ASSISTIVE DIGITAL PIANO

ECE 445: SENIOR DESIGN
PROJECT PROPOSAL
Shruti Chanumolu (shrutic2), Anna Shewell (shewell2), Jae Kwak (jaekwak2)

---

1 Introduction

1.1 Objective

Our project aims to develop an assistive digital piano for beginners who wish to teach themselves how to play. Many self-taught players lack the technical skills that other players possess. For example, self-taught players might not play at the right tempo or strength or they might rely too much on their index fingers. Therefore, we want to make an assistive digital piano that can teach the user these skills. Our piano will implement many novel features: it will use LEDs and colored gloves to highlight correct fingering; after a song has been played, it will report the timing of the notes, the speed at which the keys were hit, and the number and types of mistakes so that users can identify what they need to improve; and it will provide a strict learning mode where the piano will wait for the correct finger to press the correct key before it plays the note.

1.2 Background

Many people are interested in learning how to play the piano. However, taking lessons requires money and the availability of a good teacher. The average cost of piano lessons in the U.S. is between $15 and $40 for 30 minutes of instruction. If a student takes 1 lesson per week, that adds up to approximately $720 to $2160 per year. In addition, it's been found that many beginning students lose their motivation to play in part because they don't like their piano teacher [1]. Thus, we want to design an assistive piano that would cost little more than a regular piano, but would allow the user to move at their own pace and improve their technical skills. In order to differentiate ourselves from the many other assistive pianos on the market today, our piano will teach fingering and give the user detailed feedback on metrics like rhythm accuracy.

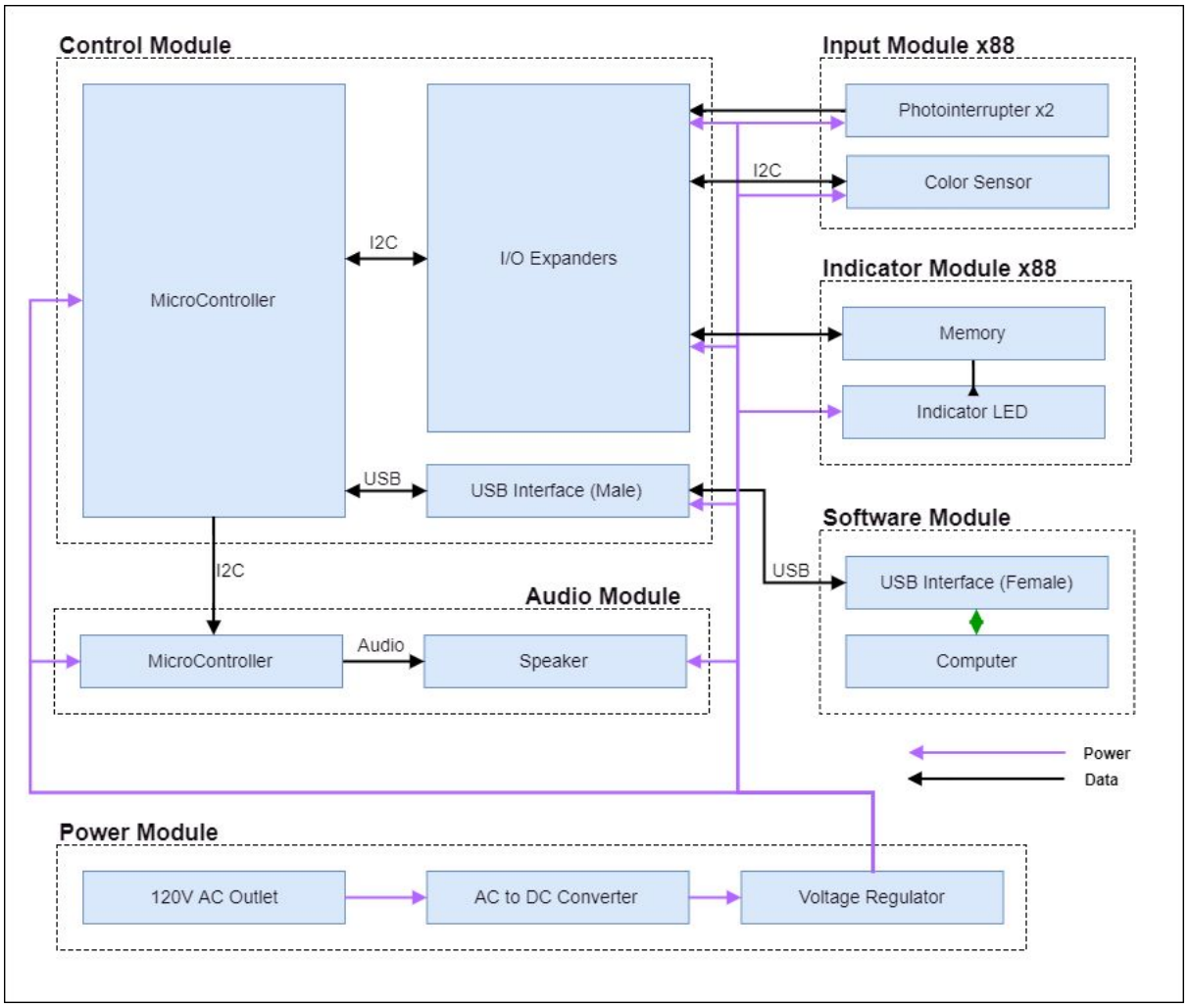
1.3 High-Level Requirements

- The piano must be able to detect key presses and play the corresponding notes from the speakers.
- The piano must be able to restrict key input to force the user to play along with a programmed song.
- The software must be able to read rhythm and keypress data from the piano and then evaluate and display timing and keypress accuracy to the user in a graphical interface.

2 Design

2.1 Block Diagram

Figure 2.1: Block Diagram



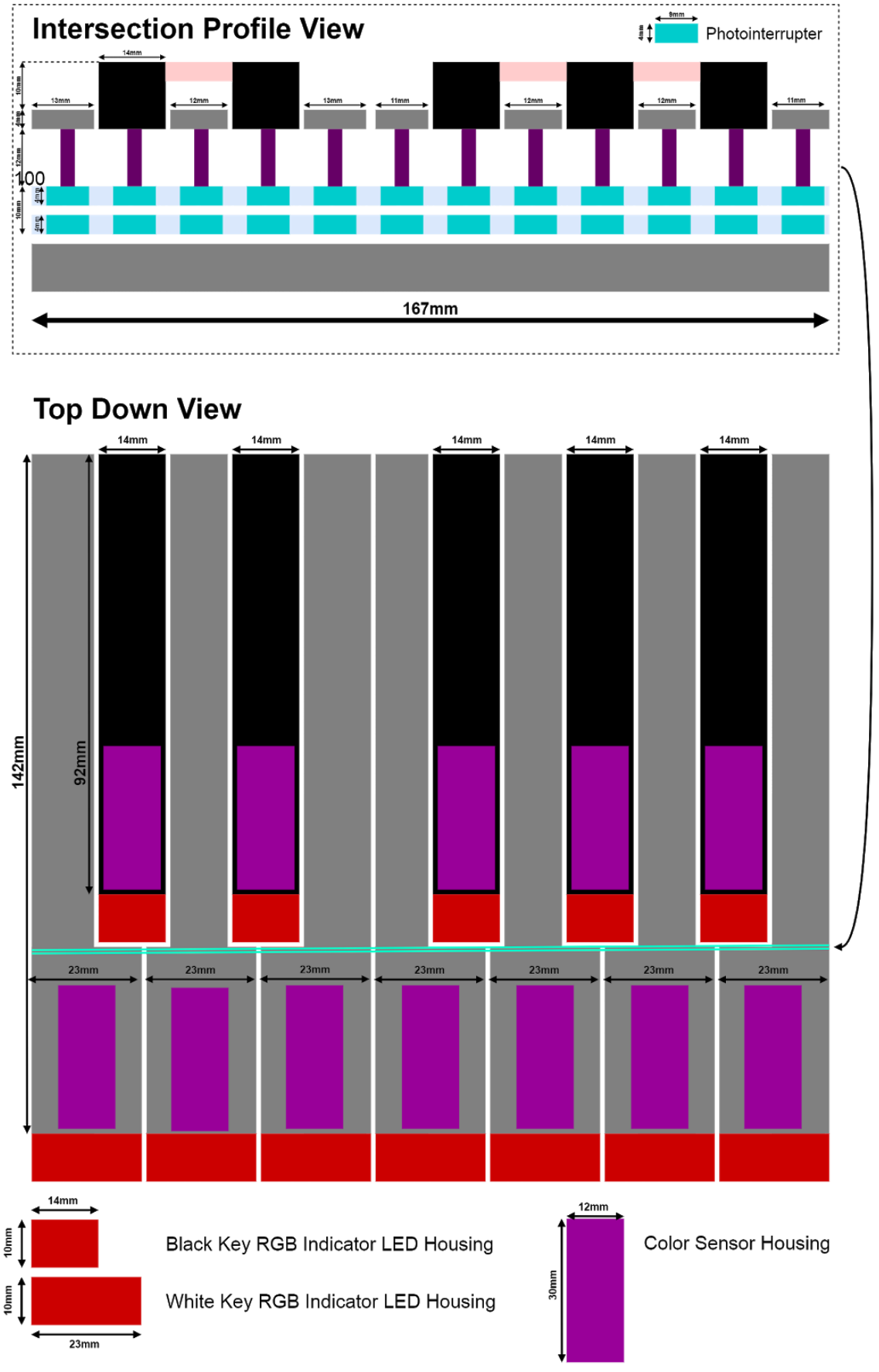
2.2 Physical Design

Keys

Two photointerrupters will be placed beneath the keys. When the key is pressed, a thin (3mm) protrusion on the bottom of the key will trigger them sequentially (see section 2.3.3 for more information on the photointerrupters).

If the piano is in “strict” mode, our microcontroller will simultaneously be reading in RGB values from the color sensors to determine which finger has been used to press the key. As the diagram below shows, the color sensor is located facing upwards around where a user’s finger will be. If the fingering or keypress of the input is wrong while in this mode, the microcontroller will not send anything to the audio module.

Figure 2.2: Key Dimensions and Sensor Layout



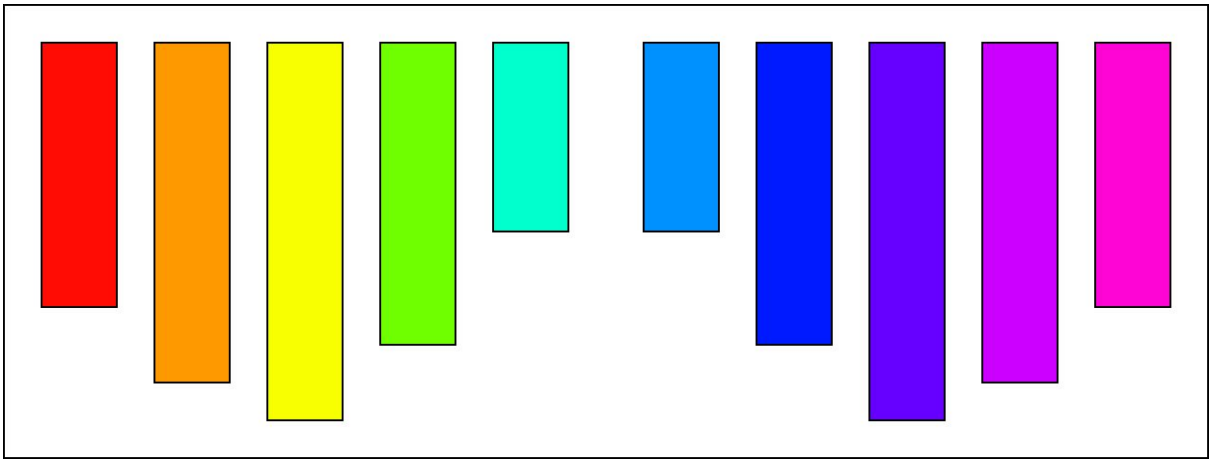
Gloves

The user will wear thin gloves with colored fingertips. The specific colors and their order are shown below in figure 2. The specific values for the colors are as follows (may change after we test the color sensors):

Table 2.1: Finger Color Values

Finger	**Color**	**Hex**	**RGB**
Left Pinky	Red	#FF0000	(255, 0, 0)
Left Ring Finger	Orange	#FF9900	(255, 128, 0)
Left Middle Finger	Yellow	#F7FF00	(255, 255, 0)
Left Index Finger	Yellow Green	#80FF00	(128, 255, 0)
Left Thumb	Green	#00FFCC	(0, 255, 0)
Right Thumb	Sky Blue	#0091FF	(0, 255, 255)
Right Index Finger	Dark Blue	#001AFF	(0, 0, 255)
Right Middle Finger	Purple	#6600FF	(128, 0, 255)
Right Ring Finger	Magenta	#CC00FF	(255, 0, 255)
Right Pinky	Pink	#FF05D5	(255, 0, 128)

Figure 2.3: Finger Colors



2.3 Functional Overview of Blocks

2.3.1 Power Module

We plan to power the system with a 120 V AC power supply. We need an AC to DC converter to convert the 120 V AC to 12V DC to power all components of our circuit. In addition, two voltage regulators are required to produce 3.3V and 5V signals.

AC to DC Converter

Table 2.2: AC To DC Converter Requirements and Verifications

REQUIREMENTS	VERIFICATION
Power from 120V AC and output a 12 ±0.2V DC with 20 ±0.1 mA	A. Connect one end of the AC-DC adapter to a 120V AC source and the other end to a DC Jack mounted on a breadboard B. Attach the Voltmeter to the DC Jack output C. Ensure readings in voltmeter is 12V ±0.2V D. Attach Multimeter to the DC jack output and read the current E. Ensure readings in current are 20 ±0.5mA

Voltage Regulator 1 (3.3V)

Table 2.3: Voltage Regulator 1 Requirements and Verifications

REQUIREMENTS	VERIFICATION
The voltage across the output terminal and ground must be 3.3 ±0.25 V for a current load up to 10 ±0.25 mA when connected to a DC supply of 12V.	A. Connect a voltmeter across the VCC pin of the the color sensor and ground.The measured voltage should be around 3.3 ±0.25 V.

Voltage Regulator 2 (5V)

Table 2.4: Voltage Regulator 2 Requirements and Verifications

REQUIREMENTS	VERIFICATION
The voltage across the output terminal and ground must be 5 ±0.25 V for a current load up to 20±0.1 mA when connected to a DC supply of 12V.	A. Connect a voltmeter across the VCC pin of the the Photointerrupters and ground.The measured voltage should be 5 ±0.25 V. B. Connect a voltmeter across the anode and cathode pins of the the LED .The measured voltage should be 5 ±0.25 V.

2.3.2 Control Module (Microcontroller)

The control module (a central microcontroller) will be responsible for handling the control logic and input/output of the piano. This will include periodically scanning the photointerrupter inputs, reading color sensor values on full keypress, outputting keypress data to the software module, and outputting keypress data to the audio module. Due to the high number of I/O requirements (88 keys in theory, each with two photointerrupters and a color sensor), we plan to use port expanders.

Figure 2.4: Microcontroller Program Flow Chart

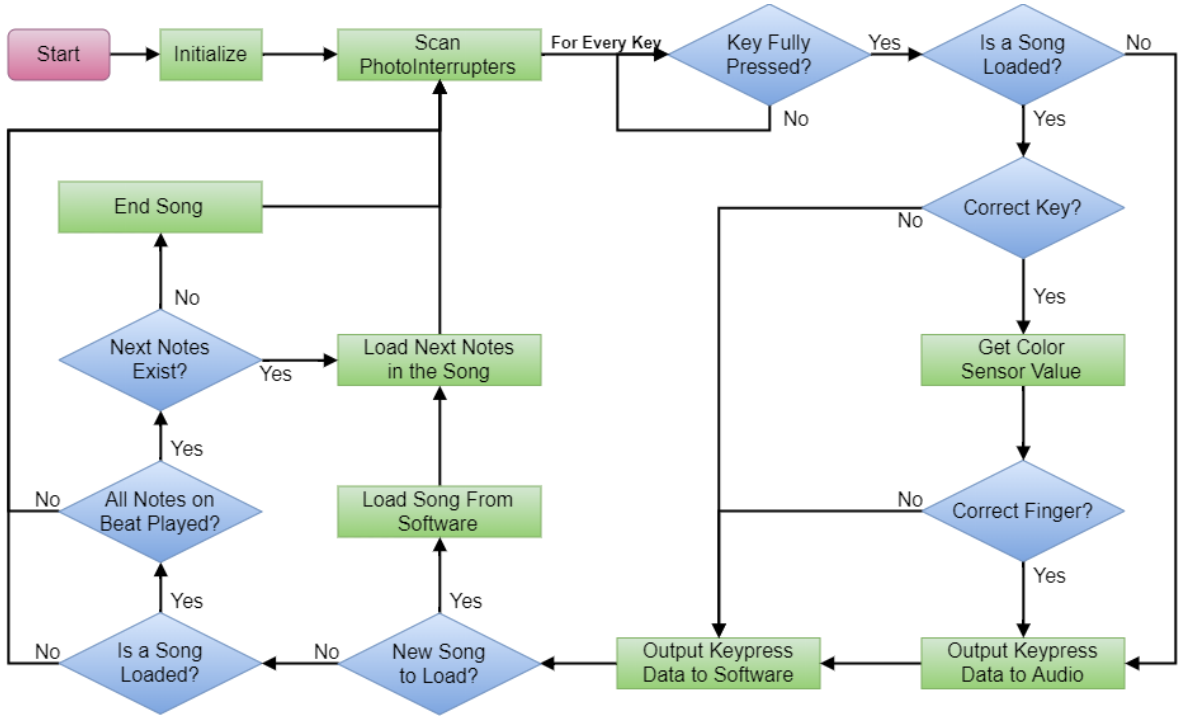


Table 2.5: Microcontroller Requirements and Verifications

REQUIREMENTS	VERIFICATION
+ I/O Expander Requirement Processes I^2C to the I/O expanders at a rate of 1.7 Mbps.	A. Connect two of the pins to a I^2C bus that is connected to a 16 bit I/O expander. B. Write 8 bits of data to port A of the I/O expander set at a rate of 1.7 Mbps. C. Read 8 bits of data from port A of the I/O expander set at a rate of 1.7 Mbps. D. Verify that the data does not change from reading/writing.
Outputs signals to correct LEDs with 100% accuracy.	A. Connect VCC pin of the microcontroller to 5 ±5% V. B. Set the output bits of the microcontroller to identify all keys of the piano, one at a time. C. Ensure that the correct LEDs turn on.
Outputs correct color codes to LEDs with 100% accuracy.	A. Connect VCC pin of the microcontroller to 5 ±5% V. B. Set the output bits of the microcontroller to identify one of the piano keys. C. Send all possible color codes to the LED. C. Ensure that the LED lights up as expected.
Must be able to correctly identify at least 10 keys pressed within 1 ms of each other.	A. Connect the microcontroller to a computer through the USB interface. B. Press any 10 keys at the same time. C. Echo key data to the computer and display that information on the screen. D. Ensure that 10 key presses have been identified and that they correspond to the respective keys pressed.

2.3.3 Input Module

The input module handles receiving input from the user. It consists of two photointerrupters that get triggered sequentially when a key is pressed for the purpose of velocity-sensitive keypress detection and a color sensor to read the color of the finger that is over the key.

Photointerrupters

There will be two photointerrupters attached sideways underneath each key that will get triggered sequentially when the key is pressed. The purpose of having two is for velocity sensing. When the first photointerrupter is triggered, the microcontroller will start timing the keypress. When the second photointerrupter is triggered, the microcontroller will stop timing the keypress. Since the two sensors are stationary and spaced a known distance apart, velocity can be computed. The computed velocity will then be used to determine the volume of the sound output.

Table 2.6: Photointerrupter Requirements and Verifications

REQUIREMENTS	VERIFICATION
Top sensor must be able to detect key press when a key is pressed 3 mm from its resting position, measured from above the sensor.	A. Supply 5 V DC power supply to the power inputs of the photointerrupter. B. Connect a digital multimeter across the output terminals of the photointerrupter. C. The multimeter must show 0±.25V if the key is pressed by more than 3mm. It should show 5V±5% if the key is pressed any shallower.
Bottom sensor must be able to detect key press when a key is pressed 9mm from its resting position, measured from above the sensor.	A. Supply 5 V DC power supply to the power inputs of the photointerrupter. B. Connect a digital multimeter across the output terminals of the photointerrupter. C. The multimeter must show 0V±.25V if the key is pressed by 9mm .It should show 5V±5% if the key is pressed any shallower.

Color Sensor

Color Sensors will be placed on the keys to detect if the correct finger is being used to press the key. This will be done by reading the colors on the fingers of a set of included gloves and sending this data to the control module using a I2C bus. The control module will then compare the RGB values of the signal sent by the color sensor to the color of the finger that was supposed to be used. Thus, it can determine if correct fingering is being used to press the key or not.

Table 2.7: Color Sensor Requirements and Verifications

REQUIREMENTS	VERIFICATION
Must be able to identify 10 distinct colors, each within ±10 nm of wavelength.	A. Take ten samples of different colors each within /+10 nm of wavelength B. Connect the color sensor to arduino board. (VCC-3.3V, GND-GND, SDL-SDL,SCL-SCL) C. ADD the color sensor library into /arduino/libraries D. Read RGB data through the library function E. Compare the intensities of these RGB data and ensure that they are different for all the 10 colors
Must be able to identify a color from 0.5 ±0.25 mm above the key.	A. Repeat B to D from above by placing the color sensor 5.26m above the color strip ensure no color is read or the color of the strip is not read.
Must be able to register color changes within 5 ±0.5 ms.	A. Make a color wheel of more than 5 colors and of radius 5cm. B. Spin the wheel such that colors are changing at 5±0.5ms form a point where the color sensor is placed. C. Repeat steps 1-B to 1-D and ensure that the output of color sensor changes.

2.3.4 Indicator Module

RGB Indicator LED

LEDs are provided on each key as a guide for users. The keys that need to be played next will light up in 1 of 10 colors corresponding to the colors on the set of included gloves (see figure 2 above). Instead of using 10 separate LEDs, we plan to use 1 RGB LED for each key. We will use a mux to feed either a full voltage signal, a 50% PWM signal, or a ground signal to each color in each RGB LED to produce 10 distinct colors [3]. Table 2.11 summarizes the select bits for those muxes. Our design for the PWM circuit (not shown here) is based off a circuit created by Electronics Tutorials [8].

Instead of sending color codes from the computer or microcontroller every time we need to light up the LEDs, we plan to store all color codes in dedicated RAM ahead of time. The microcontroller will store all the color codes at the beginning of the song and will send a signal to trigger the next output from all the memory chips, but will otherwise not be involved with the LED indicators. By doing this, we hope to reduce the strain on the microcontroller and minimize latency.

We've chosen 10 different 4-bit color codes to represent our 10 colors (we save memory space by choosing the minimum number of required bits to make 10 patterns). These color codes can be easily mapped to the 10 different 6-bit color codes that control which power signal each component of the RGB LEDs is connected to (see table 2.8). Although we consider the 6-bit color code as a whole, it's important to note that this is actually a series of three 2-bit codes, one for each component of the RGB LEDs.

Table 2.8: 4-Bit and 6-Bit Color Codes

Color	**4-Bit Color Code (Stored In Memory)**	**6-Bit Color Code (Sent To LED Muxes)**		
		R_0R_1	G_0G_1	B_0B_1
Dark Blue	0001	00	00	11
Green	0010	00	11	00
Sky Blue	0011	00	11	11
Red	0100	11	00	00
Magenta	0101	11	00	11
Yellow	0110	11	11	00
Purple	1001	10	00	11
Yellow Green	1010	10	11	00
Pink	1101	11	00	10
Orange	1110	11	10	00

Figure 2.5: Indicator Module Schematic

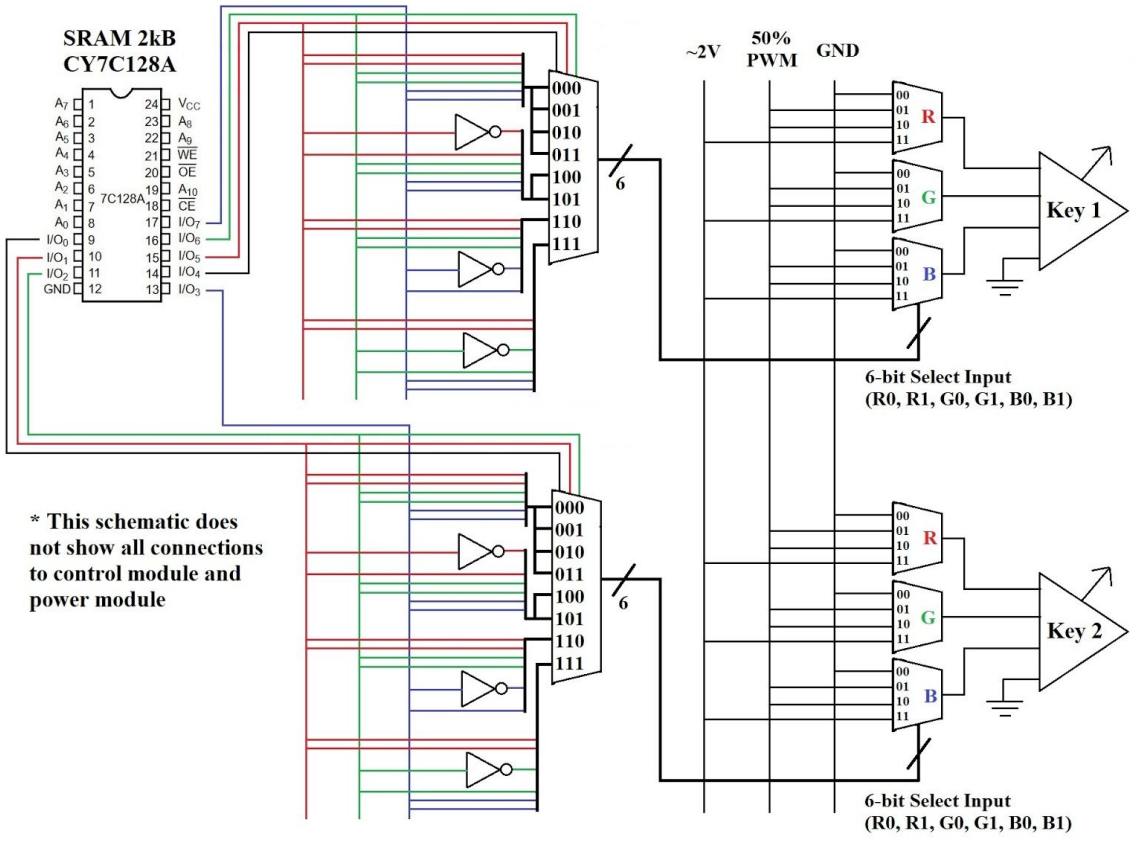


Table 2.9: Select Bits For Conversion To 6-Bit Color Code

From 4-Bit Color Code			6-Bit Signal Passed Through			Color(s)
Bit 0	Bit 1	Bit 2				
0	0	0	RR	GG	BB	Dark Blue
0	0	1	RR	GG	BB	Green, Sky Blue
0	1	0	RR	GG	BB	Red, Magenta
0	1	1	RR	GG	BB	Yellow
1	0	0	(~R)R	GG	BB	
1	0	1	(~R)R	GG	BB	Pink
1	1	0	RR	GG	(~B)B	Orange
1	1	1	RR	(~G)G	BB	Purple, Yellow Green

Table 2.10: LED Power Mux Select Bits

Sel0	**Sel1**	**Power Signal**
0	0	GND
0	1	50% PWM of ~2V
1	0	50% PWM of ~2V
1	1	~2V

Table 2.11: Indicator Module Requirements and Verifications

REQUIREMENTS	VERIFICATION
Must be able to produce 10 distinct colors matching the colors on the provided gloves such that the RGB values are within 30 degrees of each other (per RGB component).	A. Connect the color sensor to a temporary circuit with separate LEDs. This circuit will indicate the binary RGB values. B. Use the color sensor to measure the RGB values of the gloves' colors. Record these values C. Send the appropriate power signals to light up one of the RGB LEDs for each of the 10 colors. D. Use the color sensor to measure the RGB values of the LEDs. E. Compare each component individually.
Must be able to respond to microcontroller's signal and change which keys are lit within 10 ±0.5 ms.	A. Mimic the microcontroller's signal to the circuit. B. Measure the time between the signal being sent and the LED response.

2.3.5 Audio Module

The purpose of this module is to convert key presses into the corresponding waveforms that will produce the correct sounds for the speaker.

Audio Controller

The audio controller receives a key press code (sequence of bits) from the microcontroller and converts it into the corresponding waveform for that note. This waveform is sent to the speaker which should output the correct pitch for the key pressed.

Table 2.12: Audio Controller Requirements and Verifications

REQUIREMENTS	VERIFICATION
Must be able to convert the key press code into the corresponding waveform.	A. Connect the output of the audio controller to an oscilloscope and press all keys on the piano, one at a time. B. Ensure that the frequency of the waveform matches the key pressed.

Speakers

The speakers will output the sound signals given by the audio controller. Common speaker circuit elements will be used (amplifier, low-pass filter, etc.).

Table 2.13: Speaker Requirements and Verifications

REQUIREMENTS	VERIFICATION
Must be able to output the sound signals from the audio processing unit at volumes between 65 and 95 dB. The sound should be clearly audible from a distance of 1m.	A. Connect the speaker to a computer and play a song. B. Check manually by hearing the song if it is in the audible range of human hearing from at least 1m away from the Piano.

2.3.6 Software Module

Along with our physical keyboard, we will design software to send song data (which keys need to be pressed by what fingers in what order) and collect performance information (speed of key press, tempo, etc.) and display that information via a user-friendly interface. We've split the role of the software into two stages: in stage 1, the computer converts musical notes and finger data for the chosen song into binary codes and sends that data to the microcontroller (see figure 2.6); in stage 2 (occurs while the song is being played), the computer receives data from the microcontroller and processes that data for the user (see figure 2.7). We've also created an initial design for how we'll display the performance results (see figure 2.8).

Figure 2.6: Flow Chart For Stage 1 (sending song data to microcontroller)

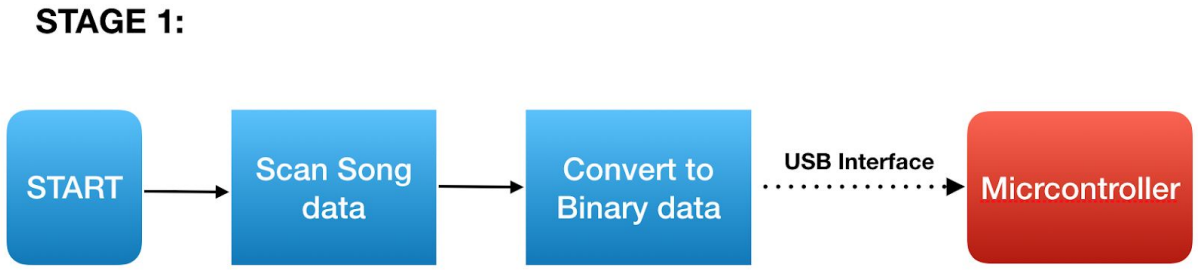


Figure 2.7: Flow Chart For Stage 2 (receiving finger and key press data from microcontroller)

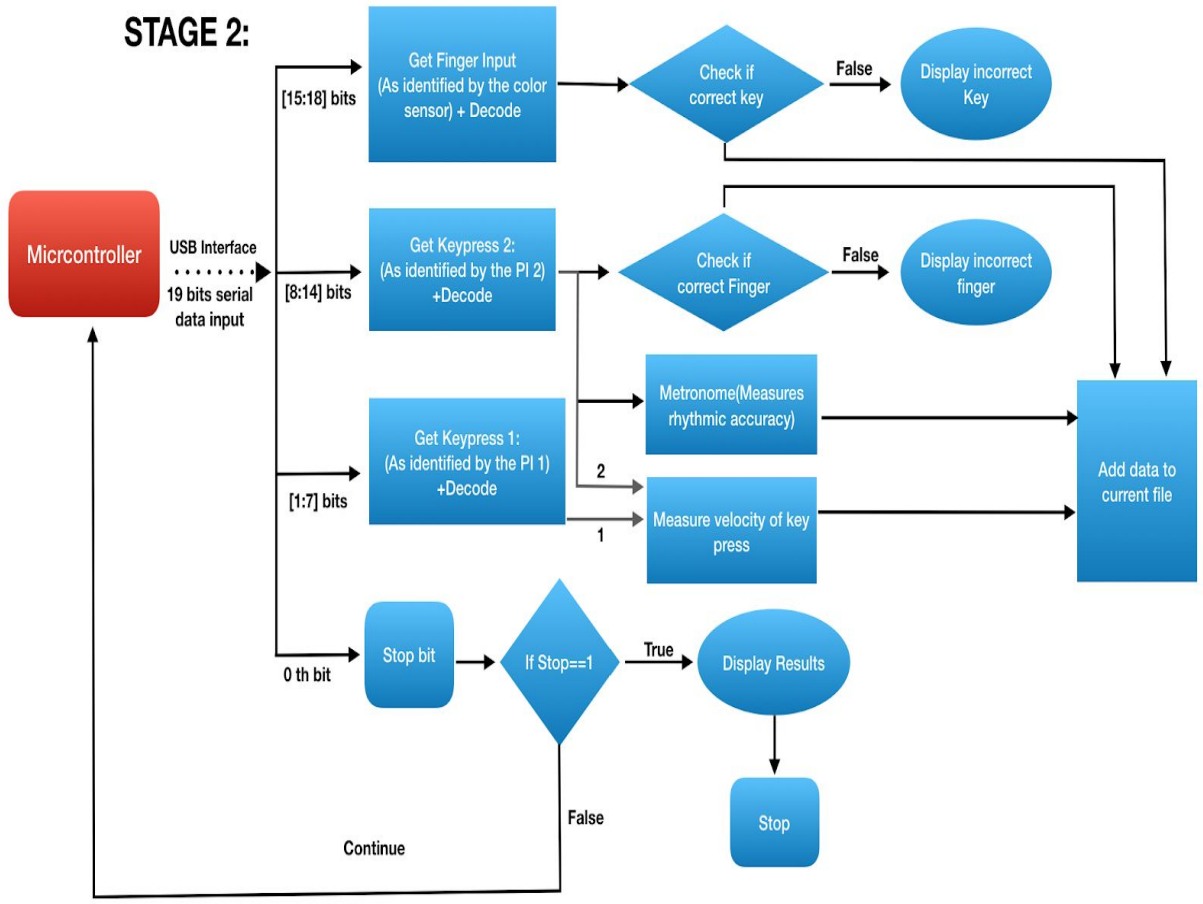
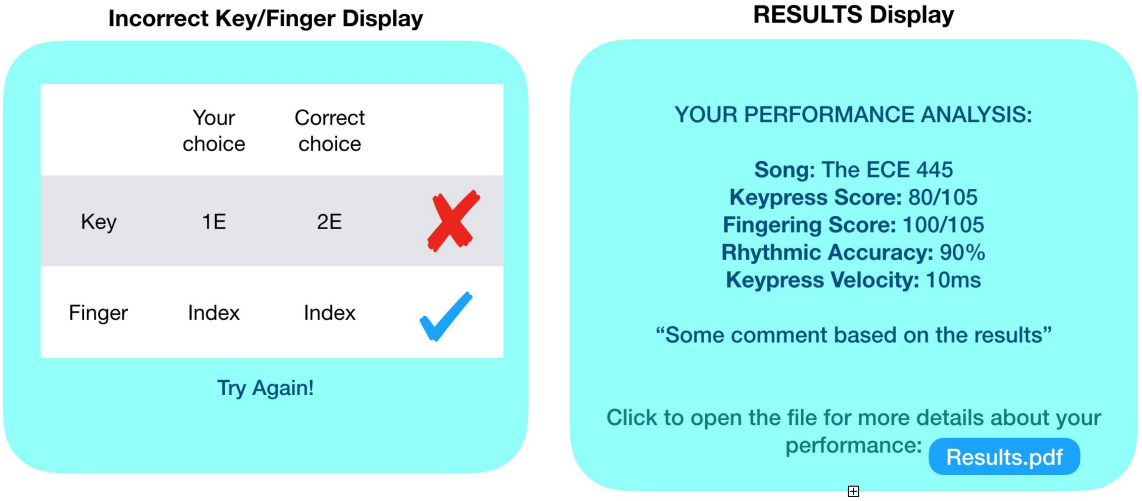


Figure 2.8: Initial Prototype For Results Display



Keypress Score: Total number of keys pressed correctly/Total number of keys pressed

Fingering Score: Total number of keys pressed with correct fingering/Total number of keys pressed

Rhythmic Accuracy: To measure rhythmic accuracy, we will use a metronome and detect if the keypress corresponds to this metronome tick (i.e. if the keypress is within ~10 ms of the metronome tick, it is considered rhythmically accurate). We will then display the percentage of rhythmically accurate key presses.

Keypress velocity: Average velocity of each key press.

Table 2.14: Software Requirements and Verifications

REQUIREMENTS	VERIFICATION
Must be able to send a set of “next keys and fingers” to the microcontroller	A. Send the next keypress data to the microcontroller. B. Verify that the microcontroller is processing the next note correctly with a rate of 99% accuracy.
Must be able to download key press data from the control module at a rate of 100 kbps.	A. Send a sequence of keypress inputs from the microcontroller to the software. B. Verify that the software detected at least 99% of the keypress inputs.

2.4 Tolerance Analysis

The most critical feature of our project is handling all of the inputs and outputs correctly in the microcontroller. Although the color sensors are risk averse, it is not exactly critical to the design and purpose of the piano - to be an assistive learning tool. It is interesting to be able to communicate to the user which finger to press which key with, but will probably not be the most often used feature. In fact, simple “learning” mode, where users are required to press the right keys, but not required to press the keys with specific fingers, is probably a more important overall feature. The user should use our piano with the expectation that if they play correctly, the piano will acknowledge that and work as expected if it is being used for learning purposes, but especially if it is only being used as a simple piano. Our product is fundamentally a piano, and should therefore reliably perform the functions of a piano, even if the assistive learning functions are buggy.

Our project includes a large number of inputs and outputs - in total, the full piano would include 174 photointerrupter inputs, 88 color sensor inputs, and 88 indicator LED outputs. The I/O for the color sensors and LEDs, specifically, are non-trivial. Our microcontroller must be able to handle these inputs and outputs very fast. Time is the key factor, since the user is expecting a sound output from the piano with minimal latency.

A grand piano has latency between depressing the key completely and the hammer hitting the string. In academic studies, this seems to be 12ms after full key depression for a fast key press and 3ms *before* full key depression for a slow key press. Though we may not go as far as to mimic this in our design, we will use this information as a benchmark for the maximal time latency from a full key press to outputting sound to the speakers. This means that our microcontroller must be able to scan the keys, compare the key/finger to the required keys/fingers in the song it is currently programmed to play, and output the information to the audio module within 12ms.

The primary component we will be analyzing is the microcontroller for our control module. We are using the ATMega328P. It has a maximum estimated speed of 20MIPs. Since we are using I/O expanders for our pins, we are limited, for every two pins of our microcontroller, to the maximal speed of the I2C connection to the I/O expander, which is 400 kbps in fast mode, or 1.7 Mbps/10 Mbps in high speed mode. Since I2C communication speed is affected by more than simply it’s maximal rating, it is critical that we are able to write our program to handle inputs and outputs faster than this.

Microcontroller Processes to Study:
Scanning photointerrupters
Outputting color data to indicator module memory
Reading in color values

First, our program needs to constantly scan the photointerrupters. For a nuanced dynamics time resolution, the entire set of 88 keys, meaning 174 photointerrupter outputs, should be scanned at least twice a millisecond. Since our microcontroller communicates with the photointerrupters through I2C to 16 pin I/O expanders, we can calculate exactly how long this would take:

Total bit transfer to read 16 Photointerrupter inputs:
8 bit I/O expander address write
+ 8 bit internal register 1 address write
+ 8 bit digital read
+ 8 bit internal register 2 address write
+ 8 bit digital read
= 40 bits (approximate)

I/O Expanders needed for 174 Photointerrupter inputs:
ceil(174 / 16) = 11 I/O Expanders

I2C buses needed for 11 I/O Expanders:
1 I/O Expander with 6 * 16 Photointerrupter inputs
1 I/O Expander with 5 * 16 Photointerrupter inputs

Data from 5 or 6 I/O Expanders to Photointerrupters (6 * 16 Photointerrupter inputs) with I2C:
5 * 40 = 200 bits
6 * 40 = 240 bits

Communication rate scanning all keys at a rate of twice a millisecond
200 [bits] * 2000 [Hz] = 400 kbps
240 [bits] * 2000 [Hz] = 480 kbps

Clearly, it will be necessary for us to use high speed mode, with a transfer rate of 1.7 Mbps, to communicate with the photointerrupters through I2C with the I/O expanders.

Total Communication time
440 [bits] / 1.7 [Mbps] = .259 ms

At a minimum polling time of .259 ms per photointerrupter input scan, we are left with a little more than .2ms to execute all other logic functions.

Next, we consider that we need to send RGB color values to the indicator LED PWM circuits. To accomplish this, we will keep the RGB values in memory in close proximity to the PWM circuit.

Total bit transfer for writing color values to memory:
8 bit device address
+ 8 bit register address
+ 4 bit color value
= 20 bits (approximate)

Minimum communication time
400 kbps: 20 / 400k = .05 ms
1.7 Mbps: 20 / 1.7M = .0117 ms

From these calculations, we can conclude that outputting color values to the indicator module memory does not take a significant amount of time, and will not affect the key scanning in a meaningful way.

Finally, we consider that we have 88 RGB color sensors that are connected to our microcontroller through I2C. Since a requirement of our microcontroller is to be able to simultaneously process 10 key press inputs, we will need to ensure that reading the RGB color values from 10 color sensors does not interfere with our keyscanning for our velocity. To do that, we will calculate how much time it takes to read the RGB color values sequentially from 10 color sensors.

Since our color sensors output data in 16 bit resolution per color, each read from a single color sensor would take:

Total bit transfer for reading a single color sensor:
(8 bit device address + 8 bit register address) * 3 colors
+ 16 bits of data * 3 colors
= 96 bits

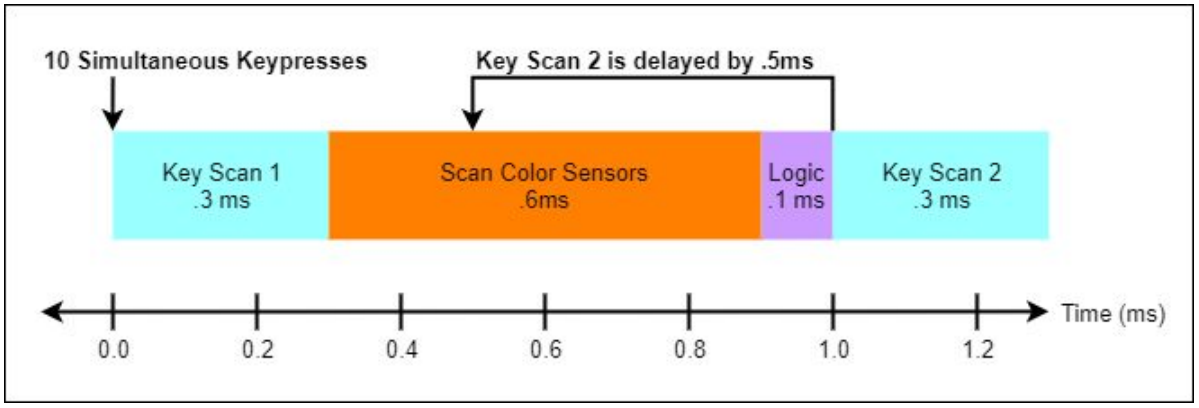
Minimum communication time
400 kbps: 96 / 400k = .24 ms
1.7 Mbps: 96 / 1.7M = .056 ms

Evidently, it takes a minimum of .056 milliseconds to read a single color sensor, meaning it will take a minimum of .56 milliseconds to read 10 of the color sensors at once. This means that when 10 keys are pressed simultaneously, we will need to sacrifice an entire half-millisecond keyscan in order to read in the color values. Moreover, since this is a minimum estimate, the lost time-resolution is likely to be even greater.

Pulling these calculations together, we can understand that our system is limited most by 1) the soft requirement to scan the 174 photointerrupter inputs every half millisecond, and 2) the requirement to be able to handle 10 simultaneous keypress inputs.

If we assume that the key scan duration is on average .3 ms (rounded up from .259 ms minimum) and the color sensor input scan for 10 simultaneously pressed keys is .6 ms (rounded up from .56 ms minimum), and all other communication and control logic takes .1 ms, we can build a worst-case timeline of the microcontroller program for this scenario:

Figure 2.9: Worst-case keyscan delay



Evidently, in the worst case, the next keyscan is delayed by one entire cycle - half a millisecond. For simplicity of analysis, we will assume that all photointerrupters were scanned instantaneously at the beginning of the keyscan. The implication of this timeline, then, is that any keypress that falls in the window of 0 ms (after key scan 1) to .5 ms (the normal execution time of key scan 2) will have an error in its dynamic output proportional to the amount of time that key scan 2 is delayed, which, in this timeline, is .5 ms. This is because the key input is read later than it normally is, resulting in a longer time differential.

Timing Error:

$$E(t) = T_{delay}$$

where $T_{delay} = T_{Key\ Scan\ 2} - (T_{Key\ Scan\ 1} + .5\ ms)$

In general, we believe that the loss of one key scan of time resolution is acceptable. Moreover, due to the flaws in human dexterity and the rarity of 10 keypress chords, it is very unlikely that 10 simultaneous key presses will ever happen in the span of a single key scan. Regardless, we came up with two methods of improvement for our algorithm from this analysis:

1. When scanning photointerrupter inputs, only scan the bottom photointerrupter if the first photointerrupter is triggered. This will improve scanning time by almost 50%.

2. Mitigate the timing error by adding $T_{delay} / 2$ to all key timings that fall in the error window. This will, assuming uniform distribution of inputs within the error window, reduce the timing error by a value of 50%.

3 Cost and Schedule

3.1 Schedule

Figure 3.1: Overall Schedule and Dependency Chart

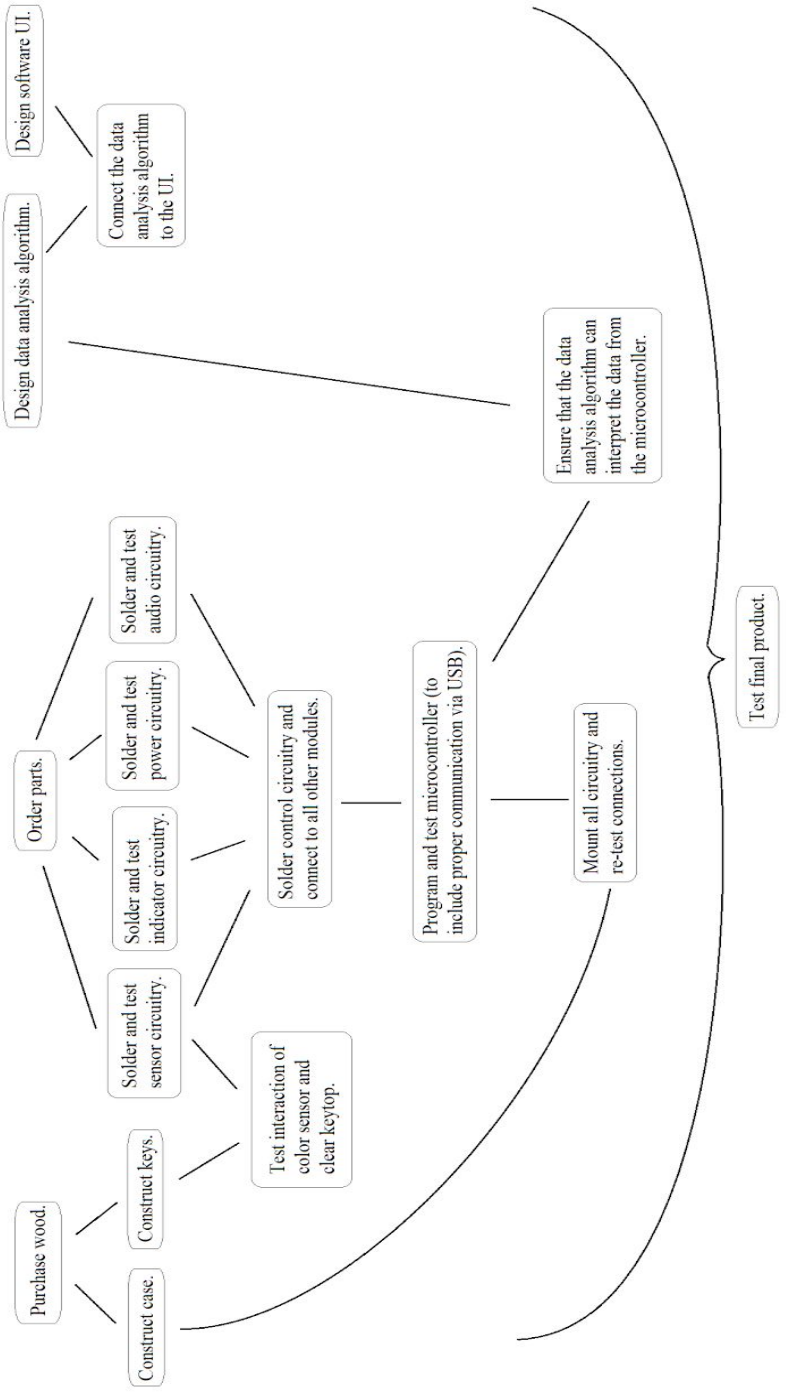


Table 3.1: Individual Task Breakdown

		Shruti Chanumolu	Anna Shewell	Jae Kwak	Expected Time Per Person
1	**2/26**	Incorporate feedback, purchase parts, and finish PCB design.			12 hrs
2	**3/5**	Design the data analysis algorithm.	Design software UI.	Construct keys.	10 hrs
3	**3/12**	Connect the data analysis algorithm to the UI and finish/edit PCB design (if not done).		Construct case.	7 hrs
4	**3/19**	Spring Break			-
5	**3/26**	Solder and test power and audio circuitry.	Solder and test indicator circuitry.	Solder and test sensor circuitry.	15 hrs
6	**4/2**	Solder control circuitry and connect to all other modules.		Test interaction of color sensor and clear keytop.	12 hrs
7	**4/9**	Program and test microcontroller (to include proper communication via USB).			10 hrs
8	**4/16**	Mount all circuitry and re-test connections.			7 hrs
9	**4/23**	Test final product, write final paper, and prepare for presentation.			10 hrs

3.2 Cost of Labor

In order to estimate a reasonable salary for our group members, we looked up the average yearly salary data for ECE graduates collected by Engineering Career Services (ECS) [6]. We chose the averages from 2013 - 2014, the most recent data available (not counting data from 2014 - 2015 which, according to ECS, is still subject to change). This data is self-reported and thus, may be subject to bias. From this average salary information, we applied the standard used by the U.S. Office of Personnel Management [7] to estimate hourly salaries.

Table 3.2: Summary of Salary Data

	Average Yearly Salary	Average Hourly Salary
CompE	$77,653	$77,653 / 2,087 hrs = **$37.21** per hr
EE	$69,342	$69,342 / 2,087 hrs = **$33.23** per hr

Based on our schedule, we estimate that each member will contribute approximately 85 hours total over the course of the project (not counting time spent until this point). Thus, the total cost of our labor will be as follows:

Table 3.3: Labor Costs

Shruti Chanumolu	EE	$33.23 * 85 = **$2,824.55**
Jae Kwak	EE	$33.23 * 85 = **$2,824.55**
Anna Shewell	CompE	$37.21 * 85 = **$3,162.85**
Total		**$8,811.95**

3.3 Cost of Parts

Power Module

Table 3.4: Power Module Costs

Description	**Part #**	**Manufacturer**	**#**	**Cost**
AC/DC Converter	MP1567DK-LF-ND	Monolithic Power Systems Inc.	1	$2.35
Voltage Regulator	AZ1117EH-3.3TRG1 DITR-ND	Diodes Incorporated	2	$0.10 * 2 = $0.20
Total	-	-	-	**$2.55**

Control Module

Table 3.5: Control Module Costs

Description	**Part #**	**Manufacturer**	**#**	**Cost**
Microcontroller	ATMega328-AUR	Microchip	1	$1.80
I/O Expanders	MCP23017	Microchip	6	$1.27 * 6 = $7.62
USB C Connector	10137062-00021LF	Amphenol FCI	1	$2.82
Total	-	-	-	**$12.24**

Input Module

Table 3.6: Input Module Costs

Description	**Part #**	**Manufacturer**	**#**	**Cost**
Photointerrupters	EE-SX1350	Omron	24	$1.86 * 24 = $44.64
Color Sensors	TCS3471	AMS	12	$3.82 * 12 = $45.84
Total	-	-	-	**$90.48**

Indicator Module

Table 3.7: Indicator Module Costs

Description	**Part #**	**Manufacturer**	**#**	**Cost**
LEDs	2228957	Jameco Valuepro	12	$0.35 * 12 = $4.20
LED Memory	242683	Major Brands	6	$1.29 * 6 = $7.74
Timer (For PWM)	27423	Major Brands	1	$0.35
Total	-	-	-	**$12.29**

Audio Module

Table 3.8: Audio Module Costs

Description	**Part #**	**Manufacturer**	**#**	**Cost**
Microcontroller	ATMega328-AUR	Microchip	1	$1.80
Speaker	COM-09151 RoHS	Sparkfun	1	$1.95
Total	-	-	-	**$3.75**

Miscellaneous

Table 3.9: Miscellaneous Costs

Description	**Cost**
Raw material for keys/case	~$20.00
Use of Fab Lab	~$32.00 (for 2 hrs)
Common ICs, Resistors, Etc.	~$20.00
Unforseen Costs	~$20.00
Total	**$92.00**

Total Parts Cost

Table 3.10: Total Parts Costs

Power Module	$2.55
Control Module	$12.24
Input Module	$90.48
Indicator Module	$12.29
Audio Module	$3.75
Miscellaneous	$92
Total	**$213.31**

3.4 Total Costs (Labor + Parts)

Table 3.11: Total Costs

Labor	$8,811.95
Parts	$213.31
Total	**$9,025.26**

3.5 Total Costs For Full-Size Piano

Our budget calculations apply only for the single octave (12 keys) that we intend to create. For a full piano with ~7 octaves (88 keys), the number of sensors and LEDs we'd need increases the cost significantly (see tables 3.11 and 3.12). This estimate may be a little high due to the cost savings of bulk ordering. In any case, our estimate falls within the average cost range of commercial full-size digital pianos (based on the products listed by [9], we estimate a range of ~$200 - ~$1,500).

Table 3.12: Total Parts Cost For Full Piano

Power Module	$2.55
Control Module	$12.24
Input Module	$90.48 * 7.333 = $663.52
Indicator Module	$12.29 * 7.333 = $90.13
Audio Module	$3.75
Miscellaneous	$92
Total	**$864.19**

Table 3.13: Total Costs (Labor + Parts) For Full Piano

Labor	$8,811.95
Parts	$864.19
Total	**$9,676.14**

4 Safety and Ethics

4.1 Safety Issues

As with anything that connects to a wall outlet, there's a chance of electrocution or fire if the plug/circuitry is tampered with, misused, or accidentally compromised. To hopefully avoid this, we will design our piano with proper circuit protections. Additionally, should we or someone else choose to market our design, we would ensure that a safety warning is included in the packaging as per the IEEE Code of Ethics #1, "...to disclose promptly factors that might endanger the public or the environment," [2].

Additionally, we need to ensure that our casing completely covers our circuitry and that our casing and keys won't break under the force of the key presses. This is to prevent accidental contact and short circuits. If a user were to come in contact with our circuitry, they could be shocked, burned, or electrocuted. Additionally, exposed circuitry and sensors are more liable to break, either from a short circuit or from physical stress. Although we will not be making our piano waterproof, our casing should prevent short circuits caused by liquid spills from endangering the user.

Our circuitry also has the potential to overheat. To prevent this, we will make sure that our power module outputs appropriate voltages and currents for our various components. We will also test our final product for varying lengths of time to determine if heat builds up. Although we may not have time to redesign our circuitry to mitigate any heat concerns at that stage, we will make note of it for future improvement and for inclusion in a safety warning.

In terms of our own safety, we run a higher risk of coming in contact with live circuitry. We'll also be using soldering irons and could burn ourselves or develop breathing issues from inhaling the fumes. To mitigate these risks, we've completed lab safety training and will follow the safety procedures outlined there along with the safety procedures we've learned from past experience.

4.2 Ethical Issues

The most likely ethical issue we'll face will be accurately reporting the capabilities of our project. It is difficult for us to tell at this stage how accurate and robust our design will be. Nevertheless, we will maintain the integrity of our work and be honest even if the final product falls below expectations in keeping with the IEEE Code of Ethics #3, "...to be honest and realistic in stating claims or estimates based on available data," [2].

Also of great importance to our project is the IEEE Code of Ethics #7, "...to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others," [2]. We will welcome criticism of our work and do our best to fix any errors in our project. In keeping both with the aforementioned IEEE policy and also with one of the University of Illinois's most important policies, we will not plagiarize anyone else's work and will give credit to outside sources whenever necessary.

Lastly, in keeping with the IEEE Code of Ethics #10, "...to assist colleagues and co-workers in their professional development and to support them in following this code of ethics, " [2], we will offer our aid to any of our fellow peers in their projects should they need it given we have the requisite knowledge and resources to do so.

References

[1] A. Mazzocchi, “Why Students Really Quit Their Musical Instrument (and How Parents Can Prevent It),” *The Music Parents' Guide*, 17-Feb-2015. [Online]. Available: http://www.musicparentsguide.com/2015/02/17/students-really-quit-musical-instrument-parents-can-preven. [Accessed: 08-Feb-2018].

[2] “IEEE Code of Ethics,” *IEEE*. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 08-Feb-2018].

[3] “How Colour Changing LEDs Work,” *Kitronik*. [Online]. Available: http://www.kitronik.co.uk/blog/how-colour-changing-leds-work. [Accessed: 08-Feb-2018].

[4] “Tempo,” *Wikipedia*. [Online]. Available: en.wikipedia.org/wiki/Tempo#Basic_tempo_markings. [Accessed: 08-Feb-2018].

[5] “Musical Keyboard,” *Wikipedia.* [Online]. Available: en.wikipedia.org/wiki/Musical_Keyboard. [Accessed: 08-Feb-2018].

[6] “Salary Averages”, ECE Illinois. [Online]. Available: https://ece.illinois.edu/admissions/why-ece/salary-averages.asp. [Accessed 22-Feb-2018].

[7] “Computing Hourly Rates of Pay Using the 2,087-Hour Divisor”, *OPM.GOV*. [Online]. Available: https://www.opm.gov/policy-data-oversight/pay-leave/pay-administration/fact-sheets/computing-hourly-rates-of-pay-using-the-2087-hour-divisor/. [Accessed 22-Feb-2018].

[8] “Pulse Width Modulation”, Electronics Tutorials. [Online]. Available: https://www.electronics-tutorials.ws/blog/pulse-width-modulation.html. [Accessed 22-Feb-2018].

[9] “Digital Pianos”, Guitar Center. [Online]. Available: http://www.guitarcenter.com/Digital-Pianos.gc. [Accessed 26-Feb-2018].