Punch Card Music Box

ASEN 5519 :: Final Project

Maurice Woods

14 December 2015

1 Introduction

The objective for this project is to develop a punch card reader system that manages the feeding, reading, and reporting of data "stored" on a punch card. The system must first receive and feed a set of one or several punch cards into a port. Upon insertion, the card is passed over a light source which, when shining through the holes of the card, is either transmitted or blocked by the card. Light that passes through the cards holes is detected by an array of photo-interrupts which measure and record the order and state of the holes as a series of rows. This process is repeated and appended to the preceding data set for lines over multiple cards. Once every row of holes for a set of cards has been recorded into RAM, the microcontroller will translate the states of each combination of bits (holes) as a series of tones to be recited using a set of piezo speakers.

Bit Pattern	Note	Frequency
00000	Silence	0
00001	A3	220
00010	B♭3	233
00011	B3	247
00100	C4	262
00101	C♯4	277
00110	D4	294
00111	E♭4	311
01000	E4	330
01001	F4	349
01010	F♯4	370
01011	G4	392
01100	G♯4	415
01101	A4	440
01110	B♭4	466
01111	B4	494
10000	C5	523
10001	C♯5	554
10010	D5	587
10011	E♭5	622
10100	E5	659
10101	F5	698
10110	F♯5	740
10111	G5	784
11000	G♯5	831
11001	A5	880
11010	B♭5	932
11011	B5	988
11100	C6	1047
11101	C♯6	1109
11110	D6	1175
11111	E♭6	1245

Figure 1: Note frequency lookup table for Note1

In the table to the left, a mapping from bit patterns, to tones, to frequency is outlined. Since each punch card has 80 rows, each with 10 punch spaces, two notes can be generated simultaneously given a pair of two 5-bit groups. Additionally, every other row can represent an extended note, with a 10-bit (realistically 7-bit) range to provide the full range of a standard piano. Each 5-bit/7-bit pattern will be mapped using a look-up table (Appendix A.) and will represent a note to be played for a specified amount of time. The amount of time that each note is played is the same for each note pair. However, that time can be varied to establish a tempo for the played song, using a potentiometer.

To begin the recording process, a pair of push-button switches will be used to progress through various modes, including "Read", "Tempo", and "Playback" modes. Upon startup, in "Read" mode, a card is placed into the mouth of the receiver and, once the push-button is pressed, the servo will feed the card through the receiver and through the photo-interrupt array. After reading and storing the card, subsequent cards can be read by loading a card in the receiver and pressing the push-button again. Proper feeding is checked by monitoring two additional photo interrupts on the side of each card, which expect to see a repeating gray(-ish) code. The stored song will play when the push-button switch is cycles to "Play" mode. Returning to "Read" mode will erase the previous data set to prepare for the next data set.

1.1 Project Goals

The major points of this project, in chronological order of completion and with appropriate off-ramps are as follows:

1. Build an enclosure for an array of 12 photointerruptor emitter/detector pairs to be held at opposition on either side of a card, to be fed by a roller
    - If roller-feeding does not work, allow for hand-fed feeding of card (triggering reads only by grey code)
2. Establish an interface structure of four function modes to be cycled by the user, including "New/Erase Song", "Read Card", "Tempo Select", and "Playback"
3. Capture the bit pattern cut into a punch card and store it as two 5-bit values on two separate, custom software stacks, followed by one 10-bit (7 bits used) value in another custom software stack.
    - If timing poses an issue with fidelity of note, reduce to two 5-bit values only
4. Allow the user to select a tempo at which the stored song will be played
5. Play the song stored in the three software stacks by simultaneously playing each trio of notes for the time designated by the tempo selection in Step 4

2 Functional Overview

2.1 Fundamental Block Diagram

The key components of the music box will include the card feeder, the card reader, the mode and tempo selection interfaces, and the speaker array.

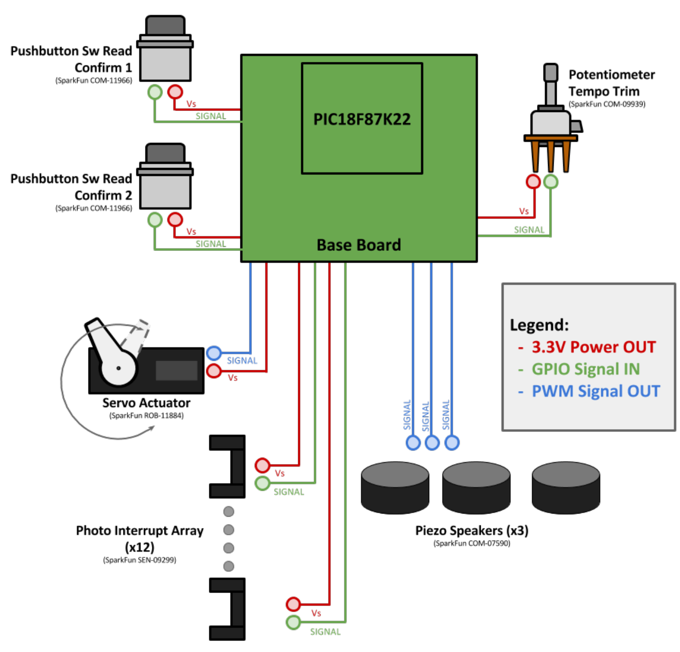

Figure 2: Fundamental block diagram

2.2 Bill Of Materials And Assembly

The following components were selected based on their price, availability, and functionality with respect to the limitations on form factor for the enclosure (e.g. the spacing of the photo-interrupters in read array). Click on a Part Numbers to follow hyperlink:

Part Name	Purpose/Description	Qty	Part Number	Vendor
Servo (Continuous Rotation)	Card receiving and feeding	1	ROB-10189	Sparkfun
10kΩ Linear Potentiometer	Emitter dimmer	1	COM-09939	Sparkfun
Pushbutton Switch (MOM)	Mode select	2	COM-11966	Sparkfun
Photo Interrupter (Emitter/Detector)	Detect holes in cards	12	SEN-00241	Sparkfun
Piezo Speaker	Speaker to produce tones	2	COM-07950	Sparkfun
Punch Cards	Song storage	30	-	ebay
Enclosure (3D Printed)	Provide dark space for scanning	1	-	Maurice
PIC18F87K22	Micro Controller Unit	1	-	CU Boulder

Figure 3: Bill of Materials

The following figure shows the assembled Music Box components from the Bill of Materials.

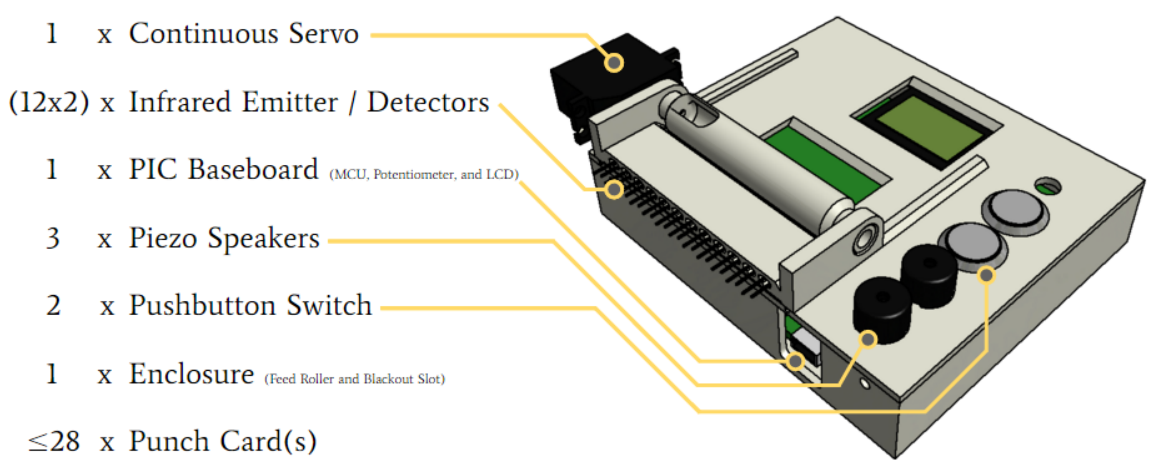

Figure 4: Assembly of enclosure, sensors, and peripherals

The enclosure was printed on a Lulzbot Mini 3D printer from white PLA plastic. Several iterations were created to ensure that the spacing between the photo-interrupters was enough to avoid cross-talk between sensors (e.g. escaping light from one channel to the other, inadvertently "closing" a signal pin that should have stayed open). Additional considerations were made that the servo-driven roller would be able to grip each card hard enough to consistently feed each card through the sensor array, but also lightly enough to where misalignment of the card's "feed trajectory" could be corrected by the guide rails located on either side of the feeder.

The punch cards themselves were cut through a two step process. First, a song would be transcribed from sheet music in terms of each note's name (e.g. Middle C is called "C4"; the next octave up is "C5", and the octave below is "C3", etc.). These values were then assigned a number value that corresponded with the value for that note's frequency (or, more accurately, it's period in units of instruction cycles) located in a lookup table (named "Note[81]" in the C source file). These numbers were then etched into each line of the punch card, in binary, using the Epilog Laser Plotter in the ITLL. During the cutting process, grey(ish) code bits were cut in the 11th and 12th position of the card for each line (the term "grey(ish)" code is used since the "zero" element of the pattern was emitted; the pattern progresses from 1, to 2, to 3, and back to 1, skipping 0. This is due to the fact that the center of a line cannot be detected if no indicating bits are present). These bits were approximately 50% the height of the bits used in the note bits so as to ensure that a line read was not triggered until all of the holes were lined up appropriately.

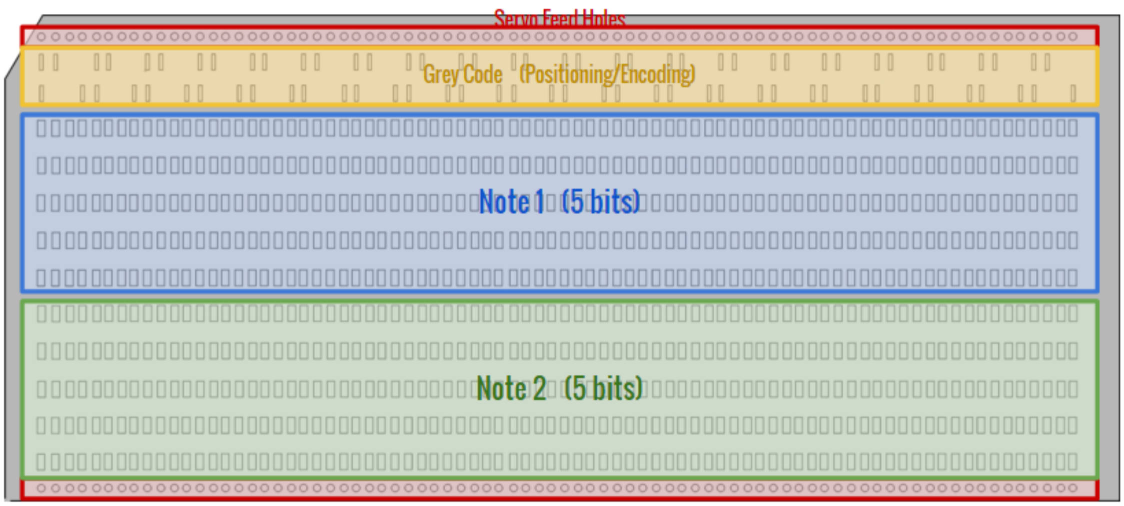

Figure 5: Punch card hole scheme (refer to Figure 1 or Appendix A)

2.3 Electrical Components And Circuitry

A total of 18 GPIO pins are needed to communicate with all of the sensors and devices used in this assembly. In particular, PORTE and PORTF were reserved for the two 5-bit photo-interrupter sets so as to simplify the process of scanning and storing information that is read from cards.

Consideration to managing idle states with pull up and pull down resistors was carefully considered. In particular, the current-limiting resistance to the photo-interrupt's emitters required precision tuning, as the detector's photogate's sensitivity is very narrow. If the emitters were too dim (limited too much), the detectors would not detect the emitted IR light and would not toggle the I/O pin. Conversely, if the emitter was too bright (not limited enough), the photogate would detect the IR light through the punch card paper, indicating no difference between high and low punch bits. This difference in resistance was on the order of less than 100Ω for an appropriate pin toggle (3.3V high, 1.2V low).

The following is the wiring schematic for each of the sensors and peripherals used:

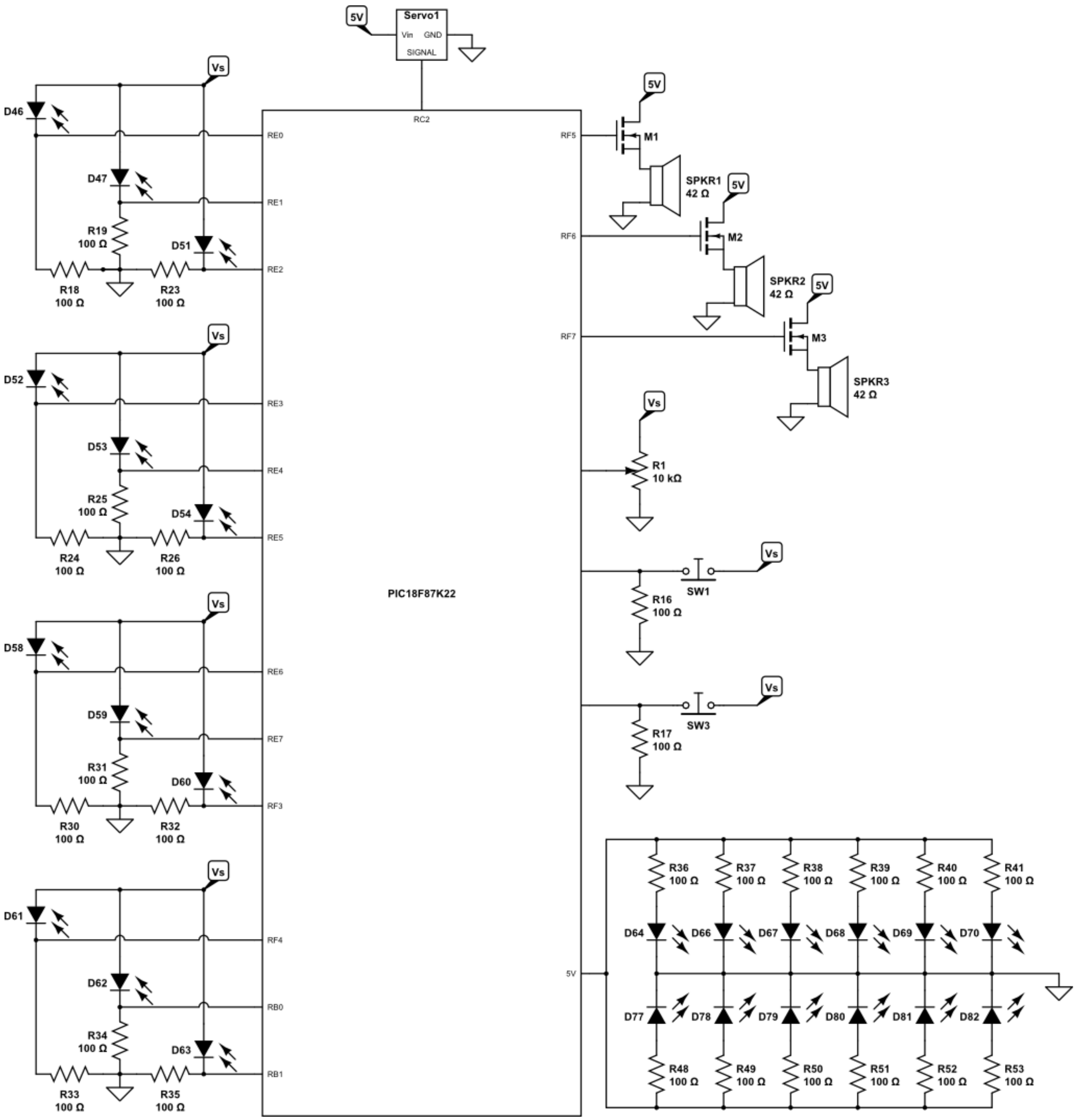

Figure 6: Electrical schematic for all sensors and devices (including integrated peripherals on the ASEN Base Board)

2.4 Software Hierarchy

The flow of the software scheme used to manage the four user modes and the timing of data collection and playback is outline in the following flow diagram:

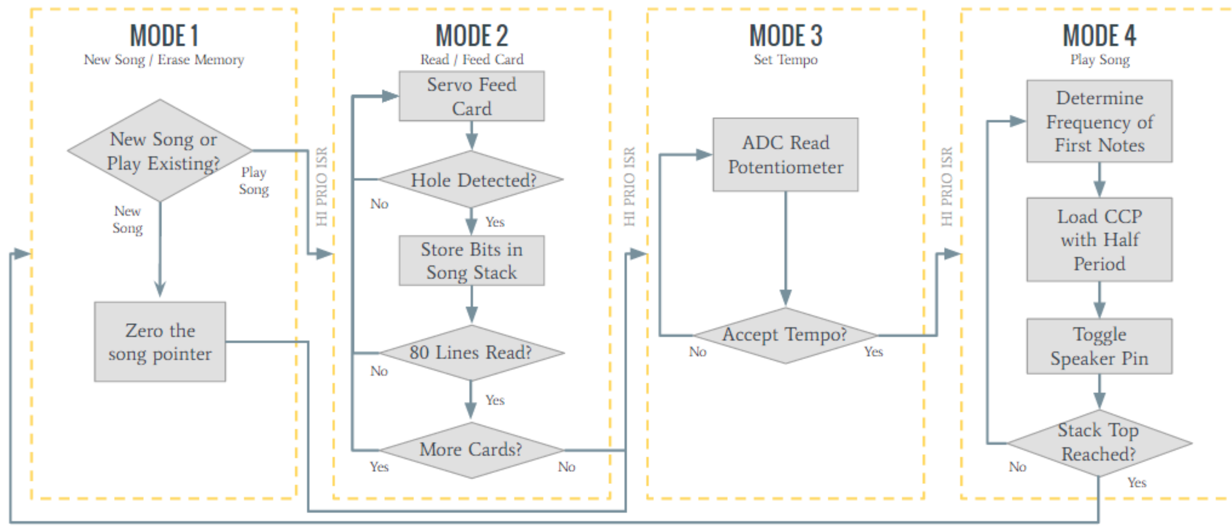

Figure 7: Software flow chart

Timing was handled by five CCP modules, used for maintaining the timing of the three note frequencies (CCP1-CCP3), the tempo separating different note sets (CCP4), and the servo, used to feed the punch cards through the photo-interrupter array (CCP5). During times when these functions were not used or needed, the respective CCP module(s) was turned off so as to avoid consuming time inside of the interrupt service routines. High priority was given to the two input triggers INT0 and INT1, which were connected to the switches that changed the operating modes.
Note storage and playback happened in the following order:

1. Two 5-bit and one 7-bit note values are read from the card and stored on three separate custom stacks, which span 928 registers in RAM each (the linker script was modified to combine banks 2 through 13 to accommodate these stacks)

2. Two stack pointers for each stack, are incremented to designate the location for the next note set to be scanned in and to designate the top of each stack

3. More notes are read and stored and each pointer is subsequently incremented until the user declares that no more cards are to be read

4. The stack pointer for each stack is reset to the bottom of each stack, leaving the top stack pointer at the last-scanned position

5. Playback is initiated and each of the stack's first values are used as a value for looking up that value's corresponding note in the Note lookup table. These values are then given to each CCP module to generate a PWM wave at the frequency specified by the value from the Note lookup table. Simultaneously, CCP4 waits for the appropriate amount of time to pass, based on the tempo/potentiometer value, and then increments all three stack pointers

6. The previous step is repeated until the address of the stack pointer is the same as the address of the top-of-stack pointer, at which point the playback stops and the user is prompted to scan a new song or replay the existing song

3 Results And Conclusions

Timing proved to be a challenging issue when constructing the playback and read features for this project. Both the ability to accurately and singularly read a single note while scanning a card was very challenging, as the edge of each grey code line indicator would often bounce (much like a switch) and cause many reads to happen for the same line. This was mitigated by incorporating a delay that allowed the card to be fed forward for a while before a new read could be triggered. Additionally, latency became a noticeable issue when using multiple CCP interrupts at once, so an effort was made to move as much timer and register setting instructions out of the ISR and into the main code loops.

Even more challenging was the mechanical assembly of the box, which required several iterations of 3D printed structures to refine the alignment of the sensor array and the smooth feeding of the card through the card slot. While, admittedly, this portion of the project was the most time-consuming aspect of the project, the successful integration of each mechanical subsystem was apparent and led to the overall success of the project.

4 Appendix

Section A :: Note Look-Up Table

Array Place	Binary Combo	Note	Freq (Hz)	Period (s)	Half-Period (instructions)
0	00000000	No Tone	0	-	-
1	00000001	D♯0/E♭0	19.45	0.05141388175	64267
2	00000010	E0	20.6	0.04854368932	60680
3	00000011	F0	21.83	0.04580852038	57261
4	00000100	F♯0/G♭0	23.12	0.04325259516	54066
5	00000101	G0	24.5	0.04081632653	51020
6	00000110	G♯0/A♭0	25.96	0.03852080123	48151
7	00000111	A0	27.5	0.03636363636	45455
8	00001000	A♯0/B♭0	29.14	0.03431708991	42896
9	00001001	B0	30.87	0.03239390994	40492
10	00001010	C1	32.7	0.03058103976	38226
11	00001011	C♯1/D♭1	34.65	0.02886002886	36075
12	00001100	D1	36.71	0.02724053391	34051
13	00001101	D♯1/E♭1	38.89	0.02571355104	32142
14	00001110	E1	41.2	0.02427184466	30340
15	00001111	F1	43.65	0.02290950745	28637
16	00010000	F♯1/G♭1	46.25	0.02162162162	27027
17	00010001	G1	49	0.02040816327	25510
18	00010010	G♯1/A♭1	51.91	0.01926411096	24080
19	00010011	A1	55	0.01818181818	22727
20	00010100	A♯1/B♭1	58.27	0.01716148962	21452
21	00010101	B1	61.74	0.01619695497	20246
22	00010110	C2	65.41	0.01528818224	19110
23	00010111	C♯2/D♭2	69.3	0.01443001443	18038
24	00011000	D2	73.42	0.01362026696	17025
25	00011001	D♯2/E♭2	77.78	0.01285677552	16071
26	00011010	E2	82.41	0.0121344497	15168
27	00011011	F2	87.31	0.01145344176	14317
28	00011100	F♯2/G♭2	92.5	0.01081081081	13514
29	00011101	G2	98	0.01020408163	12755
30	00011110	G♯2/A♭2	103.83	0.009631127805	12039
31	00011111	A2	110	0.009090909091	11364
32	00100000	A♯2/B♭2	116.54	0.008580744809	10726
33	00100001	B2	123.47	0.008099133393	10124
34	00100010	C3	130.81	0.007644675484	9556
35	00100011	C♯3/D♭3	138.59	0.007215527816	9019
36	00100100	D3	146.83	0.006810597289	8513
37	00100101	D♯3/E♭3	155.56	0.00642838776	8035
38	00100110	E3	164.81	0.006067592986	7584
39	00100111	F3	174.61	0.005727048852	7159
40	00101000	F♯3/G♭3	185	0.005405405405	6757
41	00101001	G3	196	0.005102040816	6378
42	00101010	G♯3/A♭3	207.65	0.00481579581	6020
43	00101011	A3	220	0.004545454545	5682
44	00101100	A♯3/B♭3	233.08	0.004290372404	5363
45	00101101	B3	246.94	0.004049566696	5062

Figure 8: Note Lookup Table
Blue/Purple = Note1 range, Purple/Red = Note 2 range

Array Place	Binary Combo	Note	Freq (Hz)	Period (s)	Half-Period (instructions)
46	00101110	C4	261.63	0.003822191645	4778
47	00101111	C♯4/D♭4	277.18	0.003607763908	4510
48	00110000	D4	293.66	0.003405298645	4257
49	00110001	D♯4/E♭4	311.13	0.003214090573	4018
50	00110010	E4	329.63	0.003033704457	3792
51	00110011	F4	349.23	0.00286344243	3579
52	00110100	F♯4/G♭4	369.99	0.002702775751	3378
53	00110101	G4	392	0.002551020408	3189
54	00110110	G♯4/A♭4	415.3	0.002407897905	3010
55	00110111	A4	440	0.002272727273	2841
56	00111000	A♯4/B♭4	466.16	0.002145186202	2681
57	00111001	B4	493.88	0.002024783348	2531
58	00111010	C5	523.25	0.001911132346	2389
59	00111011	C♯5/D♭5	554.37	0.001803849415	2255
60	00111100	D5	587.33	0.001702620333	2128
61	00111101	D♯5/E♭5	622.25	0.001607071113	2009
62	00111110	E5	659.25	0.001516875237	1896
63	00111111	F5	698.46	0.001431721215	1790
64	01000000	F♯5/G♭5	739.99	0.001351369613	1689
65	01000001	G5	783.99	0.001275526474	1594
66	01000010	G♯5/A♭5	830.61	0.001203934458	1505
67	01000011	A5	880	0.001136363636	1420
68	01000100	A♯5/B♭5	932.33	0.001072581597	1341
69	01000101	B5	987.77	0.001012381425	1265
70	01000110	C6	1046.5	0.000955566173	1194
71	01000111	C♯6/D♭6	1108.73	0.0009019328421	1127
72	01001000	D6	1174.66	0.0008513101663	1064
73	01001001	D♯6/E♭6	1244.51	0.0008035290998	1004
74	01001010	E6	1318.51	0.0007584318663	948
75	01001011	F6	1396.91	0.0007158657322	895
76	01001100	F♯6/G♭6	1479.98	0.0006756848066	845
77	01001101	G6	1567.98	0.0006377632368	797
78	01001110	G♯6/A♭6	1661.22	0.0006019672289	752
79	01001111	A6	1760	0.0005681818182	710
80	01010000	A♯6/B♭6	1864.66	0.0005362907983	670
81	01010001	B6	1975.53	0.0005061932747	633
82	01010010	C7	2093	0.0004777830865	597
83	01010011	C♯7/D♭7	2217.46	0.000450966421	564
84	01010100	D7	2349.32	0.0004256550832	532
85	01010101	D♯7/E♭7	2489.02	0.0004017645499	502
86	01010110	E7	2637.02	0.0003792159331	474
87	01010111	F7	2793.83	0.000357931585	447
88	01011000	F♯7/G♭7	2959.96	0.0003378424033	422
89	01011001	G7	3135.96	0.0003188816184	399
90	01011010	G♯7/A♭7	3322.44	0.0003009836145	376
91	01011011	A7	3520	0.0002840909091	355
92	01011100	A♯7/B♭7	3729.31	0.0002681461182	335

Figure 9: Note Lookup Table (cont.)
Blue/Purple = Note1 range, Purple/Red = Note 2 range

Section B :: References And Resources

Project inspiration :
"Jackson Smooth Criminal on Barrel Organ" - YouTube
Infrared Emitter/Detector Pair Hookup Guide: "Note names, MIDI numbers and frequencies"
"SparkFun Infrared Sensor Overview"
Datasheets:
SM23/33 Continuous Servo
10kΩ Potentiometer
Momentary Pushbutton Switch
Infrared Emitter
Infrared Detector
Piezo-Electric Speaker
Transcribed Sample Song (located in Stack1 and Stack2 after array element number 60):
Battle! (Trainer Battle):
Code Source Files and Linker Scripts
Github repo

Section C :: PowerPoint Presentation Slides

Card Music Box (0).png

Figure 10

Card Music Box (1).png

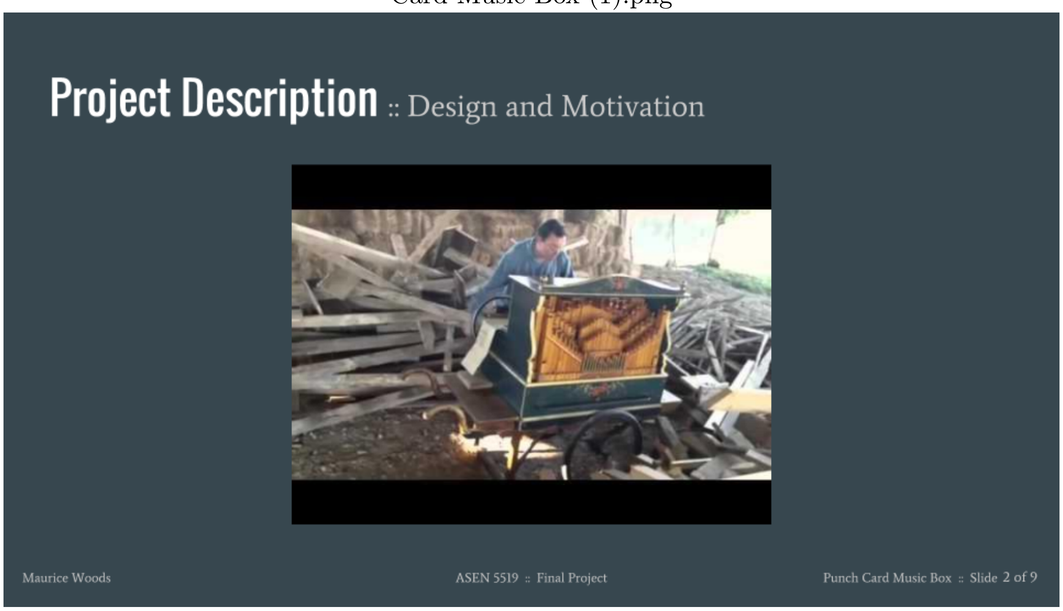

Figure 11

Card Music Box (2).png

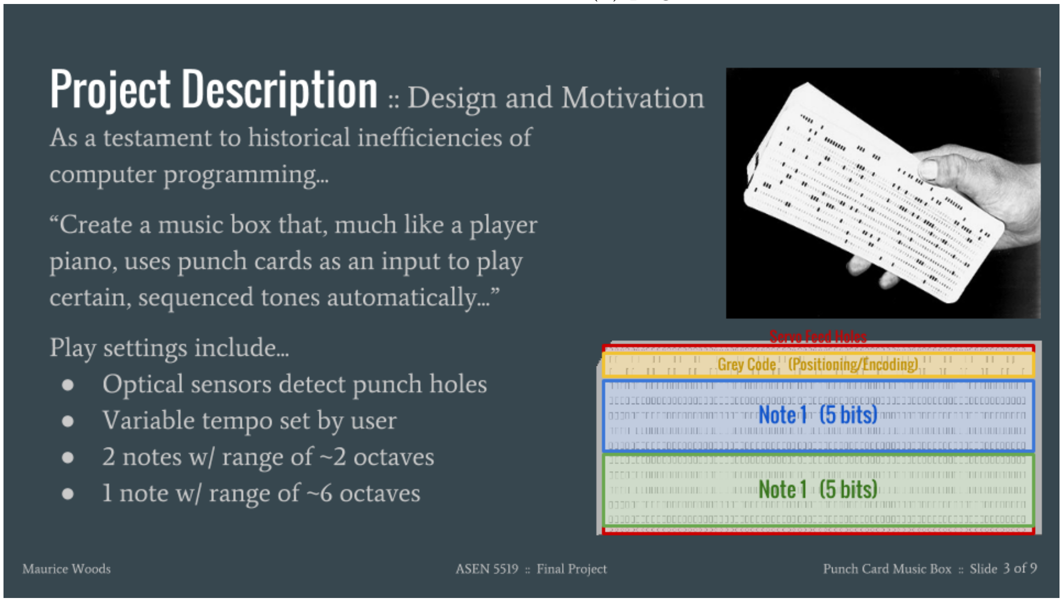

Figure 12

Card Music Box (3).png

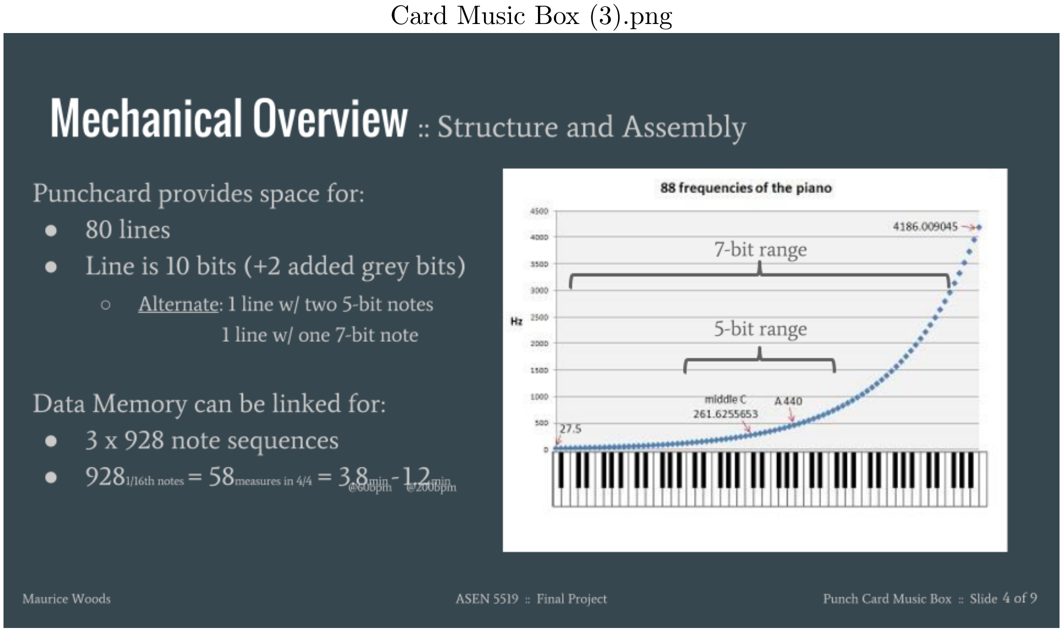

Figure 13

Card Music Box (4).png

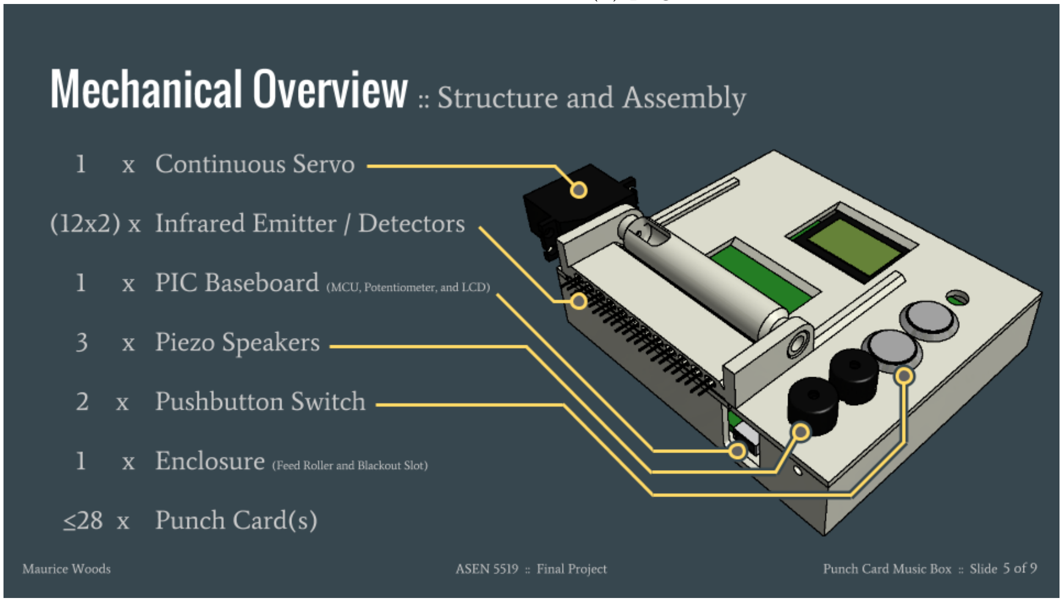

Figure 14

Card Music Box (5).png

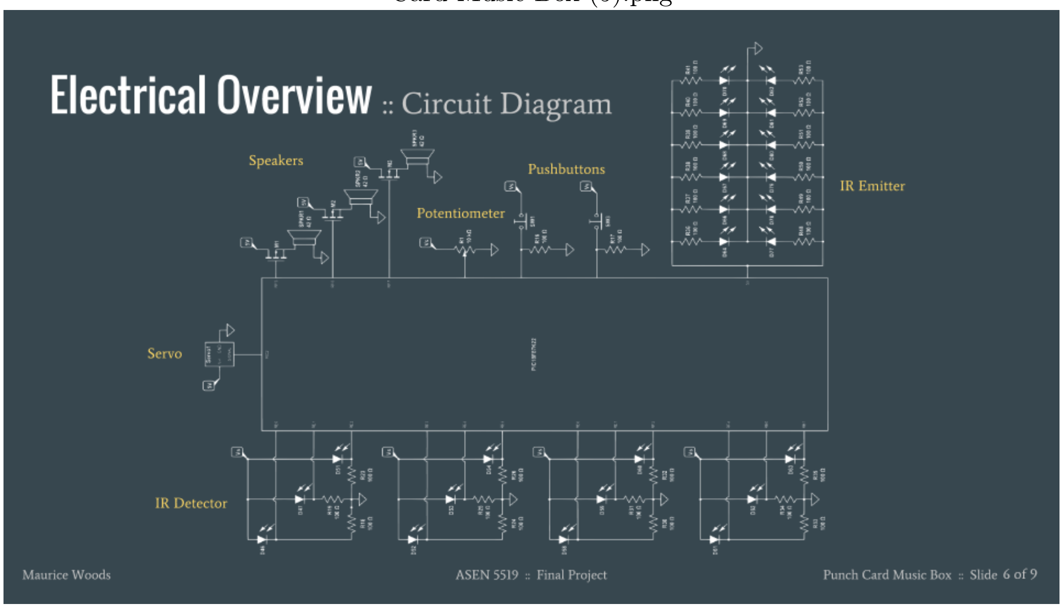

Figure 15

Card Music Box (6).png

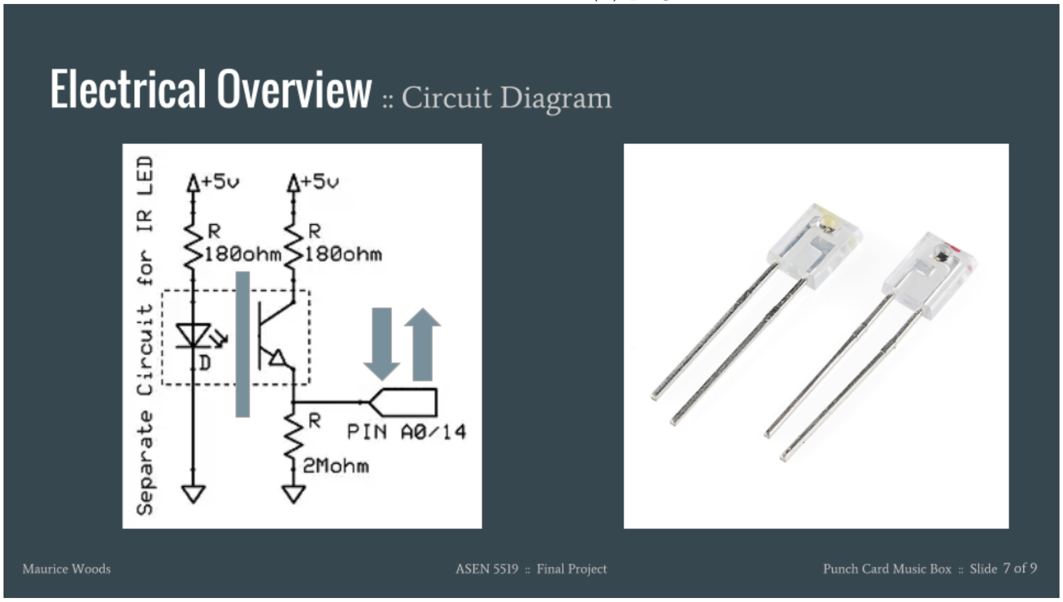

Figure 16

Card Music Box (7).png

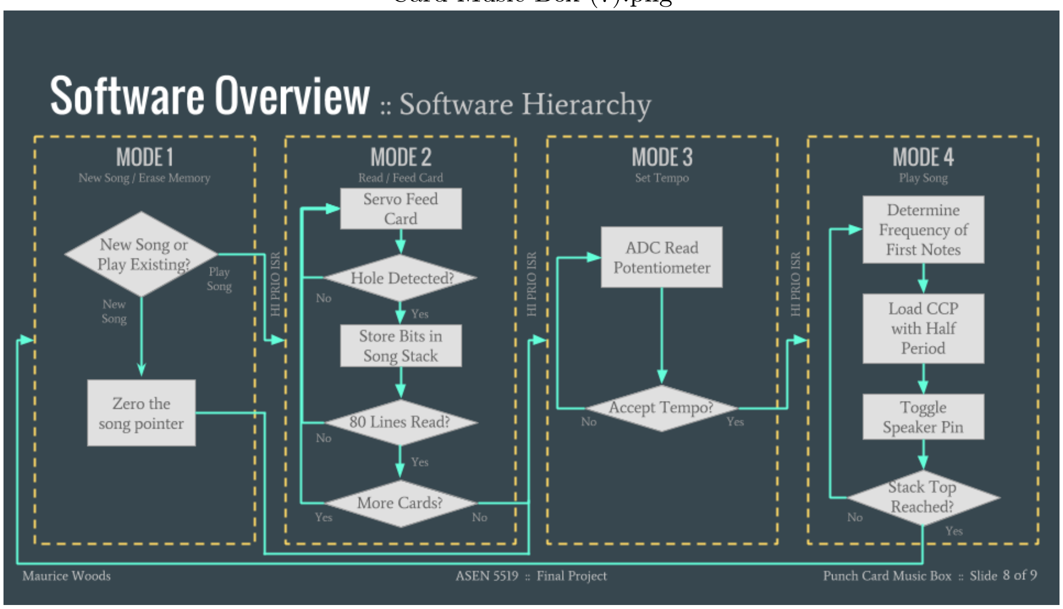

Figure 17

Card Music Box (8).png

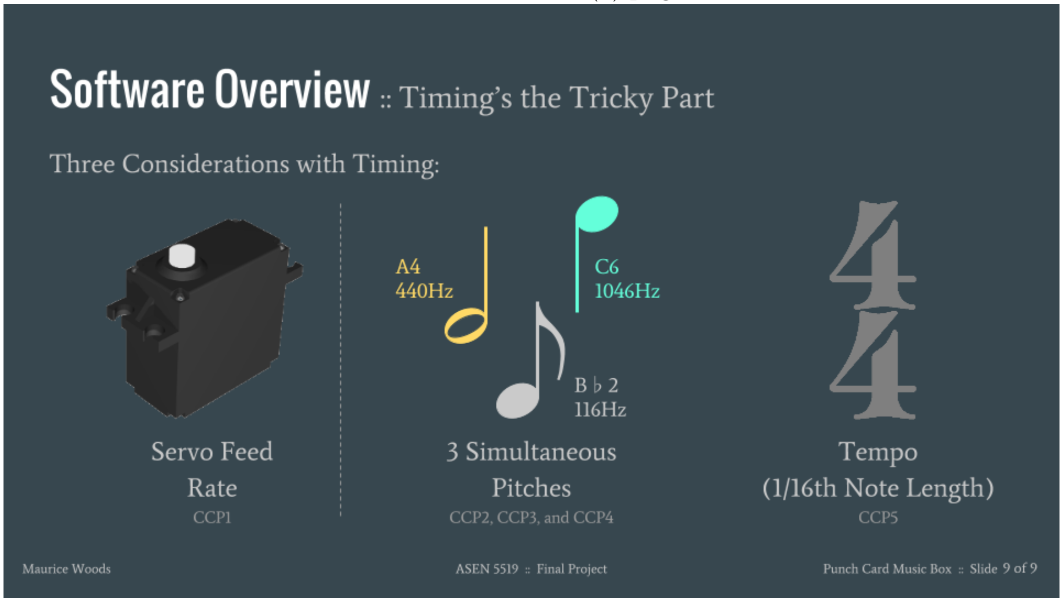

Figure 18

Card Music Box (9).png

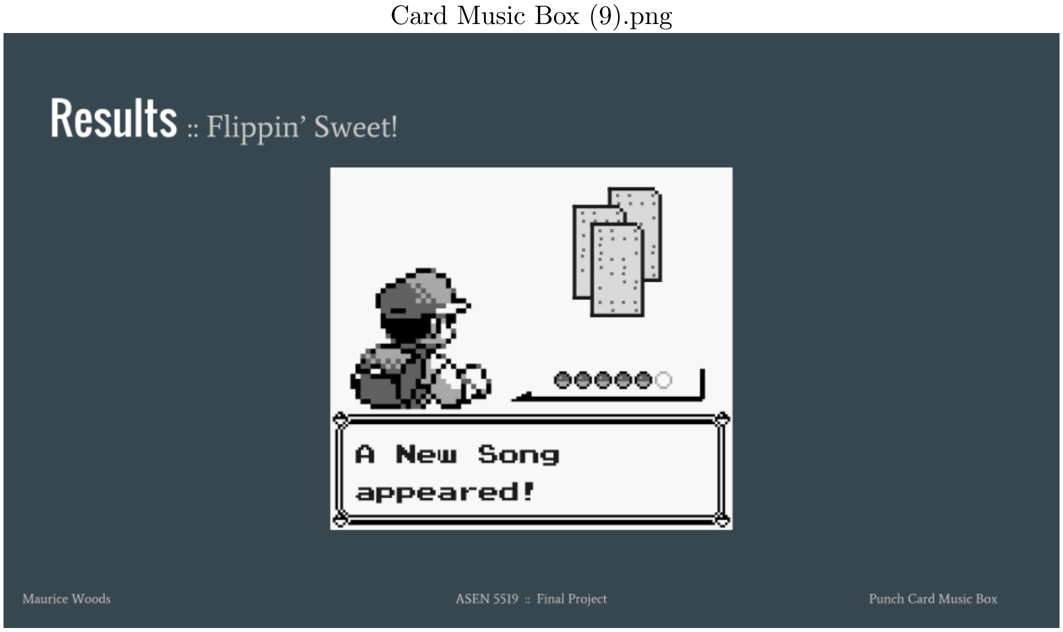

Figure 19