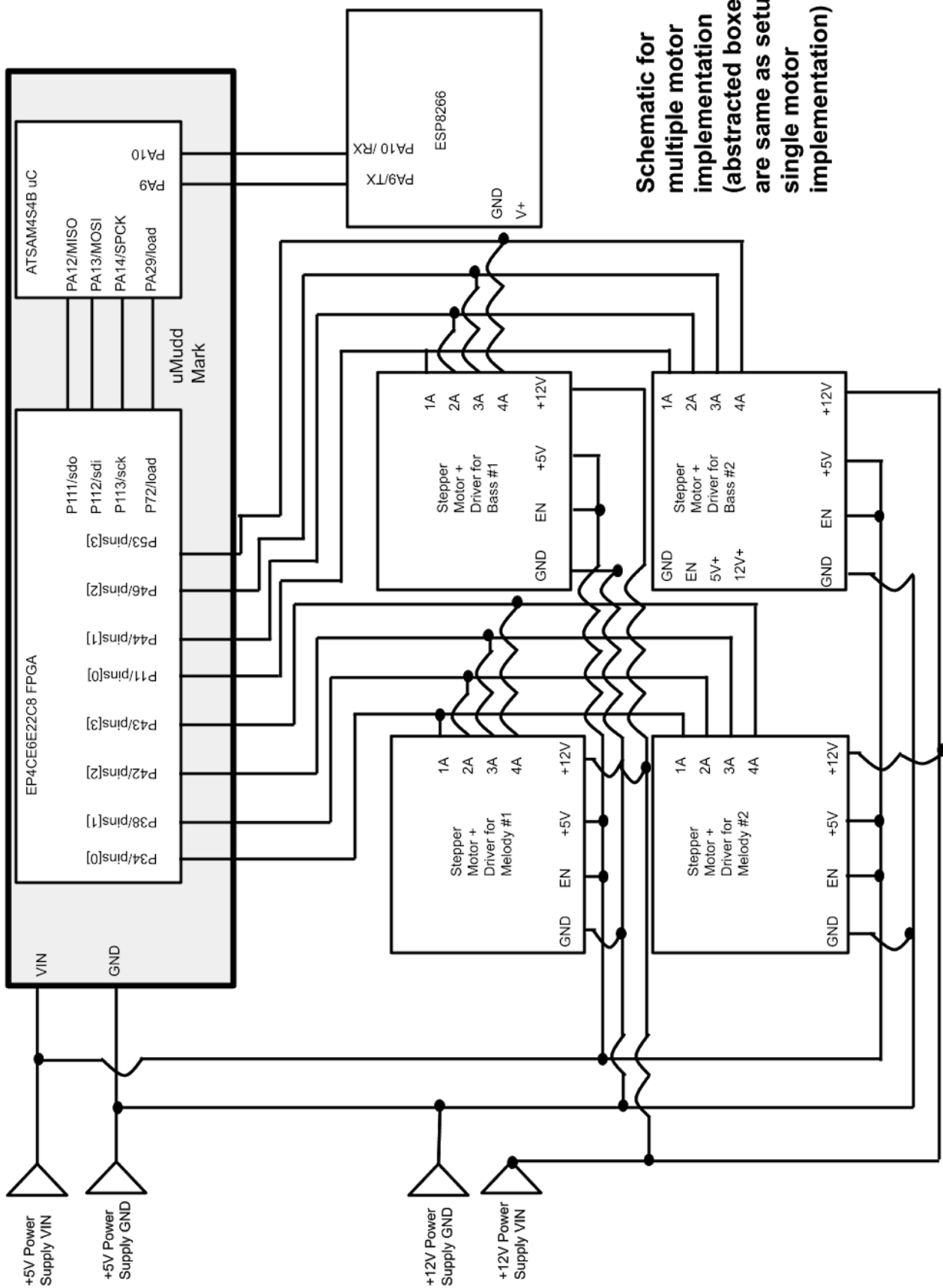
Verilog Code: Refer to pages 19 - 21

C Code: Refer to pages 22 - 68

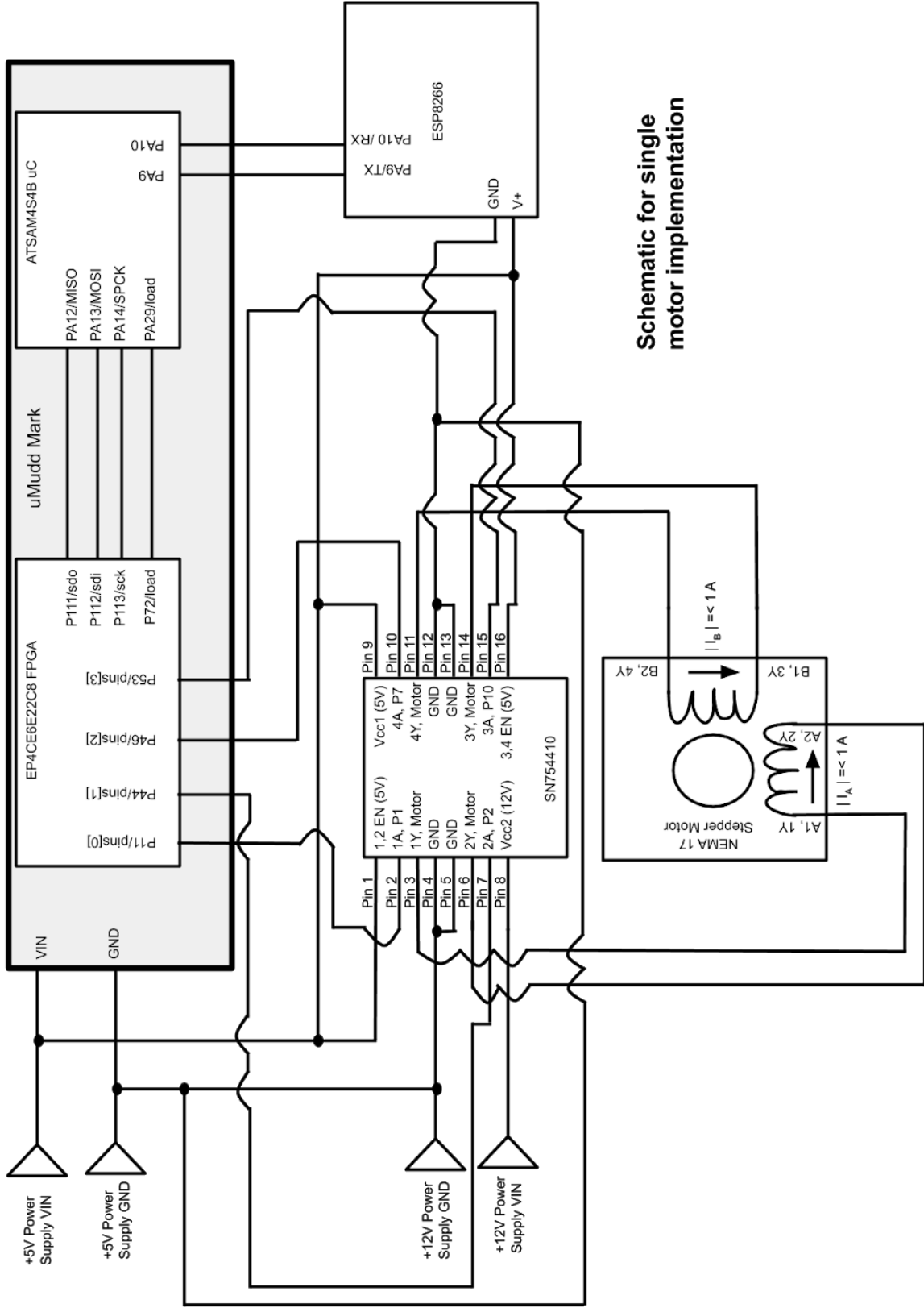
Midi Parsing Python Code: Refer to pages 69 - 70

File containing all encoded songs:

## **Appendix B. Full Size Schematics**



**Schematic for multiple motor implementation (abstracted boxes are same as setup in single motor implementation)**



**Schematic for single motor implementation**

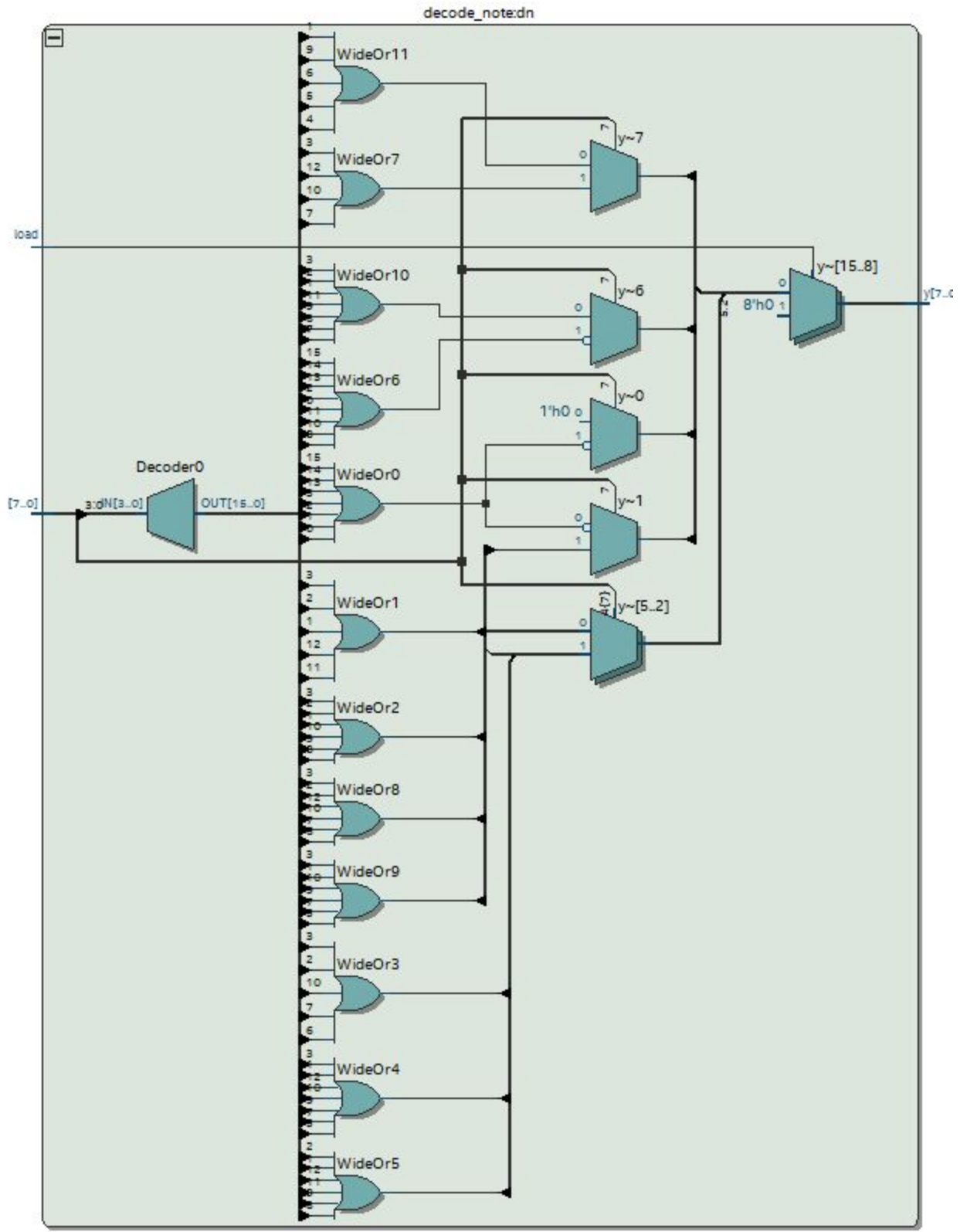
## Appendix C. Note Frequency to Slow Clock Counter Addend Table

Note	Frequency	y value exact	y value (what to add to slowclk each cycle)
E1	-	-	34
F1	-	-	36
FS1	-	-	39
G1	-	-	41
GS1	-	-	43
A1	-	-	46
A#1	-	-	49
B1	-	-	52
C2	65.41	54.86988493	55
C#2	69.3	58.13305344	58
D2	73.42	61.58915994	62
D#2	77.78	65.24659302	65
E2	82.41	69.13051853	69
F2	87.31	73.24093645	73
F#2	92.5	77.594624	78
G2	98	82.2083584	82
G#2	103.83	87.09891686	87
A2	110	92.274688	92
A#2	116.54	97.76083763	98
B2	123.47	103.574143	104
C3	130.81	109.7313812	110
C#3	138.59	116.2577183	116
D3	146.83	123.1699313	123
D#3	155.56	130.493186	130
E3	164.81	138.2526484	138
F3	174.61	146.4734843	146

F#3	185	155.189248	155
G3	196	164.4167168	164
G#3	207.65	174.1894451	174
A3	220	184.549376	185
A#3	233.08	195.5216753	196
B3	246.94	207.148286	207
C4	-	-	220
C#4	-	-	232
D4	-	-	260
D#4	-	-	276

## Appendix D. Decode Note RTL Viewer





## Appendix E. Parts List

Item	Source	Quantity
Usongshine Nema 17 Stepper Motor	Amazon	4
Large Wooden Box	Michaels	1
Small Wooden Box	Michaels	1
ESP8266 module	Sparkfun	1
SN754410 Quadruple Half-H Driver	Texas Instruments	9
MuddPi Mark V_1 Board	HMC	1

## Appendix F. Final Budget Breakdown

Item	Quantity	Cost Per Unit	Total Cost
Usongshine Nema 17 Stepper Motor	4	10.36	\$43.76
Large Wooden Box	1	\$3.73	\$3.73
Small Wooden Box	1	\$1.79	\$1.79
Wooden Box with Slits	1	\$0.85	\$0.85
<b>TOTAL</b>	-	-	<b>\$50.13</b>

```

1  ///////////////////////////////////////////////////////////////////
2  // project.sv
3  // HMC E155 16 November 2019
4  // E155 Final Project: Stepper Motor Music
5  // Samantha Tign + Andrew Q. Pham
6  // apham@hmc.edu, sting@hmc.edu
7  ///////////////////////////////////////////////////////////////////
8
9  ///////////////////////////////////////////////////////////////////
10 // testbench
11 // tests SPI communication of stepper motors notes between FPGA and ATSAM
12 ///////////////////////////////////////////////////////////////////
13
14 module testbench();
15     logic clk, load, sck, sdi, sdo;
16     logic [7:0] note, note2;
17     logic [31:0] i;
18
19     // device under test
20     project dut(clk, sck, sdi, sdo, load);
21
22     // test case
23     initial begin
24         note         <= 8'h11;
25         note2        <= 8'h91;
26     end
27
28     // generate clock and load signals
29     initial
30         forever begin
31             clk = 1'b0; #5;
32             clk = 1'b1; #5;
33         end
34
35     initial begin
36         i = 0;
37         load = 1'b1;
38     end
39
40     // shift in test vectors, wait until done
41     always @(posedge clk) begin
42         if (i == 0) load = 1'b1;
43         else if (i == 8) load = 1'b0;
44         else if (i < 8) begin
45             #1; sdi = note[7-i];
46             #1; sck = 1; #5; sck = 0;
47         end
48         else if (i == 108) load = 1'b0;
49         else if (i == 99) load = 1'b1;
50         else if (i < 108 & i > 99) begin
51             #1; sdi = note2[7-(i-100)];
52             #1; sck = 1; #5; sck = 0;
53         end
54         else if (i > 1000000) begin
55             $stop();
56         end
57
58         i = i + 1;
59     end
60 endmodule
61
62
63
64 ///////////////////////////////////////////////////////////////////
65 // project
66 // Top level module with SPI interface and SPI core
67 ///////////////////////////////////////////////////////////////////
68
69 module project(input logic clk,
70               input logic sck,
71               input logic sdi,
72               output logic sdo,
73               input logic load,
74               output logic [7:0] motor_pins);
75
76

```

```

77     logic [15:0] notes; //create array for two notes that will be played at the same time
78
79     project_spi spi(sck, sdi, sdo, notes); //read in notes via SPI interaction with uC
80     project_core core_m1(clk, load, notes[15:8], motor_pins[7:4]); //play notes for melody
part
81     project_core core_m2(clk, load, notes[7:0], motor_pins[3:0]); //play notes for bass part
82
83 endmodule
84
85 // SPI Module
86 module project_spi(input logic sck,
87                   input logic sdi,
88                   output logic sdo,
89                   output logic [15:0] notes);
90
91     always_ff @(posedge sck) begin
92         {notes} = {notes[14:0], sdi};
93     end
94     // when done is first asserted, shift out msb before clock edge
95     assign sdo = 0;
96 endmodule
97
98 // Motor Driving Module
99 module project_core(input logic clk,
100                  input logic load,
101                  input logic [7:0] note,
102                  output logic [3:0] motor_pins);
103
104     logic [24:0] q; //bus used to create slowclk
105     logic [7:0] y; //input note
106     logic [7:0] newy; //y adjusted based on raise/lower octave flags
107     logic slowclk; //slow clock used to control frequency of notes output from motor_pins
108     logic highflag; //flag used to double note frequency and raise note by an octave
109     logic lowflag; //flag used to halve note frequency and decrease note by an octave
110
111     assign highflag = note[6];
112     assign lowflag = note[5];
113
114     //Decode input note
115     decode_note dn(note, clk, load, y);
116     //Compute slowclk addend based on flags
117     assign newy = y*(1+highflag)/(1+lowflag);
118
119     //Create slow clock
120     counter counter(clk, load, newy, q);
121     assign slowclk = q[24];
122
123     //Drive motors with full step sequence
124     playnote playnote(slowclk, load, newy, motor_pins);
125
126 endmodule
127
128 // Decode Note module: determine what frequency to drive each note at
129 // y values for each octave were determined using slowclk formula as seen in Appendix C of
final report
130 module decode_note(input logic [7:0] note,
131                  input logic clk,
132                  input logic load,
133                  output logic [7:0] y);
134
135     always_comb
136         if (load == 1) begin
137             y = 0;
138         end
139         else begin
140             if (note[7]) begin //note[7] high indicates third octave
141                 case (note[3:0])
142                     4'b0000 : y <= 0;
143                     4'b0001 : y <= 110;
144                     4'b0010 : y <= 116;
145                     4'b0011 : y <= 123;
146                     4'b0100 : y <= 130;
147                     4'b0101 : y <= 138;
148                     4'b0110 : y <= 146;
149                     4'b0111 : y <= 155;
150                     4'b1000 : y <= 164;
151                     4'b1001 : y <= 174;

```

```
151         4'b1010 : y <= 185;
152         4'b1011 : y <= 196;
153         4'b1100 : y <= 207;
154         default : y <= 0;
155
156     endcase
157 end
158 else begin
159     case (note[3:0]) //note[7] low indicates second octave
160         4'b0000 : y <= 0;
161         4'b0001 : y <= 55;
162         4'b0010 : y <= 58;
163         4'b0011 : y <= 62;
164         4'b0100 : y <= 65;
165         4'b0101 : y <= 69;
166         4'b0110 : y <= 73;
167         4'b0111 : y <= 78;
168         4'b1000 : y <= 82;
169         4'b1001 : y <= 87;
170         4'b1010 : y <= 92;
171         4'b1011 : y <= 98;
172         4'b1100 : y <= 104;
173         default : y <= 0;
174
175     endcase
176 end
177 end
178
179 endmodule
180
181 //counter used to create slowclk from input 40 MHz clk
182 module counter(input logic clk,
183               input logic load,
184               input logic [7:0] y,
185               output logic [24:0] q);
186     always_ff @(posedge clk)
187         if (load)    q <= 0;
188         else        q <= q + y;
189
190 endmodule
191
```

```
1  module playnote(input logic slowclk,
2                  input logic load,
3                  input logic [7:0] y,
4                  output logic [3:0] pins);
5
6  typedef enum logic [1:0] {S0, S1, S2, S3} statetype;
7  statetype state, nextstate; // state register
8
9  always_ff @(posedge slowclk, posedge load)
10     if (load) state <= S0; //reset FSM if new notes are being loaded in
11     else state <= nextstate;
12
13 // next state logic -- always go to next state in loop regardless of inputs
14 always_comb case (state)
15     S0: nextstate = S1;
16     S1: nextstate = S2;
17     S2: nextstate = S3;
18     S3: nextstate = S0;
19     default: nextstate=S1;
20 endcase
21
22 // output logic for full step drive
23 assign pins[0] = (state==S0) && (y != 0);
24 assign pins[1] = (state==S2) && (y != 0);
25 assign pins[2] = (state==S1) && (y != 0);
26 assign pins[3] = (state==S3) && (y != 0);
27
28 endmodule
29
```

```

1 // E155: Stepper Motor Music Project
2 // Authors: Andrew Q. Pham & Sam Ting
3 // Emails: aphan@hmc.edu, sting@hmc.edu
4 // Date of Creation: 16 November 2019
5 //
6 // Description: Microcontroller code for Stepper Motor Music E155 Final Project Fall 2019
7 // Functionality: Send musical notes to hardware accelerator over SPI, create HTML webpage,
8 // where user can select a song, encode notes based on desired frequency
9
10 ////////////////////////////////////////////////////
11 // #includes
12 ////////////////////////////////////////////////////
13
14 #include "SAM4S4B.h"
15 #include <string.h>
16 #include <stdlib.h>
17 #include <stdio.h>
18 #include "songs.h"
19 #include "notes.h"
20
21 ////////////////////////////////////////////////////
22 // Constants
23 ////////////////////////////////////////////////////
24 #define LOAD_PIN 29
25
26 #define BUFF_LEN 32
27 #define NUM_MOTORS 2
28
29 //Defining the web page in two chunks: everything before the current time, and everything after the
30 current time
31 char* webpageStart = "<!DOCTYPE html><html><head><title>-----STEPPER MOTOR
32 MUSIC-----</title>\
33 <meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">\
34 </head>\
35 <body><h1>-----STEPPER MOTOR MUSIC-----</h1>";
36 char* songStr = "<p>Song Choice:</p>"
37 "<form action=\"0\"><input type=\"submit\" value=\"No Song\"></form>"
38 "<form action=\"1\"><input type=\"submit\" value=\"YMCA\"></form>"
39 "<form action=\"2\"><input type=\"submit\" value=\"Closer\"></form>"
40 "<form action=\"3\"><input type=\"submit\" value=\"Take on Me\"></form>"
41 "<form action=\"4\"><input type=\"submit\" value=\"I Write Sins But Not
42 Tragedies\"></form>"
43 "<form action=\"5\"><input type=\"submit\" value=\"All Star\"></form>"
44 "<form action=\"6\"><input type=\"submit\" value=\"99 Luftballons\"></form>"
45 "<form action=\"7\"><input type=\"submit\" value=\"Scary Monsters and Nice
46 Sprites\"></form>"
47 "<form action=\"8\"><input type=\"submit\" value=\"Yoshi Story Theme\"></form>"
48 "<form action=\"9\"><input type=\"submit\" value=\"Shape of You\"></form>";
49 char* webpageEnd = "</body></html>";
50
51 // Create song struct that holds all the notes for a song and the tempo (based on either 16th or 8th
52 notes)
53 typedef struct SONG{
54     const float ms_dur_sixteenth;
55     const float *notes;
56 }SONG;
57
58 // Create song structures to reference from website
59 const SONG NO_SONG = {0, &NO_SONG_NOTES[0][0]};
60 const SONG HOT_CROSS_BUNS = {25, &HOT_CROSS_BUNS_NOTES[0][0]};
61 const SONG LITTLE_LAMB = {20, &LITTLE_LAMB_NOTES[0][0]};
62 const SONG TAKE_ON_ME = {25, &TAKE_ON_ME_NOTES[0][0]};
63 const SONG I_WRITE_SINS = {32, &SINS_NOTES[0][0]};
64 const SONG ALL_STAR = {20, &ALL_STAR_NOTES[0][0]};
65 const SONG BALLOONS = {26, &BALLOONS_NOTES[0][0]};
66 const SONG MONSTERS = {35, &MONSTERS_NOTES[0][0]};
67 const SONG YOSHI = {20, &YOSHI_NOTES[0][0]};
68 const SONG SHAPEOFOYOU = {23, &SHAPE_NOTES[0][0]};
69 const SONG YMCA = {22, &YMCA_NOTES[0][0]};
70 const SONG CLOSER = {41, &CLOSER_NOTES[0][0]};
71

```

```

68
69 ////////////////////////////////////////////////////
70 // Function Prototypes
71 ////////////////////////////////////////////////////
72
73 void playNote(double notes[]);
74 void sendString(char* str);
75 int inString(char request[], char des[]);
76 int updateSongRequest(char request[]);
77 void updateSongChoice(int song_id, const SONG **song);
78
79 ////////////////////////////////////////////////////
80 // Main
81 ////////////////////////////////////////////////////
82
83 /**
84  * Plays a hardcoded song by encoding the notes in the song, sending
85  * them over SPI to the FPGA, and waiting for the note duration
86  */
87 int main(void) {
88     // Init peripherals on ATSAM
89     samInit();
90     pioInit();
91     spiInit(MCK_FREQ/244000, 0, 1);
92     tcInit();
93     tcDelayInit();
94     pioPinMode(LOAD_PIN, PIO_OUTPUT);
95     uartInit(4,20);
96     int song_id = 0;
97     int note_count = 0;
98     double ms_count = 0;
99     const SONG *song = &NO_SONG;
100
101     while(1){
102         // Wait for ESP8266 to send a request.
103         // Receive web request from the ESP
104         char request[BUFF_LEN] = "                "; // initialize to known value
105         int charIndex = 0;
106
107         while(inString(request, "\n") == -1) {
108             // Index to track note of song playing
109             double notes[2] = {*(song->notes + note_count*NUM_MOTORS), *(song->notes +
note_count*NUM_MOTORS + 1)};
110
111             // Wait for a complete request to be transmitted before processing
112             while(!uartRxReady()){
113
114                 if (ms_count == 0) {
115                     playNote(notes);
116                 }
117
118                 tcDelayMicroseconds(1);
119                 ms_count += 0.001;
120
121                 if (ms_count >= song->ms_dur_sixteenth){
122                     ms_count = 0;
123                     if (notes[0] != END){
124                         note_count++;
125                         notes[0] = *(song->notes + note_count*NUM_MOTORS);
126                         notes[1] = *(song->notes + note_count*NUM_MOTORS + 1);
127                     }
128                     else{
129                         song_id = 0;
130                         note_count = 0;
131                         ms_count = 0;
132                         updateSongChoice(0, &song); //begin playing newly selected song
133                     }
134                 }
135
136             }
137
138             request[charIndex++] = uartRx();

```



```

139     }
140
141     int new_song_id = updateSongRequest(request);
142
143     if (new_song_id != song_id){ //if a different song is requested
144         note_count = 0;
145         ms_count = 0;
146         song_id = new_song_id; //change song ID
147         updateSongChoice(song_id, &song); //begin playing newly selected song
148     }
149
150     // finally, transmit the webpage over UART
151     sendString(webpageStart); // webpage header code
152     sendString(songStr); // button for controlling song choice
153     sendString(webpageEnd);
154 }
155
156 }
157
158 ////////////////////////////////////////////////////////////////////
159 // Functions
160 ////////////////////////////////////////////////////////////////////
161 /**
162  * Change which song is currently being played based on webpage input
163  * @param song_id the new song id based on HTML input
164  * @param **song the pointer that should be updated with the new song request
165  * @return 8-bit signal that encodes what note to play on which motor
166  */
167 void updateSongChoice(int song_id, const SONG **song){
168     switch (song_id){
169         case 0:
170             *song = &NO_SONG;
171             break;
172         case 1:
173             *song = &YMCA;
174             break;
175         case 2:
176             *song = &CLOSER;
177             break;
178         case 3:
179             *song = &TAKE_ON_ME;
180             break;
181         case 4:
182             *song = &I_WRITE_SINS;
183             break;
184         case 5:
185             *song = &ALL_STAR;
186             break;
187         case 6:
188             *song = &BALLOONS;
189             break;
190         case 7:
191             *song = &MONSTERS;
192             break;
193         case 8:
194             *song = &YOSHI;
195             break;
196         case 9:
197             *song = &SHAPEOFYOU;
198             break;
199
200
201         default:
202             *song = &NO_SONG;
203     }
204 }
205
206 /**
207  * Encodes note and which motor to play it into 8-bit signal
208  * encoded_note[7:5] = octave of note
209  * encoded_note[3:0] = note in octave
210  * We encoded for four octaves

```

```
211 *     encoded_note[7:5] = 000 = octave 2
212 *     encoded_note[7:5] = 100 = octave 3
213 *     encoded_note[7:5] = 001 = divide note frequencies in octave 2 by 2 --> octave 1
214 *     encoded_note[7:5] = 110 = multiply note frequencies in octave 3 by 2 --> octave 4
215 * @param motor The number of the motor to play the note
216 * @param id     The id of the note
217 * @return 8-bit signal that encodes what note to play on which motor
218 **/
219 char encodeNote(int motor, double id){
220     char encoded_note = 0x00;
221     switch ((int)id){
222     case 0:
223         encoded_note |= 0x00;
224         break;
225     case C2:
226         encoded_note |= 0x01;
227         break;
228     case CS2:
229         encoded_note |= 0x02;
230         break;
231     case D2:
232         encoded_note |= 0x03;
233         break;
234     case DS2:
235         encoded_note |= 0x04;
236         break;
237     case E2:
238         encoded_note |= 0x05;
239         break;
240     case F2:
241         encoded_note |= 0x06;
242         break;
243     case FS2:
244         encoded_note |= 0x07;
245         break;
246     case G2:
247         encoded_note |= 0x08;
248         break;
249     case GS2:
250         encoded_note |= 0x09;
251         break;
252     case A2:
253         encoded_note |= 0x0a;
254         break;
255     case AS2:
256         encoded_note |= 0x0b;
257         break;
258     case B2:
259         encoded_note |= 0x0c;
260         break;
261     case C3:
262         encoded_note |= 0x81;
263         break;
264     case CS3:
265         encoded_note |= 0x82;
266         break;
267     case D3:
268         encoded_note |= 0x83;
269         break;
270     case DS3:
271         encoded_note |= 0x84;
272         break;
273     case E3:
274         encoded_note |= 0x85;
275         break;
276     case F3:
277         encoded_note |= 0x86;
278         break;
279     case FS3:
280         encoded_note |= 0x87;
281         break;
282     case G3:
```

```
283         encoded_note |= 0x88;
284         break;
285     case GS3:
286         encoded_note |= 0x89;
287         break;
288     case A3:
289         encoded_note |= 0x8a;
290         break;
291     case AS3:
292         encoded_note |= 0x8b;
293         break;
294     case B3:
295         encoded_note |= 0x8c;
296         break;
297     case C4:
298         encoded_note |= 0xc1;
299         break;
300     case CS4:
301         encoded_note |= 0xc2;
302         break;
303     case D4:
304         encoded_note |= 0xc3;
305         break;
306     case DS4:
307         encoded_note |= 0xc4;
308         break;
309     case E4:
310         encoded_note |= 0xc5;
311         break;
312
313     case E1:
314         encoded_note |= 0x28;
315         break;
316     case F1:
317         encoded_note |= 0x29;
318         break;
319
320     case FS1:
321         encoded_note |= 0x27;
322         break;
323
324     case G1:
325         encoded_note |= 0x28;
326         break;
327     case GS1:
328         encoded_note |= 0x29;
329         break;
330
331     case A1:
332         encoded_note |= 0x2a;
333         break;
334     case AS1:
335         encoded_note |= 0x2b;
336         break;
337     case B1:
338         encoded_note |= 0x2c;
339         break;
340
341     default:
342         encoded_note |= 0x00;
343         break;
344 }
345
346 encoded_note |= (motor << 4);
347
348 return encoded_note;
349 }
350
351 /**
352  * Plays a note by encoding it and sending it over SPI to the FPGA
353  *
354  * @param note_id ID number of note defined in constants section above
```

```

355  **/
356  void playNote(double notes[]) {
357      pioDigitalWrite(LOAD_PIN, 1);          // Set LOAD Pin high to initiate SPI interaction
358      for (int mtr = 0; mtr < NUM_MOTORS; mtr++){
359          char note = encodeNote(mtr, notes[mtr]);
360          spiSendReceive(note);
361      }
362      pioDigitalWrite(LOAD_PIN, 0);
363  }
364  // Sends a null terminated string of arbitrary length
365  void sendString(char* str) {
366      char* ptr = str;
367      while (*ptr) uartTx(*ptr++);
368  }
369
370  //determines whether a given character sequence is in a char array request, returning 1 if present, -1
  if not present
371  int inString(char request[], char des[]) {
372      if (strstr(request, des) != NULL) {return 1;}
373      return -1;
374  }
375
376  int updateSongRequest(char request[])
377  {
378      int song_id;
379
380      // The request has been received. now process to determine whether to turn the LED on or off
381      if (inString(request, "0")==1) {
382          song_id = 0;
383      } else if (inString(request, "1")==1) {
384          song_id= 1;
385      } else if (inString(request, "2")==1) {
386          song_id= 2;
387      }
388      else if (inString(request, "3")==1) {
389          song_id= 3;
390      }
391      else if (inString(request, "4")==1) {
392          song_id= 4;
393      }
394      else if (inString(request, "5")==1) {
395          song_id= 5;
396      }
397      else if (inString(request, "6")==1) {
398          song_id= 6;
399      }
400      else if (inString(request, "7")==1) {
401          song_id= 7;
402      }
403      else if (inString(request, "8")==1) {
404          song_id= 8;
405      }
406      else if (inString(request, "9")==1) {
407          song_id= 9;
408      }
409      else if (inString(request, "10")==1) {
410          song_id= 10;
411      }
412      else if (inString(request, "11")==1) {
413          song_id= 11;
414      }
415      else if (inString(request, "12")==1) {
416          song_id= 12;
417      }
418
419      else {
420          song_id = 0;
421      }
422
423      return song_id;
424  }
425

```

```
1
2 //Note Definitions
3 #define REST 0
4 #define C2 1
5 #define CS2 2
6 #define D2 3
7 #define DS2 4
8 #define E2 5
9 #define F2 6
10 #define FS2 7
11 #define G2 8
12 #define GS2 9
13 #define A2 10
14 #define AS2 11
15 #define B2 12
16 #define C3 13
17 #define CS3 14
18 #define D3 15
19 #define DS3 16
20 #define E3 17
21 #define F3 18
22 #define FS3 19
23 #define G3 20
24 #define GS3 21
25 #define A3 22
26 #define AS3 23
27 #define B3 24
28 #define END 25
29 #define C4 26
30 #define CS4 27
31 #define D4 28
32 #define E4 29
33 #define E1 30
34 #define F1 31
35 #define FS1 32
36 #define G1 33
37 #define GS1 34
38 #define A1 35
39 #define AS1 36
40 #define B1 37
41 #define DS4 38
42
```

```
1
2 //Song Arrays
3 #include "notes.h"
4 #include <stdio.h>
5
6 //No Song
7 const float NO_SONG_NOTES[][2] = {{END, END}};
8
9 // Hot Cross Buns, Hard-Coded
10 const float HOT_CROSS_BUNS_NOTES[][2] = {
11
12
13
14 {B2, C2},
15 {B2, REST},
16 {B2, C2},
17 {REST, REST},
18 {A2, C2},
19 {A2, REST},
20 {A2, C2},
21 {REST, REST},
22 {G2, C2},
23 {G2, REST},
24 {G2, C2},
25 {G2, REST},
26 {G2, C2},
27 {G2, REST},
28 {REST, REST},
29 {REST, REST},
30
31 {B2, C2},
32 {B2, REST},
33 {B2, C2},
34 {REST, REST},
35 {A2, C2},
36 {A2, REST},
37 {A2, C2},
38 {REST, REST},
39 {G2, C2},
40 {G2, REST},
41 {G2, C2},
42 {G2, REST},
43 {G2, C2},
44 {G2, REST},
45 {REST, REST},
46 {REST, REST},
47
48 {G2, REST},
49 {REST, C2},
50 {G2, REST},
51 {REST, C2},
52 {G2, REST},
53 {REST, C2},
54 {G2, REST},
55 {REST, C2},
56 {G2, REST},
57 {REST, C2},
58 {A2, REST},
59 {REST, C2},
60 {A2, REST},
61 {REST, C2},
62 {A2, REST},
63 {REST, C2},
64 {A2, REST},
65 {REST, C2},
66
67 {B2, C2},
68 {B2, REST},
69 {B2, C2},
70 {REST, REST},
71 {A2, C2},
72 {A2, REST},
```

```
73 {A2, C2},
74 {REST, REST},
75 {G2, C2},
76 {G2, REST},
77 {G2, C2},
78 {G2, REST},
79 {G2, C2},
80 {G2, REST},
81 {END, END}};
82
83
84 // Take On Me by a-ha, Hard-Coded
85 const float TAKE_ON_ME_NOTES[][2] = {
86 {CS4, FS3},
87 {CS4, FS3},
88 {CS4, FS3},
89 {CS4, FS3},
90 {CS4, FS3},
91 {CS4, FS3},
92 {CS4, FS3},
93 {CS4, FS3},
94
95 {CS4, FS3},
96 {CS4, FS3},
97 {CS4, FS3},
98 {CS4, FS3},
99 {CS4, FS3},
100 {CS4, FS3},
101 {CS4, FS3},
102 {CS4, FS3},
103
104 {CS3, CS3},
105 {CS3, CS3},
106 {CS3, CS3},
107 {A2, A2},
108 {A2, A2},
109 {A2, A2},
110 {A2, A2},
111 {A2, A2},
112
113 {A2, A2},
114 {A2, A2},
115 {A2, A2},
116 {A2, A2},
117 {REST, REST},
118 {REST, REST},
119 {REST, REST},
120 {REST, REST},
121
122 {REST, C3},
123 {REST, C3},
124 {REST, REST},
125 {REST, C2},
126 {REST, REST},
127 {REST, REST},
128 {REST, C2},
129 {REST, C2},
130
131 {REST, C3},
132 {REST, C3},
133 {REST, REST},
134 {REST, C2},
135 {REST, REST},
136 {REST, REST},
137 {REST, C2},
138 {REST, C2},
139
140 {REST, C3},
141 {REST, C3},
142 {REST, REST},
143 {REST, C2},
144 {REST, REST},
```

```
145 {REST, REST},
146 {REST, C2},
147 {REST, C2},
148
149 {REST, C3},
150 {REST, C3},
151 {REST, REST},
152 {REST, C2},
153 {REST, REST},
154 {REST, REST},
155 {REST, C2},
156 {REST, C2},
157
158 {FS3, B1},
159 {FS3, B1},
160 {D3, B1},
161 {B2, B1},
162 {REST, B1},
163 {B2, B1},
164 {REST, B1},
165 {E3, B1},
166
167 {REST, REST},
168 {E3, },
169 {REST, E2},
170 {E3, E2},
171 {E3, E2},
172 {GS3, E2},
173 {GS3, E2},
174 {A3, E2},
175
176 {B3, A1},
177 {A3, A1},
178 {A3, A1},
179 {A3, A1},
180 {E3, A1},
181 {REST, REST},
182 {D3, C2},
183 {REST, C2},
184
185 {FS3, REST},
186 {REST, D2},
187 {FS3, D2},
188 {REST, D2},
189 {FS3, D2},
190 {E3, CS2},
191 {E3, CS2},
192 {FS3, CS2},
193 {E3, CS2},
194
195 {FS3, B1},
196 {FS3, B1},
197 {D3, B1},
198 {B2, B1},
199 {REST, B1},
200 {B2, B1},
201 {REST, B1},
202 {E3, B1},
203
204 {REST, REST},
205 {E3, },
206 {REST, E2},
207 {E3, E2},
208 {E3, E2},
209 {GS3, E2},
210 {GS3, E2},
211 {A3, E2},
212
213 {B3, A1},
214 {A3, A1},
215 {A3, A1},
216 {A3, A1},
```



```
217 {E3, A1},
218 {REST, REST},
219 {D3, C2},
220 {REST, C2},
221
222 {FS3, REST},
223 {REST, D2},
224 {FS3, D2},
225 {REST, D2},
226 {FS3, D2},
227 {E3, CS2},
228 {E3, CS2},
229 {FS3, CS2},
230 {E3, CS2},
231
232 {REST, REST},
233 {REST, REST},
234 {REST, REST},
235 {REST, REST},
236 {REST, REST},
237
238 {D3, B1},
239 {D3, B1},
240 {D3, B1},
241 {D3, B1},
242 {D3, B1},
243 {D3, B1},
244 {CS3, B1},
245 {B2, B1},
246 {B2, B1},
247
248 {REST, REST},
249 {REST, REST},
250 {REST, REST},
251 {REST, REST},
252 {REST, REST},
253 {REST, REST},
254 {REST, REST},
255 {E2, REST},
256
257 {CS3, A1},
258 {CS3, A1},
259 {CS3, A1},
260 {CS3, A1},
261 {CS3, A1},
262 {A2, A1},
263 {REST, A1},
264 {REST, A1},
265
266 {REST, D2},
267 {FS3, D2},
268 {REST, D2},
269 {FS3, D2},
270 {FS3, CS2},
271 {FS3, CS2},
272 {E3, CS2},
273 {E3, CS2},
274
275 {D3, B1},
276 {D3, B1},
277 {D3, B1},
278 {D3, B1},
279 {D3, B1},
280 {CS3, B1},
281 {CS3, B1},
282 {B2, B1},
283
284 {B2, A1},
285 {B2, A1},
286 {REST, A1},
287 {REST, A1},
288 {REST, A1},
```

```
289 {REST, A1},
290 {REST, A1},
291 {E2, A1},
292
293 {CS3, D2},
294 {CS3, D2},
295 {CS3, D2},
296 {CS3, D2},
297 {CS3, CS2},
298 {B2, CS2},
299 {A2, CS2},
300 {A2, CS2},
301
302 {A2, REST},
303 {B2, REST},
304 {B2, REST},
305 {CS3, REST},
306 {CS3, REST},
307 {B2, REST},
308 {A2, REST},
309 {A2, REST},
310
311 {REST, REST},
312 {REST, REST},
313 {D3, REST},
314 {D3, REST},
315 {D3, REST},
316 {D3, REST},
317 {D3, REST},
318 {D3, REST},
319
320 {E3, REST},
321 {E3, REST},
322 {REST, REST},
323 {REST, REST},
324 {REST, REST},
325 {REST, REST},
326 {REST, REST},
327 {REST, REST},
328
329 {REST, REST},
330 {REST, REST},
331 {FS2, REST},
332 {A2, REST},
333 {A2, REST},
334 {A2, REST},
335 {A2, REST},
336 {A2, REST},
337
338 {A2, REST},
339 {GS2, REST},
340 {GS2, REST},
341 {GS2, REST},
342 {GS2, REST},
343 {FS2, REST},
344 {FS2, REST},
345 {FS2, REST},
346
347 {A1, A1},
348 {A1, A1},
349 {A1, A1},
350 {A1, A1},
351 {A1, A1},
352 {A1, A1},
353 {A1, A1},
354 {A1, A1},
355
356 {GS2, GS2},
357 {GS2, GS2},
358 {GS2, GS2},
359 {GS2, GS2},
360 {GS2, GS2},
```

```
361 {GS2, GS2},
362 {GS2, GS2},
363 {GS2, GS2},
364
365 {A2, A2},
366 {A2, A2},
367 {A2, A2},
368 {A2, A2},
369 {A2, A2},
370 {A2, A2},
371 {A2, A2},
372 {A2, A2},
373
374 {E3, REST},
375 {E3, REST},
376 {REST, REST},
377 {FS3, REST},
378 {FS3, REST},
379 {FS3, REST},
380 {E3, REST},
381 {E3, REST},
382
383 {A2, A2},
384 {A2, A2},
385 {A2, A2},
386 {A2, A2},
387 {A2, A2},
388 {A2, A2},
389 {A2, A2},
390 {A2, A2},
391
392 {E3, E3},
393 {E3, E3},
394 {E3, E3},
395 {E3, E3},
396 {E3, E3},
397 {E3, E3},
398 {E3, E3},
399 {E3, E3},
400
401 {FS3, FS3},
402 {FS3, FS3},
403 {FS3, FS3},
404 {FS3, FS3},
405 {FS3, FS3},
406 {FS3, FS3},
407 {FS3, FS3},
408 {FS3, FS3},
409
410 {E3, REST},
411 {E3, REST},
412 {REST, REST},
413 {FS3, REST},
414 {FS3, REST},
415 {FS3, REST},
416 {E3, REST},
417 {E3, REST},
418
419 {CS3, CS3},
420 {CS3, CS3},
421 {CS3, CS3},
422 {CS3, CS3},
423 {CS3, CS3},
424 {CS3, CS3},
425 {CS3, CS3},
426 {CS3, CS3},
427
428 {GS3, GS3},
429 {GS3, GS3},
430 {GS3, GS3},
431 {GS3, GS3},
432 {GS3, GS3},
```

```
433 {GS3, GS3},
434 {GS3, GS3},
435 {GS3, GS3},
436
437 {A3, A3},
438 {A3, A3},
439 {A3, A3},
440 {A3, A3},
441 {A3, A3},
442 {A3, A3},
443 {A3, A3},
444 {A3, A3},
445
446 {REST, REST},
447 {REST, REST},
448 {B3, B3},
449 {CS3, CS3},
450 {REST, REST},
451 {B3, B3},
452 {A3, A3},
453 {A3, A3},
454
455 {A3, A3},
456 {A3, A3},
457 {E3, E3},
458 {E3, E3},
459 {E3, E3},
460 {E3, E3},
461 {E3, E3},
462 {E3, E3},
463
464 {E3, E3},
465 {E3, E3},
466 {E3, E3},
467 {E3, E3},
468 {E3, E3},
469 {E3, E3},
470 {E3, E3},
471 {E3, E3},
472
473 {E3, E3},
474 {E3, E3},
475 {E3, E3},
476 {E3, E3},
477 {E3, E3},
478 {E3, E3},
479 {E3, E3},
480 {E3, E3},
481
482 {END, END}};
483
484 // All Star by Smash Mouth, Hard-Coded
485 const float ALL_STAR_NOTES[][2] = {
486 {DS3, REST},
487 {DS3, REST},
488
489 {DS3, REST},
490 {DS3, REST},
491
492 {AS3, DS2},
493 {AS3, DS2},
494 {G3, DS2},
495 {G3, DS2},
496 {G3, DS2},
497 {G3, DS2},
498 {G3, DS2},
499 {G3, DS2},
500
501 {F3, AS2},
502 {F3, AS2},
503 {DS3, AS2},
504 {DS3, AS2},
```

```
505 {DS3, AS2},
506 {DS3, AS2},
507 {GS3, AS2},
508 {GS3, AS2},
509
510 {GS3, G2},
511 {GS3, G2},
512 {G3, G2},
513 {G3, G2},
514 {G3, G2},
515 {G3, G2},
516 {F3, G2},
517 {F3, G2},
518
519 {F3, GS2},
520 {F3, GS2},
521 {DS3, GS2},
522 {DS3, GS2},
523 {REST, GS2},
524 {REST, GS2},
525 {DS3, GS2},
526 {DS3, GS2},
527
528 {AS3, DS2},
529 {AS3, DS2},
530 {G3, DS2},
531 {G3, DS2},
532 {G3, DS2},
533 {G3, DS2},
534 {F3, DS2},
535 {F3, DS2},
536
537 {F3, AS2},
538 {F3, AS2},
539 {DS3, AS2},
540 {DS3, AS2},
541 {DS3, AS2},
542 {DS3, AS2},
543 {C3, AS2},
544 {C3, AS2},
545
546 {C3, G2},
547 {C3, G2},
548 {AS2, G2},
549 {AS2, G2},
550 {AS2, G2},
551 {AS2, G2},
552 {AS2, G2},
553 {AS2, G2},
554
555 {REST, GS2},
556 {REST, GS2},
557 {REST, GS2},
558 {REST, GS2},
559 {DS2, GS2},
560 {DS2, GS2},
561 {DS2, GS2},
562 {DS2, GS2},
563
564
565 {AS3, DS2},
566 {AS3, DS2},
567 {G3, DS2},
568 {G3, DS2},
569 {G3, DS2},
570 {G3, DS2},
571 {F3, DS2},
572 {F3, DS2},
573
574 {F3, AS2},
575 {F3, AS2},
576 {DS3, AS2},
```

```
577 {DS3, AS2},
578 {DS3, AS2},
579 {DS3, AS2},
580 {GS3, AS2},
581 {GS3, AS2},
582
583 {GS3, G2},
584 {GS3, G2},
585 {G3, G2},
586 {G3, G2},
587 {G3, G2},
588 {G3, G2},
589 {F3, G2},
590 {F3, G2},
591
592 {F3, GS2},
593 {F3, GS2},
594 {DS3, GS2},
595 {DS3, GS2},
596 {DS3, GS2},
597 {DS3, GS2},
598 {AS3, GS2},
599 {AS3, GS2},
600
601 {AS3, DS2},
602 {AS3, DS2},
603 {G3, DS2},
604 {G3, DS2},
605 {G3, DS2},
606 {G3, DS2},
607 {F3, DS2},
608 {F3, DS2},
609
610 {F3, AS2},
611 {F3, AS2},
612 {DS3, AS2},
613 {DS3, AS2},
614 {DS3, AS2},
615 {DS3, AS2},
616 {F3, AS2},
617 {F3, AS2},
618
619 {F3, G2},
620 {F3, G2},
621 {C3, G2},
622 {C3, G2},
623 {C3, G2},
624 {C3, G2},
625 {C3, G2},
626 {C3, G2},
627
628 {REST, GS2},
629 {REST, GS2},
630 {REST, GS2},
631 {REST, GS2},
632 {DS3, GS2},
633 {DS3, GS2},
634 {DS3, GS2},
635 {C3, GS2},
636
637 {DS3, REST},
638 {DS3, REST},
639 {DS3, DS2},
640 {DS3, DS2},
641 {DS3, G2},
642 {DS3, G2},
643 {DS3, REST},
644 {REST, REST},
645
646 {DS3, REST},
647 {DS3, REST},
648 {DS3, AS2},
```

```
649 {REST, AS2},
650 {DS3, D3},
651 {DS3, D3},
652 {REST, REST},
653 {REST, REST},
654
655 {DS3, REST},
656 {REST, REST},
657 {DS3, G2},
658 {DS3, G2},
659 {DS3, AS2},
660 {REST, AS2},
661 {DS3, REST},
662 {DS3, REST},
663
664 {DS3, REST},
665 {DS3, REST},
666 {DS3, GS2},
667 {REST, GS2},
668 {G3, C3},
669 {DS3, C3},
670 {REST, REST},
671 {REST, REST},
672
673
674 {G3, REST},
675 {G3, REST},
676 {AS3, DS2},
677 {AS3, DS2},
678 {GS3, G2},
679 {GS3, G2},
680 {G3, REST},
681 {G3, REST},
682
683 {F3, REST},
684 {F3, REST},
685 {F3, AS2},
686 {F3, AS2},
687 {F3, D3},
688 {F3, D3},
689 {F3, REST},
690 {DS3, REST},
691
692
693 {G3, REST},
694 {G3, REST},
695 {DS3, G2},
696 {REST, G2},
697 {DS3, AS2},
698 {DS3, AS2},
699 {C3, REST},
700 {AS2, REST},
701
702 {DS3, REST},
703 {DS3, REST},
704 {DS3, GS2},
705 {DS3, GS2},
706 {C3, C3},
707 {AS2, C3},
708 {REST, REST},
709 {REST, REST},
710
711 {G3, REST},
712 {AS3, REST},
713 {AS3, DS2},
714 {G3, DS2},
715 {C4, G2},
716 {C4, G2},
717 {G3, REST},
718 {AS3, REST},
719
720 {AS3, REST},
```

```
721 {G3, REST},
722 {C4, AS2},
723 {C4, AS2},
724 {G3, D3},
725 {AS3, D3},
726 {AS3, REST},
727 {GS3, REST},
728
729 {GS3, REST},
730 {G3, REST},
731 {G3, G2},
732 {F3, G2},
733 {REST, AS2},
734 {F3, AS2},
735 {F3, REST},
736 {DS3, REST},
737
738 {F3, REST},
739 {F3, REST},
740 {DS3, GS2},
741 {DS3, GS2},
742 {REST, C3},
743 {REST, C3},
744 {REST, REST},
745 {AS2, REST},
746
747 {DS3, REST},
748 {DS3, REST},
749 {DS3, DS2},
750 {REST, DS2},
751 {DS3, G2},
752 {DS3, G2},
753 {DS3, REST},
754 {REST, REST},
755
756 {DS3, REST},
757 {DS3, REST},
758 {DS3, AS2},
759 {DS3, AS2},
760 {REST, D3},
761 {REST, D3},
762 {REST, REST},
763 {AS2, REST},
764
765 {G3, REST},
766 {F3, REST},
767 {DS3, G2},
768 {REST, G2},
769 {F3, AS2},
770 {DS3, AS2},
771 {C3, REST},
772 {REST, REST},
773
774 {AS2, REST},
775 {AS2, REST},
776 {AS2, GS2},
777 {AS2, GS2},
778 {REST, C3},
779 {REST, C3},
780 {REST, REST},
781 {REST, REST},
782
783 {G3, G3},
784 {G3, G3},
785 {DS3, DS3},
786 {DS3, DS3},
787 {DS3, DS3},
788 {REST, REST},
789 {DS3, DS3},
790 {C3, C3},
791
792 {DS3, DS3},
```



```
793 {DS3, DS3},
794 {DS3, DS3},
795 {DS3, DS3},
796 {DS3, DS3},
797 {REST, REST},
798 {DS3, DS3},
799 {C3, C3},
800
801 {DS3, DS3},
802 {REST, REST},
803 {DS3, DS3},
804 {DS3, DS3},
805 {DS3, DS3},
806 {DS3, DS3},
807 {DS3, DS3},
808 {DS3, DS3},
809
810 {DS3, DS3},
811 {REST, REST},
812 {G3, G3},
813 {G3, G3},
814 {G3, G3},
815 {G3, G3},
816 {G3, G3},
817 {G3, G3},
818
819 {G3, G3},
820 {G3, G3},
821 {DS3, DS3},
822 {DS3, DS3},
823 {DS3, DS3},
824 {REST, REST},
825 {DS3, DS3},
826 {C3, C3},
827
828 {DS3, DS3},
829 {DS3, DS3},
830 {DS3, DS3},
831 {DS3, DS3},
832 {DS3, DS3},
833 {REST, REST},
834 {DS3, DS3},
835 {C3, C3},
836
837 {DS3, DS3},
838 {REST, REST},
839 {DS3, DS3},
840 {DS3, DS3},
841 {DS3, DS3},
842 {DS3, DS3},
843 {DS3, DS3},
844 {DS3, DS3},
845
846 {DS3, DS3},
847 {REST, REST},
848 {G3, G3},
849 {G3, G3},
850 {G3, G3},
851 {G3, G3},
852 {DS3, DS3},
853 {DS3, DS3},
854
855 {G3, G3},
856 {G3, G3},
857 {G3, G3},
858 {G3, G3},
859 {AS3, AS3},
860 {AS3, AS3},
861 {AS3, AS3},
862 {AS3, AS3},
863
864 {GS3, GS3},
```

```
865 {GS3, GS3},
866 {G3, G3},
867 {G3, G3},
868 {F3, F3},
869 {REST, REST},
870 {F3, F3},
871 {F3, F3},
872
873 {F3, F3},
874 {F3, F3},
875 {DS3, DS3},
876 {DS3, DS3},
877 {DS3, DS3},
878 {DS3, DS3},
879 {DS3, DS3},
880 {DS3, DS3},
881
882 {DS3, DS3},
883 {REST, REST},
884 {DS3, DS3},
885 {REST, REST},
886 {F3, F3},
887 {REST, REST},
888 {DS3, DS3},
889 {REST, REST},
890
891 {G3, G3},
892 {G3, G3},
893 {F3, F3},
894 {F3, F3},
895 {F3, F3},
896 {F3, F3},
897 {F3, F3},
898 {REST, REST},
899
900 {F3, F3},
901 {F3, F3},
902 {DS3, DS3},
903 {DS3, DS3},
904 {DS3, DS3},
905 {DS3, DS3},
906 {F3, F3},
907 {F3, F3},
908
909 {G3, G3},
910 {G3, G3},
911 {DS3, DS3},
912 {DS3, DS3},
913 {DS3, DS3},
914 {DS3, DS3},
915
916 {DS3, DS3},
917 {DS3, DS3},
918 {DS3, DS3},
919 {DS3, DS3},
920 {DS3, DS3},
921 {DS3, DS3},
922 {DS3, DS3},
923 {DS3, DS3},
924 {END, END}};
925
926 // I Write Sins But Not Tragedies by Panic! At The Disco, Hard-Coded
927 const float SINS_NOTES[][2] = {
928
929 {REST, A1},
930
931 {REST, E2},
932 {REST, C3},
933 {REST, C3},
934 {REST, REST},
935 {REST, E2},
936 {REST, B2},
```

```
937  {REST, B2},
938
939  {REST, A1},
940  {REST, E2},
941  {REST, C3},
942  {REST, C3},
943  {REST, REST},
944  {REST, E2},
945  {REST, B2},
946  {REST, B2},
947
948  {REST, A1},
949  {REST, E2},
950  {REST, C3},
951  {REST, C3},
952  {REST, REST},
953  {REST, E2},
954  {REST, B2},
955  {REST, B2},
956
957  {REST, A1},
958  {REST, E2},
959  {REST, C3},
960  {REST, C3},
961  {REST, REST},
962  {REST, E2},
963  {REST, B2},
964  {REST, B2},
965
966  {REST, D2},
967  {REST, C3},
968  {REST, D3},
969  {REST, D3},
970  {REST, REST},
971  {REST, C3},
972  {REST, D3},
973  {REST, D3},
974
975  {REST, D2},
976  {REST, C3},
977  {REST, D3},
978  {REST, D3},
979  {REST, REST},
980  {REST, C3},
981  {REST, D3},
982  {REST, D3},
983
984  {REST, F2},
985  {REST, B2},
986  {REST, C3},
987  {REST, C3},
988  {REST, REST},
989  {REST, B2},
990  {REST, C3},
991  {REST, C3},
992
993  {REST, F2},
994  {REST, B2},
995  {REST, C3},
996  {REST, C3},
997  {REST, REST},
998  {REST, B2},
999  {REST, C3},
1000 {REST, C3},
1001
1002 {REST, A1},
1003 {REST, E2},
1004 {REST, C3},
1005 {REST, C3},
1006 {REST, REST},
1007 {A2, E2},
1008 {C3, C3},
```

```
1009 {B2, C3},
1010
1011 {B2, A1},
1012 {A2, E2},
1013 {A2, C3},
1014 {REST, C3},
1015 {REST, REST},
1016 {A2, E2},
1017 {C3, B2},
1018 {B2, B2},
1019
1020 {B2, A1},
1021 {A2, E2},
1022 {C3, C3},
1023 {B2, C3},
1024 {B2, REST},
1025 {A2, E2},
1026 {C3, B2},
1027 {B2, B2},
1028
1029 {B2, A1},
1030 {A2, E2},
1031 {C3, C3},
1032 {B2, C3},
1033 {B2, REST},
1034 {A2, E2},
1035 {A2, B2},
1036 {A2, B2},
1037
1038 {D3, D2},
1039 {REST, C3},
1040 {D3, D3},
1041 {REST, D3},
1042 {D3, REST},
1043 {D3, C3},
1044 {E3, D3},
1045 {C3, D3},
1046
1047 {REST, D2},
1048 {REST, E3},
1049 {A2, D3},
1050 {A2, D3},
1051 {A2, REST},
1052 {A2, C3},
1053 {G2, D3},
1054 {G2, D3},
1055
1056 {D3, F2},
1057 {D3, B2},
1058 {C3, D3},
1059 {B2, D3},
1060 {B2, REST},
1061 {C3, C3},
1062 {B2, D3},
1063 {B2, D3},
1064
1065 {A2, F2},
1066 {G2, B2},
1067 {G2, C3},
1068 {G2, C3},
1069 {G2, REST},
1070 {G2, B2},
1071 {REST, C3},
1072 {REST, C3},
1073
1074 {REST, A1},
1075 {G3, E2},
1076 {G3, C2},
1077 {G3, C2},
1078 {F3, REST},
1079 {F3, E2},
1080 {E3, D2},
```

```
1081 {E3, D2},
1082
1083 {D3, A1},
1084 {C2, E2},
1085 {REST, C3},
1086 {REST, C3},
1087 {REST, REST},
1088 {A2, E2},
1089 {C3, B2},
1090 {B2, B2},
1091
1092 {B2, A1},
1093 {A2, E2},
1094 {C3, C3},
1095 {B2, C3},
1096 {B2, REST},
1097 {A2, E2},
1098 {C3, B2},
1099 {A2, B2},
1100
1101 {D3, A1},
1102 {REST, E2},
1103 {D3, C3},
1104 {REST, C3},
1105 {D3, REST},
1106 {REST, E2},
1107 {E3, B2},
1108 {C3, B2},
1109
1110 {D3, D2},
1111 {REST, C3},
1112 {D3, D3},
1113 {REST, D3},
1114 {D3, REST},
1115 {REST, C3},
1116 {E3, D3},
1117 {C3, D3},
1118
1119 {REST, D2},
1120 {REST, C3},
1121 {A2, D3},
1122 {A2, D3},
1123 {A2, REST},
1124 {G2, C3},
1125 {G2, D3},
1126 {G2, D3},
1127
1128 {D3, F2},
1129 {D3, B2},
1130 {D3, C3},
1131 {REST, C3},
1132 {D3, REST},
1133 {REST, B2},
1134 {E3, C3},
1135 {REST, C3},
1136
1137 {C3, A2},
1138 {C3, A2},
1139 {REST, REST},
1140 {REST, REST},
1141 {REST, REST},
1142 {REST, REST},
1143 {REST, REST},
1144 {REST, REST},
1145
1146 {G3, G3},
1147 {G3, G3},
1148 {E3, E3},
1149 {E3, E3},
1150 {D3, D3},
1151 {REST, REST},
1152 {D3, D3},
```

```
1153 {C3, C3},
1154
1155 {G3, G3},
1156 {F3, F3},
1157 {E3, E3},
1158 {E3, E3},
1159 {E3, E3},
1160 {E3, E3},
1161 {F3, F3},
1162 {E3, E3},
1163
1164 {E3, E3},
1165 {D3, D3},
1166 {C3, C3},
1167 {C3, C3},
1168 {REST, REST},
1169 {REST, REST},
1170 {REST, REST},
1171 {REST, REST},
1172
1173 {D3, D3},
1174 {D3, D3},
1175 {C3, C3},
1176 {G3, G3},
1177 {G3, G3},
1178 {E3, E3},
1179 {E3, E3},
1180 {D3, D3},
1181
1182 {D3, D3},
1183 {E3, E3},
1184 {E3, E3},
1185 {REST, REST},
1186 {REST, REST},
1187 {C3, C3},
1188 {D3, D3},
1189 {D3, D3},
1190
1191 {D3, D3},
1192 {D3, D3},
1193 {C3, C3},
1194 {D3, D3},
1195 {REST, REST},
1196 {C3, C3},
1197 {REST, REST},
1198 {D3, D3},
1199
1200 {D3, D3},
1201 {E3, E3},
1202 {C3, C3},
1203 {C3, C3},
1204 {REST, REST},
1205 {C3, C3},
1206 {D3, D3},
1207 {C3, C3},
1208
1209 {D3, D3},
1210 {G3, G3},
1211 {F3, F3},
1212 {E3, E3},
1213 {D3, D3},
1214 {D3, D3},
1215 {D3, D3},
1216 {C3, C3},
1217
1218 {G3, G3},
1219 {G3, G3},
1220 {G3, G3},
1221 {E3, E3},
1222 {E3, E3},
1223 {E3, E3},
1224 {D3, D3},
```

```
1225 {D3, D3},
1226
1227 {REST, REST},
1228 {G3, G3},
1229 {F3, F3},
1230 {E3, E3},
1231 {E3, E3},
1232 {E3, E3},
1233 {E3, E3},
1234 {A3, A3},
1235
1236 {C4, C4},
1237 {C4, C4},
1238 {B3, B3},
1239 {A3, A3},
1240 {REST, REST},
1241 {REST, REST},
1242 {REST, REST},
1243 {REST, REST},
1244
1245 {D3, D3},
1246 {D3, D3},
1247 {C3, C3},
1248 {G3, G3},
1249 {G3, G3},
1250 {E3, E3},
1251 {E3, E3},
1252 {D3, D3},
1253
1254 {D3, D3},
1255 {E3, E3},
1256 {E3, E3},
1257 {REST, REST},
1258 {REST, REST},
1259 {C3, C3},
1260 {D3, D3},
1261 {D3, D3},
1262
1263 {D3, D3},
1264 {D3, D3},
1265 {C3, C3},
1266 {D3, D3},
1267 {REST, REST},
1268 {C3, C3},
1269 {REST, REST},
1270 {D3, D3},
1271
1272 {D3, D3},
1273 {E3, E3},
1274 {C3, C3},
1275 {C3, C3},
1276 {REST, REST},
1277 {REST, REST},
1278 {REST, REST},
1279 {A2, A2},
1280
1281 {A2, A2},
1282 {B2, B2},
1283 {B2, B2},
1284 {C3, C3},
1285 {C3, C3},
1286 {D3, D3},
1287 {D3, D3},
1288 {D3, D3},
1289
1290 {A2, A1},
1291 {A2, A1},
1292 {A2, A1},
1293 {A2, A1},
1294
1295 {END, END}};
1296
```

```
1297
1298 // Mary Had A Little Lamb, Hard-Coded
1299 const float LITTLE_LAMB_NOTES[][2] = {
1300
1301
1302 {B3, C2},
1303 {B3, REST},
1304
1305 {A3, C2},
1306 {A3, REST},
1307
1308 {G3, C2},
1309 {G3, REST},
1310
1311 {A3, C2},
1312 {A3, REST},
1313
1314 {B3, C2},
1315 {B3, B2},
1316
1317 {B3, C2},
1318 {B3, B2},
1319
1320 {B3, B2},
1321 {B3, B2},
1322 {B3, B2},
1323
1324 {REST, REST},
1325
1326 {A3, A2},
1327 {A3, A2},
1328
1329 {A3, A2},
1330 {A3, A2},
1331
1332 {A3, A2},
1333 {A3, A2},
1334 {A3, A2},
1335
1336 {REST, REST},
1337
1338 {B3, REST},
1339 {B3, REST},
1340
1341 {D3, REST},
1342 {D3, REST},
1343
1344 {D3, D2},
1345 {D3, D2},
1346 {D3, D2},
1347 {REST, REST},
1348
1349 {B3, C2},
1350 {B3, REST},
1351
1352 {A3, C2},
1353 {A3, REST},
1354
1355 {G3, C2},
1356 {G3, REST},
1357
1358 {A3, 125},
1359 {A3, REST},
1360
1361 {B3, B2},
1362 {B3, B2},
1363
1364 {B3, B2},
1365 {B3, B2},
1366
1367 {B3, B2},
1368 {B3, B2},
```



```
1369 {B3, B2},
1370
1371 {REST, REST},
1372
1373 {G3, C2},
1374 {G3, REST},
1375
1376 {G3, C2},
1377 {G3, REST},
1378
1379 {B3, B2},
1380 {B3, B2},
1381
1382 {A3, A2},
1383 {A3, A2},
1384 {G3, G2},
1385 {G3, G2},
1386 {END, END}};
1387
1388
1389 const float MONSTERS_NOTES[][2] = {
1390     {REST, C2},
1391     {G2, G2},
1392     {F2, F2},
1393     {G2, G2},
1394     {AS2, AS2},
1395     {AS2, AS2},
1396     {G2, G2},
1397     {D3, D3},
1398     {C3, C3},
1399     {AS2, AS2},
1400     {REST, F1},
1401     {REST, F1},
1402     {AS1, AS1},
1403     {A1, A1},
1404     {F2, F2},
1405     {DS2, DS2},
1406     {D2, D2},
1407     {C2, C2},
1408     {AS2, AS1},
1409     {F1, F1},
1410     {AS2, AS2},
1411     {G1, G1},
1412     {AS1, AS1},
1413     {D2, D2},
1414
1415     {G3, G3},
1416     {D3, D3},
1417     {AS2, AS2},
1418     {G2, G2},
1419
1420     {F3, F2},
1421     {AS2, AS2},
1422     {G2, G2},
1423     {F2, F2},
1424
1425     {REST, C2},
1426     {G2, G2},
1427     {F2, F2},
1428     {G2, G2},
1429     {AS2, AS2},
1430     {AS2, AS2},
1431     {G2, G2},
1432     {D3, D3},
1433     {C3, C3},
1434     {AS2, AS2},
1435     {REST, F1},
1436     {REST, F1},
1437     {AS1, AS1},
1438     {A1, A1},
1439     {F2, F2},
1440     {DS2, DS2},
```

```
1441     {D2, D2},
1442     {C2, C2},
1443     {AS1, AS2},
1444     {F1, F1},
1445     {AS2, AS2},
1446     {G1, G1},
1447     {AS1, AS1},
1448     {D2, D2},
1449
1450     {G3, G3},
1451     {D3, D3},
1452     {AS2, AS2},
1453     {G2, G2},
1454
1455     {F3, F2},
1456     {AS2, AS2},
1457     {G2, G2},
1458     {F2, F2},
1459
1460     {END, END}};
1461
1462
1463     const float YOSHI_NOTES[][2] = {
1464
1465     {REST ,AS2},
1466     {REST ,AS2},
1467     {REST ,AS2},
1468     {REST ,F3},
1469     {REST ,F3},
1470     {REST ,F3},
1471     {REST ,F2},
1472     {REST ,F2},
1473
1474     {REST ,AS2},
1475     {REST ,AS2},
1476     {REST ,F3},
1477     {REST ,F3},
1478     {REST ,F3},
1479     {REST ,F3},
1480     {REST ,F2},
1481     {REST ,F2},
1482
1483     {REST ,AS2},
1484     {REST ,AS2},
1485     {REST ,AS2},
1486     {REST ,F3},
1487     {REST ,F3},
1488     {REST ,F3},
1489     {REST ,F2},
1490     {REST ,F2},
1491
1492     {REST ,AS2},
1493     {REST ,AS2},
1494     {REST ,F3},
1495     {REST ,F3},
1496     {REST ,F3},
1497     {REST ,F3},
1498     {REST ,F2},
1499     {REST ,F2},
1500
1501
1502     {E3 ,AS2},
1503     {E3 ,AS2},
1504     {E3 ,AS2},
1505     {E3 ,F3},
1506     {E3 ,F3},
1507     {E3 ,F3},
1508     {REST ,F2},
1509     {REST ,F2},
1510
1511     {FS3 ,AS2},
1512     {FS3 ,AS2},
```

```
1513 {E3 ,F3},
1514 {E3 ,F3},
1515 {FS3 ,F3},
1516 {FS3 ,F3},
1517 {REST ,F2},
1518 {REST ,F2},
1519
1520 {E3 ,AS2},
1521 {E3 ,AS2},
1522 {E3 ,AS2},
1523 {E3 ,F3},
1524 {E3 ,F3},
1525 {E3 ,F3},
1526 {E3 ,F2},
1527 {E3 ,F2},
1528
1529 {E3 ,AS2},
1530 {E3 ,AS2},
1531 {E3 ,F3},
1532 {E3 ,F3},
1533 {E3 ,F3},
1534 {E3 ,F3},
1535 {E3 ,F2},
1536 {E3 ,F2},
1537
1538 {CS3 ,AS2},
1539 {CS3 ,AS2},
1540 {CS3 ,AS2},
1541 {CS3 ,F3},
1542 {CS3 ,F3},
1543 {CS3 ,F3},
1544 {REST ,F2},
1545 {REST ,F2},
1546
1547 {D3 ,AS2},
1548 {D3 ,AS2},
1549 {CS3 ,F3},
1550 {CS3 ,F3},
1551 {D3 ,F3},
1552 {D3 ,F3},
1553 {REST ,F2},
1554 {REST ,F2},
1555
1556 {CS3 ,AS2},
1557 {CS3 ,AS2},
1558 {CS3 ,AS2},
1559 {CS3 ,F3},
1560 {CS3 ,F3},
1561 {CS3 ,F3},
1562 {CS3 ,F2},
1563 {CS3 ,F2},
1564
1565 {CS3 ,AS2},
1566 {CS3 ,AS2},
1567 {CS3 ,F3},
1568 {CS3 ,F3},
1569 {CS3 ,F3},
1570 {CS3 ,F3},
1571 {CS3 ,F2},
1572 {CS3 ,F2},
1573
1574 {E3 ,AS2},
1575 {E3 ,AS2},
1576 {E3 ,AS2},
1577 {E3 ,F3},
1578 {E3 ,F3},
1579 {E3 ,F3},
1580 {REST ,F2},
1581 {REST ,F2},
1582
1583 {FS3 ,AS2},
1584 {FS3 ,AS2},
```

```
1585 {E3 ,F3},
1586 {E3 ,F3},
1587 {FS3 ,F3},
1588 {FS3 ,F3},
1589 {REST ,F2},
1590 {REST ,F2},
1591
1592
1593 {E3 ,AS2},
1594 {E3 ,AS2},
1595 {E3 ,AS2},
1596 {E3 ,F3},
1597 {E3 ,F3},
1598 {E3 ,F3},
1599 {REST ,F2},
1600 {REST ,F2},
1601
1602 {D3 ,AS2},
1603 {D3 ,AS2},
1604 {CS3 ,F3},
1605 {CS3 ,F3},
1606 {D3 ,F3},
1607 {D3 ,F3},
1608 {REST ,F2},
1609 {REST ,F2},
1610
1611 {CS3 ,AS2},
1612 {CS3 ,AS2},
1613 {CS3 ,AS2},
1614 {CS3 ,F3},
1615 {CS3 ,F3},
1616 {CS3 ,F3},
1617 {REST ,F2},
1618 {REST ,F2},
1619
1620 {B2 ,AS2},
1621 {B2 ,AS2},
1622 {A2 ,F3},
1623 {A2 ,F3},
1624 {B2 ,F3},
1625 {B2 ,F3},
1626 {REST ,F2},
1627 {REST ,F2},
1628
1629 {A2 ,AS2},
1630 {A2 ,AS2},
1631 {A2 ,AS2},
1632 {A2 ,F3},
1633 {A2 ,F3},
1634 {A2 ,F3},
1635 {A2 ,F2},
1636 {A2 ,F2},
1637
1638 {A2 ,AS2},
1639 {A2 ,AS2},
1640 {A2 ,F3},
1641 {A2 ,F3},
1642 {A2 ,F3},
1643 {A2 ,F3},
1644 {A2 ,F2},
1645 {A2 ,F2},
1646 {END, END}};
1647
1648
1649
1650 const float SHAPE_NOTES[][2] = {
1651 {CS2, CS2},
1652 {CS2, CS2},
1653 {CS2, CS2},
1654 {E2, E2},
1655 {E2, E2},
1656 {E2, E2},
```

```
1657 {CS2, CS2},
1658 {CS2, CS2},
1659
1660 {CS2, CS2},
1661 {CS2, CS2},
1662 {CS2, CS2},
1663 {E2, E2},
1664 {E2, E2},
1665 {E2, E2},
1666 {CS2, CS2},
1667 {CS2, CS2},
1668
1669 {CS2, CS2},
1670 {CS2, CS2},
1671 {CS2, CS2},
1672 {E2, E2},
1673 {E2, E2},
1674 {E2, E2},
1675 {CS2, CS2},
1676 {CS2, CS2},
1677
1678 {DS2, DS2},
1679 {DS2, DS2},
1680 {DS2, DS2},
1681 {CS2, CS2},
1682 {CS2, CS2},
1683 {CS2, CS2},
1684 {B1, B1},
1685 {B1, B1},
1686
1687 {CS2, CS2},
1688 {CS2, CS2},
1689 {CS2, CS2},
1690 {E2, E2},
1691 {E2, E2},
1692 {E2, E2},
1693 {CS2, CS2},
1694 {CS2, CS2},
1695
1696 {CS2, CS2},
1697 {CS2, CS2},
1698 {CS2, CS2},
1699 {E2, E2},
1700 {E2, E2},
1701 {E2, E2},
1702 {CS2, CS2},
1703 {CS2, CS2},
1704
1705 {CS2, CS2},
1706 {CS2, CS2},
1707 {CS2, CS2},
1708 {E2, E2},
1709 {E2, E2},
1710 {E2, E2},
1711 {CS2, CS2},
1712 {CS2, CS2},
1713
1714 {DS2, DS2},
1715 {DS2, DS2},
1716 {DS2, DS2},
1717 {CS2, CS2},
1718 {CS2, CS2},
1719 {CS2, CS2},
1720 {B1, REST},
1721 {B1, REST},
1722
1723 {E2, CS2},
1724 {REST, REST},
1725 {E2, REST},
1726 {E2, CS2},
1727 {E2, REST},
1728 {E2, REST},
```

```
1729 {REST, CS2},
1730 {E2, REST},
1731
1732 {E2, FS1},
1733 {E2, REST},
1734 {E2, REST},
1735 {REST, FS1},
1736 {E2, REST},
1737 {REST, REST},
1738 {E2, FS1},
1739 {E2, REST},
1740
1741 {E2, AS1},
1742 {E2, REST},
1743 {E2, REST},
1744 {REST, AS1},
1745 {FS2, REST},
1746 {GS2, REST},
1747 {GS2, AS1},
1748 {GS2, REST},
1749
1750 {GS2, B2},
1751 {GS2, REST},
1752 {GS2, REST},
1753 {GS2, B2},
1754 {REST, REST},
1755 {REST, REST},
1756 {REST, B2},
1757 {REST, REST},
1758
1759 {GS2, CS2},
1760 {FS2, REST},
1761 {GS2, REST},
1762 {GS2, CS2},
1763 {REST, REST},
1764 {GS2, REST},
1765 {REST, CS2},
1766 {GS2, REST},
1767
1768 {FS2, FS1},
1769 {FS2, REST},
1770 {FS2, REST},
1771 {GS2, FS1},
1772 {FS2, REST},
1773 {REST, REST},
1774 {FS2, FS1},
1775 {E2, REST},
1776
1777 {FS2, AS1},
1778 {REST, REST},
1779 {FS2, REST},
1780 {FS2, AS1},
1781 {FS2, REST},
1782 {E2, REST},
1783 {E2, AS1},
1784 {CS2, REST},
1785
1786 {CS2, B1},
1787 {CS2, REST},
1788 {CS2, REST},
1789 {CS2, B1},
1790 {REST, REST},
1791 {REST, REST},
1792 {REST, B1},
1793 {CS2, REST},
1794
1795 {GS2, CS2},
1796 {GS2, REST},
1797 {GS2, REST},
1798 {REST, CS2},
1799 {GS2, REST},
1800 {GS2, REST},
```

```
1801 {GS2, CS2},
1802 {GS2, REST},
1803
1804 {GS2, FS1},
1805 {GS2, REST},
1806 {GS2, REST},
1807 {GS2, FS1},
1808 {REST, REST},
1809 {GS2, REST},
1810 {REST, FS1},
1811 {GS2, REST},
1812
1813 {B2, AS1},
1814 {B2, REST},
1815 {GS2, REST},
1816 {GS2, AS1},
1817 {FS2, REST},
1818 {FS2, REST},
1819 {E2, AS1},
1820 {GS2, REST},
1821
1822 {GS2, B1},
1823 {GS2, REST},
1824 {GS2, REST},
1825 {GS2, B1},
1826 {FS2, REST},
1827 {E2, REST},
1828 {E2, B1},
1829 {E2, REST},
1830
1831 {E2, CS2},
1832 {E2, REST},
1833 {E2, REST},
1834 {B2, CS2},
1835 {B2, REST},
1836 {GS2, REST},
1837 {GS2, CS2},
1838 {GS2, REST},
1839
1840 {FS2, FS1},
1841 {FS2, REST},
1842 {FS2, REST},
1843 {FS2, FS1},
1844 {REST, REST},
1845 {FS2, REST},
1846 {REST, FS1},
1847 {FS2, REST},
1848
1849 {FS2, AS1},
1850 {REST, REST},
1851 {FS2, REST},
1852 {REST, AS1},
1853 {FS2, REST},
1854 {REST, REST},
1855 {E2, AS1},
1856 {CS2, E2},
1857
1858 {CS2, B1},
1859 {GS2, REST},
1860 {GS2, REST},
1861 {GS2, B1},
1862 {B2, REST},
1863 {CS3, REST},
1864 {CS3, B1},
1865 {CS3, REST},
1866
1867 {CS3, CS2},
1868 {REST, REST},
1869 {CS3, REST},
1870 {REST, CS2},
1871 {CS3, REST},
1872 {REST, REST},
```

```
1873 {B2, CS2},
1874 {REST, REST},
1875
1876 {CS3, FS1},
1877 {CS3, REST},
1878 {GS3, REST},
1879 {GS3, FS1},
1880 {FS3, REST},
1881 {FS3, REST},
1882 {FS3, FS1},
1883 {FS3, REST},
1884
1885 {E3, AS1},
1886 {FS3, REST},
1887 {FS3, REST},
1888 {GS3, AS1},
1889 {FS3, REST},
1890 {FS3, REST},
1891 {E3, AS1},
1892 {E3, E2},
1893
1894 {CS3, B1},
1895 {CS3, REST},
1896 {E3, REST},
1897 {E3, B1},
1898 {GS3, REST},
1899 {FS3, REST},
1900 {FS3, B1},
1901 {FS3, REST},
1902
1903
1904 {CS3, CS2},
1905 {CS3, REST},
1906 {CS3, REST},
1907 {CS3, CS2},
1908 {B3, REST},
1909 {FS3, REST},
1910 {GS3, CS2},
1911 {GS3, REST},
1912
1913 {FS3, FS1},
1914 {E3, REST},
1915 {E3, REST},
1916 {REST, FS1},
1917 {E3, REST},
1918 {E3, REST},
1919 {E3, FS1},
1920 {E3, REST},
1921
1922 {E3, AS1},
1923 {FS3, REST},
1924 {FS3, REST},
1925 {GS3, AS1},
1926 {FS3, REST},
1927 {FS3, REST},
1928 {E3, AS1},
1929 {E3, E2},
1930
1931 {FS3, B1},
1932 {FS3, REST},
1933 {E3, REST},
1934 {E3, B1},
1935 {CS3, REST},
1936 {REST, REST},
1937 {CS3, B1},
1938 {REST, REST},
1939
1940 {CS3, CS2},
1941 {CS3, REST},
1942 {CS3, REST},
1943 {REST, CS2},
1944 {CS3, REST},
```



```
1945 {CS3, REST},
1946 {B3, CS2},
1947 {B3, REST},
1948
1949 {CS3, FS1},
1950 {CS3, REST},
1951 {GS3, REST},
1952 {GS3, FS1},
1953 {FS3, REST},
1954 {FS3, REST},
1955 {FS3, FS1},
1956 {FS3, REST},
1957
1958 {E3, AS1},
1959 {FS3, REST},
1960 {FS3, REST},
1961 {GS3, AS1},
1962 {FS3, REST},
1963 {FS3, REST},
1964 {E3, AS1},
1965 {E3, E2},
1966
1967 {CS3, B1},
1968 {CS3, REST},
1969 {E3, REST},
1970 {E3, B1},
1971 {GS3, REST},
1972 {FS3, REST},
1973 {FS3, B1},
1974 {REST, REST},
1975
1976
1977 {CS3, CS2},
1978 {CS3, REST},
1979 {CS3, REST},
1980 {CS3, CS2},
1981 {B3, REST},
1982 {FS3, REST},
1983 {GS3, CS2},
1984 {GS3, REST},
1985
1986 {FS3, FS1},
1987 {E3, REST},
1988 {E3, REST},
1989 {REST, FS1},
1990 {CS3, REST},
1991 {REST, REST},
1992 {CS3, FS1},
1993 {REST, REST},
1994
1995 {B3, AS1},
1996 {FS3, REST},
1997 {GS3, REST},
1998 {GS3, AS1},
1999 {FS3, REST},
2000 {E3, REST},
2001 {E3, AS1},
2002 {E3, E2},
2003
2004 {CS3, B1},
2005 {CS3, REST},
2006 {GS2, REST},
2007 {GS2, REST},
2008 {B2, REST},
2009 {B2, REST},
2010 {CS3, B1},
2011 {CS3, REST},
2012
2013
2014 {CS3, CS2},
2015 {CS3, REST},
2016 {REST, REST},
```

```
2017 {REST, CS2},
2018 {REST, REST},
2019 {REST, REST},
2020 {E3, CS2},
2021 {FS3, REST},
2022
2023 {GS3, FS1},
2024 {GS3, REST},
2025 {FS3, REST},
2026 {E3, FS1},
2027 {E3, REST},
2028 {REST, REST},
2029 {FS3, FS1},
2030 {REST, REST},
2031
2032 {FS3, AS1},
2033 {FS3, REST},
2034 {FS3, REST},
2035 {REST, AS1},
2036 {REST, REST},
2037 {CS3, REST},
2038 {E3, AS1},
2039 {FS3, E2},
2040
2041 {GS3, B1},
2042 {GS3, REST},
2043 {FS3, REST},
2044 {E3, B1},
2045 {E3, REST},
2046 {E3, REST},
2047 {FS3, B1},
2048 {FS3, REST},
2049
2050 {CS3, CS2},
2051 {CS3, REST},
2052 {CS3, REST},
2053 {CS3, CS2},
2054 {REST, REST},
2055 {CS3, REST},
2056 {E3, CS2},
2057 {FS3, REST},
2058
2059 {GS3, FS1},
2060 {GS3, REST},
2061 {CS3, REST},
2062 {CS3, FS1},
2063 {E3, REST},
2064 {E3, REST},
2065 {FS3, FS1},
2066 {FS3, REST},
2067
2068 {FS3, AS1},
2069 {FS3, REST},
2070 {FS3, REST},
2071 {FS3, AS1},
2072 {REST, REST},
2073 {REST, REST},
2074 {E3, AS1},
2075 {FS3, E2},
2076
2077 {GS3, B1},
2078 {GS3, REST},
2079 {FS3, REST},
2080 {E3, B1},
2081 {FS3, REST},
2082 {FS3, REST},
2083 {FS3, B1},
2084 {CS3, REST},
2085
2086 {CS3, CS2},
2087 {CS3, REST},
2088 {CS3, REST},
```

```
2089 {CS3, CS2},
2090 {REST, REST},
2091 {REST, REST},
2092 {E3, CS2},
2093 {E3, REST},
2094
2095 {GS3, FS1},
2096 {GS3, REST},
2097 {B3, REST},
2098 {GS3, FS1},
2099 {FS3, REST},
2100 {FS3, REST},
2101 {E3, FS1},
2102 {E3, REST},
2103
2104 {FS3, AS1},
2105 {FS3, REST},
2106 {FS3, REST},
2107 {FS3, AS1},
2108 {REST, REST},
2109 {CS3, REST},
2110 {E3, AS1},
2111 {FS3, E2},
2112
2113 {GS3, B1},
2114 {GS3, REST},
2115 {FS3, REST},
2116 {E3, B1},
2117 {E3, REST},
2118 {E3, REST},
2119 {CS3, B1},
2120 {FS3, REST},
2121
2122 {CS3, CS2},
2123 {REST, REST},
2124 {CS3, REST},
2125 {CS3, CS2},
2126 {CS3, REST},
2127 {CS3, REST},
2128 {E3, CS2},
2129 {FS3, REST},
2130
2131 {GS3, FS1},
2132 {GS3, REST},
2133 {CS3, REST},
2134 {CS3, FS1},
2135 {E3, REST},
2136 {E3, REST},
2137 {FS3, FS1},
2138 {FS3, REST},
2139
2140 {FS3, AS1},
2141 {FS3, REST},
2142 {FS3, REST},
2143 {FS3, AS1},
2144 {FS3, REST},
2145 {FS3, REST},
2146 {E3, AS1},
2147 {FS3, E2},
2148
2149 {GS3, B1},
2150 {GS3, REST},
2151 {FS3, REST},
2152 {E3, B1},
2153 {FS3, REST},
2154 {FS3, REST},
2155 {FS3, B1},
2156 {CS3, REST},
2157
2158 {CS3, CS2},
2159 {REST, REST},
2160 {CS3, REST},
```

```
2161 {REST, CS2},
2162 {E3, REST},
2163 {REST, REST},
2164 {E3, CS2},
2165 {REST, REST},
2166
2167 {FS3, FS1},
2168 {REST, REST},
2169 {FS3, REST},
2170 {REST, FS1},
2171 {GS3, REST},
2172 {REST, REST},
2173 {GS3, FS1},
2174 {REST, REST},
2175
2176 {GS3, AS1},
2177 {GS3, REST},
2178 {GS3, REST},
2179 {GS3, AS1},
2180 {GS3, REST},
2181 {GS3, REST},
2182 {E3, AS1},
2183 {FS3, E2},
2184
2185 {GS3, B1},
2186 {GS3, REST},
2187 {FS3, REST},
2188 {E3, B1},
2189 {FS3, REST},
2190 {FS3, REST},
2191 {FS3, B1},
2192 {CS3, REST},
2193
2194 {CS3, CS2},
2195 {REST, REST},
2196 {CS3, REST},
2197 {REST, CS2},
2198 {E3, REST},
2199 {REST, REST},
2200 {E3, CS2},
2201 {REST, REST},
2202
2203 {FS3, FS1},
2204 {REST, REST},
2205 {FS3, REST},
2206 {REST, FS1},
2207 {GS3, REST},
2208 {REST, REST},
2209 {GS3, FS1},
2210 {REST, REST},
2211
2212 {GS3, AS1},
2213 {GS3, REST},
2214 {GS3, REST},
2215 {GS3, AS1},
2216 {GS3, REST},
2217 {GS3, REST},
2218 {E3, AS1},
2219 {FS3, E2},
2220
2221 {GS3, B1},
2222 {GS3, REST},
2223 {FS3, REST},
2224 {E3, B1},
2225 {FS3, REST},
2226 {FS3, REST},
2227 {FS3, B1},
2228 {CS3, REST},
2229 {CS3, REST},
2230 {CS3, REST},
2231 {CS3, REST},
2232 {CS3, REST},
```

```
2233 {CS3, REST},
2234 {CS3, REST},
2235 {CS3, REST},
2236 {END, END}};
2237
2238
2239 //99 Luftballoons
2240
2241 const float BALLOONS_NOTES[][2] = {
2242     {REST, E2},
2243     {REST, E2},
2244     {REST, E2},
2245     {REST, E2},
2246     {REST, E2},
2247     {REST, E2},
2248     {REST, E2},
2249     {REST, E2},
2250     {REST, E2},
2251     {REST, E2},
2252     {REST, E2},
2253     {REST, E2},
2254     {REST, E2},
2255     {REST, E2},
2256     {REST, E2},
2257     {REST, E2},
2258     {REST, E2},
2259     {FS2, E2},
2260     {FS2, E2},
2261     {GS2, E2},
2262     {E2, E2},
2263     {E2, E2},
2264     {E2, E2},
2265     {GS2, E2},
2266     {GS2, E2},
2267     {FS2, CS2},
2268     {FS2, CS2},
2269     {E2, CS2},
2270     {CS2, CS2},
2271     {CS2, CS2},
2272     {CS2, CS2},
2273     {CS2, CS2},
2274     {CS2, CS2},
2275     {A2, E2},
2276     {A2, E2},
2277     {A2, E2},
2278     {A2, E2},
2279     {A2, E2},
2280     {A2, E2},
2281     {A2, E2},
2282     {A2, E2},
2283     {A2, FS2},
2284     {A2, FS2},
2285     {GS2, FS2},
2286     {FS2, FS2},
2287     {FS2, FS2},
2288     {E2, FS2},
2289     {E2, FS2},
2290     {E2, FS2},
2291     {FS2, E2},
2292     {FS2, E2},
2293     {GS2, E2},
2294     {E2, E2},
2295     {E2, E2},
2296     {E2, E2},
2297     {GS2, E2},
2298     {GS2, E2},
2299     {FS2, CS2},
2300     {FS2, CS2},
2301     {E2, CS2},
2302     {CS2, CS2},
2303     {CS2, CS2},
2304     {CS2, CS2},
```

```
2305     {CS2,   CS2},
2306     {CS2,   CS2},
2307     {E2,   E2},
2308     {E2,   E2},
2309     {CS2,   E2},
2310     {E2,   E2},
2311     {E2,   E2},
2312     {E2,   E2},
2313     {CS2,   E2},
2314     {CS2,   E2},
2315     {E2,   FS2},
2316     {CS2,   FS2},
2317     {CS2,   FS2},
2318     {E2,   FS2},
2319     {E2,   FS2},
2320     {E2,   FS2},
2321     {E2,   FS2},
2322     {E2,   FS2},
2323     {FS2,   E2},
2324     {FS2,   E2},
2325     {FS2,   E2},
2326     {GS2,   E2},
2327     {E2,   E2},
2328     {E2,   E2},
2329     {E2,   E2},
2330     {E2,   E2},
2331     {FS2,   CS2},
2332     {FS2,   CS2},
2333     {E2,   CS2},
2334     {CS2,   CS2},
2335     {REST,  CS2},
2336     {REST,  CS2},
2337     {REST,  CS2},
2338     {CS2,   CS2},
2339     {A2,   E2},
2340     {A2,   E2},
2341     {A2,   E2},
2342     {A2,   E2},
2343     {A2,   E2},
2344     {A2,   E2},
2345     {A2,   E2},
2346     {A2,   E2},
2347     {A2,   FS2},
2348     {A2,   FS2},
2349     {GS2,   FS2},
2350     {FS2,   FS2},
2351     {FS2,   FS2},
2352     {FS2,   FS2},
2353     {E2,   FS2},
2354     {E2,   FS2},
2355     {FS2,   E2},
2356     {FS2,   E2},
2357     {GS2,   E2},
2358     {E2,   E2},
2359     {E2,   E2},
2360     {E2,   E2},
2361     {GS2,   E2},
2362     {GS2,   E2},
2363     {FS2,   CS2},
2364     {FS2,   CS2},
2365     {E2,   CS2},
2366     {CS2,   CS2},
2367     {CS2,   CS2},
2368     {CS2,   CS2},
2369     {CS2,   CS2},
2370     {CS2,   CS2},
2371     {E2,   E2},
2372     {E2,   E2},
2373     {CS2,   E2},
2374     {E2,   E2},
2375     {REST,  E2},
2376     {REST,  E2},
```

```
2377     {CS2,  E2},
2378     {CS2,  E2},
2379     {FS2,  FS2},
2380     {FS2,  FS2},
2381     {FS2,  FS2},
2382     {FS2,  FS2},
2383     {FS2,  FS2},
2384     {FS2,  FS2},
2385     {E2,  FS2},
2386     {E2,  FS2},
2387     {REST, REST},
2388     {REST, REST},
2389     {REST, REST},
2390     {REST, REST},
2391     {REST, REST},
2392     {REST, REST},
2393     {REST, REST},
2394     {REST, REST},
2395     {E2,  REST},
2396     {REST, REST},
2397     {E2,  REST},
2398     {CS2,  REST},
2399     {CS2,  REST},
2400     {E2,  REST},
2401     {E2,  REST},
2402     {CS2,  REST},
2403     {CS2,  REST},
2404     {G2,  REST},
2405     {GS2,  REST},
2406     {FS2,  REST},
2407     {E2,  REST},
2408     {REST, REST},
2409     {E2,  REST},
2410     {CS2,  REST},
2411     {E2,  REST},
2412     {REST, REST},
2413     {E2,  REST},
2414     {CS2,  REST},
2415     {CS2,  REST},
2416     {E2,  REST},
2417     {E2,  REST},
2418     {CS2,  REST},
2419     {CS2,  REST},
2420     {E1,  REST},
2421     {E1,  REST},
2422     {E1,  REST},
2423     {E1,  REST},
2424     {REST, REST},
2425     {E1,  REST},
2426     {CS2,  REST},
2427     {E2,  REST},
2428     {REST, REST},
2429     {E2,  REST},
2430     {CS2,  REST},
2431     {CS2,  REST},
2432     {E2,  REST},
2433     {E2,  REST},
2434     {CS2,  REST},
2435     {CS2,  REST},
2436     {G2,  REST},
2437     {GS2,  REST},
2438     {FS2,  REST},
2439     {E2,  REST},
2440     {REST, REST},
2441     {E2,  REST},
2442     {CS2,  REST},
2443     {E2,  REST},
2444     {REST, REST},
2445     {E2,  REST},
2446     {CS2,  REST},
2447     {CS2,  REST},
2448     {E2,  REST},
```

```
2449     {E2,  REST},
2450     {CS2,  REST},
2451     {CS2,  REST},
2452     {E1,  REST},
2453     {E1,  REST},
2454     {E1,  REST},
2455     {E1,  REST},
2456     {REST, REST},
2457     {B2,  REST},
2458     {CS3,  REST},
2459     {E3,  REST},
2460     {REST, REST},
2461     {E3,  REST},
2462     {CS3,  REST},
2463     {CS3,  REST},
2464     {E3,  REST},
2465     {E3,  REST},
2466     {CS3,  REST},
2467     {CS3,  REST},
2468     {G3,  REST},
2469     {GS3,  REST},
2470     {FS3,  REST},
2471     {E3,  REST},
2472     {REST, REST},
2473     {E3,  REST},
2474     {CS3,  REST},
2475     {E3,  REST},
2476     {REST, REST},
2477     {E3,  REST},
2478     {CS3,  REST},
2479     {CS3,  REST},
2480     {E3,  REST},
2481     {E3,  REST},
2482     {CS3,  REST},
2483     {CS3,  REST},
2484     {B2,  REST},
2485     {B2,  REST},
2486     {B2,  REST},
2487     {B2,  REST},
2488     {REST, REST},
2489     {B2,  REST},
2490     {CS3,  REST},
2491     {E3,  REST},
2492     {REST, REST},
2493     {E3,  REST},
2494     {CS3,  REST},
2495     {CS3,  REST},
2496     {E3,  REST},
2497     {E3,  REST},
2498     {CS3,  REST},
2499     {CS3,  REST},
2500     {G3,  REST},
2501     {GS3,  REST},
2502     {FS3,  REST},
2503     {E3,  REST},
2504     {REST, REST},
2505     {E3,  REST},
2506     {CS3,  REST},
2507     {E3,  REST},
2508     {REST, REST},
2509     {E3,  REST},
2510     {CS3,  REST},
2511     {CS3,  REST},
2512     {E3,  REST},
2513     {E3,  REST},
2514     {CS3,  REST},
2515     {CS3,  REST},
2516     {B2,  REST},
2517     {B2,  REST},
2518     {B2,  REST},
2519     {B2,  REST},
2520     {REST, REST},
```



```
2521     {B2,  REST},
2522     {B2,  REST},
2523     {B2,  REST},
2524     {FS2,  E1},
2525     {FS2,  E1},
2526     {FS2,  E1},
2527     {FS2,  E1},
2528     {GS2,  E1},
2529     {GS2,  E1},
2530     {E2,  E1},
2531     {E2,  E1},
2532     {E2,  E1},
2533     {E2,  E1},
2534     {E2,  E1},
2535     {E2,  E1},
2536     {GS2,  E1},
2537     {GS2,  E1},
2538     {GS2,  E1},
2539     {GS2,  E1},
2540     {FS2,  FS1},
2541     {FS2,  FS1},
2542     {FS2,  FS1},
2543     {FS2,  FS1},
2544     {E2,  FS1},
2545     {E2,  FS1},
2546     {CS2,  FS1},
2547     {CS2,  FS1},
2548     {CS2,  FS1},
2549     {CS2,  FS1},
2550     {CS2,  FS1},
2551     {CS2,  FS1},
2552     {REST, FS1},
2553     {REST, FS1},
2554     {CS2,  FS1},
2555     {CS2,  FS1},
2556     {A2,  A1},
2557     {A2,  A1},
2558     {A2,  A1},
2559     {A2,  A1},
2560     {A2,  A1},
2561     {A2,  A1},
2562     {A2,  A1},
2563     {A2,  A1},
2564     {REST, A1},
2565     {REST, A1},
2566     {REST, A1},
2567     {REST, A1},
2568     {A2,  A1},
2569     {A2,  A1},
2570     {A2,  A1},
2571     {A2,  A1},
2572     {A2,  E1},
2573     {A2,  E1},
2574
2575     {END, END}};
2576
2577
2578
2579
2580
2581
2582
2583     const float CLOSER_NOTES[][2] = {
2584         {AS3,AS3},
2585         {GS3,GS3},
2586         {GS3,GS3},
2587         {AS3,AS3},
2588         {AS3,AS3},
2589         {GS3,GS3},
2590         {GS3,GS3},
2591         {C4,C4},
2592         {C4,C4},
```

```
2593 {GS3,GS3},
2594 {GS3,GS3},
2595 {AS3,AS3},
2596 {AS3,AS3},
2597 {GS3,GS3},
2598 {GS3,GS3},
2599 {AS3,AS3},
2600 {AS3,AS3},
2601 {GS3,GS3},
2602 {GS3,GS3},
2603 {AS3,AS3},
2604 {AS3,AS3},
2605 {GS3,GS3},
2606 {GS3,GS3},
2607 {C4,C4},
2608 {C4,C4},
2609 {GS3,GS3},
2610 {GS3,GS3},
2611 {AS3,AS3},
2612 {AS3,AS3},
2613 {GS3,GS3},
2614 {GS3,GS3},
2615 {AS3,AS3},
2616 {AS3,AS3},
2617 {C4,C4},
2618 {GS3,GS3},
2619 {AS3,AS3},
2620 {AS3,AS3},
2621 {GS3,GS3},
2622 {GS3,GS3},
2623 {C4,C4},
2624 {C4,C4},
2625 {GS3,GS3},
2626 {GS3,GS3},
2627 {AS3,AS3},
2628 {AS3,AS3},
2629 {GS3,GS3},
2630 {GS3,GS3},
2631 {AS3,AS3},
2632 {AS3,AS3},
2633 {GS3,GS3},
2634 {GS3,GS3},
2635 {AS3,AS3},
2636 {AS3,AS3},
2637 {GS3,GS3},
2638 {GS3,GS3},
2639 {C4,C4},
2640 {C4,C4},
2641 {GS3,GS3},
2642 {GS3,GS3},
2643 {AS3,AS3},
2644 {AS3,AS3},
2645 {GS3,GS3},
2646 {GS3,GS3},
2647 {AS3,AS3},
2648 {AS3,AS3},
2649 {GS3,GS3},
2650 {GS3,GS3},
2651 {AS3,AS3},
2652 {AS3,AS3},
2653 {GS3,GS3},
2654 {GS3,GS3},
2655 {C4,C4},
2656 {C4,C4},
2657 {GS3,GS3},
2658 {GS3,GS3},
2659 {AS3,AS3},
2660 {AS3,AS3},
2661 {GS3,GS3},
2662 {GS3,GS3},
2663 {AS3,AS3},
2664 {AS3,AS3},
```

```
2665 {GS3,GS3},
2666 {GS3,GS3},
2667 {AS3,AS3},
2668 {AS3,AS3},
2669 {GS3,GS3},
2670 {GS3,GS3},
2671 {C4,C4},
2672 {C4,C4},
2673 {GS3,GS3},
2674 {GS3,GS3},
2675 {AS3,AS3},
2676 {AS3,AS3},
2677 {GS3,GS3},
2678 {GS3,GS3},
2679 {AS3,AS3},
2680 {AS3,AS3},
2681 {C4,C4},
2682 {GS3,GS3},
2683 {AS3,AS3},
2684 {AS3,AS3},
2685 {GS3,GS3},
2686 {GS3,GS3},
2687 {C4,C4},
2688 {C4,C4},
2689 {GS3,GS3},
2690 {GS3,GS3},
2691 {AS3,AS3},
2692 {AS3,AS3},
2693 {GS3,GS3},
2694 {GS3,GS3},
2695 {AS3,AS3},
2696 {AS3,AS3},
2697 {GS3,GS3},
2698 {GS3,GS3},
2699 {AS3,AS3},
2700 {AS3,AS3},
2701 {GS3,GS3},
2702 {GS3,GS3},
2703 {C4,C4},
2704 {C4,C4},
2705 {GS3,GS3},
2706 {GS3,GS3},
2707 {AS3,AS3},
2708 {AS3,AS3},
2709 {GS3,GS3},
2710 {GS3,GS3},
2711 {AS3,AS3},
2712 {END, END}};
2713
2714
2715
2716
2717 const float YMCA_NOTES[][2] = {
2718
2719 {B2, C2 },
2720 {A2, C2 },
2721 {G2, C2 },
2722 {A2, C2 },
2723 {B2, C2 },
2724 {D3, C2 },
2725 {D3, C2 },
2726 {REST, C2 },
2727 {B2, C2 },
2728 {D3, C2 },
2729 {D3, C2 },
2730 {REST, C2 },
2731 {E3, C2 },
2732 {E3, C2 },
2733 {REST, C2 },
2734 {REST, C2 },
2735 {REST, C2 },
2736 {REST, C2 },
```

```
2737 {REST, C2 },
2738 {REST, C2 },
2739 {REST, C2 },
2740 {REST, C2 },
2741 {REST, C2 },
2742 {B2, C2 },
2743 {A2, C2 },
2744 {G2, C2 },
2745 {A2, C2 },
2746 {B2, C2 },
2747 {D3, C2 },
2748 {D3, C2 },
2749 {REST, C2 },
2750 {B2, C2 },
2751 {D3, C2 },
2752 {D3, C2 },
2753 {REST, C2 },
2754 {E3, C2 },
2755 {E3, C2 },
2756 {C3, C2 },
2757 {C3, C2 },
2758 {REST, C2 },
2759 {REST, C2 },
2760 {REST, C2 },
2761 {REST, C2 },
2762 {REST, C2 },
2763 {REST, C2 },
2764 {REST, C2 },
2765 {C3, C2 },
2766 {B2, C2 },
2767 {A2, C2 },
2768 {B2, C2 },
2769 {C3, C2 },
2770 {E3, C2 },
2771 {E3, C2 },
2772 {REST, C2 },
2773 {G3, C2 },
2774 {E3, C2 },
2775 {E3, C2 },
2776 {REST, C2 },
2777 {FS3, C2 },
2778 {FS3, C2 },
2779 {FS3, C2 },
2780 {REST, C2 },
2781 {REST, C2 },
2782 {E3, C2 },
2783 {E3, C2 },
2784 {E3, C2 },
2785 {REST, C2 },
2786 {REST, C2 },
2787 {D3, C2 },
2788 {D3, C2 },
2789 {D3, C2 },
2790 {D3, C2 },
2791 {REST, C2 },
2792 {C3, C2 },
2793 {C3, C2 },
2794 {C3, C2 },
2795 {C3, C2 },
2796 {REST, C2 },
2797 {REST, C2 },
2798 {B2, C2 },
2799 {B2, C2 },
2800 {B2, C2 },
2801 {A2, C2 },
2802 {A2, C2 },
2803 {REST, C2 },
2804 {D3, C2 },
2805 {B2, C2 },
2806 {B2, C2 },
2807 {REST, C2 },
2808 {REST, C2 },
```

```
2809 {REST, C2 },
2810 {REST, C2 },
2811 {REST, C2 },
2812 {REST, C2 },
2813 {REST, C2 },
2814 {B2, C2 },
2815 {A2, C2 },
2816 {G2, C2 },
2817 {A2, C2 },
2818 {B2, C2 },
2819 {D3, C2 },
2820 {D3, C2 },
2821 {D3, C2 },
2822 {B2, C2 },
2823 {D3, C2 },
2824 {D3, C2 },
2825 {REST, C2 },
2826 {E3, C2 },
2827 {B2, C2 },
2828 {END, END}};
2829
2830
2831
2832
2833
2834
2835
2836
```

```
from mido import MidiFile

note_encodings = {
    0: "REST",
    28: "E1",
    29: "F1",
    30: "FS1",
    31: "G1",
    32: "GS1",
    33: "A1",
    34: "AS1",
    35: "E1",
    36: "C2",
    37: "CS2",
    38: "D2",
    39: "DS2",
    40: "E2",
    41: "F2",
    42: "FS2",
    43: "G2",
    44: "GS2",
    45: "A2",
    46: "AS2",
    47: "B2",
    48: "C3",
    49: "CS3",
    50: "D3",
    51: "DS3",
    52: "E3",
    53: "F3",
    54: "FS3",
    55: "G3",
    56: "GS3",
    57: "A3",
    58: "AS3",
    59: "B3",
    60: "C4",
    61: "CS4",
    62: "D4",
    63: "DS4",
    64: "E4"
}

midifile_name = 'pirates.mid'
track_num = 1

mid = MidiFile(midifile_name)
f = open("temp.txt", "w+")

motor_notes = [0, 0]
```

```

motor_on = [0, 0]
deltat_skipped = 0
deltat_per_sixteenth = mid.ticks_per_beat/4;

for i, track in enumerate(mid.tracks):
    if i == track_num:
        print('Track {}: {}'.format(i, track.name))
        for msg in track:
            if (msg.type == 'note_on'):
                if not motor_on[0]:
                    motor_notes[0] = msg.note
                    motor_on[0] = 1
                elif not motor_on[1]:
                    motor_notes[1] = msg.note
                    motor_on[1] = 1
                else:
                    deltat_skipped += msg.time
            elif (msg.type == 'note_off'):
                if(msg.note == motor_notes[0] or msg.note == motor_notes[1]):
                    for i in range((msg.time+deltat_skipped)/deltat_per_sixteenth):
                        f.write("{}\\" + note_encodings[motor_notes[0]] + "\", \\" +
                                note_encodings[motor_notes[1]] + "\",\n")

                    if(msg.note == motor_notes[0]):
                        motor_notes[0] = 0
                        motor_on[0] = 0
                    elif (msg.note == motor_notes[1]):
                        motor_notes[1] = 0
                        motor_on[1] = 0
                    deltat_skipped = 0
                else:
                    deltat_skipped += msg.time

        print(msg)

f.write("{}\\"END\", \\"END\"}\n")
f.close()

```