Facilitated Instrument Learning

By

Christopher Chen

Theodore Lao

Jiajun Xu

Final Report for ECE 445, Senior Design, Spring 2018

TA: Zhen Qin

2 May 2018

Project No. 5

Abstract

This report documents our design, implementation, and verification of our facilitated instrument learning system. We discuss the initial requirements for our design and how we satisfied those requirements. Furthermore, we explain the algorithms behind pith detection used in the system and also the hardware design of it. It is concluded with a description of the accomplishments and future work.

Contents

1 Introduction

Musicians spend a substantial amount of time learning the positions of chords and notes on new instruments they are interested in learning. Facilitated instrument learning will allow one to sing, hum, or play another instrument and the notes being played will be mapped onto the new instrument in real time. This allows a beginner, with little musical background, to sing a melody they wish to play and learn it on a new instrument and also allows a professional, with an extensive background on musical theory and other instruments, to compose music on new instruments.

The built solution contains three main modules: Power, Audio, and Control. All components require either a 3.3V or 5V source. Therefore, the power module contains buck converters and linear regulators to provide these two voltages. The power module's source are two rechargeable lithium ion battery which operates between 6.3 to 8.4 V. The Audio unit contains an acoustic input implemented as an electret condensor microphone going through an amplifier. The signal is sent to the control system which contains an ARM M4 microcontroller with a built in ADC. The control system samples the data and performs the analysis on it. It then uses a GPIO pin to send serial commands to a string of LEDs, each corresponding to a certain pitch on a piano key. The LCD and joystick peripherals in the control unit enhance the user experience by displaying the current note that is being played and enabling record and replay modes.

1.1 Block Diagram

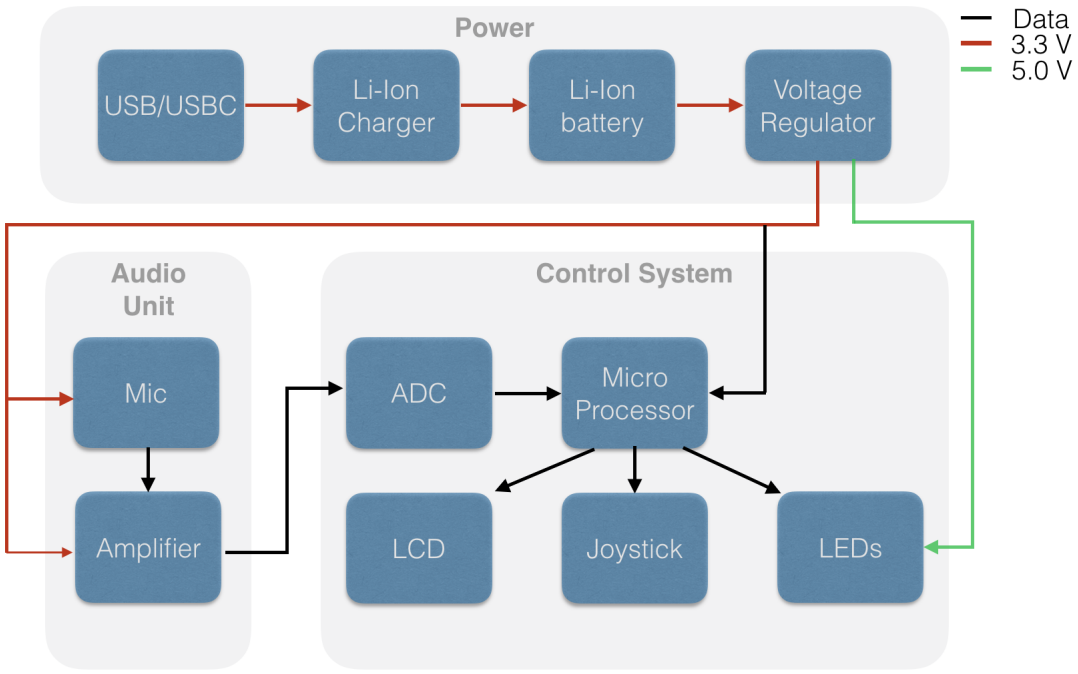

Figure 1: Project Block Diagram

1.2 High Level Requirement Lists

The requirements listed below were the initial goals of this project. They were created to quantify the quality of our system as we were building it. The specifications ensure that if we meet those criteria, the system will operate well enough to ensure a quality user experience.

- Detect pitches accurately at least 80 percent of the time,
- Be able to detect pitches in less than 200ms
- Record at least two instruments, voice and guitar, and accurately map the notes to the piano.

1.3 Competing Products

There are other products in the market which help people earn instruments but they have different limitations which this project overcomes.

SCI V9000 KEY/NOTE VISUALIZER. This $2000 product is limited because notes must be pre-selected on the piano for students to learn from later. With the proposed project, users produce the melodies with their voice or an instrument and the notes on the destination instrument will be shown in real time.

Synthesia This piano learning subscription service is limited to select songs the company has pre-transcribed for its users, so customers are unable to write songs with melodies they have come up with. Our system allows users to make up melodies on the spot and learn them on the target instrument instantly.

2 Design

2.1 Power Unit

Our device has very specific power requirements, which we have to implement as efficiently as possible. When considering the design, we first took a look at the voltage requirements for our circuit. All of the ICs could be run off of 3.3V, but the LEDs required a 5V connection due to its maximum brightness. Our power input would thus consist of a two-cell lithium-ion battery, which has a nominal voltage rating of 7.4V. From resources online, it appeared that each battery has a maximum voltage of 4.2V, and shouldn't be run under 3.0V, meaning we needed to supply a 5V and 3.3V connection from an input that could range from 6.0V to 8.4V. With almost 60mA per LED, and a maximum of 88 LEDs, we had to find a compromise and provide a power circuit that could efficiently supply a large amount of current.

A linear regular would not be a good choice for this application. Stepping 8.4V down to 5V and drawing as much current as we expect, too much power would be lost as heat through a linear regulator. We ended up deciding to use the TPS62153 buck converter from Texas Instruments. This specific integrated circuit provides a very consistent 5V power from a large source voltage range, and was relatively simple to implement. The ripple for the 5V rail was not as important, since the LEDs aren't as sensitive to it as, for example, our microprocessor was. During our testing, this IC was able to maintain a consistent 5.034V from an input between 5V and 17V. We also decided to turn down the brightness of the LEDs, since the default maximum brightness was too bright for comfortable viewing. By using this combination, we were able to increase the battery life and efficiency of our device.

The microprocessor, amplifier, and our external peripherals all required a 3.3V connection. The power draw for these devices isn't nearly as much as the LEDs, and the cleanliness of the voltage was important, since the MCU is highly sensitive to ripple currents. Because of this, we chose to use a 5V to 3.3V low dropout linear regulator to meet the demands. The schematic can be found in Figure 2 and our tests for both the 5V and 3.3V units can be seen in Figure3. The battery voltage range is indicated on the graph.

We also decided to use two separate sets of 5V and 3.3V converters. The TPS62153 has an enable pin, which can be supplied also from a large voltage range. The MCU has a very low power mode that only responds to basic GPIO inputs, essentially drawing little to no current. We decided to add a solder jumper, allowing us to connect the MCU to the enable pin of the second set of voltage converters. Any device that did not need to be on when not in use, including the audio amplifier, LEDs, and peripherals, would then have the power cut. The buck converter could be enabled through the GPIO pins on the MCU when usage of these devices are necessary. If this did not work, we could solder the enable pin to the V+ pin, hard enabling the voltage converters. This created a robust platform that allowed room to correct errors.

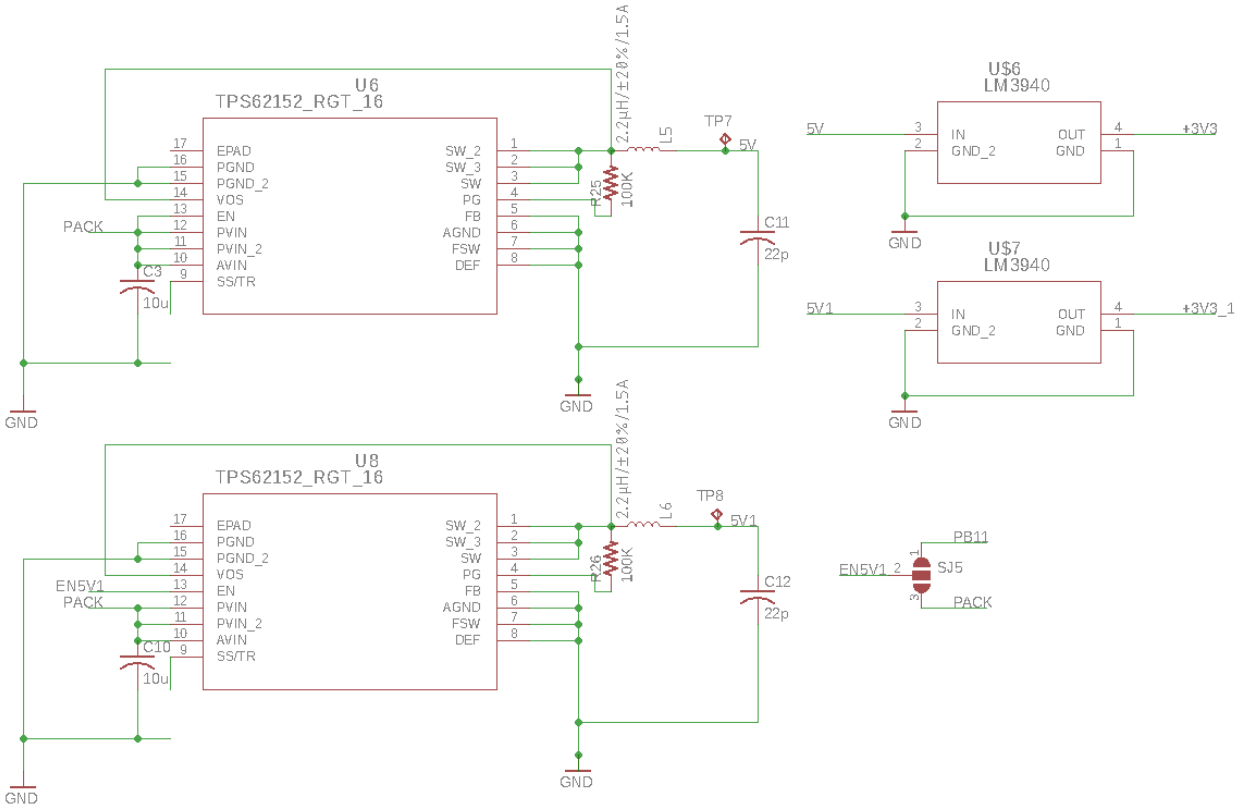

Figure 2: Power Unit Schematic

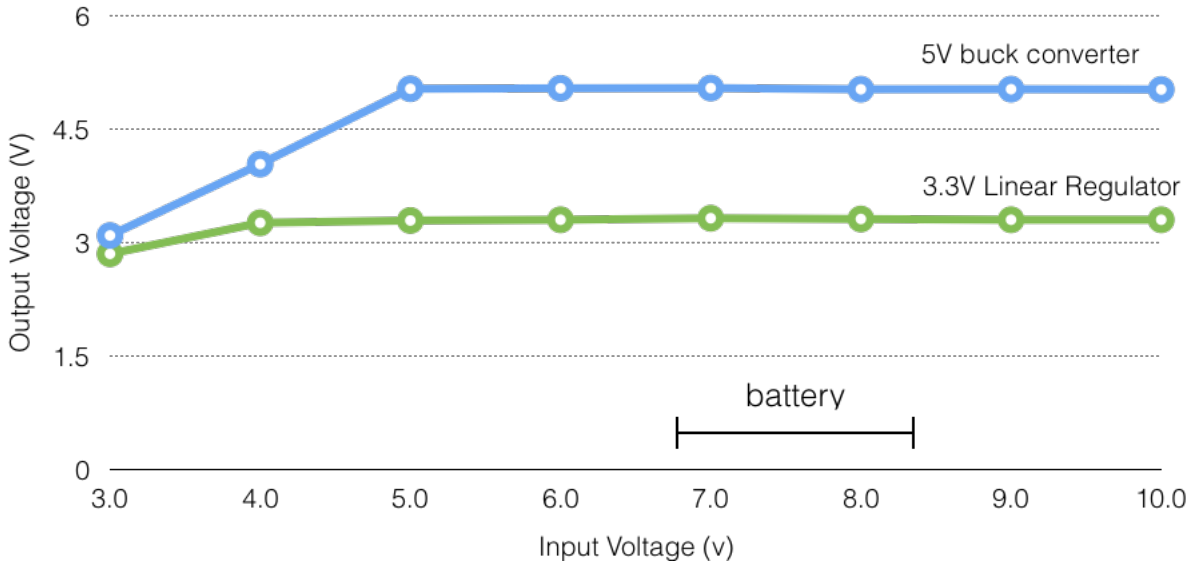

Figure 3: Power Unit 3.3V and 5V tests

2.2 Battery Charging Unit

The design of our battery charging circuit incorporates a BQ24103 switch mode Li-Ion battery charger that supports up to 2A charging on a two cell battery. We initially wanted to charge through a standard USB port, however we dismissed that idea, due to the fact that the voltage and current supply on a standard USB port is very small and would not realistically charge our device in a reasonable amount of time. Instead, our power is provided through a 9-12V port towards the bottom of the PCB, and implements power switching MOSFETs to charge the battery at higher currents. Ideally, this configuration allows the charging of our 7000mAh system in around 2-3 hours. The schematic for this system can be found in Figure 4.

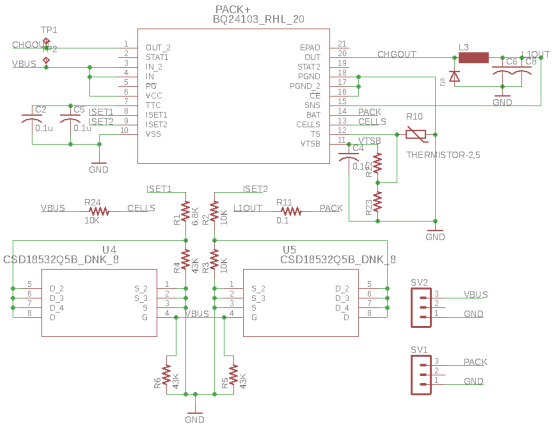

Figure 4: Battery Charging Schematic

2.3 Audio Unit

For our device, we needed an audio unit that could take input from a wide variety of input instruments and provide a clean signal for our algorithms. The lowest note on a piano is at around 30 Hz, while the highest pitch has a frequency of about 4000 Hz. We needed to be able to sample with a microphone within this range, so we chose the CMA-4544PF, which has a frequency range of 20Hz - 20kHz, while still operating at our voltage of 3.3V. The other thing to consider was the volume of the input signal. The user playing the initial instrument could be sitting very close to the device, or potentially a few feet away. They could play the instrument with either a louder or softer attack. With a typical microphone amplifier, a sound too loud in volume could result in a clipped signal, which would be a challenge to analyze. Thus, we chose the MAX9814 amplifier due to its automatic gain capabilities. Since exact replication is not as important as preserving the frequency data of the signal, we found this to be much more suitable for our application. The rest of the components are a result of design considerations found on the datasheet.

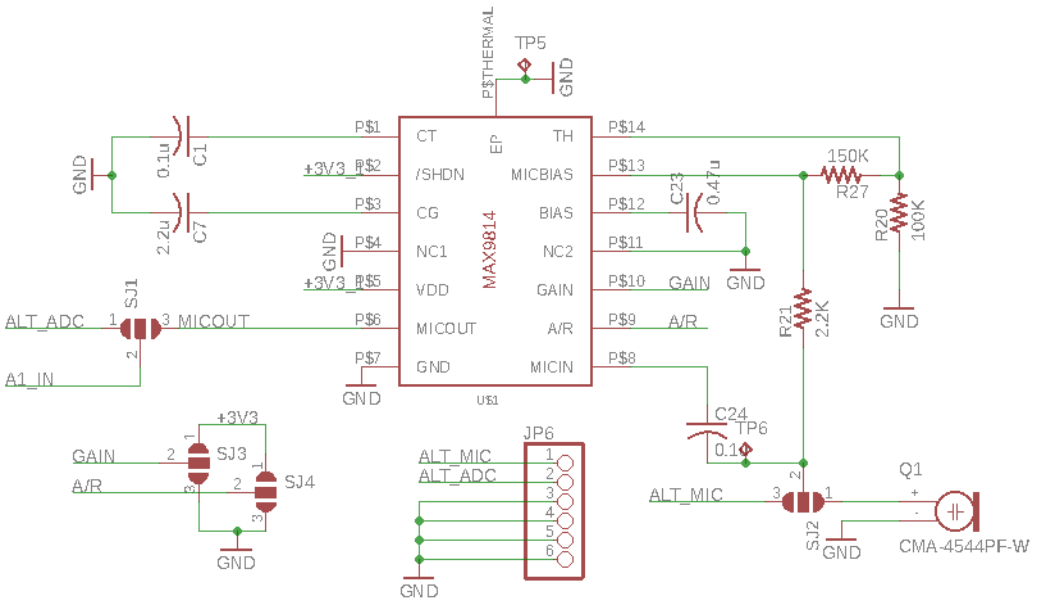

Figure 5: Audio Unit Schematic

2.4 Control Unit

The core functionality of the control unit of the system is to take the output of the audio unit as the input, run the signal processing algorithm on the audio inputs, and output the LED signals. It also has the ability to use the joystick to initiate the recording of the played notes. All the control logic are implemented on our microprocessor chip.

2.4.1 Microprocessor

Portability, while having sufficient computing power is crucial for our design. An Atmega328 would be the simplest MCU to implement, however it doesn't have the necessary computing power and features that we are looking for. After doing some research, the microprocessor we chose is the Pearl Gecko series EFM32PG1B200F256GM32. It is a 32-bit, ARM Cortex-M4 based single core processor. The design of this power is centered around low power applications, providing four energy modes for use in various situations. One of the requirements of our device was portability, and using an energy efficient processor provides a great step towards this goal. This processor also includes a built in 12 bit ADC, which is a very useful feature since we can process analog data that is being fed directly into the MCU. One of the improvements of the ARM Cortex-M4 architecture over the M3 is the inclusion of enhanced DSP libraries, which provides optimized libraries for many useful DSP functions, with FFT being one of them. It provides the perfect all-in-one package for our implementation, while including a feature packed IDE, Simplicity Studio, to debug and monitor energy usage. The wiring schematic for the microprocessor and its power circuit can be seen in Figures 6 and 7.

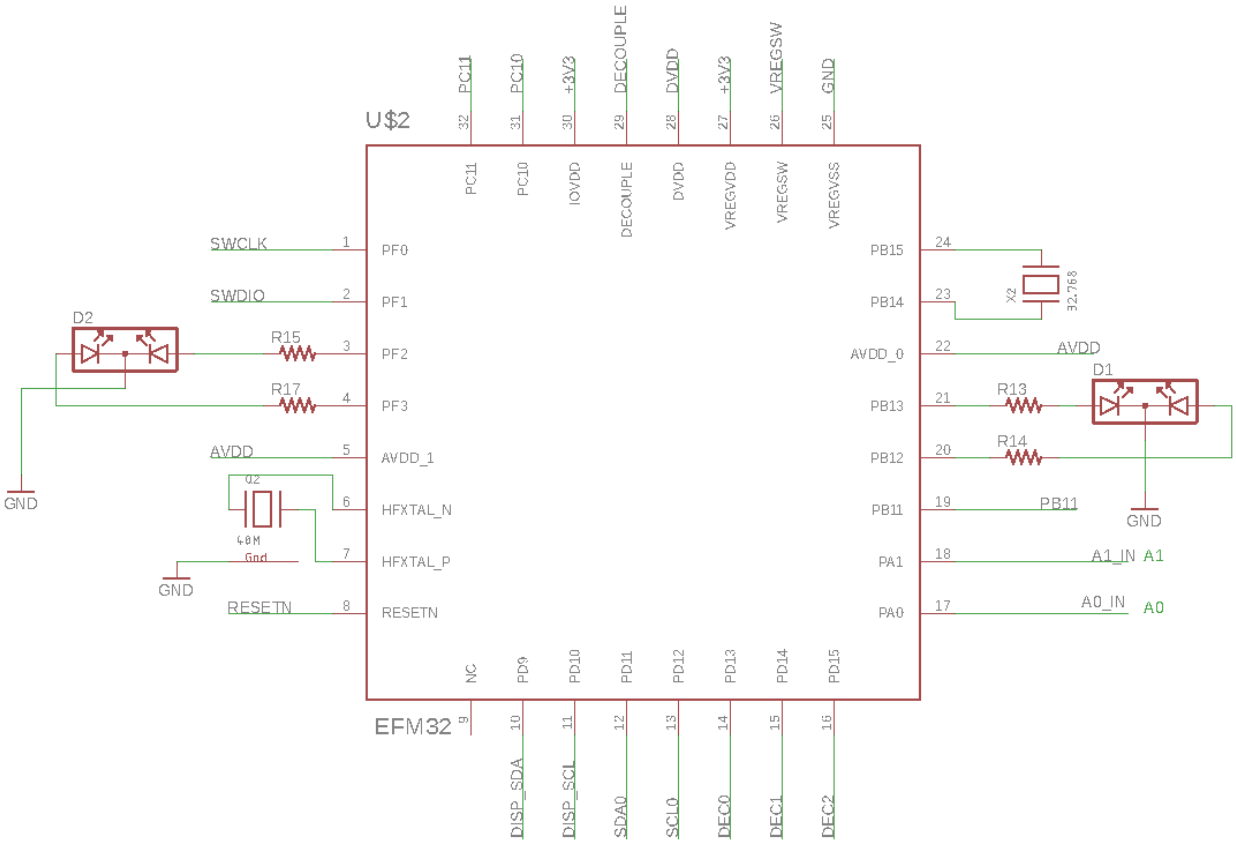

Figure 6: MCU Schematic

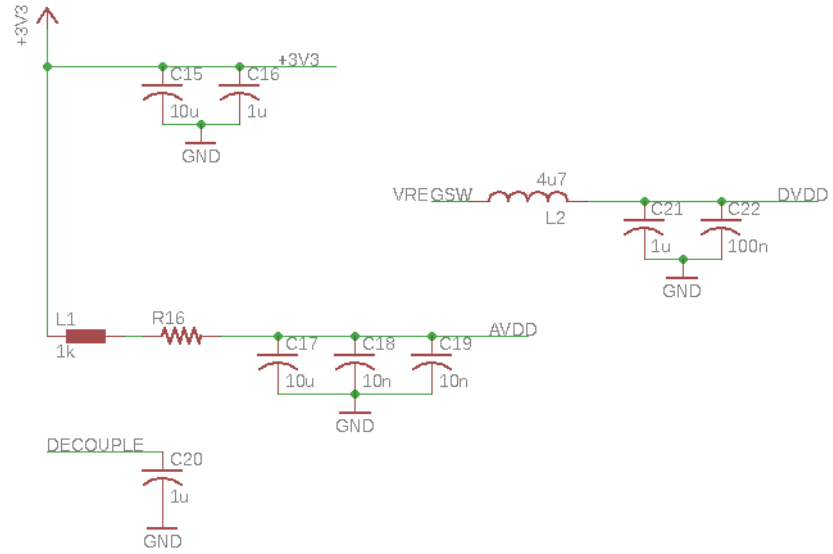

Figure 7: MCU Power Supply Schematic

2.4.2 Control Logic

There are three main parts from input to output: sampling, pitch analysis, note indication. During the sampling phase, the voice/instrument outputs are first converted from analog signals to digital signals, and then are put in the data buffer. In our current algorithm, we need the sampling to run at 16 kHz, and 512 samples are analyzed for the pitch detection. During the pitch analysis phase, the data buffer goes into the pitch detection algorithm, and the algorithm will output the note corresponding to the samples. During the note indication phase, the output of our pitch analysis algorithm is converted to the signals to light up the corresponding LEDs. Additionally, when the button turns on the record mode, the control logic stores the recorded notes in memory to be played at a later time when the user presses the replay button.

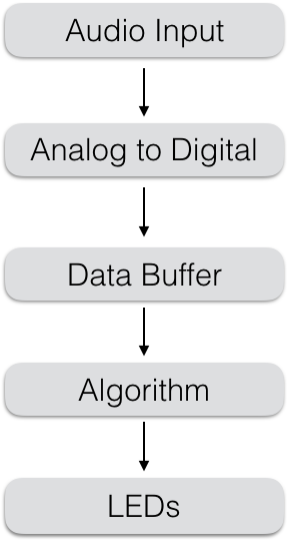

Figure 8: Control Logic Diagram

Due to budget and time constraints, we picked a single core microprocessor, which forces us to run the sampling and the pitch analysis phases in sequence. The consequence of this sequential structure is that when the system is sampling from the microphone, it's not doing the analysis; when the system is doing the analysis, it's not sampling from the microphone. That means we aren't analyzing all the sound that's generated form the instruments. So the performance of our system depends on how fast the sampling speed and the algorithm speed. With our implementation of the Yin algorithm on the microcontroller, it can finish detect one pitch in about 30 ms. In our observations, it works pretty well – we don't notice missing notes if the instruments are being playing at a normal speed.

One potential alternative to our microcontroller is a multi-core controller. We can run 2 threads concurrently to solve the above issue. One thread will be dedicated to sampling from the microphone, and the other thread will be dedicated to perform the pitch detection algorithm. In this case, all the microphone data will be sampled by the sampling thread, and all the sampled data can be analyzed by the analysis thread.

2.4.3 Peripherals

At the most basic level, our board needs to include a 4 pin port that interfaces with the external LEDs. A separate 3 pin port should be used as a power connection. However, we decided to include many extra headers in case we need to add additional devices. both 5V and 3.3V connections have headers, as well as multiple SCL and SDA ports for I2C communication. An additional separate 4 pin port is included on the opposite side, in case a second strip of LEDs are necessary. By adding these ports, we ensure that any errors and roadblocks can be rectified on the spot, instead of having to redesign an additional board.

We also include several IO components, including a 4 direction joystick with center select, and an LCD display. By allowing the user to provide input to select various settings and algorithms, or to perform certain actions such as recording, we are able to create a much more immersive experience. The connectors for these components can be seen in Figure 9.

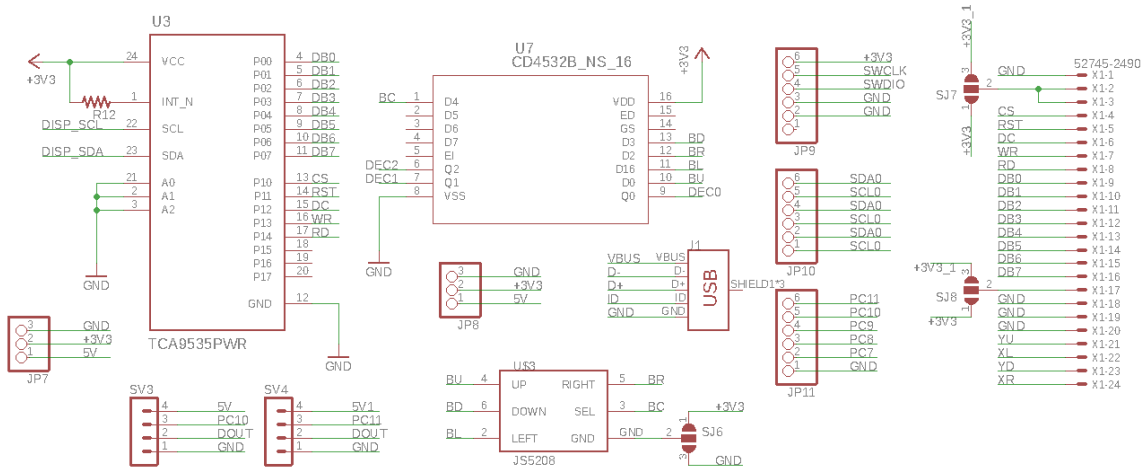

Figure 9: Schematic of Peripherals and Connectors

2.4.4 LEDs

We are using a series of WS2812bs, individually addressable LEDs, for notes indication. In the piano implementation there are 12 LEDs per octave. The LEDs are connected serially, allowing them to be modular. Four pins are used to connect each LED board to the next: Vdd, Vss, Din, and Dout.

WS2812's require precise timing on the data line which can be determined from its data sheet. There is only a single line which contains the data to 1-255 for red, green, and blue colors for each LED. A logical HIGH is determined when the line is high for a 600-900 ns and a logical LOW is determined when the line is HIGH for 250-400 ns.

We used an SPI's (Serial Peripheral Interface) MOSI line on the microcontroller at 3.33 MHz to satisfy the strict timing requirements. Therefore, each bit is on the line for 300 ns. Each bit will take Therefore a HIGH is on the line when the bit stream is 110 and a LOW is on the line when the bit stream is 100.

2.5 PCB and Enclosure Design

Packaging our device was also an important aspect of our project. Since our device is purposed for a specific application, it's necessary for us to develop a design that's intuitive while still attractive. We designed our PCB such that the external ports would be located on the outside of the board. That way, if we needed access to these ports, we could implement cutouts at these locations. Our final enclosure design only allowed space for three external ports: a charging port, the LED connector port, and the programming port.

The three part design consisted of a base layer which allocated space for the power switch and batteries, a middle layer which consisted of a tray for the PCB, isolating it from the battery, and the top cover, which had cutouts for the LCD, buttons, status LEDs, and microphone. The base, second layer, and PCB were held together using two screws from the top of the PCB through the battery layer, and the top cover snapped on. The assembly of the 3 layers can be seen in Figure 10.

Figure 10: Main device enclosure design

The design of the LED enclosure was also important for our device to provide full functionality. We needed the LEDs to sit on top of the piano keys and to be able to clearly indicate the key that it intended the user

to press. To do so, we spaced the LEDs on the PCB such that each LED lined up with the center of each black and white key. The enclosure for the LEDs had holes for each LED, and legs such that one module sat on one octave from note C to B. Many keys could be simultaneously pressed while still allowing each LED module to stand on the rest of the non pressed keys. The enclosure design can be found in Figure 11.

The device and the LED holders are shown in Figure 12 on a keyboard.

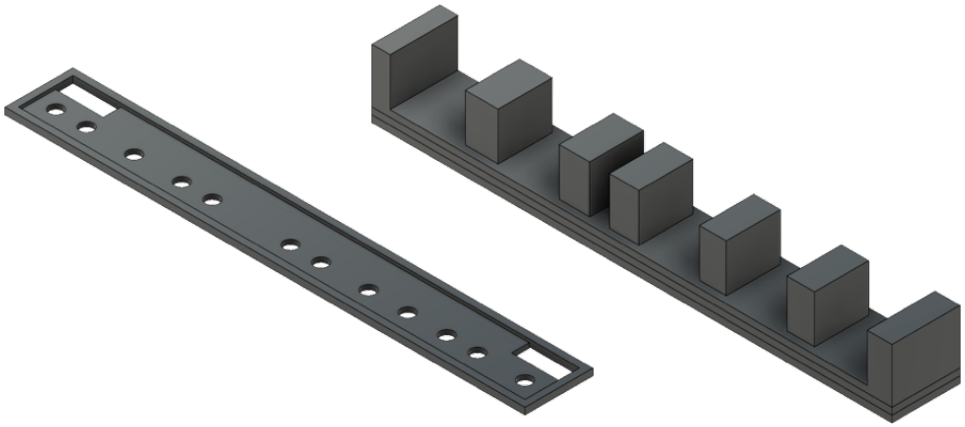

Figure 11: LED module enclosure

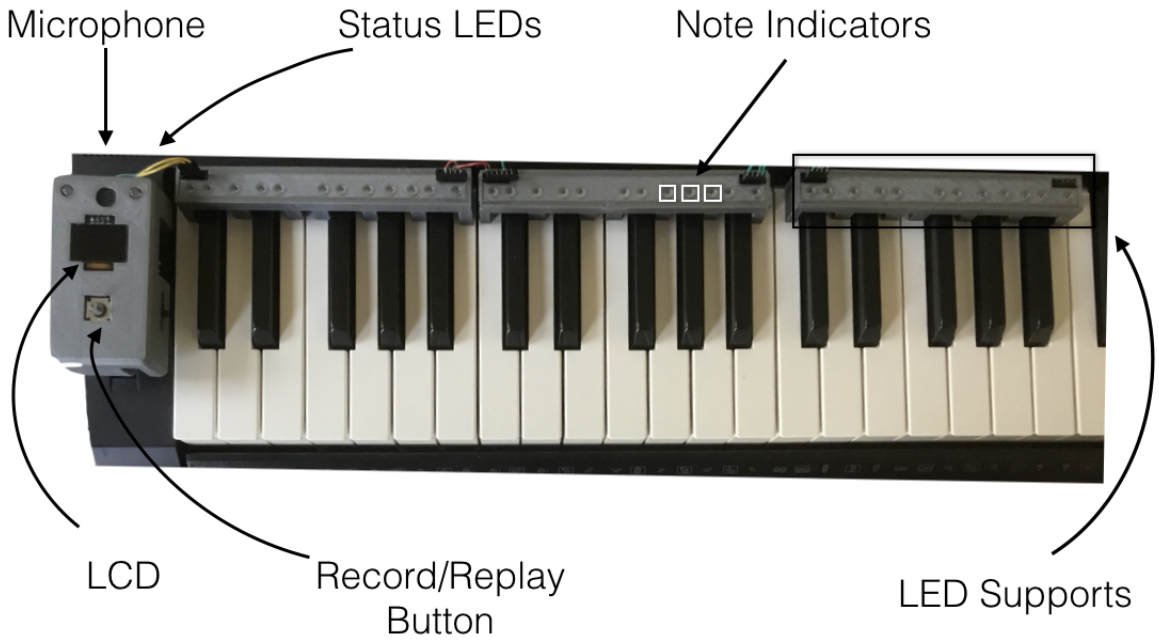

Figure 12: Complete System Design

2.6 Signal Processing Algorithms

Algorithms programmed into the system handle the pitch (fundamental frequency) detection. Given an audio waveform, the fundamental frequency should be detected accurately. There are many obstacles to fundamental frequency detection such as noise, harmonics and overtones, pitch precision, and instrument timbre. Instrument timbre is extra frequency content variation between different instrument. Sources of these extra variations can be due to resonating tones in the unique body of an instrument.

After the algorithms determined fundamental frequencies in the audio, the LED index was determined by finding the number of notes away it was from a reference note[4].

$$f_n = f_0 * 2^{\frac{1}{12n}} \tag{1}$$

where f_n is the detected frequency, f_0 is the reference note's frequency, and n is the number of musical steps away from it.

Three algorithms were researched and implemented. The Yin algorithm is a single fundamental frequency pitch detector and performs time domain analysis. The Harmonic Amplitude summation and non-negative matrix factorization are meant for polyphonic, multiple fundamental frequency, pitch detection. Yin is an accurate, robust algorithm for monophonic analysis and Harmonic Amplitude Summation had improved accuracy over non-negative matrix factorization.

2.6.1 Yin Algorithm [1]

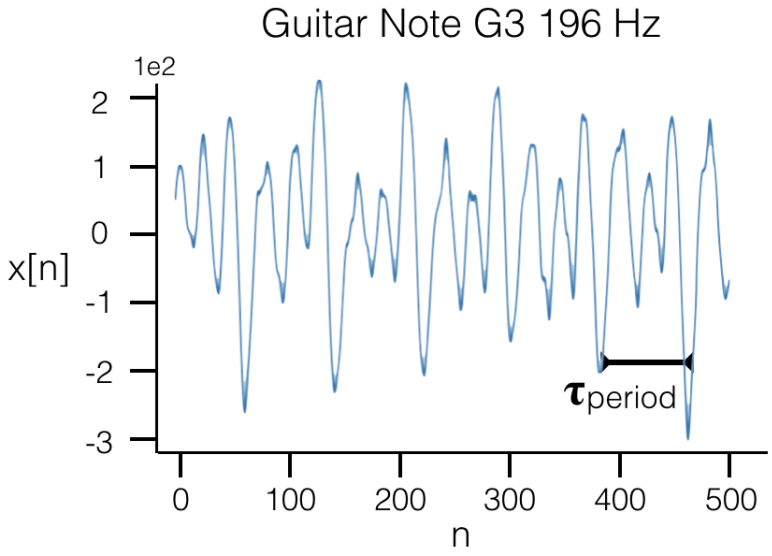

Figure 13: Audio Waveform 293Hz

Yins objective is to determine the fundamental period of a given audio signal. Because the audio is a musical pitch, the waveform will be periodic. When the period is known, the fundamental can be determined by dividing the sampling rate by the fundamental period. The fundamental period, tau, is determined in four steps. First, a squared difference is calculated between the original signal and its shifted self. Then the

result is normalized. When a periodic signal is shifted by a period and subtracted from its initial self, the result is zero since its the same signal repeating itself. Therefore, there will be minimums. We will search for these minima under a certain threshold we select and the shift that corresponds to that minimum is the fundamental period tau.

The squared difference (2) contains minima close to zero when the signal is shifted to integer multiples of its fundamental period. An ideal periodic signal with no noise will have a squared difference of zero because a periodic signal shifted by a period is just the initial periodic signal. This squared difference is calculated for all shifts up to half of the total length of the buffer, W. There are other local minima in the data that are a result of harmonics.

$$d_t(\tau) = \sum_{j=1}^{W} (x_j - x_{j+\tau})^2 \tag{2}$$

where W is half of the buffer size and τ is the shift.

$$d'_t(\tau) = d(\tau) \Big/ [(1/\tau) \sum_{j=1}^{\tau} d_t(\tau)] \tag{3}$$

$$F_0 = \frac{F_s}{\tau_{period}} \tag{4}$$

where F_0 is the detected fundamental frequency or pitch of the note, F_s is the sampling rate, and τ_{period} is the determined minimum's delay value.

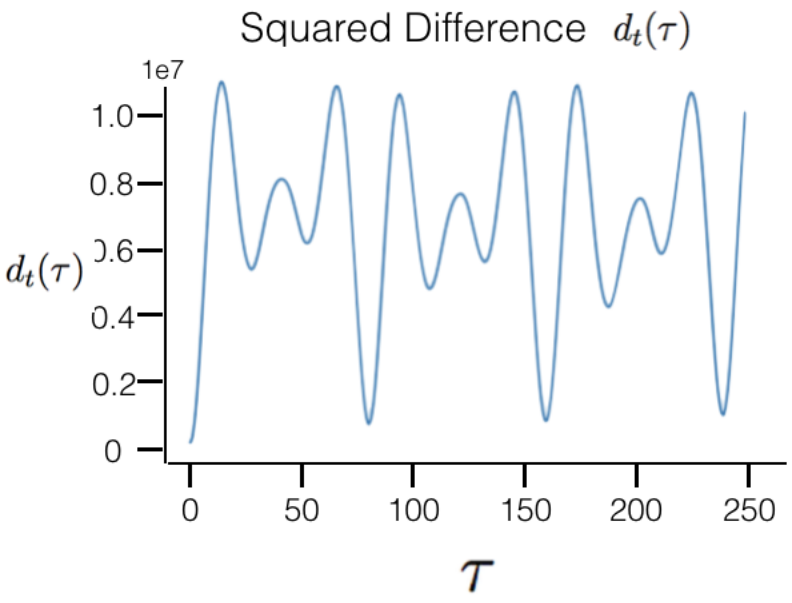

Figure 14: Squared Difference

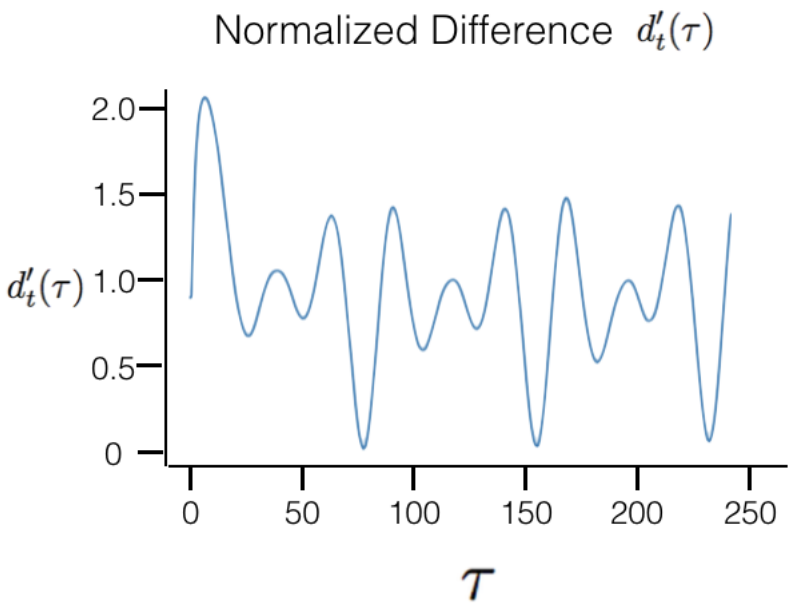

Figure 15: Normalized Squared Difference

Afterwards, the squared difference is normalized by computing the cumulative mean normalized difference (3). The most visible effect of this are the 0th multiple of the period is no longer a minimum. The cumulative mean normalized difference computes the mean up to each shift and divides the squared difference by it. This causes the harmonic minima, which are near the mean of the squared difference to have a value near one, while the minima will have values below 0.3.

Now that the signal is normalized, the first global minima should be under a controllable threshold value. The index where the minima occurs is found when the search finds a value below the threshold. After this point the search continues until the signal begins to increase. Then it uses the values near the minima to perform parabolic interpolation to find a more precise value for the period.

With the determined fundamental period the fundamental frequency of the signal is computed by dividing the sampling rate by the period.

The Yin algorithm is robust enough that successfully detected pitches for all the instruments used as inputs: guitar, vocal, and piano. This is because Yin requires no training and solely searches for the period in a given signal.

The drawback of the Yin algorithm is that it only detects a single fundamental frequency, meaning it only detects melodies.

2.6.2 Harmonic Amplitude Summation Algorithm[3]

Summing harmonic amplitudes in the frequency domain is intended to increase the number of detectable fundamental frequencies when a chord is played. The FFT bins associated with each possible fundamental frequency F_0 along with its harmonic bins are pre-calculated before the algorithms begins processing real time audio data.

There are three main steps in this algorithm that each audio frame goes through. First, the frame is pre-processed with spectral whitening. Then the saliences (5), or un-normalized probabilities that each F_0 exist in the spectrum, are calculated.

A hanning window is applied to the signal and is then zero padded to twice its length before applying the fourier transform. Next, triangular power filters are applied to the spectrum to reduce the effects of instrument timbre.

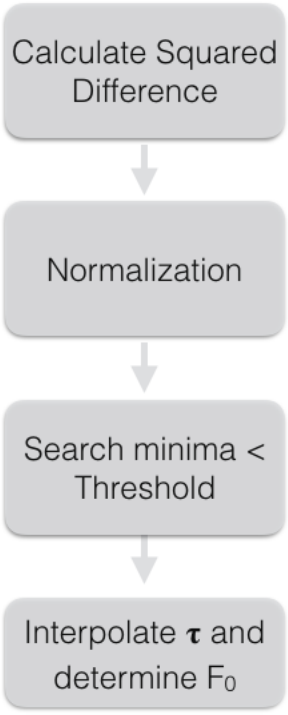

Figure 16: Harmonic Amplitude Summation

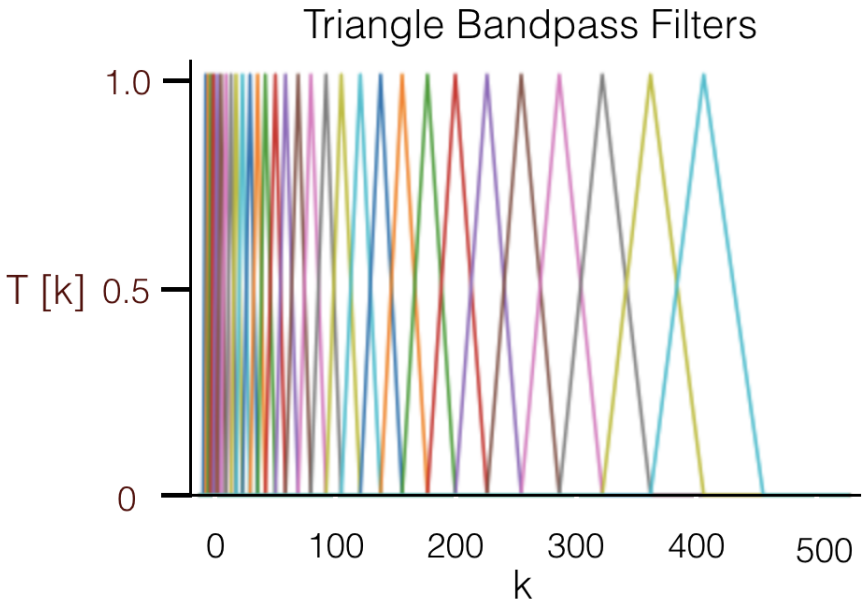

Figure 17: Triangular Filters

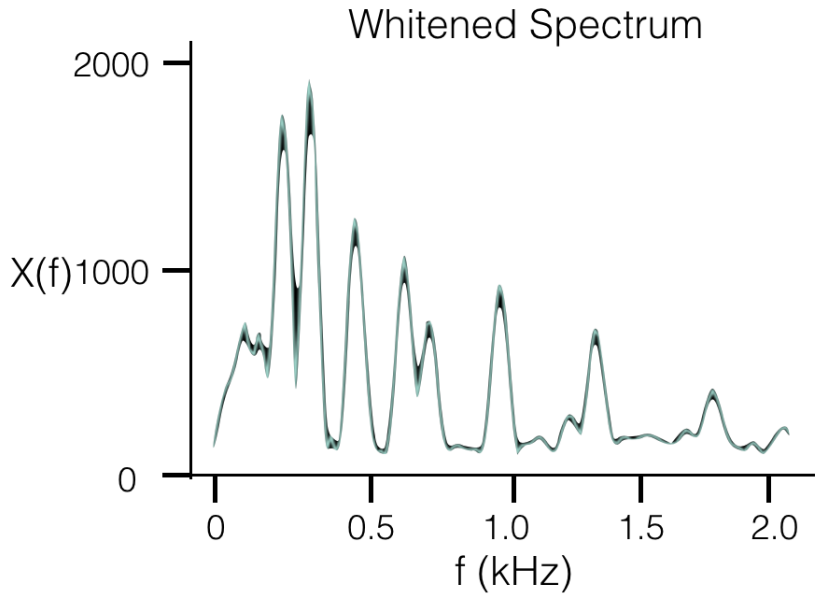

Figure 18: Whitened Spectrum

As mentioned earlier, the saliences are calculated for each potential fundamental frequency (5). The weight g (6) is a function of the fundamental frequency and its harmonics along with two constants *alpha* and *beta*. Afterwards, the fundamental frequency with the highest salience that is greater than a threshold is determined to be in the spectrum.

$$\hat{s}(\tau) = \sum_{m=1}^{M} g(\tau, m) \max_{k \epsilon k_{\tau,m}} (|X[k]|) \tag{5}$$

$$g(\tau, m) = \frac{f_s/\tau + \alpha}{m f_s/\tau + \beta} \tag{6}$$

where τ is the index of the fundamental frequency being analyzed, m is the mth harmonic of the fundamental, and k is the bin indices belonging to the τ fundamental frequency and its harmonic bins.

If a fundamental frequency is determined to be in the spectrum, it is then attenuated from the spectrum. Then the residual spectrum goes back to the salience calculation to search for more fundamental frequencies. If another fundamental is determined, it is attenuated. These two steps repeat until there are all saliences are below the threshold.

2.6.3 Non-negative Matrix Factorization (NMF)

This algorithm requires pre-recorded training data for the all notes it wishes to identify. This data is used to compute a template library from **M** by computing feature vectors from each recording [8]. Each column of which represents the feature vector for a a training example from a particular instrument. All of the notes represented in the columns **M** form the search space from which the algorithm is looking for the notes.

This algorithm represents the input audio as an 1-dimensional feature vector, **V**. The algorithm then solves the pitch detection problem by finding an activation vector, **H**. Each element of **H** is the probability of the existence of the note for the corresponding column in the template matrix, **M**. When **H** is multiplied together with **M**, it will approximate the input feature vector **V**. The cost function used in this method is explained in [5].

$$\mathbf{V} \approx \mathbf{MH}. \tag{7}$$

The algorithm then determines the notes played by identifying the notes whose probabilities are greater than a certain threshold. A flowchart for this algorithm is shown below.

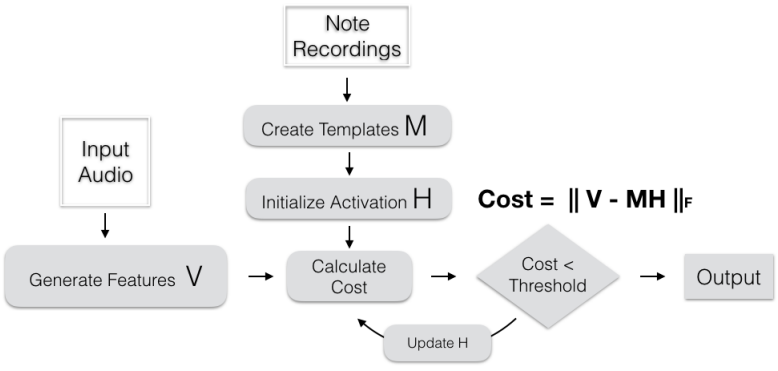

Figure 19: Non-negative Matrix Factoriazation

3 Design Verification

In this section, we discuss how the system performs with respect to the high level requirements we set at the beginning of the project. these are the system's accuracy, latency, and robustness to allow multiple source instruments.

3.1 Accuracy Requirement

As mentioned in the high level requirements, accuracy is crucial for a well-performing instrument learning system. Tests were performed for all three algorithms. The Yin algorithm was tested on the system itself. However, non-negative matrix factorization was only tested in python because accuracy was too low to perform well. Harmonic amplitude summation was tested in python because the micro controller ran out of memory when porting the converted C code onto it.

The accuracy was split into two categories. One is an accuracy excluding harmonic errors. This excludes detected harmonics as errors. Then the other is accuracy including harmonic errors. This includes detected harmonics as errors.

To test the Yin algorithm, the notes from E2 to E5 (48 notes) were played on the guitar for five times each. The number of correctly detected notes divided by the total number of tests 48*5 = 360.

The accuracy excluding harmonic errors successfully detected 342 out of 360 total tests correctly. This resulted in a 95 percent accuracy. When harmonic errors are considered errors the system correctly identified 288 notes out of 360 total tests, resulting in 80 percent accuracy. Both of these values are above the specification we set for the system.

During the testing of the other two algorithms, three note chords were played on the guitar at a time. This was repeated for five different chords and repeated seven times. The accuracy for polyphonic algorithms is calculated as the number of correctly detected notes divided by the total number of notes played: five chords * three notes * seven times = 105 notes.

Summing harmonic amplitudes had 58 percent accuracy for chord detection when excluding harmonic errors and 45 percent accuracy when including harmonic errors. Non-negative matrix factorization had 43 percent accuracy when excluding harmonic errors and 33 percent when including harmonic errors.

Both algorithms did not reach the 80 percent accuracy specification set for the project. However polyphonic pitch detection is still an unsolved problem and most papers claim around 50 percent accuracy. Therefore, we declared harmonic amplitude summation to be a good implementation as opposed to non-negative matrix factorization.

After the tests, the Yin algorithm was robust enough for a quality user experience and harmonic amplitude summation had relatively good results but is not ready for consumers.

3.2 Real-time Requirement

Our system is required to be able to detect notes being played within 200 ms. In this section, we are going to show that our implementation, in practice, satisfies that requirement.

Our sampling step, which runs at 16 kHz, fills the buffer of size 512 in

$$512 * \frac{1}{16000} = 0.0032s \tag{8}$$

, which is about 30 ms. The buffer size and sampling rate were the same for all three algorithms we analyzed.

We tested the three algorithms' average run-times. The Yin algorithm took 30 ms to run on the microcontroller and it also took 30 ms to run on python. Harmonic amplitude summation took 31 ms, and harmonic takes also about 30 ms in python.

Our note indication step involves lighting up 36 LEDs. Each LED needs 24 bits (8 bits for each of the Red, Green, Blue channel) to light up. Every bit takes a pulse of length $1.2\mu s$. So to light up 36 LEDs at once, we need

$$36 * 24 * 1.2 = 1036.8\mu s \tag{9}$$

, which is about one ms.

Since the above three steps are run in sequence, the latency of our note indication from the time a note is sound to the note is indicated on the LEDs is about 60 ms, using the yin algorithm, which is well below our initial requirement of 200 ms.

Below is a table summarizing the run-time and accuracy specifications. Accuracy E.H is the accuracy excluding harmonics as errors. Accuracy I.H is the accuracy including harmonics as errors.

Table 1: Algorithms Comparison

Specification	**Yin**	**HAS**	**NMF**
Accuracy E.H	95%	58%	43%
Accuracy I.H	80%	45%	33%
Run-time (ms)	31	32	817

3.3 Various Instrument Inputs

The algorithms selected for monophonic and polyphonic pitch detection are robust enough to handle various instrument inputs. These two algorithms are the Yin algorithm and the harmonic amplitude summation algorithm. Non-negative matrix factorization requires training data from user's instrument and therefore is not going to be implemented.

As mentioned in the algorithms section, the yin algorithm looks for the fundamental period of a given musical signal. Pitches being played from any instrument or source will have a period related to it and therefore Yin can determine the fundamental frequency.

During spectral whitening of harmonic amplitude summation, the instrument timbre is suppressed by applying triangular filters at certain frequencies to suppress frequency variations between different instruments. This allows the algorithm to determine fundamental frequencies of all different source instruments.

4 Cost And Schedule

4.1 Schedule

Time	Tasks	Ted	Chris	Jiajun
02/22	Finish the design document	x	x	x
03/02	Board Layout		x	
	Develop pitch detection algorithm on	x		x
	instrument recordings			
	Order all parts	x		
03/09	Test pitch detection algorithm	x		x
	Verify power supply components		x	
	Verify ADC, FFT, and LCD on board and LEDs	x	x	
03/16	Finalize/order PCB Design Rev1		x	
	Verify mic and amplifier			x
	Make algorithm process in real time	x		x
	Stream battery status and current instrument on LCD		x	
03/23	SPRING BREAK			
	Solder components on board	x	x	x
	Verify PCB & Debug	x	x	x
	Port algorithm to C and reduce run-time	x		x
03/30	Order final PCB		x	
	Test algorithm on hardware	x	x	x
04/06	Debug and finish remaining items	x	x	x
04/13	Finish Technical Work	x	x	x
04/28	Finish Presentation	x	x	x
05/2	Finish Paper	x	x	x

Table 2: Time Table and Task Distribution

4.2 Parts

Table 3: Parts Costs

Part	**Vendor**	**Unit Cost($)**	**Quantity**	**Cost ($)**
WS2812B	Digi-Key	4.5	5	22.5
EFM32PG	Digi-Key	3.3	1	3.3
40MHz Oscillator	Digi-Key	0.68	1	0.68
32.7 kHz Oscillator	Digi-Key	0.92	1	0.92
RG LED	Digi-Key	0.36	2	0.72
TCA9535PWR	Digi-Key	1.2	1	1.2

Continued on next page

Table 3 – continued from previous page

Part	Vendor	Unit Cost($)	Quantity	Cost ($)
Button Decoder	Digi-Key	1.2	1	1.2
LM3940	Digi-Key	1.65	2	3.3
BQ24103	Digi-Key	4.8	1	4.8
10 uF Caps	Digi-Key	0.15	7	1.05
1 uF Caps	Digi-Key	0.08	3	0.24
0.1 uF Caps	Digi-Key	0.04	4	0.16
0.47 uF Caps	Digi-Key	0.04	4	0.16
2.22 uF Caps	Digi-Key	0.13	1	0.13
2.22 uF Caps	Digi-Key	0.13	1	0.13
4.7uH Inductor	Digi-Key	0.3	1	0.3
Ferrite Bead Inductor	Digi-Key	0.1	1	0.1
150k Resistor	Digi-Key	0.1	1	0.1
100k Resistor	Digi-Key	0.1	1	0.1
2.2k Resistor	Digi-Key	0.1	1	0.1
MAX9814 Amp	Digi-Key	1.51	1	1.51
CMA-4544PF-W	Digi-Key	0.82	1	0.82
i2c Display	Amazon	7.99	1	7.99
Total				51.52

4.3 Labor

3 persons x 15 hrs/person/week x $50/hr x 13 weeks = $29,250

5 Conclusion

5.1 Accomplishments

By the end of the semester, we developed a portable, rechargeable, modular and completely functional system for instrument learning on piano. It's able to detect notes from various input sources, including human voice, guitar, and piano. The accuracy of detected notes for monophonic sounds is 80 up to 95 percent. The system also works in real time with 60 ms latency. Additional features is a joystick button which enables recording and replaying features and status LEDs for each.

5.2 Uncertainties

The current system does not have source separation implemented into the algorithm. Therefore, if the destination instrument is played, the system will also detect that note. When there is background noise, the accuracy decreases.

As it was explained in the section 3.1, the current implementation misses the input data when it's analyzing pitches. Even though in practice we don't notice obvious problems, it is still an area to improve on.

5.3 Ethical considerations

5.3.1 Ethics

To follow the #7 of IEEE code of ethics, "to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others"[6]. We will be designing our own hardware module and software algorithms. We offered constructive advice to our team members and consider the advice given to us by other students, professors, and TA's to improve the project.

This project does not illegally use other people's musical work, so there isn't any copyright infringement.The only way to misuse this product is through mishandling the lithium ion batteries and is discussed below.

5.3.2 Safety

Li-ion batteries can explode when they are overcharged or overheated[7]. To prevent that, we ensured battery is correctly charged and properly separated from the circuit board. In use, the battery charging IC handles the voltages when charging, and the user should always be present when handing a connected battery. Users should ensure no water gets near the battery, and make sure the battery isn't short circuited or handled near a fire.

5.4 Future work

Algorithm research will be a large focus on future work. Polyphonic pitch detection is still a difficult, unsolved problem so more research will be done on polyphonic algorithms. The monophonic algorithm can also use improvements to avoid detecting harmonics. Another algorithm that can be implemented is source separation to solely focus on the input instrument and not perform analysis on the target instrument's audio.

For hardware development, the first task is to add additional memory to the micro controller in order have enough memory for the chord detection algorithm. We will also look for an affordable multi-core processor to reduce system latency and reduce missed data. Additionally, we'd like to develop LED holders for different instruments such as guitar. On instruments where it is difficult to visualize the positions such as trumpets, we could introduce haptics to the buttons.

Appendix A Requirement and Verification Table

Table 4: System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. Li-Ion Charging IC (a) Battery charger should follow CC/CV charging	(a) Deplete battery and attach a power source to the IC and battery pack (b) Record battery charging current and voltage until fully charged (c) Verify circuit follows CC/CV charging scheme	N
2. Li-Ion Battery (a) Battery pack should be $8.4 \pm 0.1V$ fully charged, and $5.8 \pm 0.2V$ fully discharged	2. Verification (a) Measure battery voltages at full and empty, and verify it falls within battery constraints	Y
3. DC/DC Conversion (a) 5V rails should be $5 \pm 0.1V$ (b) 3.3V rails should be $3.3 \pm 0.05V$	3. Verification (a) Measure open voltage and verify correct voltages (b) Simulate resistive load, and verify output voltages	Y
4. ADC (a) Discretize analog signal from 0-5 V to 2^16 values	4. Verification (a) Connect pin PC10 of the microcontroller to a DC power supply (b) Monitor output readings from ADC by supplying 0.0V from the DC supply (c) Repeat previous step for 1.0 V, 2.0V, 3.0V, and 4.0V (d) Values for each voltage value should be separated 13000 - 14000 digital units apart	Y

Continued on next page

Table 4 – continued from previous page

Requirement	Verification	Verification status (Y or N)
5. Microprocessor (a) Perform FFT on digital data from ADC (b) Determine single fundamental in monophonic digital data (c) Determine fundamentals of polyphonic sound at 65 Hz, 164 Hz and 196 Hz	5. Verification (a) Connect a waveform generators output to microcontrollers digital input (b) Generate 50 Hz sine wave (c) FFT results should indicate peaks at n 50Hz where $n = 1, 2, 3...$ (d) Play an A4 on the guitar and the algorithm should find the fundamental at 440 Hz 10 Hz (e) Repeat for different notes B4, C4, D4.. (f) Play an open C chord on the guitar (g) Algorithm should find fundamentals at$65 \pm 10Hz, 164 \pm 10Hz$, and $196 \pm 10Hz$) (h) Repeat for different chords and vocal recordings	Y
6. LEDs (a) Microcontroller programmable	6. Verification (a) Supply 5.0V to the LEDs Vcc with 0.1 microFarad bypass capacitor (b) Program microcontroller to address the first LED to dataIn of the LED chain (c) First LED should turn on (d) Repeat for other LEDs on the chain	Y
7. MEMS Mic (a) generate correct values for the input signal	7. Verification (a) Supply 3.3 VDC to + Vs (b) Generate a -10.0 dB sound (c) Monitor the output voltage modulation (d) Repeat for -15.0, -20.0 dB	Y
8. Amplifier (a) Amplifies the output of the MEMES Mic	8. Verification (a) Attach the amplifier to the MEMS Mic, and monitor the output voltage of the amplifier	Y

All hardware modules besides the Li-ion charging IC are successfully implemented into the design and passed verification. The charging IC burned out during the last week of testing due to incorrectly connecting the battery pack under reverse polarity. The voltage regulation modules survived the blunder, however the charging IC failed to recover.

The polyphonic algorithm was verified but only up to around 50 percent using the harmonic amplitude summation algorithm. More information is found in the algorithms section under design.

References

[1] Alain de Cheveigne. *YIN, a fundamental frequency estimator for speech and music*. Wakayama University 2002. Available: `http://audition.ens.fr/adc/pdf/2002_JASA_YIN.pdf`

[2] Anssi Klapuri. *Signal Processing Methods for the Automatic Transcription of Music*. Tampere University of Technology, Tampere, Finland, 2004. Available: `https://pdfs.semanticscholar.org/1970/5f4c41c2ee8411aa7b940dbed c5b864070a6.pdf`

[3] Anssi Klapuri. *Multiple Fundamental Frequency Estimation by Summing Harmonic Amplitudes*. Tampere University of Technology, Tampere, Finland, 2006. Available: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.78.8558rep=rep1type=pdf`

[4] Bryan Suits. *Physics of Music - Notes*. Michigan Technology University, Houghton, Michigan. Available: `https://pages.mtu.edu/ suits/NoteFreqCalcs.html`

[5] Arnaud Dessein, Arshia Cont, Guillaume Lemaitr *REAL-TIME POLYPHONIC MUSIC TRANSCRIPTION WITH NON-NEGATIVE MATRIX FACTORIZATION AND BETA-DIVERGENCE*.Paris, France

[6] IEEE.org, *IEEE Code of Ethics*. Available: `https://www.ieee.org/about/corporate/governance/p7-8.html`

[7] Battery University. *Safety Concerns with Li-ion* . Available: `http://batteryuniversity.com/learn/article/safety_c`

[8] John Hartquist. *REAL-TIME MUSICAL ANALYSIS OF POLYPHONIC GUITAR AUDIO* .California Polytechnic State University

[9] Titze, I.R. (1994). Principles of Voice Production, Prentice Hall (currently published by NCVS.org) (pp. 188)