Vinyl Player 2.0

Group 15

Martin Do (CPE)
Jose Medina (CPE)
Micaiah Reid (EE)
Daniel Weinberg (EE)

Mentor

Chung Yong Chan

Table of Contents

Figure List

Table List

Equation List

1 Executive Summary

Vinyl records, once a nearly obsolete product, have been rising in popularity. Though vinyl records and record players have many drawbacks when compared to modern digital music consumption methods, including inconvenience of use, difficulty of distribution, and increased production and manufacturing time, there are also many advantages to the record players. One major advantage of the record player is that the record player's system for generating sound is completely analog; the sound is never sampled and digitized, giving it a sound quality that is truer to the original recordings. The main goal of the Vinyl Player 2.0 is to remove as much of the first disadvantage – inconvenience of use – as possible in using a record player.

The main inconveniences in using a record player are tedious setup, difficulty in song selection, and the need to physically go to the record player to change songs, flip the record, lower the volume, or turn off the record player. Though the tediousness of setup will not be addressed by the Vinyl Player 2.0 (as in, the user will still need to place the record on the platter, etc.), the difficulty in playing songs and the need to be *at* the record player to control it will be solved with the Vinyl Player 2.0.

The Vinyl Player 2.0 will consist of two main systems: the record player and the mobile application that will control the record player. The record player maintains the ability for the user to use all manual device controls, such as moving the tonearm, and using the physical buttons on the record player. Additionally, the mobile application will be able to control the record player's power, rotation speed, and what specific song on the record is being played. Whenever a new song is selected, the record player arm will be lifted off of the record, moved to the appropriate location over the record, and then gently lowered onto the record to continue playing music. This will allow the user to have nearly full control over the record player remotely. Some new controls were also added to the record player including a 'go home' function that lifts the tonearm off of the record and moves it to the tonearm's stand and a 'anti-skip' function that lifts the tonearm and moves it over one grove to avoid skipping.

This document fully outlines the implementation of the Vinyl Player 2.0. The Project Description chapter will describe generally the motivations for this project, as well as the requirement specifications that will be met. The Project Research chapter will explain all of the research that went into this project, including models for motor, record player, and microcontroller components, software technologies that will be used to create the application, and market conditions to evaluate the need for a product like the Vinyl Player 2.0. The following chapter, Design Constraints and Standards, will discuss just that, the constraints and standards that were overcome and met when implementing this project. The Project Design chapter walks through all of the design choices that were made and how they were implemented and tested, both through hardware and software. The Project Operation chapter explains in detail how the Vinyl Player 2.0 should be used by a user, and the Administrative Content chapter outlines the cost, both in time and price, of this project. Finally, the Conclusions chapter outlines the final outcome of this project, as well as what was learned and gained from implementing it.

2 Project Description

This section will give a general overview of the Vinyl Player 2.0. The motivations, goals, and objectives section will outline why this project is a good idea in today's market and what the group would like to accomplish through creating this project. The specifications section dictates specific features that the Vinyl Player 2.0 should have and generally how that should be accomplished. The house of quality section will show how the user and engineering requirements in this project sometimes conflict and sometimes work together, and how those conflicts will be resolved in implementing this project.

2.1 Motivations, Goals & Objectives

Vinyl records have been experiencing a resurgence in recent years and sales are currently at the highest they've been since 1988. This gust of wind in a format that was considered antiquated just 10 years ago can be attributed to a variety of factors such as sound fidelity, collectability, personability, and backlash against the rapid digitization of music. Yet, even with the increasing interest in the analog medium there are still some inherent issues that have the potential to stunt the growth that the vinyl record industry is experiencing.

One fault that might not be evident but provides a glaring weakness in the ability of vinyl records scaling to a wider audience, especially younger users, is the difficulty to consume content from a record in the manner that they are accustomed to. Actions that are considered compulsory when listening to music nowadays such as skipping, restarting, or jumping to your favorite part of a track require the user to interface with the record player itself, ultimately tethering a user to the device to perform these actions. Imagine having to go up to the tv every time you wanted to change what you were watching or wanted to add subtitles on Netflix.

The inconvenience to the listening experience is further compacted with the complications a typical user might have when trying to cue the record to the part they want. First simply identifying what song is where can prove to be a challenge as all a user sees is the many grooves that encompass the record. Secondly, the arm/needle of the turntable requires delicate handling to assure the record is neither cued to the wrong portion of the record or damaged due to excessive force, which is a task that can escape many listeners.

All of the above reasons served as incentive to design a system that will rid a listener of the inconveniences of playback, while using the analog format, and simplify the vinyl experience as a whole. This system will provide the best of the analog and digital music consumption experiences by not obstructing the current functionality of the turntable, thus allowing users to cue the record themselves, but adding the option to interface with the record player in a modern way to make its usage more accessible. The ultimate goal of simplifying vinyl

record usage to open the industry will be achieved through a few subsystems that each control a critical component.

The first is the subsystem that attaches to the arm of the record player and will move the arm both horizontally and vertically to position the arm/needle to the location on the record that is desired. This attachment to an existing record player should be small enough as not to obstruct the style of the player and not add an excessive amount of force when cueing the record. There will be a controller that wirelessly communicates with the mobile application and formulates how much to move the arm.

The second subsystem will perform image recognition on the record that is currently on the platter of the record player to reliably pull information on that record. This second subsystem will function within the mobile application that accompanies the system and will utilize the user's camera to scan either the record itself, which contains identifying information on its label, or the cover image of the record. Obtaining information about the record such as name of artist, name of album, etc. A pre-existing record database will be queried and more information about the record will be pulled such as song lengths, year of release, etc. The third subsystem, which is also harboured within the mobile application, will take the information that is pulled and present in an intuitive music player fashion that a user can interface with for playback.

Once a particular song/time is selected then the mobile application will wirelessly communicate with the controller of the system attached to the turntable arm, which will utilize factors such as rotations per minute (rpm), record diameter, song lengths, etc. to move and cue itself close to the song/time that the user desires. Close in this situation means a maximum offset of a few grooves from where the desired location is. For a record playing at 33 rpm, most common rpm rate, this offset will translate to a minimal amount of seconds from where the record was desired to start from.

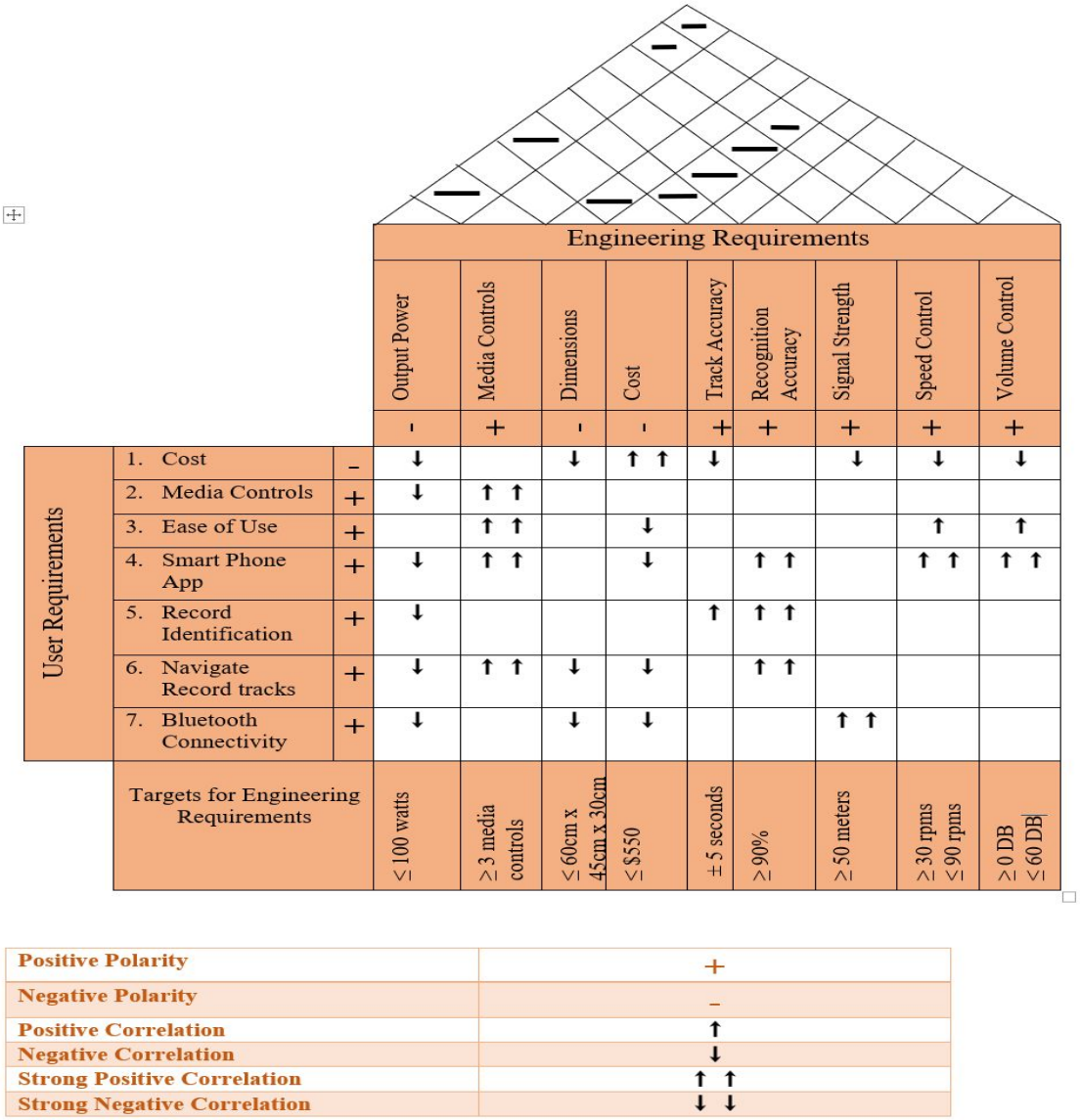
2.2 Specifications

1. Device should not significantly increase the size of the pre-modified record player.
2. Phone application
    a. Connects wirelessly to controller.
    b. Should allow users to use their phone to take a picture of a record and find the scanned record in the database.
    c. Allows the selection of a specific track found on the record from a database.
    d. Allows user to insert record data into database.
    e. Lets the user start and stop the playback of a record.
    f. Allows the user to add records from the database to their personal collection for easy access.
    g. Allows user to adjust the volume of the record player.
3. Arm Movement

    a. A powered stepper motor will adjust the radial location of the record player's arm. A motor will raise and lower the contact pin of the record player.
    b. The stepper motor should be able to place the needle within ±5 seconds of the desired time (maximum error of about 2 grooves of the record in either direction).
4. Record database
    a. Stores record specific details.
        i. Which tracks are on the record.
        ii. Times at which the tracks start and stop.
        iii. The corresponding grooves on the physical groove on the record. (this information is used by the servo to place the pin)
        iv. Information to allow scanner to identify record.
    b. Allows user to upload record data to database.
5. Controller
    a. Connects wirelessly to phone application.
    b. Connects to power outlet and acts as a power supply to all other components.

2.3 House of Quality

The house of quality gives us an overview of the products user capabilities and engineering requirements. It provides a graphical way to identify the positive and negative dependencies between the capabilities and requirements. The house of quality is an important tool because it lists the engineering requirement constraints and allows us to identify problematic areas that hinder us from achieving our goals. We can plan for the inevitable and identify creative solutions.



			Output Power	Media Controls	Dimensions	Cost	Track Accuracy	Recognition Accuracy	Signal Strength	Speed Control	Volume Control
			Engineering Requirements								
			-	+	-	-	+	+	+	+	+
User Requirements	1. Cost	-	↓		↓	↑ ↑	↓		↓	↓	↓
	2. Media Controls	+	↓	↑ ↑							
	3. Ease of Use	+		↑ ↑		↓				↑	↑
	4. Smart Phone App	+	↓	↑ ↑		↓		↑ ↑		↑ ↑	↑ ↑
	5. Record Identification	+	↓				↑	↑ ↑			
	6. Navigate Record tracks	+	↓	↑ ↑	↓	↓		↑ ↑			
	7. Bluetooth Connectivity	+	↓		↓	↓			↑ ↑		
	Targets for Engineering Requirements		≤ 100 watts	≥ 3 media controls	≤ 60cm x 45cm x 30cm	≤ $550	± 5 seconds	≥ 90%	≥ 50 meters	≥ 30 rpms ≤ 90 rpms	≥ 0 DB ≤ 60 DB

Positive Polarity	+
Negative Polarity	-
Positive Correlation	↑
Negative Correlation	↓
Strong Positive Correlation	↑ ↑
Strong Negative Correlation	↓ ↓

Figure 1. House of Quality

2.4 Block Diagram

Figure # shows the high-level block diagram. This diagram shows how each module connects to one another. The division of labor relating to each of the specified blocks is also shown in this diagram. As this is the high-level diagram, this only shows the the components in an abstract sense. Additional information regarding each of the block and specifics of the interactions will be shown later in the report.

From the diagram, the division of labor can be seen. The software components will be handled by the computer engineers of the group, Jose and Martin. Since these tasks are integrated so tightly, responsibility between these blocks will be spread between the two of them evenly. For the hardware side, the

electrical engineers will be responsible. The power supply and record player controls will chiefly be the responsibility of Micaiah. Meanwhile, the stepper motor controls will primarily be the responsibility of Daniel. The microcontroller block tasks will be evenly distributed between Daniel and Micaiah.

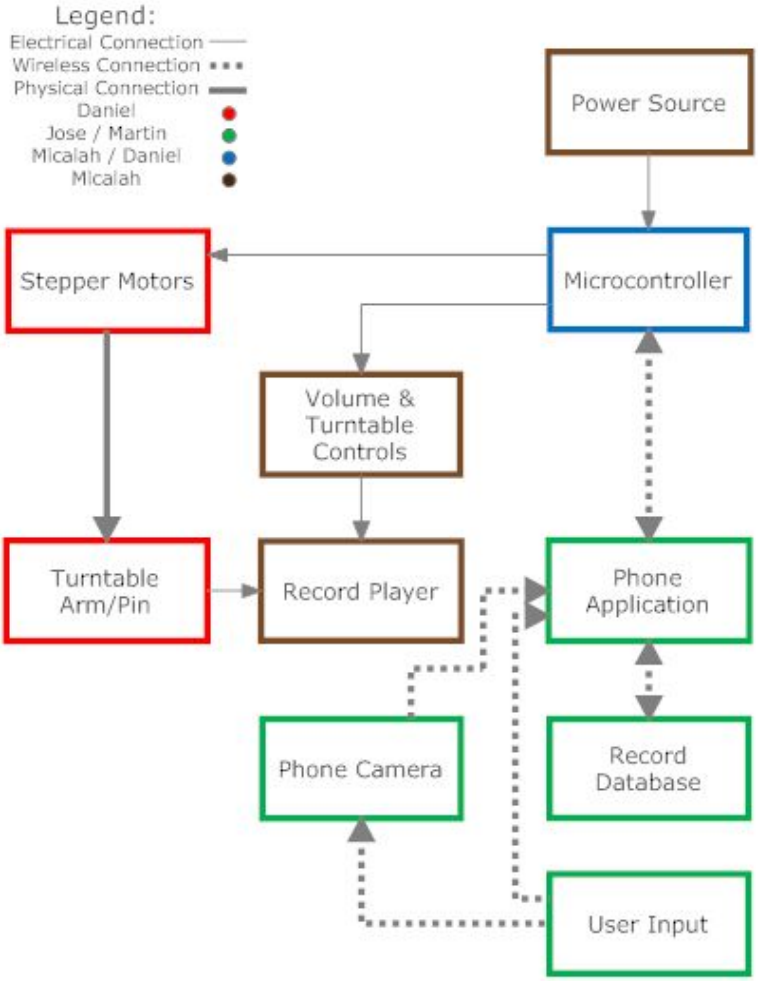


Fig 2. Block Diagram

3 Project Research

Before starting work on Vinyl Player 2.0, a variety of technologies had to be researched, both for the hardware and software portions of the project. The technologies researched include record players, microcontrollers, high-accuracy motors, application frameworks, application hosting, computer vision, and open-source music application programming interfaces (API).

3.1 Record Player Components

Vinyl record players needed to be researched from two different perspectives: the individual components of the record player that allow it to function and what make/model of record player would best suit our needs for a record player that could be disassembled and repurposed. First, the individual components of a record player – the turntable, stylus, cartridge, tonearm, and the amplifier and preamplifier – will be expounded.

3.1.1 Turntable

The turntable of a record player is the round component that actually moves the record. The turntable is commonly referred to as the “platter”. The center of the turntable has a small, metal rod that the center hole of the record fits into. The main portion of the turntable is made from some metal covered with rubber to prevent from damaging the record. A heavier metal is preferred for the turntable, as this provides higher momentum and stability which can reduce vibrations and thus a better sound. The turntable is ideally at least 90% of the size of a record because this provides more support for the record. There are two main configurations of motors to power the turning of the turntable – belt drive and direct drive.

3.1.1.1 Belt Drive

In a belt drive, there is a belt with a bearing attached on one end (on which the platter of the turntable rests) and a motor attached on the other end. For this configuration, the motor does not directly turn the platter, but rather turns the belt, which turns the bearing, and in turn rotates the platter. This separation from motor to platter helps absorb vibrations created from the rotating motor, leading to reduced noise and an improvement in the sound quality. This improvement in sound quality does, however, come with a price. Belt drives tend to have less torque and less control of the speed of rotation.

3.1.1.2 Direct Drive

The direct drive configuration consists of a motor directly attached to the platter. This direct motor-to-platter connection allows for highly consistent rotation speeds, and allows for finer control of the speed. It also decreases the effect of outside forces (the stylus/tonearm or user input) on the rotation speed.

For the purposes of the Vinyl Player 2.0, a direct drive configuration is ideal, as it will allow for control of the speed and direction of the rotation, which can be controlled from the mobile application.

3.1.2 Stylus

The stylus is the needle the rests on top of the record, and is made up of a needle that picks up the vibrations in the record and a flexible metal strip that connects the needle to the cartridge and tonearm. The needle is ideally a very hard piece of metal in a conical shape. A very hard metal is used because, though the arm and needle don't apply a very large force onto the record, the small size of the needle causes a very large pressure (26 tons per square inch) to be exerted on the needle and record. The flexible piece of metal that attaches to the needle allows the needle to smoothly rise and fall with the divots on the record.

In the context of the Vinyl Player 2.0, tone and sound quality are not of extreme importance. While different types, shapes, and materials of the stylus can have an extreme effect on the tone of the record player, those details are not relevant for this application.

3.1.3 Cartridge

The cartridge connects to the stylus, and converts the vibrations picked up by the stylus into an electrical signal. The cartridge contains small magnets that generate a magnetic field. The stylus, which is connected to a magnet, interrupts the magnetic field created in the cartridge, which leads to a current in the leads connected to the cartridge. Similarly to the stylus, the cartridge can have a substantial impact on the quality of sound generated by the record player, but a high quality cartridge is not necessary for the purposes of the Vinyl Player 2.0.

3.1.4 Tonearm

The tonearm connects the stylus and cartridge to the base of the record player. It can rotate smoothly about its point of contact with the base, and often has a counterweight to adjust the force of the stylus on the record. The tonearm can be curved or straight, and contains all of the wiring that connects the cartridge to the preamp and amplifier.

In the Vinyl Player 2.0, the main functionality provided is precise control over the placement of the stylus on the record via the tonearm.

3.1.5 Preamplifier & Amplifier

While many record players do not contain built-in speakers, many do contain a preamp to boost the signal generated by the stylus and cartridge. The quality of these preamps greatly affect the sound quality of the record player. For the purposes of the Vinyl Player 2.0, preamp quality is not important, but the presence of a preamp may be necessary in order to remotely control the volume of the output of the record player.

3.2 Record Player Models

For the purposes of the Vinyl Player 2.0, a record player will be purchased and modified to add features. It will be crucial to find a record player with the appropriate specifications to be modified for use in this project. Below, a few record players will be compared and qualified based off of turntable motor type, platter material, configurability, and cost.

3.2.1 Audio-Technica AT-LP60

The Audio-Technica AT-LP60 is an affordable, belt drive turntable with an aluminum platter. The motor is DC servo-controlled and can operate at 33-⅓ RPM and 45 RPM. It has two built-in preamps and a diamond stylus. This model has a straight tonearm and costs $99.00.

3.2.2 Audio-Technica AT-LP3BK

The Audio-Technica AT-LP3BK, though slightly more expensive than the AT-LP60, adds a few useful features. It is belt driven with an aluminum platter. Its motor is DC servo-controller and can operate at 33-⅓ RPM and 45 RPM. Like the AT-LP60, it has two built-in preamps and a diamond stylus. The main feature that is added to this model is the hydraulically damped lift control. This allows the arm to be raised and lowered gently on pressing the Start and Stop button. Reusing this capability could help simplify the build process of the Vinyl Player 2.0. This model has a straight tonearm and costs $249.00.

3.2.3 Audio-Technica AT-LP120BK-USB

The Audio-Technica AT-LP120BK-USB has a direct-drive turntable with an aluminum platter. The motor is a DC motor and can operate at 33-⅓ RPM, 45 RPM and 78 RPM. It uses an electronic braking system, has two built-in preamps, and a diamond stylus. Like the AT-LP3BK, this record player has a hydraulically damped tonearm lift. This model also has the capability to connect to a computer through USB to create digital versions of the record (which is an unnecessary feature for the Vinyl Record 2.0). Additionally, the

AT-LP120BK-USB can spin the turntable in either direction, for forward and reverse play. This model has a curved tonearm and costs $299.00.

3.2.4 Numark TT250USB

The Numark TT250USB has a direct-drive turntable with an aluminum platter. The motor is a quartz-controlled direct drive motor. Like the AT-LP120BK-USB, this model has a damped tonearm lift and can connect to a computer via USB to convert the records to a digital format. This model has a curved tonearm and costs $189.95.

3.2.5 Stanton T.62 M2

The Stanton T.62 M2 has a direct-drive turntable. The motor can be run at two speeds, 33-⅓ RPM and 45 RPM. This model has a straight tonearm and costs $159.00.

3.2.6 Record Player Model Final Comparison

Though the AT-LP60 has a great, low price, it does not have direct drive, and must, therefore, be ruled out. The AT-LP3BK can also be ruled out, as it doesn't have direct drive. The AT-LP120BK-USB was not chosen because of its high price. The TT250USB was also ruled out because of its unnecessary USB technology and higher price. In the end, the T.62 M2 was purchased. Below is a table comparing the different models described above.

	AT-LP60	AT-LP3BK	AT-LP120BK-USB	TT250USB	T.62 M2
Cost	$99	$249	$299	$190	$160
Direct Drive	No	No	Yes	Yes	Yes
Preamp	Yes	Yes	Yes		
Tonarm	Straight	Straight	Curved	Curved	Straight
Configurability	High	Unknown	Unknown	Unknown	High

Table 1 - Record Player Model Final Comparison

3.3 Microcontrollers

The microcontroller is a crucial component of the Vinyl Player 2.0. The microcontroller will be used to communicate with the user's phone application and to control the location of the tonearm, the speed and direction of rotation of the turntable, the output volume, and the master on/off switch of the record

player. Because the microcontroller will be located inside of an already manufactured record player, it will be necessary to choose a microcontroller that is as small as possible in order to ensure that it can fit inside the record player. Additionally, the microcontroller will be required to have bluetooth, a good sdk, and low cost. Some microcontroller models will be compared below on those features.

3.3.1 MSP430F4618

The MSP430F4618 is an ultralow-power microcontroller made by Texas Instruments. It uses only 400 uA/MHz in an active mode, 1.3 uA in standby mode, and .22 uA in Off Mode. The wakeup time is an impressive 6 microseconds. It can be run at frequencies as high as 8 MHz, and it has 116 KB of nonvolatile memory and 8 KB of RAM. This microcontroller has 80 GPIO Pins, which is more than what is necessary to control all of the motors, the power, and the amplifiers.

Additional features available with the MSP430F4618 are an LCD system, real-time clock, watchdog, temperature sensor and a brown out reset. The package size is a 16mm x 16mm LQFP package. Another advantage of the MSP430 is that everyone involved in the project has experience programming these microcontrollers. A disadvantage is that it doesn't have built-in bluetooth capabilities. One MSP430F468 would cost $13.14.

3.3.2 CC2540F128RHAT

The CC2540F128RHAT is a low-power microcontroller made by Texas Instruments. With 5 power modes, this microcontroller can run anywhere from 19.6mA in Active mode to .4uA in Power Mode 3 (which allows external interrupts). The wake-up time is about 3 microseconds. It has 8 KB of RAM and 21 GPIO pins, which should be an appropriate number of pins to power all of the motors, power systems, and amplifiers for the Vinyl Record 2.0.

A very important feature of this microcontroller is that it has built-in Bluetooth capabilities, with a Bluetooth v4.0 Compliant Protocol Stack. The maximum data rate is 1000 kbps, the security used is 128-bit AES. The package size is a 6mm x 6mm QFN40 package. Though no one in the group has experience with this microcontroller, there is ample documentation for it. One CC2540F128RHAT would cost $3.92.

3.3.3 Nordic nRF52832

The Nordic nRF52832 is a powerful, ultralow-power system-on-a-chip made by Nordic Semiconductors. This microcontroller has 3 power modes: OFF mode, which uses .3uA at 3V, OFF mode with full RAM retention, which uses .7uA at 3V, and ON mode, which uses 1.9uA at 3V. It has two memory options - 512kB flash with 64kB RAM or 256kB flash with 32kB RAM. There are 32 GPIO

pins, which should be more than enough to power all of the motors, power systems, and amplifiers for this project.

This microcontroller has built-in bluetooth capabilities, complying to Bluetooth 5 standards. There are two options of data rates: 1Mbps and 2Mbps. For the purposes of this project, 1Mbps would be more than enough, as the only information needed to be sent through bluetooth is simple instructions and times. There are also two package size options: 6mm x 6mm QFN48 or a 3mm x 3.2mm WLCSP package. One Nordic nRF52832 would cost about $5.73.

3.3.4 ESP8266EX

The ESP8266EX is highly integrated system-on-a-chip made by Espressif Systems. This microcontroller has 3 power modes: deep-sleep mode, running at 20uA, sleep mode, and on mode. The CPU clock speed runs at 80 MHz and can reach a maximum value of 160 MHz. It has 96KB of RAM and 1MB of flash memory. It has 16 GPIO pins. This should be enough pins to control all of the motors, power systems, and amplifiers for this project, though it might be cutting it close.

This chip has Wi-Fi and bluetooth capabilities. As Wi-Fi is not necessary for the Vinyl Player 2.0, this might make the ESP8266EX overkill. The package size is a 5mm x 5mm QFN32. One ESP8266EX costs $2.88.

3.3.5 Arduino Uno Rev3

The Arduino Uno Rev3 is a full microcontroller board made by Arduino. This device differs from the others discussed because it has more than just the chip; it also includes the full board, making powering and connecting the device to I/O much simpler. The microcontroller used on the board is the ATmega328P. It has 32 KB of flash memory and 2KB of SRAM. The clock runs at 16 MHz, and it has 14 digital I/O pins. This may not be a sufficient number of pins to control all of the motors, power supplies, and amplifiers for this project.

This device does not have built-in bluetooth capabilities, but could be paired with a bluetooth transceiver. Since it is a full board, it is larger than the other microcontrollers discussed, with dimensions of 68.6mm x 53.4 mm. One Uno Rev3 costs $22.

3.3.6 Redbear Labs Blend 2

The Redbear Blend 2 is an integrated development board that uses the Nordic nRF52832 (discussed above). The advantage of the Blend 2 is that many of the peripherals that would be necessary in setting up the nRF52832, such as antennas for bluetooth, are already implemented. Also, the software development kit that comes with the Blend 2 is very easy to use. There are open source Arduino libraries available for the chip used. The board has an input voltage ranging from 6.6V to 12V and has dimensions of 69mm x 53 mm. It has 26 Digital I/O pins, 8 PWMs and 6 ADCs. One Blend 2 costs $29.90.

3.3.7 Redbear Labs BLE Nano v2 and MK20 USB Board

The BLE Nano v2 has many of the same advantages of the Blend 2, described above, without all of the extra features that are not necessary for this project. Also, since the BLE Nano is not a full development board like the Blend 2, it has lower voltage requirements, operating from 1.8 to 3.6 V. The MK20 USB Board can be easily attached to the BLE Nano in order to upload code to the chip. This is a major advantage over some of the other microcontrollers.

3.3.7 Microcontroller Final Comparison

As stated above, the ideal microcontroller for the Vinyl Player 2.0 would be a low-power device with 20-30 available pins, at least 32KB of RAM and 256KB of flash memory, bluetooth capabilities, small in size and affordable in price. The following table is a breakdown of those qualities of the preceding microcontrollers.

	V_{in}	Current	Blue tooth	Pins	RAM (KB)	Size (mm)	Price
MSP430F4618	1.8 – 3.6	400 uA	✘	80	8	16 x 16	$13.14
CC2540F128RHAT	3	19.6 mA	✓	21	8	6 x 6	$3.92
Nordic nRF52832	1.7 – 3.6	1.9 uA	✓	32	32-64	3 x 3.2	$5.73
ESP8266EX	2.5 – 3.6	170 mA	✓	16	96	5 x 5	$2.88
Arduino Uno Rev3	7 – 12	50 mA	✘	14	2	69 x 53	$22
Redbear Blend 2	6.6 – 12	1.9 uA	✓	26	64	69 x 53	$29.90
Redbear BLE v2	1.8 – 3.6	1.9 uA	✓	26	64	69 x 53	$16

Table 2. Microcontroller Final Comparison

Besides the specifications listed above, another factor that was taken into consideration in choosing a microcontroller is ease of use and setup. Because of its ease of use, ample memory, pin count, and its bluetooth capabilities, and lack

of extraneous peripherals, the Redbear BLE v2 was chosen as the microcontroller.

3.4 Bluetooth Transceivers

Though an ideal microcontroller for this project would include built-in bluetooth capabilities, as some of the viable microcontrollers considered did not include bluetooth capabilities, some bluetooth transceivers were also researched.

3.4.1 CC2560

The CC2560 is a complete Bluetooth transceiver system that is designed for ease of use and to reduce time in implementation made by Texas Instruments. This device has 5 power modes, ranging from 1 uA in shutdown mode, 40 uA in deep sleep mode, to 112.5 mA in Continuous Transmission. This device could be coupled with a microcontroller that doesn't have built-in bluetooth capabilities. It complies with Bluetooth 4.1 protocols. One of these devices would cost $6.29.

3.4.2 HC-05

The HC-05 is also a Bluetooth transceiver that is commonly paired with the Arduino Rev3. It runs at 3.3 V and consists of a Bluetooth serial interface and Bluetooth adapter.

3.4.3 Bluetooth Transceiver Final Comparison

Because a microcontroller with built-in Bluetooth capabilities was chosen, a Bluetooth Transceiver was not deemed necessary for this project.

3.5 Motor Research

After receiving user input from the phone application, the device must be able to control the tonearm's position. The method to accomplish this requires the use of several motors. These motors will be positioned to move the horizontal and vertical location of the tonearm to allow the user to select the correct track on the album. In this section, different types of motors will be mentioned and discussion will take place to decide the motor most suitable for this application. In addition to motor selection, the control methods of these motors will also be discussed and compared to find the solution.

3.5.1 DC Motors

DC motors continuously spin as long as it is connected to a power source. The speed of these motors are controlled by cycling on and off the power supply.

These motors are capable of running at high RPM and are typically inexpensive. The high RPM and lack of real position control makes these types of motors unsuitable for this application.

Typical brushless DC motors work by having permanent magnets on the outside and a spinning armature on the inside. The magnets remain stationary while the armature rotates. The armature creates a magnetic field when current flows through it. This causes the armature to repel and attract the stationary magnets (stator). This causes the armature (rotor) to spin. To keep the motion of the motor spinning, the poles of the armature must be changed. The brushes handle the change in polarity. The brushes make contact with two spinning electrodes attached to the armature to flip the polarity of the electromagnet as it spins. Brushless remove the need for brushes by "turning the motor inside out". This refers to putting the permanent magnet on the rotor and moving the electromagnet into the stator. This gives two main advantages. First, the brushes eventually wear out, lowering the lifespan of the motor. The second, the brushes limit the maximum speed of the motor. Limiting the maximum speed of the motor is not a big deal for this application, but limiting the motor's lifespan can be severely detrimental. To compensate for the brushless version of a DC motor not having these side effects, brushless DC motors have a higher initial cost.

3.5.2 Servo Motors

The position of servo motors are controlled precisely unlike the position of a DC. Also unlike DC motors, servo motors do not rotate freely. Instead, servo motors rotate back and forth 180 degrees. Servo motors are very quick and have a high holding torque, both of which are not necessarily needed for this application.

Servo motors work the same as how a DC motors operate. This is because most servo motors contain a DC motor. To differentiate themselves from DC motors, servo motors contain a control circuit and also a potentiometer. The motor is attached by gears to the control wheel. As the motor rotates, the potentiometer resistance changes, so the control circuit can precisely regulate how much movement there is and in which direction.

3.5.3 Stepper Motors

Stepper motors are controlled through energizing inductive coils to move toothed electromagnets arranged around a central gear to a specific position. setting up a stepper motor is more complicated than implementing a DC or servo motor because stepper motors require energizing coils in specific sequence. Due to the design of the internal electromagnetic gear, stepper motors move in specific angle increments referred to as step angle. The minimum deviation (without microstepping), or step angle, varies depending on the design of the motor. Due to the precise angles at which stepper motors rotate to, they are perfect for this application. Stepper motors are either unipolar or bipolar. A

unipolar stepper motors direction is controlled by a central tap which controls the direction of the magnetic field. This lets the user control the direction of rotation of the stepper motor without having to change the current flowing through the inductive coils. Contrary to bipolar stepper motors which controls the magnetic field by controlling the direction of the current going through the coils. Bipolar are more popular due to higher supplied torque however are typically harder to control. Stepper motors are the slowest of the three mentioned motors, but for this application speed is not important. For this application, stepper motors are the most suitable for the task and will be the focus of the research portion of this project.

3.5.4 Motor Requirements

In order to lift and move the arm of the record player for the chosen design, three stepper motors are required. The first stepper motor will be used to swivel an assembly controlling two of the tonearm's horizontal movement. This motor must be able to provide enough torque to move the entire assembly and potentially both of the pitch motors. This motor must display exceptional precision in order to move the needle within 1 groove of the desired groove. This mean the motor must have precision of at least 0.2 degrees. This means that the step angle of the chosen stepper motor must have a minimum step angle of 0.2 degrees. In addition to these qualities, the horizontal motor should be as inexpensive as possible to meet the designated price budget. Some stepper motors might meet the torque requirement but do not meet the precision requirement. These motors still can be used if the motion of the motor is either geared down sufficiently or able to be microstepped to the desired location.

The second type of motor used in the chosen design is used to move the tonearms up and down. For each of the tonearms, a separate motor must be used to move each of the two tonearms up and down. As specified in the design overview for Design A, the arms lie on an armrest which is moved by the motors. The tonearms are not attached to the armrest, but rather they are held in place by the counterweight balancing the tonearms. As the motor moves the arm rest downwards vertically, the arm will eventually make contact with the record. At this point the motor can continue to spin without influencing the arm. This will allow the arm to rest on the record without applying additional pressure on the record. For this motor, precision is not an important factor as any overshoot/undershoot is either unimportant or has no impact on the final resting place of the arm. This motor only needs to provide enough torque to lift and lower the arm.

3.5.4.1 Torque Requirement

There are two different types of torques that are given in stepper motor datasheets. These torques are detent torque and holding torque. Detent torque is the torque produced by the motor when the coils are not energized. When the coils are not energized, it is desired for the motor to move as freely as possible

so the arm moves freely along the rotating turntable. For this project, the desired detent torque for the yaw motor should be as low as possible to reach the desired functionality. The opposite is true for the pitch motor. The pitch motor should have enough detent torque to hold the record player arm even when not energized. Holding torque is defined as the amount of torque produced when the coils are energized. The amount of holding torque is not massively important for this application, because if the detent torque (which is normally much smaller than holding torque) is sufficient for the project, then the holding torque should be enough.

3.5.4.2 Precision Requirement

The size of the records the record player play are 12" in diameter. Most vinyl records are played back on a turntable rotating at 33.33 revolutions per minute or .5555 revolutions per second. On average, a vinyl record plays roughly 20 minutes of audio. Using these numbers, this means each groove along the 6" radius of the record is roughly .009" apart. For Design A, the angle the arm must move for each groove changes depending on how far radially the desired groove is along the record. For example, the smallest change of angle to move a single groove would be the grooves along the outer edge of the record. For this worst case scenario, the smallest angle difference per record groove would be .086 degrees. To meet the specification of being off by at most two grooves, the yaw stepper motor must have a maximum step angle of 0.2 degrees. Most stepper motors have a step angle of much greater than 0.2 degrees, but this can be overcome by gearing down the input. In addition to additional gearing, the step angle of a stepper motor can be further decreased by "microstepping". However to perform microstepping, the stepper motor would require a stepper motor driver which will be further expanded upon later in the report.

3.5.5 Stepper Motor Selection

The selected motors must meet the previously mentioned specifications. The following sections will list the pros and cons of the selected researched motors. Following that, one or two of the motors will be selected for use in the application. One motor will be used to control the horizontal movement of the tonearm and another to control the vertical movement.

3.5.5.1 17HS16-2004S

The NEMA 17 17HS16-2004S is a bipolar stepper motor. With a step angle of 1.8 degrees, this motor does not initially meet the requirements for the horizontal motor. In order to meet the requirements, this motor must be geared down 1:10 in order to meet the .2 degree requirement. This in addition to microstepping should meet the requirement. This motor also has a high current rating of 2A allowing it to get sufficient torque to meet the rest of the

specifications. At a price of 12.99$ this motor is also the cheapest of the 4 potential stepper motors.

3.5.5.2 ROB-09238

The NEMA 17 ROB-09238 is a bipolar stepper motor. With a step angle of 1.8 degrees, the motor does not initially meet the requirements for the horizontal motor. In order to meet the requirements, this motor must be geared down 1:10 in order to meet the .2 degree requirement. This in addition to microstepping should meet the previously mentioned requirement.

3.5.5.3 PG35S-D48-HHC2

The PG35S-D48-HHC2 is a bipolar stepper motor. With a step angle of 0.212 degrees, this stepper motor meets the precision specification for the horizontal stepper motor without any down gearing. This combined with microstepping should be ample precision to exceed the needed specification. The price of this motor is 36.61$ which is quite expensive stepper motor wise.

3.5.5.4 26M048B2U-V31

The 26M048B2U-V31 is a unipolar stepper motor. . With a step angle of 0.212 degrees, this stepper motor meets the precision specification for the horizontal stepper motor without any down gearing. This combined with microstepping should be ample precision to exceed the needed specification. At a price of 40.23$, this stepper motor is the most expensive of the four researched stepper motor options. Since this motor is unipolar, it will provide significantly less torque than the other bipolar stepper motors This could be a potential issue if the tonearm assemble needs extra torque to move properly.

3.5.5.5 17HS19-1684S-PG19

The Nema 17 17HS19-1684S-PG19 stepper motor is a bipolar stepper motor with a 19:1 planetary gearbox. Normally this stepper motor would have a step angle of 1.8 degrees, however the planetary gearbox brings the step angle down to .094 degrees step angle. Even with no additional gearing and microstepping, this stepper motor/gearbox combination requires no additional modification to meet the step angle requirement. At the price of 26.88$, this stepper motor is more expensive than the other NEMA 17 motors, but it comes with the planetary gearbox making it fairly cost efficient. Using a gearbox specifically made for a specific motor is the simplest solution for stepping down the step angle of a motor without additional mechanical knowledge.

3.5.5.6 Stepper Motor Comparison

For the purpose of this project, three stepper motors are required, two to control the pitch of both arms and one to control the horizontal movement of the arms. The pitch control motors do not need to be very precise so the cheaper motors should be sufficient. Therefore, the best option out of the five researched motors for the pitch motors would be the NEMA 17 17HS16-2004S stepper motor. The horizontal movement of the arms will be controlled by one very precise servo motor. Due to the low cost of 26.88$ and the small step angle of .094 degrees. The best choice of the five researched motors for the horizontal motor is the NEMA 17 17HS19-1684S-PG19 with 19:1 planetary gearbox.

	17HS16-2004S	ROB-09238	PG35S-D48-HHC2	26M048B2U-V31	17HS19-1684S-PG19
Step Angle	1.8 degrees	1.8 degrees	0.212 degrees	0.25 degrees	0.094 degrees
Number of Leads	4	4	4	6	4
Cost	12.99$	14.95$	36.61$	40.23$	26.88$
Polarity	Bipolar	Bipolar	Bipolar	Unipolar	Bipolar
Dimensions	1.7"x1.7"	1.7"x1.7"	1.378" Dia	1.030" Dia	42 x 42mm
Current Rating	2A	330mA	250mA	110mA	1.68A

Table 3. Stepper Motor Final Comparison

3.5.6 Motor Controls

The stepper motor used in the project are controlled through four wires connected to the motor. Each pair of these two wires connect each end of two separate coils of the motor. To move the motor, both of these coils are controlled by generating a current through each of the inductive coil. To move the motor is more than one direction, this current must be able to be generated in either direction. There are two suggested ways to accomplish this either through the use of a pre-built stepper motor driver, or through a H-bridge circuit.

Since there are two separate motors with 4 connectors each, it could be considered to use 8 separate pins on the microcontroller. However, this is not necessary. Because each of the motors are not required to move simultaneously, four of the pins can be used by both of the motors with and additional pin used as an enable for each of the circuits. This can be accomplished by using two and gates and an inverted to turn off the Vcc that powers the motor that is not being controlled. However, this method can be considered unnecessary due to the chosen microcontroller exceeding the required pins for this project.

3.5.6.1 H-bridge

One method to control the stepper motors is to use multiple H-bridge circuits. H-bridge circuits allow a designated current to flow either direction through an inductive load through the use of MOSFET transistors. H-bridges use two control wires to control which direction of current flow. Since the chosen design for this project requires three separate stepper motors each with two coils each, a total of 6 H-bridge circuit would be required to control all the motors. Without using some kind of ASIC, this method could become needlessly complicated and cumbersome. However, this circuit is the foundation for many pre-built drivers that are currently on the market.

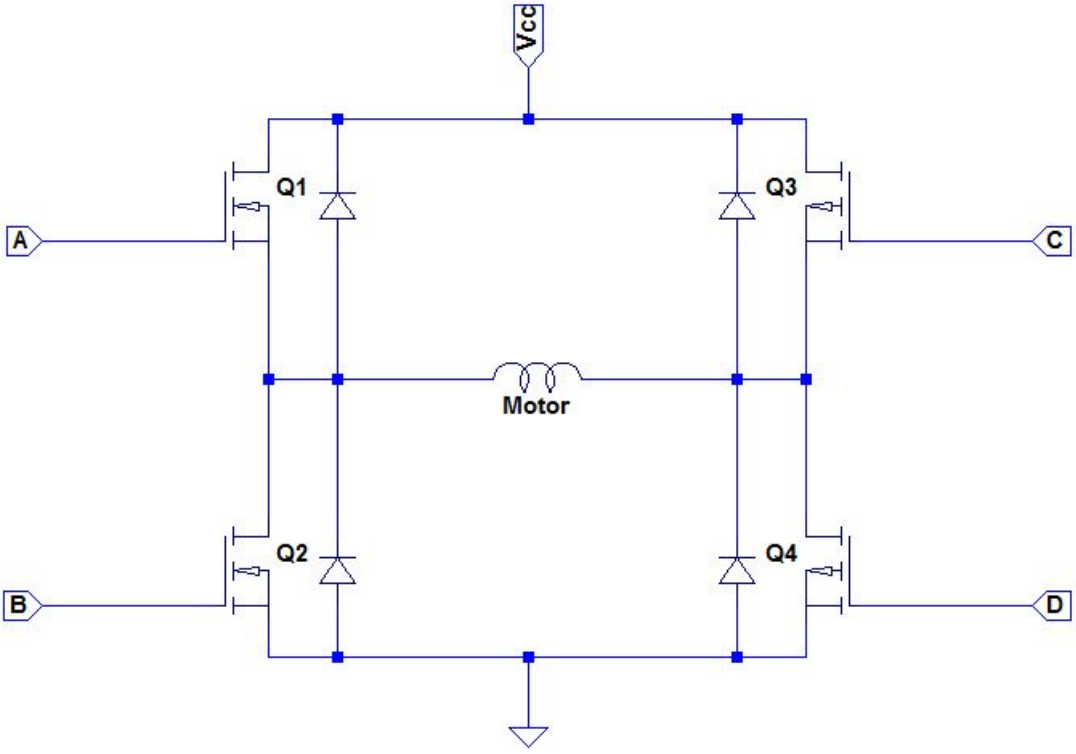


Figure 3. H-bridge Circuit Diagram

Above is the circuit diagram for an H-bridge circuit. When Q1 and Q4 are turned on, current will flow left to right through the motor's coil. Inversely when Q3 and Q2 are turned on, current will flow the other direction through the motor's coil. Q1 and Q2 as well as Q3 and Q4 should never be turned on at the same time. If this were to occur, the VCC and ground would be shorted, potentially damaging the transistors. This is prevented by properly connecting connections A, B, C, and D so that this never happens. If the current were to switch directions quickly, the voltage across the inductive coil will generate an extremely large

voltage, potentially damaging the circuit. This is prevented by placing diodes across each of the transistors.

3.5.6.2 Pre-Built Stepper Motor Drivers

Rather than creating the H-bridge circuits to control the stepper motors, another option is to use a pre-built stepper motor driver to control the coils of the motor. Typically these drivers use H-bridge circuits anyway to control the coils inside the stepper motor. Using a driver not only simplifies the design but also comes with the advantage of being able to "microstep" the motor. This microstepping allows the motor to move less than a full step which allows the motor to move at angle smaller than the step angle of the motor. This function is very advantageous for this design, as being able to move in increments of small angles is desired for this application. The project design uses three stepper motors to move the tonearms, so three stepper motor drivers will be required in this project.

3.5.7 Stepper Motor Driver Selection

The goal of this section is to research a select group of stepper motor drivers to find the best for this application. For the two different types of motors controlling the tonearm, one or two different types of stepper motor drivers will be required. The ideal driver should support microstepping to increase the precision of the horizontal movement of the tonearm by decreasing the step angle. As a minimum of three drivers will be required for this application, the ideal driver should be as cheap as possible to lower the overall price of the project.

3.5.7.1 V44 A3967

The V44 A3967 stepper motor driver allows for up to 8th step microstepping. This driver can support stepper motors with 4, 6, and 8 control leads. This is beneficial, but not useful for this application as the stepper motors chosen for this application all use 4 control leads. One downside of this driver it does not come with pre-soldered pins which may cause some difficulty during the breadboard testing.

3.5.7.2 A4988

The A4988 stepper motor driver carrier allows for up to 16th step microstepping. Of the three researched motor drivers, this driver supports the highest degree of microstepping. While not completely necessary to meet the precision requirements for this application, this high degree of microstepping will be useful to reach a great deal of precision. The vendor for this driver also gives options for pre-soldered pins, allowing for easier breadboard testing/debugging.

3.5.7.3 L293D

The L293D dual H-bridge driver for DC and Stepper motors is designed to drive a current through an inductive load in either direction. Unlike the other drivers researched for this application, the L293D does not support microstepping. This is made up for by being the cheapest of the three options at a price of 2.95$. Because of the lack of microstepping, this driver is not suitable for driving the horizontal stepper motor. However because of the low price, this driver could be suitable for the pitch motors.

3.5.7.4 Stepper Motor Driver Comparison

	V44 A3967	A4988	L293D
Supported Leads	4,6,8	4	4
Microstepping	Yes, 8th step	Yes, 16th step	No
Cost	6$	5.95$	2.95$
Operation Range	3-5.5V	8-35V	4.5-36V
Current Output	750mA	1A	600mA

Table 4. Stepper Motor Driver Final Comparison

3.5.8 Electromagnetic Clutch

When playing a record, the horizontal movement of the tonearm should not be held completely in place. Unfortunately, the detent torque of the horizontal stepper motor will hold the arm in place even when the coils are not energized. One method to counter act this is to use an electromagnetic clutch to disengage the motor with the mechanism holding the tonearm while the record is playing. The motor selected to control the horizontal movement of the tone arms is 8mm. Therefore, the electromagnetic clutch desired for this application should have a bore size of 8mm.

3.5.8.1 Tiny-Clutch M50

The Tiny-Clutch M50 magnetic spring clutch is a high torque capacity spring clutch with the control of a conventional electric clutch. This clutch has a Bore size between 1/8" and 5/16". This clutch has a max torque of 2.8Nm and a max speed of 3000RPM which is significantly higher than this project could possibly need. The biggest problem with this clutch is that there is no vendor which this clutch can be easily purchased from.

3.5.8.2 Micro EM Clutch 2.5

The Micro EM Clutch 2.5 is a Bi-directional electromagnetic clutch. It can obtain speeds up to 300 RPM which is much more than required for this application. This clutch also has a bore size of 6mm. This clutch operates at 24V and 3.4 Watts so there would need to be some form of voltage amplification if it is desired to run just from the microcontroller. Like the previous clutch, there is no vender which this clutch can be easily purchased.

3.5.8.3 102-02-13

The 102-02-13 electromagnetic clutch operates at 24V so there would need to be some form of voltage amplification if it is desired to run just from the microcontroller. This clutch has a bore size of 8mm and a maximum torque of .4Nm. This clutch can run at a max speed of 10000 RPM which is much more than the required speed for this application. Like all the previous clutches, there is no vendor which this clutch can be easily purchased.

3.5.8.4 Electromagnetic Clutch Selection

The table below shows a comparison of the three clutches researched for this project. Although any of these clutches would be suitable for this application, none of the clutches have a vendor which could prove problematic attempting to purchase one of them. All three clutches operate basically the same way and have very similar dimension and bore sizes. All three would be suitable in terms of max torque and max rotational velocity. However, the M50 appears to have the highest max torque and would be the clutch of choice if a suitable vendor is found.

	M50	MIC-E	102-02-13
Bore Size	1/8"-5/16"	6mm	8mm
Dimensions	1.25"x1.17"	29x26x4mm	39x39x18mm
Max Torque	2.8Nm	.25Nm	.4Nm

Table 5. Electric Clutch Final Comparison

3.6 PCB

A PCB (short for printed circuit board) is just that, a circuit board that is printed. Rather than loose wires connecting circuit components, all connections are manufactured into the board. This allows for more reliable circuits. This following section outlines the parts and key features of a PCB.

3.6.1 PCB Layers

A PCB is made up of many layers of material that are made into a single object through heat and adhesive. Each layer serves a specific purpose. The main layers involved are a Silkscreen, a Soldermask, Copper, and a Substrate.

3.6.1.1 Substrate

The substrate, usually made out of fiberglass, is used to give the PCB its rigidity and thickness. Usually a PCB ranges from .8mm to 1.6mm thick. The material that is used for most substrates is “FR4”. FR4 can be more expensive than other substrates, but it is much more durable. FR4 is also less likely to melt when a soldering is held too long on the board. FR4 actually describes the grade of the glass-reinforced epoxy laminate used in manufacturing, and the “FR” stands for flame resistant.

3.6.1.2 Copper

The copper portion of the PCB is a thin foil of copper that is laminated to the board using heat and adhesive. Many PCBs are double sided, meaning that this copper foil is applied to both sides of the substrate. PCBs can have 1 to 16+ layers, meaning that there are more and more layers of substrate and copper. The thickness of the copper is different from manufacturer to manufacturer, but it tends to be specified by weight (ounces per square foot). Most PCBs have 1 ounce of copper per square foot, but PCBs that need to handle higher power would have a thicker layer of copper. Every ounce per square foot corresponds to roughly 35 micrometers.

3.6.1.3 Soldermask

The soldermask is the layer that is on top of the copper foil. For most PCB manufacturers the soldermask is green, though it can be a variety of other colors. This layer is on top of the copper in order to help insulate the copper from accidental contact with solder or some other metal. Whenever a piece of solder connects two pieces of a circuit, it is called a solder bridge. A solder bridge being out of place can ruin a circuit, so the solder mask is an extremely important layer of the PCB. It also helps the person responsible for soldering components onto the PCB solder to the correct places. The least expensive solder masks are an epoxy liquid. Other types are liquid photoimageable soldermask and are sprayed on the PCB

3.6.1.4 Silkscreen

The silkscreen layer is placed on top of the soldermask. This layer adds letters, numbers, and symbols onto the visible side of the board in order to make

assembly easier. It can also make it easier for a user to understand the board. The silkscreen is usually used to label pins, leds, part numbers, warnings, descriptions of the board as a whole, or even company logos. The silkscreen can be any color (usually white), though the same color is used on the whole board. The silkscreen does add to the cost of the PCB, however, so it shouldn't be used unless necessary.

3.6.1.5 Layer Stackup

Though the layers described above make up the layers of a PCB, some of these layers can be repeated to make a multi-layered PCB. These additional layers make it possible to have more components and therefore more complex circuitry. The following table shows how additional would be stacked to make a multi-layered PCB design.

Layer Number	Layer Description
1	Top Silkscreen
2	Top Soldermask
3	Layer 1 Copper
4	Substrate
5	Layer 2 Copper
...	...
n – 1	Substrate
n	Layer n Copper
n + 1	Bottom soldermask
n + 2	Bottom Silkscreen

Table 6 – Multilayered PCB Design

As this table shows, copper and substrate layers can be stacked again and again between the outer layer to add more and more depth to the PCB.

3.6.2 Component Packages

There are multiple types of packages that each component can come in, and these different packages come with their own advantages and disadvantages. The main types of packages used are

thru-hole packages, SMD packages, and BGA packages.

Thru-Hole packages are the component packages that are used most often when prototyping circuits, but is still used in finalized products. These components have pins that are meant to be inserted through a plated hole in the printed circuit board. On the opposite side from which the component was inserted, the component is soldered in order to semi-permanently secure the device onto the board. An advantage of these thru-hole packages is that the component can be removed melting the solder that attaches the pin to the board. Thru-hole packages, however, do make boards more expensive to produce, as they require the boards to be drilled. In modern PCBs, thru-hole packages tend to be used for larger components such as electrolytic capacitors.

Figure x – thru-hole package

SMD packages, short for surface mount device, are components that are soldered on the same side of the board that the component was placed on. This is advantageous because these components can be mounted on both sides of the board. Additionally, these devices are smaller than thru-hole components, which can allow for the whole board design to be smaller. Because SMDs are smaller and mounted directly onto the board, these devices perform better under vibration conditions. Because of their size and because holes don't need to be drilled onto the PCB to mount them, among other reasons, these packages tend to be cheaper to manufacture.

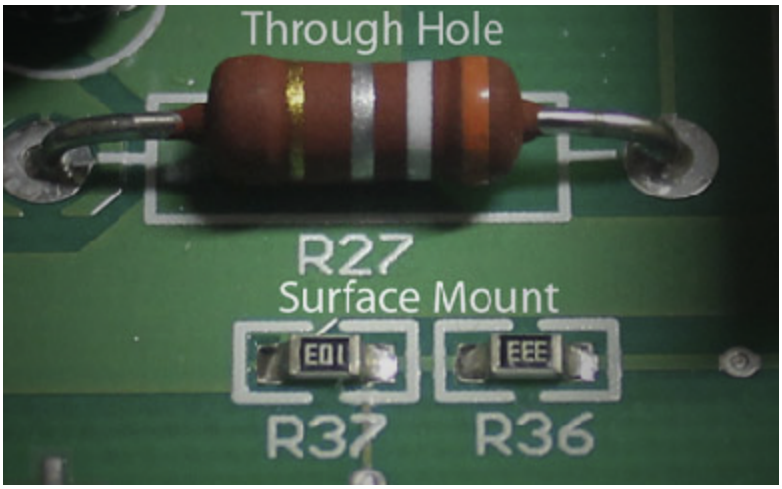


Figure 4 – SMD vs Thru-Hole size comparison

BGA, or ball grid arrays, are used to permanently mount devices, such as microprocessors. BGAs tend to be used for high-density pin ICs. Soldering these to a PCB requires specialized machinery because the pins are made of solder balls that have to be melted in order to make an electrical contact with the pads. The components are very popular in hardware like motherboards and video cards.

3.6.3 Other PCB Terminology

PCBs are a very complex technology, and a lot of research was done on how to develop a PCB for this project. While researching PCBs, many new ideas

had to be learned. This section defines some key ideas that were learned in researching PCBs.

3.6.3.1 Pads

The pad is the portion of exposed metal on the surface of the PCB that the component is soldered to. These pads mechanically support the components that are connected to them, and also provide a connection to the board (and therefore the rest of the components). There are two types of pads – thru-hole and SMD – each of which accommodates a different component package type. The thru-hole pads have holes in the circuit board while the SMD pads don't.

3.6.3.2 Copper Tracks & Vias

Copper tracks are conductive paths that connect multiple points in the PCB. The widths of these copper tracks varies based off of the current that would flow through the copper track. In very high frequency devices, the width of the track can play a large role in the performance of the device.

Plated holes holes in the board that have an annular ring that is plated all the way through the board. Plated holes are used whenever components located on different layers of the PCB need to be connected. These connections between the layers of the board can also be called a via, which stands for Vertical Interconnect Access. Technically, a via is a plated hole that allows for current to be passed through the board. Tented vias have a soldermask covering over them. This protects them from being soldered to accidentally. Untented vias are not covered, and are often used for components that might need soldering later. Most vias will connect from the top layer to the bottom layer.

There are blind vias, however, that connect from an external layer and end on an internal layer. It is possible to see if a via on a PCB is blind by placing it against a source of light. If the light can be seen through a via in the PCB, it is a through-hole via. Otherwise, it is a blind via. Blind vias help to conserve space when making PCBs with many components. Buried vias are similar to the blind vias, except the via starts and ends in an internal layer of the PCB. Like the blind vias, buried vias are helpful in preserving space on the board. In the figure below, a through-hole, blind, and buried via are depicted from left to right.

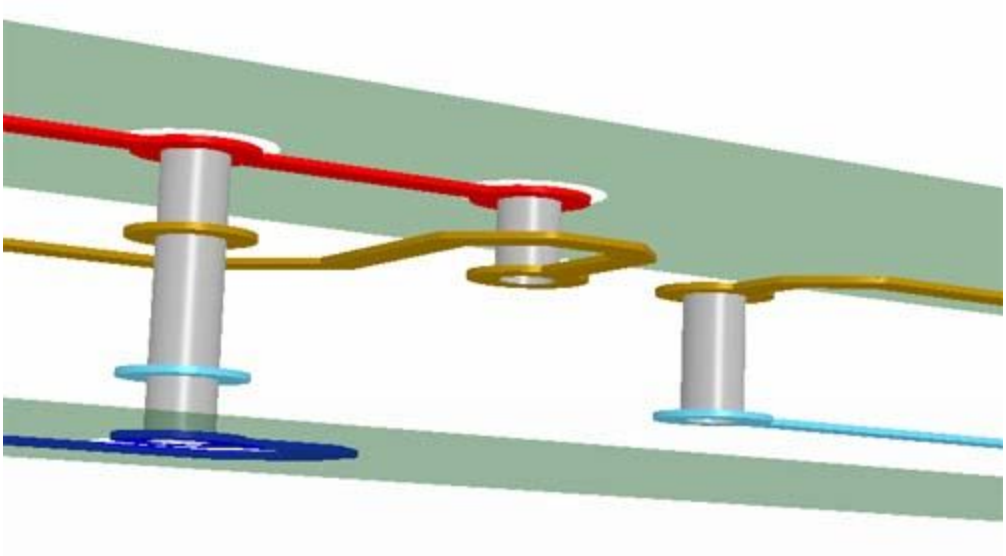


Figure 5 – Through-hole, blind, and buried vias

3.6.4 PCB Design Software

Autodesk's Eagle application will be used to design the PCBs for the Vinyl Player 2.0. This software facilitates in designing the schematic for the circuit to be manufactures, as well as placing the components in a way that makes sense. The schematic design portion of Eagle works similarly to many circuit simulator softwares such as Multisim and LTspice. When a new schematic is made, a button can be clicked that opens a window containing all of the elements that can be placed on the circuit. This list contains thousands of components, as there are many component types with slight variations that are listed for each manufacturer that makes the component. Additionally, libraries of other components can be imported into Eagle, which would add more components that are available for use.

To make a schematic, the components required for the circuit can be selected one at a time and placed on the workspace. It is important to note that for any given component, the package type and size for the component also needs to be taken into account. Once the components have been placed on the workspace, they can be moved, rotated, and arranged on the workspace to create a design that makes sense using the many formatting tools offered by the Eagle software. Once the components are laid out on the workspace, they can be connected with wires in the schematic. Common connections can be named to make the schematic as easy as possible for another person viewing the circuit to understand. Once the schematic has been wired together, circuit components can be assigned a specific value where applicable (i.e. resistors and capacitors).

After the schematic has been completed, the PCB board can be designed. A button can be clicked that will generate a board file that contains all of the components with their wiring as specified in the schematic file. An outline of the

maximum board size will be displayed, and each of the components can be dragged and dropped in the space allotted. It is important to arrange these components in a way that makes sense for the PCBs use case (i.e., IO pins on the outside edges), as this file determines exactly how the PCB layout will be manufactured. As the board is designed, specific layers can be specified for the copper tracks, vias, and components. Additionally, holes that need to be drilled (for mounting, etc.) and silkscreen to be applied are specified at this stage.

Once the board has been designed, the PCB can be finalized by choosing the actual manufacturer for each component. Each component will have a different price, quantity available, minimum purchase quantity, and perhaps different specifications, so it is important to take note of these details. Also, some components will have a one-time fee that needs to be made to purchase the product. These fees can add up to make a big impact on the price of the PCB. Once these details have been specified, each layer of the PCB design can be exported as a Gerber file. This file type is a standard across the industry. Most PCB manufacturers will have a website where a folder containing the PCBs Gerber files can be uploaded to verify the design. Also, the website should provide a quote of the price of the PCB.

3.7 Proximity Sensors

One of the main difficulties in making the record player play two sides of the record is making the tonearm and stylus apply a consistent force onto the underside of the record. When playing from the top of the record, gravity makes the stylus to rest on the record. When playing from the underside of the record, however, the stylus would need to assert the same the same force on the record while working against gravity. Many ideas were discussed in achieving this, one of which was using proximity sensors in a feedback loop in order to continually measure the distance from the cartridge (which is attached to the stylus), thus maintaining a constant force onto the record.

3.7.1 Proximity Sensors for Two-Sided Play

The main types of proximity sensors are Inductive, Capacitive, Photoelectric, Retro-reflective, Diffuse, and Ultrasonic sensors. Inductive sensors are only suitable for ferrous targets, and since the target (vinyl record) is not ferrous they were not researched further. Also, since Photoelectric, Retro-reflective, Diffuse, and Ultrasonic sensors aren’t as appropriate for close-range detection, Capacitive sensors were the main devices researched.

Capacitive sensors have two conduction plates at different potentials in the sensing head. The plates are connected to an oscillator, a Schmitt trigger, and an output amplifier, creating an output signal when the capacitance of the two plates increases from an object entering the sensing zone. One disadvantage of capacitive sensors was the slow response time from the device sensing a nearby target to outputting a signal. To maintain a constant force on

the record from the arm, a quick response time is crucial. Also, though these sensors can be very accurate, their cost can be high. In the end, it was decided that other methods were more reliable for maintaining a constant force on the record when playing from the bottom side.

3.7.2 Proximity Sensors for Tonearm Calibration

Another potential use for proximity sensors is in calibrating the motor that would control the tonearm's radial motion. Initially, the motor would move the tonearm to the user's inputted location on the record. Once the tonearm has been placed, however, the motor would disengage from the tonearm, allowing it to move freely. As the record turns, it would slowly move the tonearm. Then, when the user gives input to the app to change songs, the motor would re-engage the tonearm and would no longer have a reference of there the tonearm is along the record. This would then require the motor to recalibrate in order to regain its bearing. One idea for how to accomplish this is to add a proximity sensor on the outside edge of the record and moving the tonearm to this proximity sensor after every time a song is chosen. Once the proximity sensor registers the tonearm, a signal would be sent to the controller indicating that the tonearm is at the outside edge. From their, the tonearm would be moved and lowered as usual.

Because extreme precision for this case is not necessary for this application, photoelectric sensors were the main proximity sensors researched for this application. Though there are many variations of types of photoelectric sensors, all have a source that emits light, a photodiode or phototransistor that detects the emitted light, and an amplifier to output the signal.

3.7.2.1 Through-beam Sensors

In a through-beam photoelectric sensor, the emitter and the receiver are separate and installed across from each other. The emitter continually sends a beam of infrared light to the receiver, and the target would move somewhere between the two. If the target passes through the beam of infrared light, the receiver no longer receives the signal and therefore outputs a signal to the controller. These sensors are very reliable and accurate. The main draw-back of a through-beam sensor is that it would require the installation of two separate devices. Especially since the sensor cannot be housed within the record player, it would be ideal for the sensor for calibration to be as unobtrusive as possible.

3.7.2.2 Retro-reflective Sensors

Unlike the through-beam sensor, retro-reflective sensors have both the emitter and the receiver are located in the same housing. The emitter sends a constant beam of light, which is projected off of a reflector, causing the beam to be aimed at the receiver. Like the through-beam sensor, a constant beam of light is provided to the receiver, and an output signal is generated whenever that

signal is broken. This occurs when the target disrupts the laser that is between the emitter-receiver pair and the reflector. Retro-reflective sensors have accuracy similar to through-beam sensors. The fact that the emitter and receiver are housed together does provide the benefit of being able to wire them together; nevertheless, this solution would require two sets of devices to be set up at two different spots on the record player, which is not ideal.

3.7.2.3 Diffuse Sensors

Like the retro-reflective sensors, the emitter and receiver are housed in the same device for diffuse sensors. In diffuse sensors, however, the target acts as the reflector. So, the emitter sends out a beam of light in all directions. If the target enters the area where these beams are sent, the light reflects off of the target and returns to the receiver. The receiver then outputs a signal to indicate that the target is within range. While in the through-beam and retro-reflective sensors the beam of light is constantly hitting the receiver, and the target is registered when the beam is broken, the diffuse sensor works in the opposite way; the receiver doesn't register the beam of light until the target moves in range. Though these sensors can be slightly less reliable and accurate when compared to the through-beam and retro-reflective sensors, they have the advantage of being located in one compact device. For that reason, the only photoelectric sensor models that were researched further are diffuse sensors.

3.7.2.4 OT 18 M 1000 N4-B4

The OT 18 M 1000 N4-B4 sensor is a diffuse reflective sensor. It has a screw design and emits an infrared light. It has a operating distance of 1 mm, which is appropriate as a calibration sensor for the Vinyl Record 2.0. The sensor does have high power consumption, operating at 24V DC and about 30mA. Sensitivity can be adjusted through a potentiometer, and has a switching output of 100 mA.

3.7.2.5 KT10-8-H-8

The KT10-8-H-8 is also a diffuse sensor with a screw design that emits infrared light. It has a detection range of .5mm to 10mm. A major disadvantage of this device is that it requires a signal transformer, so this device will not be used.

3.8 Bluetooth

We are using bluetooth technology to enable communication between our smartphones/tablets and smart vinyl player. Bluetooth will be able to transmit the commands (media controls, power, track selection, etc.) and information from our mobile application to our vinyl player. The microcontroller unit on the vinyl player should be able to interpret and process the given commands from our mobile app and perform the necessary actions. In the following paragraphs we are going to

be discussing the different flavors of wireless communication, the pros and cons of each technology, and why we chose to use bluetooth.

3.8.1 Bluetooth Adoption

Bluetooth is one of the most commonly used technologies for mobile communication. Bluetooth's flexible and robust system ensures a strong and reliable connection with any bluetooth enabled device. Any bluetooth device must be evaluated and registered by the Bluetooth Special Interest Group to ensure standard practices and protocols are being followed. Ultimately, this means that every bluetooth enabled device will be compatible and work as expected. It makes sense for us to utilize bluetooth technology because we will be able to market our product to a wide audience of mobile users.

Many companies have integrated bluetooth technology into a variety of devices like smartphones, tablets, microcontroller units (MCU), etc. This means that there is a wide range of devices that we can use to interface with our smart vinyl player. This gives us some flexibility in choosing which environment we want to develop our mobile application, Apple or Android or Windows. Our mobile application must interface with a compatible MCU to receive and process our commands. We also require a compatible MCU to interface with our mobile application. Many manufacturers (Raspberry Pi, Arduino, Texas Instruments) offer cheap MCUs with bluetooth enabled technology, giving us a wide range of options to choose from.

3.8.2 Bluetooth Documentation

Since Bluetooth has been widely adopted for mobile communication, there is a great amount of support and documentation from the Bluetooth SIG and third-party developers. The Bluetooth SIG outline the requirements, design specifications, and constraints of a Bluetooth enabled device. Every Bluetooth enabled device must be certified through BT SIG. Bluetooth standards are beneficial because they ensure that every Bluetooth enabled device is compatible with each other. They also provide useful information about the capabilities that Bluetooth has to offer like profiles, protocols, and Assigned Numbers. Having this information readily available to developers makes it easier for us to learn and understand the features of Bluetooth, consequently, making our applications much more efficient and easier to use.

Apple and Android provide information and resources about their Bluetooth APIs that developers like us can reference as we develop our mobile application. The documentation provides information and examples on how to implement Bluetooth functionality in the most efficient way possible. Some of the features noted inside the documentation include Bluetooth setup, connecting two devices, device discovery, and necessary permissions. Important topics to note as we begin developing our mobile application to connect to the smart vinyl player.

Developers have been working with Bluetooth for a long time. Seasoned developers understand the ins and outs of Bluetooth and how it operates. Because of their past experiences, they are able to analyze a problem and provide a hint or solution to a problem. All of their hints and solutions can be documented on forums like stackoverflow. Developers new to Bluetooth can reference the past problems and solutions to solve their current problems, speeding up Bluetooth development. We are going to have problems as we develop our mobile application to communicate with our smart vinyl player through Bluetooth. We will most likely encounter a problem that another developer has faced before and be able to refer to someone's solution rather than solving it on our own.

3.8.3 Bluetooth Network Topologies

There are different types of bluetooth technology that exists for different scenarios. There is Bluetooth Low Energy (BT LE) Broadcast which establishes a one - to - many device communications. This mean that we can turn a single device into a beacon that would transmit information to multiple devices. This network topology doesn't apply to our project because our phone/tablet isn't meant to communicate with multiple devices, just one vinyl player. The next one is BT LE Mesh which establishes a many - to - many device communications. This translates into multiple device communicating with multiple devices at any given moment. This topology doesn't apply to our particular case because we only need a one - to - one device communication. The two network topologies that would apply to our particular case would be Basic Rate/Enhanced Data Rate (BR/EDR) and BT LE Point - to - Point. Both of these topologies offer a one - to - one device communication, perfect for our scenario. Both of the topologies fit our requirements but they both have their advantages and disadvantages.

3.8.4 Bluetooth Redundancy

Bluetooth has redundancy features (error correction) to mitigate any loss of data when transmitted to another Bluetooth enabled device. Transmitted data can be corrupted or lost due to external factors like interference from other Bluetooth enabled devices or radio frequencies. BT redundancies ensure that our mobile device can effectively communicate and transmit data to our smart vinyl player. For a good user experience, the user should be able to transmit commands (media/speed controls) with no delay or error. The mobile device should also be able to transmit record information (# of songs on the album, duration of the songs, album name, etc.) to the microcontroller unit (MCU) where the MCU will process the information and execute the corresponding commands. If the MCU was given garbage inputs, the MCU will produce garbage outputs and execute the wrong commands, possibly disturbing the user experience or destroying the vinyl player.

Bluetooth provides two error correction schemes (forward error correction code (FEC) and automatic repeat request (ARQ)) to protect against data loss.

There currently exists ⅓ FEC and ⅔ FEC schemes that corrects against errors. ⅓ FEC works by repeating each bit three times. ⅔ FEC encodes data using the (15,10) shortened hamming code. In the (15,10) scheme, 10-bits are used for information and 5-bits are for error correction. This allows correction of all single errors and detection of double bit errors. In the ARQ scheme, the transmitting device continues to transmit packets until it receives an acknowledgement from the receiver that it has successfully received the packet.

Error correction is very useful to mitigate any unforeseen issues/ data losses in congested environments but they add overhead to each packet, reducing the overall throughput.

3.8.5 Bluetooth Security

Bluetooth achieves device connection and data transfers using radio waves. Unfortunately, this make Bluetooth susceptible to being hacked. Hackers could interrupt or disrupt the signal and hijack the Bluetooth enabled device. What this means is that random users could block your connection from the mobile device to the smart vinyl player or worse, hack into your vinyl player/smart phone and play unwanted songs or steal personal information. Luckily, Bluetooth SIG and device manufacturers have developed security features to mitigate any unwarranted access. Bluetooth utilizes five security services; confidentiality (only authorized devices can access and view transmitted data), authentication (verify the identity of Bluetooth devices by their address), message integrity (verification of transmitted data), and pairing/bonding (creating and storing shared secret keys for subsequent connections) procedures to provide basic and foundational security. Any other means of security will be provided by the manufacturers of the Bluetooth enabled device. You can also make your Bluetooth enabled device “non-discoverable” to stop any Bluetooth connections.

3.8.6 Bluetooth Android Development

We have chosen to develop our application in the Android environment because Android provides well documented resources on their software & APIs, better integration support for third-party software, wide variety of debugging and simulation tools, java based syntax, and our team only have PCs. This makes it easier to begin software development because of the reduced learning curve and ease of access to resources and support from the open-sourced community. This allows us to create a feature rich and efficient application and increase the success of our project. Androids Bluetooth Low Energy API provides a guide to setup Bluetooth, establish a connection between two devices, transmit data between the devices, and finally close the connection.

The first thing we are required to do is setup the Bluetooth permissions on our device in order to communicate (request a connection, accept a connection, or transfer data) with other Bluetooth devices. We must declare “android.permissions.BLUETOOTH” and if we require device discovery or manipulation of Bluetooth settings, “android.permissions.BLUETOOTH_ADMIN.”

We would only like our mobile application to connect to Bluetooth Low Energy devices only, so we must declare “android.hardware.bluetooth_le.”

The second thing we need to is verify BT LE is supported on our mobile device. However, today’s smartphones and tablets come standard with BT LE and the device we’re using will have BT LE support. However, we need to check in case an older device wants to try to use our smart vinyl player. The next thing would be to obtain the “BluetoothAdapter” class on the mobile device and check if the mobile device has Bluetooth enabled.

The next step is to search/query for Bluetooth Low Energy enabled devices. You would call the “startLeScane()” method to search for any BT LE enabled devices. In order to preserve power, stop scanning when you find the target device, never scan on a loop, and only scan for a predetermined amount of time. You have the option to scan for specific type of peripherals by scanning for specific universally unique identifier (UUID).

Once you have found the device, you can now begin connecting to it. To connect to the Bluetooth enabled device, you are connecting to the device’s Generic Attribute (GATT) profile server, the GATT profile defines how the device works in the particular application. To enable connection with the GATT server, you call upon the “connectGatt()” method. In our project, the phone (GATT client) will call the “connectGatt()” method and initiate a connection with the smart vinyl player. If successful, the smart vinyl player will return an acknowledgement to the GATT client that the connection was successful. We can begin transmitting information, attributes about the GATT profile, and any status changes to the Android application.

Once we are done communicating with the smart vinyl player (finished listening to music), we should end the communication between the two devices. Doing so will preserve battery power and allow the mobile device to connect to other Bluetooth enabled devices. To break up the Bluetooth connection, we would call the “close()” method and the application will end the connection with the GATT profile server and release the occupied resources.

3.9 Mobile Communication Technologies

There are many types of wireless communication technologies on the market today which gives us a range of options to choose from. We are going to identify the most prominent features of each technology and weigh the costs vs. reward of each technology and decide the best fit for our project.

3.9.1 Bluetooth Table Comparison

Below is a table that are comparing four different mobile wireless technologies that exist today. Each one of these technologies have key features/components that pertain to our goals outlined in the “house of quality.”

Bluetooth Low Energy is the newest generation of Bluetooth that exists on most modern mobile technologies.

BR/EDR is the foundation of Bluetooth with many generations of upgrades that make it relevant in today's technology.

A promising technology created by WiFi Alliance to compete with Bluetooth technology. It is a promising technology that easily rivals Bluetooth and you can see why in the table below.

NFC is another wireless mobile technology that currently exists in the market. The table below might discourage the use of NFC but they do have a place in today's market.

	Low Energy	**BR/EDR**	**Wi-FI Direct**	**NFC**
# of Channels	40	79	N/A	N/A
Operating Frequency	2400 - 2483.5 MHz	2400 - 2483.5 MHz	2400 MHz & 5000 MHz	13.56 MHz
Powered by Coin cell battery	Yes	No	No	N/A
Current Consumption	< 15ma (read and transmit)	N/A	N/A	< 15ma (read)
Throughput	1 Mbps	≥ 2Mbps	≤ 250 Mbps	106 - 424 Kbps
GATT Profile Support	Exclude HSP, OBEX, A2DP, VDP, FTP	Everything	N/A	N/A
Required Pairing	Yes	Yes	Yes	No
Standby time	.6 - 1.2 ms	22.5 ms	N/A	N/A
Connection time	3 ms	≤ 100 ms	N/A	≤ 100ms
Range	Up to 100M	Up to 100M	≤ 180M	10 - 20 Cm
Android &	Yes	Yes	No	Yes

Apple Support?				
Security	128-bit AES	E0/SAFER+	256-bit AES	Limited
Backwards Compatibility	Yes	Yes	Yes	N/A
Cost of Tag	$5.00	N/A	N/A	$0.10

Table 7: Bluetooth Comparison

3.9.2 Bluetooth Legacy

Bluetooth was formally introduced in May of 1998 and since then every BT generation has been evolving to include faster speeds, more features and better reliability/robustness. That is why Bluetooth 4.x has become the standard for mobile devices. For our project, we have decided to develop with Bluetooth 4.x (4.2 to be exact) because of its speed and reliability as well as the low power consumption. Our mobile application would be compatible with older generations of Bluetooth due to the built in backwards compatibility developed by BT SIG but connecting to older version of BT could hinder our goals outlined in the "house of quality."

The following table outlines the most notable features from different generations. The following generations inherit most of the notable features and protocols but can be dropped if they interfere or clash with another feature. The most notable feature that we have to outline from BT 4.x was the Low Energy mode and single-mode and dual-mode chips. Both single-mode and dual-mode offer lower energy options but the difference between the too is cost. If we need to lower costs, we can choose the single-mode chip but will be limited to low energy options. However, having dual-mode, we can switch between low energy low throughput for efficiency and change to BR/EDR if we require a higher throughput (higher power consumption).

Bluetooth Generation	Features	Description
BT 4.x	Low Energy	Powered by coin cell battery
	Dual Mode	BR/EDR and Low Energy support
	GATT & SM	Generic Attribute profile, Security manager
	AES Encryption	Security Update

	Internet Protocol Support Profile	Allows connection to the internet
	Single-mode chips	Low power idle operation, device discovery, lightweight link layer, etc. at the lowest cost
BT 3.x	Alternative MAC/PHY (AMP)	Bluetooth + 802.11 supports 24Mbps
	Unicast Connectionless Data	Sends data without establishing L2CAP channel
	L2CAP enhanced modes	Enhanced Retransmission mode that implements L2CAP channels
	Enhanced Power Control	Removes inefficiencies and ambiguity caused by open loops power control and EDR modulation schemes and adds closed loop power control

Table 8: Bluetooth 4.x Features

3.9.3 Bluetooth Final Table Comparison

Each one of these technologies has features outlined in the “house of quality.”
Some of the biggest features to note from the house of quality is cost, power consumption, range, and throughput. Other features that we would like to note is Android support, backwards compatibility, and network connection. These are the features that would complement our smart vinyl player and make it enjoyable for the everyday users.

After careful consideration, we have chosen to go with Bluetooth Low Energy for a variety of reasons. We have chosen BT LE because of its low power consumption (15 mA), range (up to 100M), and excellent throughput (1Mbps). Our reason for choosing LE vs. BR/EDR or Wi-Fi Direct was because of power consumption. Bluetooth Low Energy consumes about 15 mA which is orders of

magnitude less than BR/EDR and Wi-Fi Direct. The throughput and range is impressive on these two technologies but we are more concerned with power consumption for our project. Wi-Fi Direct isn't cross compatible with Android and Apple which makes it an issue. NFC provides the lower power consumption, low costs, and Android & Apple support we require but it's appalling range and throughput makes it less ideal.

We had chosen Bluetooth 4.2 because it is the most versatile and updated version of Bluetooth. Currently, it is the standard on all modern mobile devices. However, the main reason we have chosen Bluetooth 4.2 because it offers low energy power consumption and single/dual mode chips. The dual-mode chips allows us to alternate between low power/low throughput and high power/high throughput. We can switch modes depending on the current situation and whether the situation calls for it. Unfortunately, this comes at a higher cost. Single-mode chips offers only low power/low throughput but at the cheapest price possible, most suitable for our project.

3.10 Computer Vision

A fundamental component to the Vinyl Player 2.0 software system is the ability to take an image of a record label and recognize identifying information about it . Incorrectly identifying the album, artist, release or other information that is pulled from the label may result in the app showing the wrong information to the user virtually rendering its functionality useless until the software system can accurately fingerprint the record to be played. The field of interest for this task is computer vision, which deals with processing, analyzing, interpreting digital images.

3.10.1 Concepts & Algorithms

Becoming a field of interest in the late 1960s, computer vision has a plethora of solution approaches to the same problem and a variety of implementations for the same solution. Retrofitting the right concepts to the challenge of identifying records can make the process much more streamlined and ultimately produce a better product.

3.10.1.1 Viola-Jones Algorithm

The first object detection algorithm, the Viola-Jones algorithm proposed in 2001 by Paul Viola and Michael Jones is to this day one of the most powerful computer vision algorithms in the world [7]. Originally introduced to tackle the problem face detection, this algorithm has been expanded to general object detection. Some of the features of the Viola-Jones algorithm that make it one of the most useful of its kind is its robust nature, simplicity and applicability to real-time detection [8].

The process of this algorithm can be broken into for four stages: Haar feature selection, integral image creation, training of the system and cascading classifiers [8]. Haar features developed by Alfred Haar, a Hungarian mathematician, are properties that are shared among all (or nearly all) of the instances of the object that is trying to be detected.

Even though this technology is stable, supported and still popular today there has been much research in the field of computer vision. This has led to other more sophisticated solutions to the same problems that the Viola-Jones algorithms solves and thus has shown a slow decrease in this technique's popularity.

3.10.1.2 Optical Character Recognition (OCR)

Optical character recognition (OCR) is the process of converting characters, handwritten or electronically printed, contained in an image into "machine encoded text" [23]. A subfield of pattern recognition, this collection of techniques is highly applicable to the software system of the Vinyl Player 2.0 as the main goal is to recognize text on the record label to identify the record that is going to be played.

The first step in optically recognizing text in an image is segmenting the image so characters and words are isolated, thus making the image simpler to analyze. A simple method of implementing segmentation is to "isolate each connected component", where a connected component is a word or continuous line of words [22]. Once the image has been segmented the next step to OCR is to perform some pre-processing on the segmented images to reduce noise that may reduce the accuracy of the character recognition. A common pre-processing that occurs is smoothing, which entails eliminating small gaps and reducing the width of the segmented image. Another pre-processing technique is normalization, a process that will render all the characters in an image to be on the same size, rotation and slant.

After simplifying the task of optical character recognition through segmentation and pre-processing it is time to perform the actual classification of the characters. An early method of implementing this task is known as pattern matching or pattern recognition, which involves comparing the image to a stored glyph, an elemental symbol intended to represent a character, pixel-by-pixel. This method will have trouble when encountering new fonts, meaning it might not be optimal for the Vinyl Player 2.0 seeing that it is almost guaranteed that a large variety of fonts and styles will appear on record labels. A more sophisticated method for performing the classification is feature extraction. This technique uses features or piece of information that is common to the object that is being classified instead of glyphs effectively improving the computational efficiency of the process. Finding features to use for classification for a certain class of object is in itself a step that needs to be taken with feature extraction. Multiple runs of classification may be desirable to increase accuracy to OCR algorithm.

The final step in OCR is to perform some post-processing techniques to again increase the assurance of the results of optical character recognition is perform some post-processing. Grouping is a post-processing technique that groups characters that are sufficiently close to each other to create words, a technique that seems almost compulsory for this application. A post-processing technique that might not be as necessary would be restricting the possible words to a lexicon, the English language for example. While this can increase the accuracy of the OCR process it fails to account for proper nouns, which is likely to appear very often in the images that this software system will receive.

3.10.1.3 Deep Learning

Perhaps the most cutting edge technology that is replacing the Viola-Jones algorithm and other traditional computer vision techniques is deep learning, specifically the usage of convolutional neural networks. CNNs is a class of deep neural network, meaning that is has multiple hidden layers to represent complex models and when coupled used in image recognition has been shown to highly reduce error rate [10].

Using a deep learning approach as opposed to traditional programming are two fundamentally different tactics to getting a software system to accomplish the goal that is desired. Where traditional programming is virtually comprised of listing rules that the program follows to determine whether the image that it is searching for is there or not, programming with a convolutional neural network will simply be giving it the program an architecture that can take in input from a dataset, declaring what it is needed as output and allowing the system to learn and function on its own. This is one of the greatest advantages when it comes to working with CNNs because the system does not have to be altered every time there is a unique scenario, as it adapts itself based on the dataset. The specific techniques and tools that will be utilized to execute the deep learning implementation of the Vinyl Player 2.0 computer software system are discussed in section 3.14.

3.10.2 Computer Vision Libraries

All of the concepts and algorithms in the section above are core to the field of computer vision and also have been around for quite some time, with many of them being at least 20 years old. As a result of this the implementations of these algorithms and concepts are very sophisticated, meaning it would be much more efficient for the process of producing a computer vision application to use these existing implementations instead of attempting to build every component from scratch. This is the main incentive for utilizing libraries that provide easy access to the functionality of these algorithms and concepts in their best form.

3.10.2.1 OpenCV

Started in 1999 as an Intel Research initiative, OpenCV is an pioneer computer vision library that has a focus on computational efficiency and real time applications. Due its long and influential existence this library is much more comprehensive than its counterparts and has a huge community of around 47,000.

While being written in C and C++, OpenCV provides a number of interfaces for developing in other languages and can run on multiple platforms meaning that there would be no roadblocks developing an Android application using this library. This collection of computer vision programming functions is also open-sourced and released under a BSD license, which allows for free usage and alteration of the source code. Another feature bolstered by OpenCV is it's sublibrary for general purpose machine learning, Machine Learning Library (MLL). Seeing that training has an equally important part as image detection and recognition in a computer vision software system the MLL sublibrary provides a good infrastructure to for building the system.

3.10.2.2 MATLAB Computer Vision Toolbox

MATLAB itself is not a library, but a matrix oriented programming language developed in the 1970's and maintained by MathWorks. Like many programming languages there is a vast variety of extensions to the language for specific applications, where of course the Computer System System Toolbox is the library for computer vision applications. Some of the key features of this toolbox is support for deep learning, which a subset of machine learning, and object detection & recognition, both central components to the software side of the Vinyl Player 2.0 [5]. MATLAB is fairly simply to debug, highly documented and has a massive community of about 2 million as of 2017, although with the vast amount of applications of MATLAB it is likely that the community that specializes in the usage of the Computer Vision System Toolbox is a mere fraction of that [3].

An issue with this approach is that MATLAB is proprietary software and thus would require the purchase of a license and each individual toolbox, as they are sold separately from the environment and programming language. Considering student discounts and a minimum of one toolbox needed the lowest total cost for a single member of the team to develop with MATLAB would be $39.00, which may make this approach an unnecessary cost for this application [4]. Another drawback is that MATLAB is closed-source having the ability to alter the source to adapt to the Vinyl Player 2.0 would not be a viable option and actual understanding of the underlying functionality would be limited. In addition to MATLAB being closed-source and pricey it is not a very portable and would require some extra work to actually be used in an Android application.

3.10.2.3 Point Cloud Library (PCL)

PCL (Point Cloud Library), with its focus on 3D image and point cloud processing, is another option of computer vision library [6]. This modular library is cross-platform and has been successfully deployed on Android, the development platform for the Vinyl Player 2.0 software system. Similarly to OpenCV, this library also uses a BSD license giving free control to cater the code to the purpose of this application, yet PCL is not as highly documented as its contemporaries. A deficient in documentation would mean that there is more of time and effort needed to decipher the source code, a task that may not be the best use of resources.

As it was previously stated PCL has a focus on 3D images, which is an aspect of the system that would be highly beneficial for robotics or some autonomy application, but for the function of the Vinyl Player 2.0 this is an area that will most likely be unused. Point Cloud Library's focus on 3D is also showcased through the high emphasis on point cloud processing. A point cloud is a set of data points, typically in a 3D coordinate system, which can be visualized from the camera. Seeing that there is a sizeable focal point on 3D computer vision it would seem that using it for the core of the Vinyl Player 2.0 software system is a misguided step, as there are other libraries that more highly cater to this application.

3.10.2.4 SimpleCV

SimpleCV is unique from the other technologies mentioned thus far in that it is a framework that builds off of other computer vision libraries instead of being a library itself. Through aggregating other libraries, including OpenCV, this framework gives an easy way to interact and gain the benefits of using multiple libraries. The underlying functionality of all the libraries SimpleCV takes advantage of is accessed through a relatively simple Python interface. The high level interface provided by this technology would allow for quick prototyping of the computer vision software component of the Vinyl Player 2.0 system. Fast and easy prototyping coupled with the open-source nature of the framework and underlying usage of robust technologies makes SimpleCV an attractive prospect, yet it is not without its drawbacks.

A limitation of using this high-level and user friendly framework is that it is mainly designed for desktop applications and while all the underlying libraries that are encompassed in SimpleCV can be utilized in Android development there isn't much indication in the documentation that this platform is a strength of SimpleCV. This could mean that the benefits of simplifying the creation process may be offset by simply attempting to run the system on Android. Another issue with using SimpleCV is a lack of community and documentation, which again will stunt the development of the software, especially if the there is a need to work with the source code of the Python interface. Also, it can be observed that even though SimpleCV is open source, actual alteration of

the source is limited as the functionality is contained within the libraries that are contained in the framework and not the framework itself.. So it seems that while the SimpleCv framework provides a useful tool to get accustomed with some computer vision concepts and prototype them quickly, using it for the actual implementation of a computer vision system is not the best route to go.

3.10.2.5 Computer Vision Library Comparison

While all of the computer vision libraries shown in this section can in one way or another can be used has the tool that builds the system to pull identifying information from a record label, it is clear that OpenCV is best and most reliable library for this application. The dedicated and active community that grows every day simply dwarfs that of PCL and SimpleCV. MATLAB also has a huge community and great documentation, but of course the area that OpenCV defeats this technology in is the price, or lack thereof. With all of this and also being more comprehensive and often performing better than its counterparts, it is evident that OpenCV should be the computer vision tool of choice for the application of the Vinyl Player 2.0, as well as most other applications.

	OpenCV	MATLAB's CV Toolbox	Point Cloud Library	SimpleCV
Supports Android	Yes	Yes	Yes	Unlikely
Cost	$0	~$39 per developer	$0	$0
Community Size	~50,000	2,000,000 (not exclusively for computer vision)	~5,000	12,000

Table 9. Computer Vision Library Comparison

3.10.3 Datasets

One of the less glamorous parts of creating an image recognition, image detection or just general computer vision software system is training the system to give it some of knowledge of potential patterns in the data. The actual size of training set to effectively give the computer vision solution new insight is highly variable, but generally very large and more data is almost never a deficient to the training process. A dataset that accurately represents the problem that is being presented and gives the correct answers to that problem that the system can learn from will give the system a backbone to build off of and is one of the best ways to quickly and efficiently get the system outputting the desired results. There are a variety of resources to obtain data, which can be categorized into

open-source preexisting datasets and datasets that are built from the ground up mainly for the purpose of training your computer vision system.

3.10.3.1 Open-Source Datasets

One of the greatest problems when wanting to train a system is that it typically requires a great amount of well annotated data to learn from. Creating a dataset of this nature can be tedious and take a fair amount of time for a single team, which is why publicly available datasets are such a great asset to the community. There are a surplus of reliable open-source datasets with some of the most popular being Common Objects in Context (COCO), Udacity, Open Images by Google, ImageNet and many more than provide segmentation, classification and many properties of an image than can be used for application.

Probably the biggest fault that these form of datasets have is that since that in their nature they are just general datasets of many compiled images, they generally will lack the full information that is desired for the computer vision system. This aspect of publicly available datasets can actually detriment the training because the data being fed does not fully represent the data expected. For these reasons it is best not to fully depend of open-source datasets and use them to start testing the algorithms of the system as opposed to using them for the fully matured product.

3.10.3.2 Creating A Dataset

The much more painstaking, but rewarding option for gathering data to train a computer vision software system would be to create a dataset from scratch that is perfectly fits the challenge of the application. Proprietary datasets is what sets apart some implementations from others and is typically needed for more complex systems.

The simplest and slowest solution to creating a dataset, annotated in such a way to optimize training of the model, is by just add training examples to the dataset one by one with the correct annotations in mind. The process of creating each individual training example (i.e. taking a picture of a record label) and annotating it with the correct output (i.e. album name and artist) is absolutely not a scalable technique as it is very time consuming, but will train the system to produce exceptional accuracy.

Another method of creating a custom dataset to train the computer vision system of the Vinyl Player 2.0 is to attempt to crowdsource data. This will almost inevitably cause the quality of the data to go down, but greatly increases the scalability of creating a custom dataset and reduces the amount of time to do so. To minimize poor quality data coming when crowdsourcing the dataset it is important to give some guidelines about the annotations of training examples in the training set and potentially even use an annotation tool to provide an environment that will allow the community to add to the dataset with a similar format.

The creation of a custom dataset coupled with the use of publicly available data is the general way to gather data that will be used to train the system. This is also the method that will be used for the Vinyl Player 2.0 as there can be data about records pulled from open-source datasets, but the ultimate goal of OCR on a record label will require some more custom data.

3.11 Deep Learning

As discussed in the previous section, deep learning is one of the most modern techniques that tackle the computer vision challenge of the Vinyl Player 2.0. Presented here are fundamental concepts that lead to the convolutional neural networks, which is the class of neural network shown to work exceptionally well for image recognition. Also showcased will be a discussion and comparison of all the tools in contention for use in this application.

3.11.1 Concepts & Algorithms

Deep learning is encompassed within the field of machine learning and the algorithms and techniques of this subfield have breathed new life into the machine learning and AI as it has given the area many more practical uses. Just like with many complex concepts and algorithms, the sophisticated field of deep learning is built off of a number of smaller, slightly more intuitive ideas. These building blocks are discussed below.

3.11.1.1 Artificial Neural Networks (ANNs)

An artificial neural network, also known as just neural network, is a system that pulls inspiration from biological neural networks present in brains and uses neurons as the building block for the network. A neuron is virtually a node that receives one or more weighted inputs, representing dendrites, and produces a single output, representing the axon. Each neuron will sum up all the weighted inputs and apply an activation function to the result, which will define the output for the node. The core structure of an artificial neural network is an input layer of neurons, where the input into those neurons could be data from a dataset or data taken from external sensors, a series of hidden layers consisting of neurons and an output layer of neurons. The purpose of the hidden layers in an ANN is to transform the input signal to produce new information about the input data and move closer to the desired output.

The true power of neural networks is the ability to take the output of the network or expected value and determine if it matches the actual value, learn from the experience and tune the network to perform better. This is done by updating the weights of connections between neurons to better mold the results of the neural network to the results of the dataset. Doing multiple passes of the training set into the ANN will tune the weight values to very closely reproduce the actual value.

3.11.1.2 Gradient Descent

Gradient descent is an optimization algorithm for finding the minimum of a function and is utilized in neural networks on the cost function, which represents the accuracy of the neural network's output compared to the actual output. In the mathematical calculation of the gradient there is a variable for the learning rate of the algorithm, α, which determines how big of a step the weights of the neural network take towards the values that will minimize the cost function. Learning rate, α, is an important component of the gradient descent algorithm because if it is too small then there will be a performance hit as there needs to be many iterations of the algorithm for the weights to reach the desired values and inversely if the learning rate is too large then there is the possibility that the algorithm will never converge as the weights will always jump over the values that will make the cost function a minimum.

A property of gradient descent is that it finds local minimums and not absolute minimums, which in some problems could cause a non-optimal result as the true minimum is not returned. This is not an issue when the cost function is the squared difference of the actual value and expected value, which is the popular choice for cost function, as this function only has one local minimum, meaning that the optimization algorithm will always work towards the absolute minimum.

A method to remedy the issue of only locating the first local minimum in gradient descent is a stochastic version of the algorithm, dubbed Stochastic Gradient Descent (SDP). The main difference between the stochastic and nonstochastic versions of gradient descent is that SGD updates the weights in the neural network after each training example in a training set, whereas regular gradient descent simply updates the weight after all the training examples in a training set have gone. As stated before, the stochastic gradient descent method avoids the problem of the algorithm stopping at local minimum. Performing SGD is also surprisingly faster than the typical gradient descent due to not having to load as much memory.

3.11.1.3 Backpropagation

Backpropagation is a process used in gradient descent to finalize taking the step towards minimizing the cost function of the artificial neural network by updating the weight of each connection. An impressive aspect of this algorithm is that all of the weights are updated simultaneously. This property of backpropagation is due to the fact that the cost function takes all the weights in the network as a parameter and that new weights are found through first-order derivation. By taking the partial derivative of the cost function with respect to each variable weight in the neural network then all of the weights can be updated without having to traverse the network.

3.11.1.1 Convolutional Neural Networks (CNNs)

Convolutional neural networks are a class of artificial neural network that have exemplary performance when it comes to the fields of image recognition and classification, which makes it an ideal deep learning system to use for the task of performing optical character recognition on record labels to determine information about the record.

A CNN is implemented through four steps with the first being convolution, a process derived from the mathematical operation between two functions to produce a new function that gives insight the other two did not. The main goal at the convolution step of a CNN is to reduce the amount of data that needs to be processed by the neural network, while keeping the important features of the image and the spatial relationship in the pixel matrix that represents the image. This reduction is done by performing convolution on the pixel matrix of the image and a feature detector or filter which is a small matrix, typically 3x3, that when convolved with the original image will produce a smaller pixel matrix that maintains focus of the features that were outlined in the filter. In a convolutional neural network the convolution layer will be a multitude of feature maps that will all take the image from the training set and perform convolution to produce a multitude of the images, each with a different feature focal point. Note that the features that the filters in the convolution layer look out for are not determined by the developer, but by the network itself, which again is one of the reasons why this technology is so powerful.

While not as powerful of a step as convolution, the rectifier linear unit (ReLU) layer is still an integral portion of the convolutional neural network workflow. The ReLU layer is performed directly after the convolution layer and has the purpose of increasing the nonlinearity in the pixel matrix of an image because images themselves are highly nonlinear. This is executed simply by replacing all negative pixel values with zero.

The next layer in a CNN is the max pooling layer, where the objective is to reduce the dimensionality, allow for the CNN to avoid issues like distortion in an image and prevent over-fitting. Max pooling is implemented by defining a spatial neighborhood, 2x2 for example, and sliding it throughout the feature map that was obtained from the convolution layer. The maximum value for each neighborhood is obtained and put into a pooled feature map maintaining both spatial relationship and important features.

Each pixel in a pooled feature map is what is used as input to an artificial neural network, which will be the last step of the convolutional neural network known as full connection. In the ANN contained within a CNN the hidden layers are now known as fully connected layers due to the fact that while in a typical ANN the nodes between layers did not have to be fully connected, in a CNN they do. The main incentive of having an artificial neural network on the tail end of a convolutional neural network is to use the features of the images to create new properties of the image to give more insight about the classification of the image.

Just as with ANNs, each training example will run through the network, the expected result will be checked against the actual result and the network will be updated using this knowledge. Note that in a CNN will updates the features in the convolution layer as well as the weights in the ANN.

All of these steps and processes make convolutional neural networks very useful for problems of image recognition and classification, which is why these concepts can be fitted very nicely for the implementation of the computer vision component of the Vinyl Player 2.0.

3.11.2 Deep Learning Libraries

While most of the computer vision libraries discussed in the previous section contain machine learning libraries, they are generally too low-level for an object recognition system with deep learning at the center of it. This is remedied by a number of deep learning libraries that assist in the creation of convolutional neural networks.

3.11.2.1 TensorFlow (TF)

Conceived by Google Brain and originally released in 2015, TensorFlow is one of the most popular machine learning software libraries out on the market today. Due to its open-source nature and brand recognition, this library has a large & active community, extensive documentation and plenty of examples to learn from. The mass adoption of TF provides a strong incentive for its usage.

A majority of TensorFlow is written in C++ and CUDA, a programming language developed by Nvidia for GPUs, but has APIs for several of the most popular programming languages. It is important to note that even though there are multiple APIs, all language interfaces besides Python are experimental and not covered under the same "stability promises" that Python, which will ultimately add complexity when attempting to utilize this library for Android development, whether that be in the form of dependencies or making the CNN in Python and using a native interface to use in the mobile app [11].

3.11.2.2 Theano

Theano is one of the oldest of the stable deep learning libraries being around since 2007. Being a Python only library gives it similar complications as other libraries when attempting to develop for mobile. Theano is built for numerical computations optimization and is tightly integrated with the NumPy, a Python library for scientific computation, which made this library desired for deep learning in its inception.

The biggest drawback of using the pioneer deep learning library would be that it was announced on September 28th, 2017 that development of the library would cease after its next release and that maintenance would stop a year after that [13] . Of course it is essential to take into consideration the longevity and support of the underlying technologies when developing a software

system of any form and so this news greatly diminishes the incentive to use this technology for the Vinyl Player 2.0.

3.11.2.3 Torch

This deep learning framework is infamous for being used both for Facebook Research and DeepMind, prior to being acquired by Google [14].

Torch is unique from other deep learning resources in that it is based and uses programming language Lua, designed primarily for embedded systems, as opposed Python. While this aspect of the framework may give it advantages in memory efficiency and easy to interface with C & C++, it also produces the issue of having to learn a new programming language. The time learning a new language is a resource that may be better allocated in other tasks, especially since this framework is somewhat low-level and work require a deep understanding to build all necessary components of the system.

3.11.2.4 Keras

With its first initial release in 2015, Keras is the second fastest growing deep learning library behind TensorFlow. The library is written in Python providing and uses the language as its main interface like most of the libraries discussed thus far, but Keras sets itself apart by being is the most high-level of all the libraries. This makes Keras the most user friendly and minimizes the steps to implement common case neural network algorithms, while still allowing a user to dig into the source and gain a better understanding or alter the existing code to better work for the application.

Another aspect of this highly adopted deep learning library is that it can be layered on top of other deep learning libraries such as TensorFlow and Theano. This is where the library thrives rather than being used for end-to-end functionality. This feature could be very useful for the Vinyl Player 2.0 in that a more powerful deep learning library such can be utilized, but can be implemented at a high-level giving more time for non-common case components of the system.

3.11.2.5 Machine Learning Library Comparison

Due to the fact that Theano will no longer be developed on and will be unsupported within a year timespan it must be the first to be withdrawn from consideration. With the candidates that are left for choice of machine learning library that will be used to implement a convolutional neural network for computer vision purposes it appears optimal to go with the industry darling TensorFlow. The main reasoning for this that this library is the most adopted by a wide margin and the help that an active community coupled with good documentation is an a great asset. The fact that Torch requires the studying of Lua also detriments its case for being the library of choice, while Keras still could very well be used on top of TensorFlow to simplify development.

3.12 Online Database

Once a new vinyl record has been correctly identified, the next step in the software system of the Vinyl Player 2.0 is to obtain data about the record that is seen in most modern day music players including song titles & lengths, cover art, year of release, etc. This will be done by querying an online database using a REST (Representational State Transfer) request from the server, which will subsequently transfer the data to the android application.

3.12.1 Last.fm

Started in the UK in 2002, the purpose of Last.fm is to track the music that a user listens too and subsequently act as a recommender system based on their musical taste. The database contains around 12 million individual audio tracks and is updated both via the Last.fm staff and through data transferred from the music player the user is listening to music through [17]. The latter method of adding songs and albums to the Last.fm database is worrying as it relies on the user having their digital copy of the music tagged correctly. This is not much a concern for most users, creating the potential issue of displaying poorly tagged content in the application.

The usage of the Last.fm API is limited to those that have an API account, which can be obtained simply by filling out a form. There are methods to obtain information regarding a user, artist, track, etc. but the only real area of concern for the Vinyl Player 2.0 is the albums. This is accomplished with the "album.getInfo" method that has parameters for the artist name and album name and returns an information about the album in a XML or JSON format. Some of the information is the list of tracks in the album with a name and duration, in seconds, for each track. Ultimately this API is a simple way to pull information about records identified by the computer vision system, but is susceptible to bad data.

3.12.2 Discogs

Created in 2000 as a hobby project Discogs, short for discographies, is one of the most extensive music databases to exist with over 9 million recordings and 150 million individual tracks [19]. This database is used as the backbone for an online marketplace for physical audio recordings of all formats. A majority of the contributions to the database come from and are rated for accuracy by a large community on the website, providing greater assurance for the authenticity of the information that is seen. Discogs' API has clients for Python, Ruby, PHP and Node.js and also utilizes REST requests for pulling information.

The focus that Discogs has on physical audio recordings is a quintessential feature for the application of the Vinyl Player 2.0. The reason for

the power of this focus is due to the fact that there can be multiple vinyl releases for a single album thus introducing a possible element of variance. Records for the same album can have altered track listings, time lengths, cover arts, etc. for each different release of the album and thus using the information from the most popular or first or most recent iteration of an album may give inaccurate results.

Discogs fixes this issue do to the database storing all the different versions of a vinyl record, or any audio recording, and giving easy access to all of them via the API. It is done using a REST GET request for an album's "Master Release Versions", which will return the list of all different releases for a single record in a JSON format.

3.12.3 MusicBrainz

A community-maintained database for music metadata, MusicBrainz was started in 2000 in response to GraceNote's Compact Disc Database (CDDB), which charged users to lookup audio CD information on the internet. MusicBrainz has a considerable size with 1.5 million releases and around 22 million individual tracks [20]. As stated before this database is community run like Discogs, but it appears to be even more stringent on accuracy of content than its competitors, providing style guides for preferred formatting of metadata and more strict community voting.

Development for the Vinyl Player 2.0 using MusicBrainz would go through a XML web service. MusicBrainz also solves the issue of multiple releases that correlate to a single record with the concept of a release group. MusicBrainz also allows for a more in depth search with parameters such as instrument, record label, place,etc.

3.12.4 Online Database Comparison

Since the main concern for the Vinyl Player 2.0 when querying the online database is whether the record requested will be found, the main focus for the selection of database relies heavily on number of entries in the database. This is the reason why Discogs is the prime candidate to be utilized for this application, not to mention that it allows for a more in depth information about a record due to Discogs logging each individual release of a record.

	Last.fm	**Discogs**	MusicBrainz
# of Individual Tracks (millions)	~12	**~150**	~23
Supports Multiple Releases Per Album	No	**Yes**	Yes
Data Format	JSON or	**JSON**	JSON

	XML		

Table 10. Online Database Library Comparison

3.13 Market Research

An additional area that needed to be researched in making the Vinyl Player 2.0 was whether there was an actual demand for the product. Many see analog record players as an obsolete way to consume music. There is, nevertheless, a resurgence of this form of music consumption.

Vinyl record sales have been growing quickly lately; what's more, sales have been growing more and more quickly every year. According to Forbes, record sales are "up about 18% in 2012, 32% in 2013 and 51% in 2014". Record player sales go hand-in-hand with record sales.

3.13.1 Similar Products

As a part of researching whether the Vinyl Player 2.0 is a viable product on the market, it was also necessary to see if a similar product exists. While there are countless brands and models of record players, there are fewer that can play both sides of a record. The main feature that the Vinyl Player 2.0 introduces is the ability to choose any song on the record without needing to move the tonearm manually. No other products were found that offer this feature, but some other record players that are new to the market are described below. Both of the products listed did very well on crowdfunding sites, though they provide less features than the Vinyl Record 2.0.

3.13.1.1 LOVE Turntable

The LOVE turntable is a creative design for a turntable, where the device turns on a stationary record rather than the record turning. The device can be controlled through an app (power, volume, and rotation speed), has built-in speakers, and is portable.

3.13.1.2 RokBlok

Similar to the LOVE Turntable, the RokBlok works on a stationary record. Unlike the LOVE, however, the RokBlok doesn't rotate around the record, but rather rolls radially over the record, slowly moving towards the center. This design makes the RokBlok very portable.

3.14 Application Server

For the application to store user information and work with the music database's API, it will be necessary to set up a server. Whenever the user adds a new record to their catalog, the application will send a request to this server to

see if that record's information is already available in the server's database. If not, the server will send a request to the music database's API, saves the information locally on the server, and forwards that information to the user's application.

In setting up the server, the Framework, database, and hosting tools used will need to be decided. The following section will expound upon these topics.

3.14.1 Server-Side Web Frameworks

Server-side web frameworks make it much easier to setup servers that can respond to web requests, and to scale this servers as its use grows. While it is possible to create a web server without using a framework, it is much more difficult and time consuming. Server-side web frameworks allow direct work with HTTP requests and responses. They also allow for very simple and easy routing of requests to handlers and accessing data in the requests. There are many available frameworks to choose from, and they all have their advantages and disadvantages. The main points that could be taken into consideration when choosing a server-side web framework are ease of use, performance (of both the framework and the language used), caching support, scalability, and security. As the Vinyl Record 2.0 is mainly to be a prototype, scalability and caching support are not as important.

3.14.1.1 Ruby on Rails

The main idea behind Ruby on Rails is "convention over configuration", meaning getting a Rails project up and running is meant to be very easy. Rails is the server-side web framework, and Ruby is the language.

3.14.1.2 ASP.NET

ASP.NET is a framework made by Microsoft for building web services. The ASP.NET framework supports multiple languages, including Visual Basic and C#. The framework is essentially a library of open source tools that can be imported to be used for the project. One disadvantage of ASP.NET is that some of the products that can host web servers do not support ASP.NET.

3.14.1.3 Node.js/Express.js

Node.js is a framework built off of the JavaScript language that allows for quick and easy setup of web servers. Node is made up of many open source libraries, called node modules, that can be imported into the web server. One of those node modules, express.js is specifically for creating web servers and handling/routing server requests. Because of the Vinyl Player 2.0 team's experience with Node, this is the framework that was chosen.

3.14.2 Server Database

PostgreSQL is a high speed database and was chosen for this project. The main reason this database was chosen is because members of the group already had experience in creating a database with PostgreSQL.

3.14.3 Server Hosting

Once the web server has been created, it will need to be hosted. This server will be hosted on Heroku. Heroku is a cloud hosting company that allows for quick and easy deployment of web servers through their CLI (command line interface). Heroku handles all scaling and certification for sites that they host. Heroku works very well with the framework (Node js) and the source control (github) chosen for this project. Also, the free plan allows the server database to contain up to 10,000 rows of data, which is more than enough for the purposes of this project.

3.15 Mobile Development

We have chosen to use our mobile devices (phones/tablets) to control our smart vinyl player. In order to use our mobile devices, we need to create a mobile application that will interface with our smart vinyl player. There currently exists three types of environments on the market today, Android, Apple, and Microsoft. Each of these environments are supported by some of the biggest companies in software & technology and offer unique features that make it contender. We are going to be researching each of these mobile environments and choosing the environment that best suits our project.

3.15.1 Android

Below is a table that outlines the main components and statistics of the Android mobile operating system.

Feature	Android
# of active users	More than 2 billion
Open - sourced?	Yes
Operating System	Linux
Manufacturers support	HTC, Google, Samsung, LG, Sony, etc.
Programming Language	Java
Development Hardware	PC, Mac, Linux

Compatibility Issues w/ other Android devices	Yes

Table 11 : Android Feature

3.15.2 IOS

Below is a table that outlines the main components and statistics of the IOS mobile operating system.

Feature	Apple
# of active users	More than 1 billion
Open - sourced?	No
Operating System	Unix
Manufacturers support	Apple
Programming Language	Swift
Development Hardware	Mac
Compatibility Issues with other Apple Devices	No

Table 12 : IOS Feature

3.15.3 Windows Mobile

Below is a table that outlines the main components and statistics of the Windows Mobile operating system.

Features	Microsoft
# of active users	~ 70 million
Programming Language	C++
Manufacturers Support	Microsoft
Development Hardware	PC, Mac, Linux
Open - Sourced	No
Compatibility Issues with other Windows devices	Yes

Table 13 : Windows Feature

3.15.4 Mobile Operating System Final Comparison

IOS, Android, and Windows Mobile are excellent operating systems for mobile application but from our initial impressions and research, we have decided to go with Android. We have chosen Android for four main reasons; open-sourced, number of active users, programming language, and development hardware.

Developing with an open-sourced software provides lots of flexibility and customizability you don't get with closed-source. We have the freedom to modify the native Android code to fit the needs of our project if we should need to. Open source software also has more support than closed source software. For example, Android is supported by Google and they're responsible for major features and upgrades but developers like us can develop add-ons and other modifications to make Android more useful. Since IOS and Windows Mobile are closed source, we have to rely only on Apple and Microsoft to provide major features and upgrades and must develop within the constraints of the operating system.

The large number of active users means that we can reach more mobile users with our application. The more people we can reach, the better chance our smart vinyl player system sells in the market. Since Android has over 2 billion active users vs. Apple's 1 billion active users vs. Microsoft measly 70 million active users, we predict that we would be able to sell more smart vinyl player systems with an Android application.

Android supports the Java programming language vs. Apple's support for Swift programming language vs. Microsoft's support for C++. The reason why we prefer Android's Java programming language because we are more situated and comfortable with that language. We have been taught the Java language by our professors and understand the syntax and inner workings of Java. We don't have to spend precious time to learn and understand a new language, we can just begin developing our application as soon as possible. Microsoft C++ programming language is similar to Java and would make it the second easiest language to learn but requires some time to learn the syntax and other nuances of the language. If we were to learn Apple's Swift language, we would have to set aside a good amount of time to learn the syntax of Swift and how it operates.

Android as well as Windows Mobile allows developers to use any kind of machine (PC, Mac, Linux) to develop Android/Windows applications. Apple requires a Mac computer in order to develop IOS applications which becomes a hinderance. Since our team consists of only PC users, it would be difficult to develop an application without a Mac Computer.

3.16 Version Control

One of the biggest and most important tools in the toolbelt of the software developer is version control software. Version control software hosts a central

repository where the developer can store and edit developmental code. Version control software also serves as a collaboration tool that makes it easier to develop and share code with other teammates. Version control software can track all changes and modifications that occur with the software as well as revert your code base to a previous point in time if your updated code breaks. We are going to be comparing four popular version control software and determine the best one for our project.

3.16.1 Version Control Table Comparison

Below you will find a table that compares features of different version control software from different companies. The features that are being compared are important features that we are looking for as well as features that pertain to our specifications outlined in the “house of quality.”

GitHub is one of the most popular version control softwares that exist on the market. Professors, major companies, and students use this version control software for their software projects. They come highly recommended for their ease of use, excellent documentation, and safety procedures (prevent accidental deletion or overwrite of code).

Microsoft is one of the biggest software companies in the world today. They are responsible for building some of the most well known products (windows operating system, xbox, and Cortana). These projects would most likely have failed without the use of their version control software, Team Foundation Version Control (TFVC).

HelixTeamHub made by Perforce is a lesser known brand but not inferior to other version control software. This software more/less matches the caliber of major version control software.

Amazon Web Services is one of the biggest cloud services that exists today. One of their biggests services is CodeCommit. It’s not known to be the best version control software but we will examine the basic features of CodeCommit in the table below and how it matches to other version control software.

Features	**GitHub**	TFVC	HelixTeamHub	CodeCommit
Costs	**Free**	Free	Free	Free
Company	**GitHub**	Microsoft	Perforce	Amazon

GUI	**Yes**	Yes	Yes	Yes
Code Review	**Yes**	Yes	Yes	Yes
Agile Issue Tracking	**Yes**	Yes	Yes	Yes
Third Party Plugins	**Yes**	Yes	Yes	Limited
IDE Integration	**Yes**	Yes	Yes	Yes
Cloud Storage	**1GB**	Paid	1 GB	50 GB
# of users	**unlimited**	Up to 5	Up to 5	Up to 5
Multi- Level Permissions	**Yes**	Yes	Yes	Yes
OS Support	**Windows, Mac**	Windows, Mac	Windows, Linux, Unix, Mac	Windows, Linux, Unix, Mac
Transfer Protocol	**SSH & HTTP**	SSH & HTTP	SSH & HTTP	SSH & HTTP
Backups	**Yes**	Yes	Yes	Yes

Table 14 : Version Control Comparison

3.16.2 Version Control Final Table Comparison

The major source control softwares that are available on the market are very similar to one another. They all have the major features one would expect with a few minor differences. These differences include GUI design, configuration, ease of use, etc. We have chosen GitHub as our version control software. All of the other types of source control software are comparable to GitHub but we have ultimately chose this software due to familiarity. We already have experience with this software and have been previously installed on our computers. This reduces setup and configuration times and allows us to begin developing as soon as possible.

3.17 Android Studio

Before we can begin developing an Android application, we must download the Android Studio Integrated Development Environment (IDE) and the Android Software Development Kit (SDK). The Android Studio IDE comes pre-configured out of the box with no extra configuration required. Since Android Studio is based on the IntelliJ IDEA, an IDE that we have become familiar with, we can begin development right away with no down time.

3.17.1 Intelligent Code Editor

Android Studio provides an intelligent code editor that is able to predict and auto populate code as the user writes. It is able to reference different classes, methods, fields, and keywords within your project and offer relevant suggestions as you type. The user can focus more on the logic and methodology instead of researching the Android library and syntax. In the long run, the user will have an increased in productivity with minimal errors.

3.17.2 Emulator

Android Studio offers the Android Emulator, a tool that simulates your mobile application on any android device (phone, tablet, wearables, and TV). It provides almost all of the capabilities of the android device without having the need to buy one. The emulator allows the user to conduct GUI tests to make sure the mobile application responds and resizes correctly with each mobile device. The emulator also allows the user to verify that the device hardware can interface with the mobile application like storing data onto the phone or SD card. It goes one step further and simulate real world events like receiving phone calls/text messages and throttling network usage. Unfortunately the emulator will only allow us to test the stability and performance of our mobile application, none of the peripherals like the Bluetooth or Wi-Fi (supported on API level 25 system images).

3.17.3 Testing

Android offers multiple options to test your mobile application for any bugs, performance issues, or anything that affects the user experience. Android Studio allows the tester to setup JUnit tests to verify your mobile device performs as it was designed. For example, if your program was given an input of 10 random numbers and outputs the 10 numbers from least to greatest. The JUnit test would input 10 test numbers and check if the program sorted it out correctly. Android Studio also supports third party testing frameworks like Mockito (test API calls) and Espresso/UI Automator (GUI tests) to test your android application for

any bugs or issues. The point of testing your Android application in the early stages of development because it saves you time and money if you can find the bug/issues early on. In the later stages, you discover a bug/issue, it becomes more costly to refactor, test, and integrate the patch back into the software system.

There are two ways to test the GUI of the Android application. One, we can manually test the GUI on the emulator or install our Android application onto an Android compatible phone. We can test if the GUI is responsive and responds correctly to the actions performed and redirects to the correct pages depending on the user's input. This technique is good when you are only testing your Android application periodically and only a handful of GUI items. However, this can become problematic depending on the size of our project. Based on the scope of our project, we are planning to test multiple GUI items and frequently to make sure nothing breaks within our application. The manual technique can be time consuming and inefficient. Luckily, Android Studio provides a "Testing Support Library" that provides a set of APIs (JUnit 4 and functional UI tests) and tools (Espresso). We can create a suite of tests to analyze the functionality of our mobile application and confirm our GUI operates as it was designed. We wouldn't want our mobile application to redirect to the wrong page or have a button perform the wrong action. We also want to test the responsiveness as well as the robustness of our GUI to ensure that the user experience is not degraded and any unexpected errors or crashes resulting from the GUI. We can also save time and be efficient by utilizing our suite of tests. Espresso allows us to automatically execute a suite of common GUI tests every time we make any alteration to the mobile application. This ensures that the GUI runs as expected all the time and when something breaks, we are notified of the issue and we can quickly patch the problem. Best of all, this saves us time from manually testing the GUI, giving us more time for development.

The next stage in testing is to test the inner logic and external callouts of our mobile application. We need to make sure our mobile application can provide the correct output/results given a specific input by the user or the system. We also want to make sure that our mobile application can interface with external sources like our smart vinyl player and databases, be able to communicate and transfer information. Some of the major components (computer vision, database requests, and time calculations) are the foundation of our smart vinyl player. Each component must operate correctly, separately, and seamlessly work together in unison. A suite of JUnit tests will test the inner logic of our mobile application by pulling information from our local & remote database and process the given information and our tests will verify the output. We are going to write tests that will test if the application can correctly identify and pull the information from the vinyl label. Lastly, we are going to use "Mockito" to test if the information from the mobile application can be transferred to our local database and to the vinyl player. This all can be done manually but it can be repetitive and time wasting but these tools will streamline the development process and make it easier to identify any bugs/issues.

3.17.4 Layout Editor

Android Studio provides a native tool, Layout Editor, that allows the user to create a graphical user interface by clicking and dragging widgets onto the canvas. The user doesn't have to waste time writing the XML code by hand, the Layout Editor does it automatically. Once you are finished with your design in mind, you can view your layout on other Android devices to see how it would look and if it resizes properly. This makes it easier to prototype multiple designs and decide the most fitting design for your mobile application.

3.17.5 GitHub Integration

Android Studio allows third party software to be integrated into their software. One of the most useful and necessary plugins we require is GitHub, ourc chosen version control software. It provides a seamless and automatic update to/from the GitHub repository. Any changes made on the GitHub will update my project to reflect the new changes and vice versa. This prevents loss of time due to code conflicts or compatibility issues.

3.17.6 Developer Workflow

As we begin to develop our mobile application, we require some general knowledge on how to approach a software development project. Android has developed the "Developer Workflow Basics" to guide us to build a robust and efficient mobile application. Android has outline the main processes; Setup, Write, Build & Run, Iterate, and Publish.

The first stage "Setup," we must setup our workstation (computer) before we start developing the Android application. We need to install the Android Studio integrated development environment (IDE) and all of it's dependencies (software development kit, plugins, etc.). Once we have configured our IDE, we can create a new software project which we will name "SmartVinylPlayer."

The second stage "Write," we can begin developing our code. Using the tools mentioned before, intelligent code editor and layout editor, we can create the graphical user interface (GUI) and the code to support the GUI. When we first create the Android application, the Android Studio IDE sets up the basic foundation of every Android application. We don't have to waste time setting up the project settings, we can begin programming our project right away. With the intelligent editor and layout editor, we can develop our application much faster, giving us more time to test and debug in the end.

The third stage "Build & Run," is to build (compile) our software. Once we build the software without any errors or issues, we can begin loading the mobile

application onto an emulator or mobile device. Although it would be more preferred to load your mobile application onto an emulator to save time. Once we have successfully loaded the application onto the emulator, it is time to test it.

The fourth stage “Debug, Profile, and Test,” our mobile application. We are going to test the accuracy and efficiency of our mobile application to verify that it fits our requirements. We also will be testing the application for any bugs/issues. We want our application to run smoothly without any faults or errors. As we begin testing, any bugs/issues that we encounter will be dealt with immediately. Regression testing is extremely important because as we continue to find and fix bugs, we need to make sure that our fixes aren’t affecting or breaking other parts of the code. We would refer to regression testing techniques, creating a suite of JUnit and UI tests to check our software after each modification. This also saves us time from having to manually execute these tests as well, giving us more time to develop.

The last stage “Publish,” would mark the end of our software development. Once our application has passed our tests, we can have our application available on the Android app store for people who purchased our product.

3.17.7 Camera Integration

One of the major features of the smart vinyl player would be to take a picture of the vinyl player label and the mobile application would be able to identify the album name, date, etc. Our mobile application needs to be able to interface with a camera and relay the picture taken from the camera to our computer vision/ machine learning program. Android framework provides API support for a variety of cameras and camera features for mobile devices (Samsung, LG, Google, etc.). Some things to consider as we develop our application, we are going to declare that the mobile device has a camera inside the “manifest declarations.” We believe the camera will be a good addition to our mobile application and enhance the user experience. We also have the option to use existing camera apps or create our own specialized version. For the purpose of our project, we will most likely be using existing camera apps to save time and effort. We don’t need any special features with our camera app, we only need a clear well-defined picture. Any pictures taken within our app will be accessible to other applications on the user’s phone (gallery, facebook, third-party applications). The pictures need to be accessible by our machine learning/ computer vision program which will receive and process the given images and relay the information back to our android application.

The class, Intent, will allow us to call upon an existing camera application to take quick pictures. The existing camera application has been developed and tested by other developers, we can be sure we won’t have any compatibility issues implementing it in our program. This will save us time and

effort from developing our own camera application. The process to take a photo is request camera feature > take photo with app > get thumbnail > save photo > add to gallery > (optional) decode scaled images.

The first thing we want to do is set a requirement that all mobile devices that download our app should have a camera. We will add this line (uses-feature android:name="android.hardware.camera" android:required="true") to our application manifest. We believe the camera is essential to the user experience. However, we might decide to make the camera optional and the user would be able to operate the application without using the camera by setting the “android:required = false.” We could make the vinyl scanning option available to users with a camera with this statement, “hasSystemFeature(PackageManager.FEATURE_CAMERA).”

The next thing would be to create an “Intent” object to call an external activity (camera application). The external activity will execute and process the image resulting from the external activity.

The next thing would be to retrieve the photo from the camera. The Android camera would encodes the photo in a format as small as BITMAP. Once the image has been encoded in a specific format, the ImageView object will display the image in the specified format.

The last thing would be to store the encoded image. We have two options, we can store the image in a location that is accessible by other applications or have the image only accessible to the pertaining application. To save the image in a public location where third party applications like the gallery, Facebook, etc., you would call the “getExternalStoragePublicDirectory(DIRECTORY_PICTURES).” If you would like the image available only to the specific application, you would call “ getExternalFilesDir()” method. The last thing would be to invoke a method to provide a name for that file. The method should have a way to give the file a unique name. To make the image discoverable by the Android gallery and third - party applications, you would invoke the system’s media scanner to add the new image to your Media Provider’s database.

3.17.8 Mathematical calculations

We are going to be using the powerful hardware on our mobile device to do all of the necessary calculations. We don’t have to buy expensive processors or equipment to supplement our processing needs or worry about memory/power constraints. We can just rely on existing resources (mobile device) to process any and all calculations.

One of our main features, being able to drop the needle in the correct spot, requires us to do some calculations in order to achieve this. We need to take into account a few variables, size of the vinyl record, the number of grooves on the record, how many songs are on the vinyl record, and how long are each song. This information will allow us to determine how far we need to bring the needle from a predefined place.

For example, if we analyze a 12 inch vinyl record (4 inches belong to the label), that leaves us 8 inches of grooves where the music is stored. If we were to assume each record have about four songs and assume the four songs take up all 8 inches of vinyl space, we can calculate how far from the edge we need to go if we were to pick a specific song. If each song was 2 minutes long each, that means each song would take about 2 inches (radial distance) of the vinyl record. If we want to pick the third song, we would move the needle 4 inches (radial distance) to start the third track.

We need to take into account that not all songs will be uniform or that each vinyl record may have more or less than 4 songs. Our solution would be to scan the vinyl label to determine the number of songs on the vinyl record as well as find the duration of each song. Once we know the information we can calculate the radial distance for each track and relay that information to the microcontroller on or smart vinyl player to execute.

3.17.9 Android Operating Systems

The Android mobile operating system has many generations since its official inception in 2008. Today, the mobile operating system is developed and maintained by Google. With each iteration of Android, Google continues to provide more interactive features and updates. The cause for software improvement comes partly from the advanced hardware capabilities of the mobile market. Other reasons could include security patches, virtual reality, support for other mobile devices, etc. We have chosen to work with Android Nougat because we believe the latest and greatest Android operating system provides all the necessary functionality we require as well as the latest software updates to ensure speed and efficiency.

3.17.10 Android Nougat

Some of the notable upgrades of Android Nougat is increased performance, battery & data conservation, and privacy & Security. These features are the foundation of any mobile operating system and must be continuously upgraded. Some of our goals outlined in the “House of Quality” benefit from these improvements.

Android Nougat introduced a new JIT compiler. This means as the developer (us) writes the code, the compiler will work in the background to optimize our written code. This will increase runtime performance, faster, and reduce the size of our software. Memory management is important for performance, to reduce application crashes and faults, and power consumption.

Android also provides a battery saving feature “Doze,” which keeps your device in low power usage as you move around your home. We want to limit the amount of power consumed from a lit screen or while using the bluetooth capability. We only want to power the basic necessities. Android also provides a “Data Saver” capability that limits the apps in the background from accessing cell

data. For example, our mobile application will be using cell data to make requests to external database to pull information about vinyl albums. We don't want the mobile application to continue making requests to database in the background or constantly retrieve information if it's not required.

Privacy and security is always a top priority when developing any software. Android provides some key features that keeps the identity of the user hidden as well as his content. Some new features include seamless software updates, file-based encryption, and scoped folder access. Any mobile application would benefit from constant software patches and upgrades to reduce any security risks. Any security risk that goes unpatched and unnoticed can be problematic in the future. For example, if a hacker was to exploit a security risk that would have been prevented if the user had updated, they would have access to the user's phone and personal vinyl record collection. Android also introduced file-based encryption to encrypt each and individual file for better isolation and protection. This means that each file created by our application (vinyl records and user records) would be encrypted and if someone wanted to gain your entire vinyl record collection, they would have to decrypt each and every file. Scoped folder access allows the user/developer to provide access to specific folders instead of all the memory. This means we can limit access to our vinyl records and user records to be used by our application only. No other application would have access to our records, keeping your information private.

3.17.11 Programming Flexibility : C/C++ Code

Android studio includes an option to include C and C++ code in your Android project. This can be particularly useful because other developers might be more well rehearsed in these languages instead of Java or Kotlin. This will allow us take advantage of the benefits of C/C++ code and combine them with the Java code to make our code much better. We only need to add the C/C++ code into the cpp directory of the project, and when we build the project, the code will be compiled into a library that the Gradle can pack with the Android Application Package (APK). The Java/Kotlin code can just call those C/C++ functions using the Java Native Interface (JNI). When creating native libraries (C/C++ code), it is recommended we use CMake, a cross-platform and open-software to handling the build process.

3.17.11.2 Enable C/C++ into Android Program

To add C/C++ code, also known as native code, we can add/create native code and import it into the Android Studio project. Second, we need to configure CMake to build the native code into a library that the JNI can use. The last thing would be to configure the Gradle (add path to CMake or ndk-build script file) to import the source code and package the native library into the APK. The tools we need are the Android Native Development kit (NDK), CMake (version 3.7+), and Android Studio debugger for native code (LLDB).

3.17.12 Java Native Interface : JavaVM & JNIEnv

A java program that defines the procedure for interacting with native code. It has the ability to load code from dynamic shared libraries in the most efficient way. JNI use two data structures “JavaVM” and “JNIEnv” which are basically pointers to function tables. JavaVM provides an interface to invocate functions, allowing the developer to create and destroy a JavaVM. JNIEnv provides most of the JNI functions and it is the first argument in the native functions. JNIEnv is used as thread-local storage which prevents threads from sharing the JNIEnv. One way to get the JNIEnv, the user would get the JavaVM and use the “GetEnv” method to find the thread’s JNIEnv. Please note that C and C++ have different JNIEnv and JavaVM declarations which means the “jni.h” includes different typedefs for C and C++. Convention says to avoid including JNIEnv arguments in header files to avoid unnecessary configuration or development work.

3.17.12.2 JNI: jclass, jmethodID, jfieldID

To access an object field from native code, you would call “FindClass” to get the class object reference, “GetFieldID” to find the field ID for a field, and get the contents by calling the “GetFooField.” Calling a method follows the same procedure, get the class object reference then the method ID (pointers to runtime data structures). All the references and IDs are valid for as long as the class is loaded, as long as all the classes associated with a ClassLoader are being used. To increase performance it is wise to cache some fields and IDs by using a “nativeClassInit” method in the native code. We will invoke the method once when the class gets initialized and keep invoking the method every time a new class instance is created.

3.17.12.3 JNI: Local & Global References

Arguments that are passed to a native method and objects returned by the JNI function are known as a “local reference.” Local references exist only during the execution of a native method in the current thread. After the execution of said method, the object may continue to exist but will no longer be accessible. This rule applies to all sub-classes of “jobject”, “jclass”, “jstring”, and “jarray.” To preserve references outside the native function/method call, you call the functions “NewGlobalRef” and “NewWeakGlobalRef.” The functions will take the local reference and transform the reference into a global reference and will be valid until you call the “DeleteGlobalRef” function. One thing to note, references to the same object may contain different values and to check if they are the same, you would use the “IsSameObject” function (never ==). We shouldn’t assume object references are constant or unique and because of this we should avoid using “jobject” values as keys. Developers should be mindful when creating

many local references to preserve precious resources (up to 16 local references). If we require more than 16 local references, we should reserve more by calling “EnsureLocalCapcity” or “PushLocalFram” or to delete local references as soon as we’re done with them. You can free local references by making a function call to “DeleteLocalRef.” If we don’t call the function, JNI will automatically delete local references but we have no control on when it will be deleted. On a special note, if you use “AttachCurrentThread” method to attach a native thread, JNI will not delete local references until the thread has been detached. The job of deleting local references belong to the developer instead of the JNI. Since jfieldIDS and jmethodIDS are opaque types (not object references), they cannot be passed to the “NewGlobalRef”.“GetStringUTFChars” and “GetByteArrayElements” return raw data pointers, not objects references, meaning they can’t be passed to “NewGlobalRef” as well.

3.17.12.4 JNI: Exceptions

It is advised that you don’t call JNI functions while handling an exception. There are some JNI functions that are acceptable which include but not limited to DeleteGlobalRef, DeleteLocalRef, ExceptionCheck, ExceptionClear, PushLocalFram, ReleaseStringChars, etc. You should check for exceptions when making JNI calls to ensure the return value is valid, unless the the return value is obvious. Please not Android doesn’t support C++ exceptions. JNI handles exceptions by using “Throw” and “ThrowNew” instructions to set an exception pointer in the current thread. Once the program returns from the native method call, it will handle the exception accordingly. However, the developer can directly falg the exceptions by enacting the “ExceptionCheck” and “ExceptionOccured” calls and once the program has dealt with the exception, the developer would call the “ExceptionClear” to clear the exception. If you need to log the exception message, you need to find the “Throwable” class and look up and call the “getMessage ‘()Some/Foo/Exception;’” and print the return value from the function.

3.17.12.5 JNI: Extended Checking

Android offers a mode called CheckJNI which converts the JavaVM and JNIEnv function table pointers to tables of functions that perform a series of tests before calling the standard implementation. The purpose of CheckJNI is to search for errors in native code to prevent software crashes.ChckJNIchecks for bad pointers, critical calls, exceptions, references, type safety, arrays, class names, jfieldIDs, jmethodIDs, etc. CheckJNI is enabled by default when using an emulator but if you are using a rooted device, the user would need to input a sequence of commands to enable it. An alternative way would be to set “android:debuggable” attribute in the application’s manifest.

3.17.12.6 JNI: Unsupported Features/Compatibility

All JNI 1.6 features are supported except “DefineClass.” Android cannot process bytecodes or class files, so using binary class data is ineffective. Dynamic lookup of native functions for Android version 2.0 and below requires using explicit registration or moving native methods out of inner classes. Detaching threads using the “pthread_key_create” function and suppressing the “thread must be detached before exit” check isn’t possible for Android 2.0 and below. Versions older than Android 2.2 rejected the use of weak global references and versions older than Android 4.0 required weak global references to be passed to “NewLocalRef”, “NewGlobalRef”, and “DeleteWeakGlobalReg” functions. Later iterations allowed weak global references to be used as an JNI reference. Versions older than Android 4.0 used local references as direct pointers. Newer versions added indirection to support better garbage collectors but hid a lot of JNI bugs that are prevalent on older releases. Local references as direct pointers also made it impossible to implement “GetObjectRefType” correctly. To determine the reference type, the function would look through the weak globals table, the arguments, the locals table, and the global table sequentially. If the direct pointer was found, the function would return the reference type at that particular moment in time and not necessarily its true reference type.

3.17.13 LINT Tool

The LINT tool, introduced in ADT 16 and Tools 16, analyzes the Android project for any bugs. The tool also identifies any problems that affect the structural quality of the code and assigns a severity level and message. The developer can prioritize the issues within the code and fix the problems that have a severe impact. The program can provide optimization suggestions for correctness, security, performance, usability, accessibility, and internalization. The application source files consist of the Android project which includes Java and XML files, icons, and ProGuard configuration files. The lint.xml configuration file is used to specify specific lint checks that we want to include and exclude and the severity of each problem. The lint tool is static code that is tasked with analyzing for structural code problems.

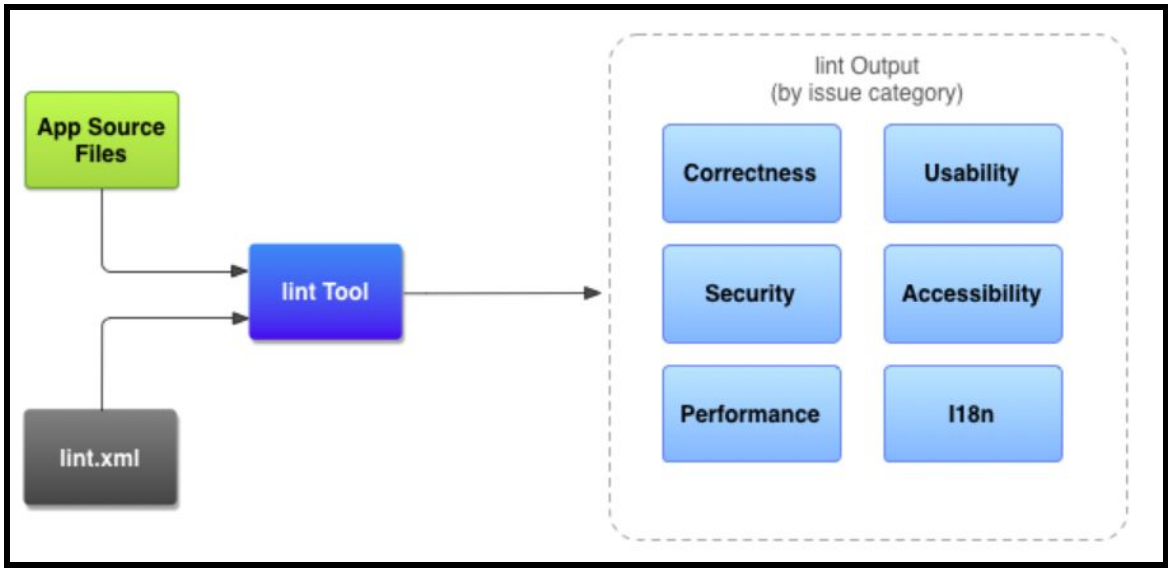


Figure 6: Lint Tool Process

The tool can be accessed numerous ways, command line, standalone tool, Android Studio plugin (ideal way to use the tool), eclipse, etc. The tool can be executed at will through the command prompt or every time we build the Android application in Android studio.

3.17.14 Vector Asset Studio

The tool helps add material icons and import Scalable Vector Graphic (SVG) and Adobe Photoshop Document (PSD) as vector drawable resources. The benefit of using vector drawables instead of bitmap images because it reduces the size of the APK. This happens because the images can be resized for different screen densities without loss of image quality. Android 4.4 and lower doesn't support vector drawables, however, Vector Asset Studio can convert vector drawables into different bitmap sizes for each screen density at build time. For backward-compatibility, Vector Asset Studio can generate raster images of the vector drawable. Depending on the API level, your application will interpret Drawable (java code)/@drawable (xml) objects as vector or raster image. You have the option to only support vector drawables but it requires Android Support Library 23.2 or higher. The developer must modify the "build.gradle" in order to use the "VectorDrawableCompact" class in the support library. Using the method will support VectorDrawable in Android 2.1 and above.

3.17.15 Network Profiler

The Network Profiler displays your application's network activity showing data that has been sent and received as well as the number of connections. This can give insight on how your app transfers data and how you can optimize your code to reduce the bandwidth. The Network Profiler can also be used to look for

frequent spikes in network activity which means the phone is either turning on/off the mobile/WiFI radios frequently or remaining on to handle many short requests. The developer should take note and optimize the application by batching network requests. Ultimately, this will reduce the number of times the mobile/WiFI radios turn on and preserving precious energy resources. The mobile/WiFi radios can also take advantage of low-power mode in the longer gaps between batched network requests. The Network Profiler currently only supports “HttpURLConnection” and “OkHttp” libraries for network connections.

3.17.16 Profiler GPU Rendering

A visual tool that highlights user interface issues like unnecessary rendering work or executing long thread and GPU operations and displays it as a scrolling histogram. It displays the time it takes to render the frames of a UI window relative to 16ms per frame. The tool can analyze the GPU and determine when the GPU is overwhelmed and lagging behind to draw the pixels of each frame. The developer can choose to optimize the code to run intense graphics or lower the video/graphic quality to ease the load. Please note, this tool doesn’t work with any apps that utilize the native development kit (NDK).

You can visually analyze your Android application for signs of overdraw (when the application draws the same pixel more than once within the same frame).

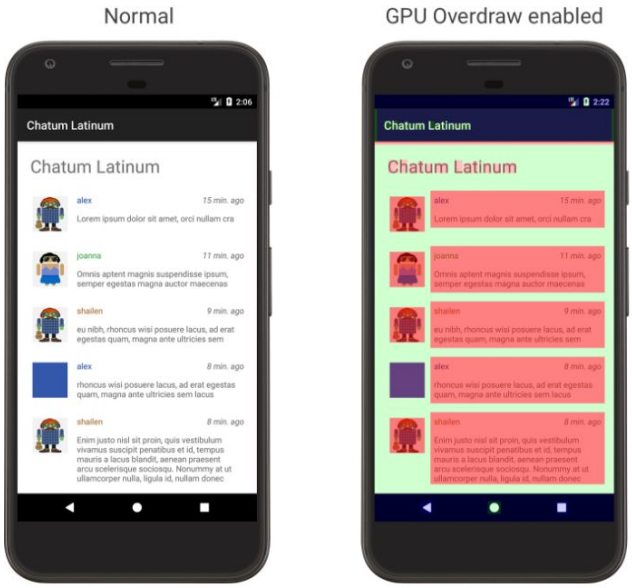


Figure 7: GPU Overdraw

If the UI elements have not been overdrawn, they show the original color. If the UI elements have been overdrawn once, they are highlighted in blue. They are highlighted in green if they’re overdrawn twice. Pink, if they have been overdrawn 3 times and red if they have been overdrawn 4 or more times. The Overdraw tool

is particularly useful because you can identify and correct any GPU issues to preserve energy and GPU resources.

3.17.17 Memory Profiler

The Memory Profiler identifies memory leaks and memory churn (using large amount of processing resources for garbage collection) that can lead to stutters, freezes, and app crashes. Some of the tool features include a realtime graph of the application's memory use, capturing the heap dump, force garbage collections, and track memory allocation. The developer will use this tool to analyze memory allocation patterns and correct any issues. Identify memory leaks by dumping the Java heap over a period of time and analyzing which objects take up memory. Recording memory allocations during regular and intense utilization of the app can show where the program is allocating too many objects or if some of the objects are getting leaked.

3.17.18 CPU Profiler

We can analyze the application's CPU usage and thread activity and record method traces with the CPU Profiler. We want to optimize and minimize CPU usage to provide a fast and smooth user experience as well as preserve energy resources. Being efficient also allows your application to be compatible with newer and older devices as well. We can identify what methods are being executed over a period of time and how much of the CPU resources are used for each method. We can use method traces to identify callers (method that invokes another method) and callees (method that is invoked by another method) to trace and find out what methods call resource-heavy tasks and use the information to optimize the code.

3.17.19 Battery Stats and Battery Historian

The Battery Stats tool collects battery data on the Android enabled device and saves the information onto your local machine using Android Debug Bridge (ADB). Battery Historian will take the information saved on your local machine and create a visualized report that can be displayed in your browser. The report will show where and how process are drawing energy from the battery and identify tasks in the Android application that could be deleted or put to sleep. Please note, to use this tool, it requires the device to be using Android 5.0 and higher with USB Debugging enabled.

3.17.20 Android Device : Samsung Galaxy 8 (Android 7.0)

For demonstration purposes of the mobile application in a real-world environment, we are going to be selecting the Samsung Galaxy 8 smartphone. The smartphone contains the latest updates which equates to more functional features and better graphical user interface. The smartphone also packs 8 cores,

operating at 2.45 GHz, of processing power that makes it easy to handle mathematical problems. It also provides 4GB of RAM and 64GB of storage that gives us plenty of memory for the Android application to operate. Having relaxed constraints, in the beginning, as we develop allows us to focus on attaining a solution. Once we achieve a solution, we can begin to optimize our solution in order for us to achieve some of the qualities outline in the “house of quality.”

3.17.21 Android Device : Samsung Galaxy Tab (Android 4.2)

For demonstration purposes of the mobile application in a real-world environment, we are going to be using the Samsung Galaxy Tablet. The tablet will be used to demonstrate our applications capabilities and compatibility with older Android enabled devices. If we want to reach a broad audience, we need to make our application available to as many phones as possible. The tablet also offers a bigger screen size which makes it easier to display and operate our application. The fact that the tablet is an older mobile device using older software, some of the power saving and efficient features would be disabled. But the overall functionality of our application should continue to work as expected.

3.18 Computer Vision Development

As noted in section 3.x.x, Python programming language is the most natural language to implement the computer vision and deep learning components of the Vinyl Player 2.0 software system, largely due to its simplicity to implement and lack of support for other languages. Initiated in 1991 by Guido van Rossum, Python is a powerful interpreted language that supports multiple programming paradigms and has an emphasis on readability.

3.18.1 Python Core Features

There are a variety of reasons that attribute to the popularity of Python and thus it being a first choice language for the tools that this projects utilizes other than the simplicity of usage and this section will focus on some aspects of the language that sets it apart from other object-oriented approaches and how it may give advantage when applied to the task that the Vinyl Player 2.0 tackles.

3.18.1.1 Interpreted Language

Unlike other programming languages that are supported, to varying extents, by the computer vision and machine learning libraries needed for this task, Python is an interpreted language as opposed to a compiled language. A programming language being interpreted means that a program produced using the language will not be translated into machine level instructions, rather it will be executed directly on a separate program called an interpreter, written in the native language on the platform it's on. An advantage of using a programming

language with an interpreted implementation is that the code tends to be more portable, which is a useful aspect when attempting to deploy on multiple platforms. Another benefit of Python being an interpreted language is that it supports dynamic typing, a core feature of Python that is discussed in section 2.x.x. One downside of this aspect of Python is that compiled languages are typically faster than interpreted ones because it produces the native machine code and allows for optimizations in the native language.

3.18.1.2 High-Level

While some of Python's popular contemporaries such as Java, C#, C++, etc. ease the task of producing instructions that are executed on a machine, Python takes this to another degree with its high-level nature. Python provides complex data structures and has human readable language in mind with its easily understandable syntax, which both allow for faster prototyping and experimentation in the implementation of the Vinyl Player 2.0 software system.

3.18.1.3 Dynamic Typing

Dynamic type checking is the operation of verifying the type of a variable at runtime as opposed to compile time, which is what static type checking does. Python being an interpreted language does not have a compiler, and thus no compile time, which allows it to have this characteristic. This is yet another aspect that gives Python the advantage for developing and prototyping at a quicker rate than statically typed programming languages because where a compiler would not allow an object to have a dynamic type Python does need to worry about it. This also gives an additional dimension of freedom when implementing the system, yet somewhat reduces the reliability of the system compared to statically typed languages.

3.18.2 Development Environment

It is important to have the right tools to create a product with and that is what is achieved through usage of libraries such as OpenCV & TensorFlow, but a component to development of a software system that also holds great weight is the environment that is used to bring these tools and the code together. Below is the environment that will be utilized to create the computer vision portion of the Vinyl Player 2.0 and a description of just a handful of the useful characteristics contained.

3.18.2.1 Anaconda

Anaconda is an open source Python distribution and package manager that is geared towards data science and is popular for applications such as the optical character recognition task that the Vinyl Player 2.0 computer vision

software deals with. An aspect of Anaconda that aids its wide adoption in the data science development community is that the distribution is very comprehensive when it comes to the basic tools needed for large-scale data applications. This means that quintessential Python libraries such as numpy, pandas as well as many others already come pre-loaded in the distribution, which is convenient for the development team.

In addition to the libraries used for Python development, Anaconda also comes packaged with resources such as an IDE for programming and web application for prototyping. The Spyder IDE, formerly known as Spydee, is a lightweight and has an exemplary variable explorer that could be useful for debugging purposes. The web application that allows not only for fast prototyping of the Vinyl Player 2.0 software system , but also gives the opportunity for teams to have live shareable documents is Jupyter Notebook. This is a great asset to have package in Anaconda has it provides for powerful and fast visualizations that will yield room for fast experimentation when creating the system.

3.18.3 Python Integration

Python provides some unique advantages of other programming languages and is the most popular choice for developing with libraries like OpenCV and TensorFlow, although there is an additional step that is produced through the choice of developing with this programming language. Python does not have the ability to directly run on Android, just as any programming language that is not Java or Kotlin doesn't and thus there is a need to link the functionality that is given by the Python script with the Android application that will host it. The best approach to remedy this is to package the Python scripts and bundle it with the Android Application Package (APK), the file format used for mobile Android apps. As this isn't a process related to the core functionality of the Vinyl Player 2.0 it is best to outsource this duty to a preexisting technology.

There isn't a wide range of choice to perform the specific duty of running Python script on an Android platform, so the most viable option is Kivy's Python-for-Android library. Kivy itself is a library for creating multi-platform applications, but since it is not desired to write the totality of the mobile application in Python it is best to solely utilize the sublibrary to give the Android app access to the computer vision and deep learning functionality. An additional issue with trying to run Python scripts in Android is that the mobile platform does not come out-of-box with a Python interpreter so one must be included with the build of the application, which will increase the memory size of the application.

4 Design Constraints and Standards

The following section outlines the constraints that were present and had to be taken into consideration while developing the Vinyl Player 2.0. Additionally, this section details the standards that were met in the design of this project.

4.1 Constraints

There were various design constraints in creating this project, all of which are outlined in this section.

4.1.1 Mechanical

As with many full fledged products there is a number of areas of knowledge that are required for the complete system that may not be a strong suit of the members on the team. This is the case with mechanical work for the Vinyl Player 2.0 and as the team for this project consists of computer and electrical engineers.

The result of this is an attempt to limit the amount of work that happens in this area of engineering as some of the enhancements from mechanical work would be both time consuming due to the time it takes to gain the expertise in the field and not directly correlated to the main use case of the goal of the Vinyl Player 2.0, which is to increase the accessibility of vinyl player usage.

One feature that was contemplated but ultimately not pursued due to it almost purely consisting of mechanical work was a system that could carry and switch out records on the platter to give have more of a library of albums that could be accessed via the Vinyl Player 2.0 as opposed to just one. This is one instance where it idea was interesting but the overwhelming complexity on the mechanical side to have a system that can identify, lift and switch records was one of the main reasons for it to be dismissed.

4.1.2 Time

Of course the amount of time allotted to finish the project is a limiting factor in the process of building the Vinyl Player 2.0, yet this is a factor for most any project that happens in industry or personal development. There are a number of stepping stones to reach that will aid in seeing how the project is moving with the time that the team has to research, design, build and test the Vinyl Player 2.0. Some of these goals that help in assuring the project is progressing smoothly are testing the motors that are used to move the tonearm of the record player, training the computer vision system against a dataset of vinyl record images to obtain a system that can accurately identify album and artist, creating wireframes of the mobile app, etc.

4.1.3 Financial

Another common constraint is the budget that is given to build the project. Although the very generous sponsorship of SoarTech gives the team the added benefit of not needing to source the money needed for the projects themselves, which would almost assuredly be much less, there is still times where a certain functionality must find a different implementation or be scrapped entirely simply because the budget does not support the expense. While this is a constraint, it can also aid the project as the incentivizes the team to come up with more creative solutions to issues or functionalities that are too pricey to implement.

An example of a financial restraint can actually be found by looking back at the selector system that was discussed in the mechanical constraint section (Section 4.1.1). There are systems that perform this exact functionality described in the mentioned section found in certain models of jukeboxes, which would have nearly eliminated the mechanical workload that comes with building a complex system of that nature. The financial constraint again dashed the possibility of this supplemental component is that buying this selector costs over $1000 on the low end, which would far exceed budget and not allow for monetary allocation for any of the other more essential components of the project. This was another reason for not proceeding with this idea.

4.1.4 Environmental

While portable turntables exist, the model that was chosen to build off of for the Vinyl Player 2.0 is not one of them and thus is limited to the same environmental conditions a regular record player that would rest on a shelf. Some conditions that may not be optimal for the system to function is rain, snow, humid weather and excessive heat.

4.2 Standards

There are many standards that are regulated and enforced in power supplies and electronics. Likewise, there are standards that are used in application development that vary from language to language. Standards help protect consumers from unsafe products, ensure portability and reusability of code, protect manufacturers from lawsuits, and allow for many products to coexist without interfering with each other. The following section outlines the many standards that were adhered to in the creation of the Vinyl Record 2.0.

4.2.1 Power Supply Standards

Though the Vinyl Record 2.0 does not require large amounts of power in order to operate, its power supply will need to step down relatively high voltages from wall outlets to provide appropriate power to the rest of the connected devices. When working with large voltages, it is particularly important to strictly

adhere to industry standards to ensure safety in operation for users. Both fire and electric shock can be caused from poor implementation in power supplies.

The main agencies that create electrical safety standards are the International Electrotechnical Commission (IEC) and the International Organization for Standardization (ISO), though there are many more agencies that provide standards and certification that are specific to specific countries or continents. Whenever identifying if a product meets safety standards, it will likely have a code for the standard met, as well as a code to indicate the country in which the product received its certification. There are many different power supply standards, but only some specifically pertain to the Vinyl Player 2.0.

4.2.1.1 IEC60950-1

This standard applies to battery-powered information technology equipment and to machines with a rated voltage that is less than 600 V. Its purpose is to prevent injury and damage to property from electric shock and fires. The standard defines three classes of equipment that relate to how their power supplies isolate the rest of the device from dangerous AC voltages.

- Class 1 – This equipment uses basic insulation and protective earth grounding to protect against electric shock. All conductive parts that could potentially have a hazardous voltage must be connected to a protective earth conductor in case that the basic insulation fails.
- Class 2 – This equipment provides protection using double or reinforced insulation. Because of this, no ground is required.
- Class 3 – This equipment operates from a Safety Extra Low Voltage supply circuit. This means that the circuit inherently protects against electric shock since it is impossible for hazardous voltages to be generated within the equipment.

In understanding the above classifications, it was necessary to understand some important definitions: ***Hazardous Voltage*** is defined as any voltage that exceeds 42.2 V AC peak or 60 V DC without a limited current circuit. ***Extra-Low Voltage*** is a voltage that is in a secondary circuit that is less than 42.2 V AC peak or 60 V DC, with the circuit being separated through basic insulation (at least) from any hazardous voltages. A ***Safety Extra-Low Voltage Circuit*** is a secondary circuit that cannot reach a hazardous voltage through any two accessible ports and must be separated from hazardous voltages by at least two levels of protection (double insulation or basic insulation with an earthed conductive barrier). ***Limited Current Circuits*** are circuits that are designed to ensure that hazardous currents cannot be drawn even in a fault condition, while also maintaining all of the segregation rules of Safety Extra-Low Voltage Circuits. ***Protective Earth Conductors*** connect exposed parts of the circuit and helps protect from electric shock by bringing these exposed parts of the circuit to lower

potential. This standard also defines types of insulations that should be used to separate components from hazardous voltages.

- Operation/Functional Insulation is the minimum insulation for equipment to function, but does not protect against electric shock.
- Basic Insulation is insulation that is connected to live parts and adds basic protection against electric shock.
- Supplementary Insulation is insulation added in addition to basic insulation to give extra protection against electric shock in the case that the basic insulation fails.
- Double insulation uses both basic and supplementary insulation.
- Reinforced Insulation is a single piece of insulation that gives protection against electric shock that is equal to double insulation.

The standard dictates that the minimum insulation requirements are defined as follows. ***Primary to secondary*** defines reinforced insulation with as having dielectric strength of at least 3000 V_{rms}. ***Primary to ground*** defines basic insulation as having a dielectric strength of at least 1500 V_{rms}.

4.2.1.2 UL 60065

UL 60065 is the standard for audio, video and similar electronic apparatuses, so it specifically pertains to the Vinyl Player 2.0. This standard is mainly for applications that are intended for household use and that generate, record, or reproduce audio or video. The main requirements to meet this standard is that the apparatus should not present any danger when used as is intended, both when operating with and without fault. More specifically, there should be protection against hazardous currents flowing through the user's body (also known as electric shock), excessively high temperatures, harmful radiation, the possibility of implosion or explosion, mechanical instability, and the equipment starting a fire.

4.2.2 Soldering Standards

Soldering is the process of joining two leads of a circuit by melting a metal onto the circuit. In this project, there are many times that soldering will be necessary, including to connect the microcontroller pins to the PCB, connecting the motors to the drivers, and connecting the drivers to the PCB. It is important to be very careful when soldering, as the soldering iron (used to melt the filler metal) runs at very high temperatures and because any mistakes in soldering could cause shorts in the circuit.

4.2.3 C Programming Standards

The C Programming language will be used for the programming of the microcontroller, which will control all of the motors in the Vinyl Player 2.0. There

are many standards in the C language. These standards help promote readability, portability and functionality of the code.

4.2.3.1 Naming Conventions

The naming of functions and variables is an extremely important part of any programing. The following table shows different data types and proper naming conventions. When naming function, the function name should clearly indicate what action that function performs. For example, if a function is created to connect to a user's device via bluetooth, an appropriate name for the function would be *connect_via_bluetooth*. Function names should be all lowercase with underscores separating words. For most generic variables, camel casing should be used. Camel casing is where the the first word of the variable name starts lowercase, and each subsequent word starts with an uppercase character. If the variable is referencing a unit, it should include that unit in its name.

When naming a pointer variable, a "*" should be appended to the front of the variable name. When naming a global variable, the name should be prepended with "g_", designating that it is a global variable. Global constants should be in all uppercase letters with an underscore separating each character.

4.2.3.2 Formatting

There are many formatting categories, including brace placement, comments, separation of keywords, character limits per line, and specific function formatting. While there are some differing views on how brackets should be placed, for the Vinyl Player 2.0, brackets will be used to enclose all if, while, and do statements regardless of how many lines are in the statement. Any keywords that are used should have a space between them and the next part of the statement. This helps keep the keyword separated to prevent it from being confused with another variable. A single line of code should not exceed 78 characters, as it would make the line difficult to read, especially on smaller monitors.

In if/else statements, for this project, a new line will be made after the closing brace of a line for the next part of the statement. Also, in an if condition, if a variable is compared to a constant, the constant should always be on the left side of the comparator. A single line of code should only contain one statement. This helps make the code clearer and easier to read.

4.2.3.3 Comments

Comments in code can be helpful in understanding what the code is doing. Nevertheless, the code itself should be very clear and easy to understand without comments, so comments should only be used when necessary – on parts of code that are a bit more difficult to understand. Comments can be helpful in giving an overview of a section of code and of the whole application, in giving an explanation of a specific function, or explaining specific parts of code that may

be confusing. It is also important to document decisions with comments. If there is a specific section of code where other solutions seem possible, it can be important to explain why that specific solution was chosen.

4.3.1 Android Application Guidelines

Application development guidelines and standards, developed by Google's software engineers, are designed to help developers like us to develop high quality applications. They outline the best practices when approaching a given solution and what practices we should avoid as we begin programming. The basic guidelines and standards cover user interface, functionality, compatibility, performance & stability, and security. All of the criterias contribute to the longevity and success of our Android application. If our application fails any of these criteria, users will take notice of the issues and use our application less. So it is imperative we make a high quality application from the start.

4.3.2 Android Application Guidelines: Design Standards

Android provides common user interface design standards, navigation, and notifications. It is recommended that we don't alter any of the system icons and it's corresponding behaviors. If we must alter them, they should resemble the old icon and behavior. Any deviation from the expected icon/behavior can cause confusion among the user which would lead to a bad user experience. The app needs to support Android's "back button" functionality and dismiss any and all dialogs when the user presses the back button. No matter what stage/process within the app the user has executed, they must be able to navigate to the home screen of the device anytime the user presses the home button. Notifications must be stacked into a single notification object, and must only be persistent if they are related to ongoing events such as music playing in the background or phone calls. Application notifications cannot contain any advertisement or unrelated content unless the user has given us permission. This means we shouldn't promote new vinyl records or vinyl players to the user until we request permission from the user. We should only use notifications to alert the user that something has occurred or relaying information from an ongoing event.

4.3.3 Android Application Guidelines: Functionality

The next important criteria we must address is functionality. We must ensure that our Android application provides the functionality we promise to our users within the given permissions. Our application must only require the bare minimum permissions to support the application's core functionality. We should also avoid requesting permission to private data like text messages or contacts and paid services like banking or stock trading. If the mobile device has an SD card, it's preferable to install it on the SD card especially if the size of the

application is larger than 10MB. Audio shouldn't play when the screen has been turned off, locked, or over another application unless this is a feature of your application and should resume when you reopen the application. The application should support both orientations (landscape and portrait) in full mode without any rendering issues. The application shouldn't operate any service in the background unless it's part of the functionality. For example, the application shouldn't maintain the GPS/cell network connection while it is in the background unless it's necessary. The application should be able to restore/resume its previous state after the user leaves the application, locks the screen, or from recent apps and if isn't possible to save the state, notify the user.

4.3.4 Android Application Guidelines: C.P.S

Compatibility, performance, and stability could be arguably the most important criteria of an application. If the application suffers from these issues, the user would become frustrated and possibly forget the application. The application must be stable enough to not crash, fault, freeze, or any unexpected behavior. The application must be able to load quickly or provide some feedback otherwise. While "StrictMode" enabled, the application must be able to perform without raising any red flashes (performance warnings). The application should run the latest software development kit (SDK) without crashing or reducing functionality. The application must be able to support power management features created by Android unless it's absolutely necessary to disregard. Media playback must run without any issues (lag, use and load, pixelated). The application must provide decent quality graphics with no distortion in all multiple settings and display text in an acceptable manner.

4.3.5 Android Application Guidelines: Privacy & Security

In order for our application to be published in the Google Play Store, we must adhere to user data policies. Some data policies include storing private data inside the app's internal storage and accessing any data inside an external storage must be verified. "Intents" must be explicit, use and enforce correct permissions, and verified before use. No sensitive information will be saved to the system or app log. Application components that share data with other apps can be exported, define appropriate permissions, and use "android:protectionLevel="signature."" Our mobile application must declare a network security configuration and send any network traffic through SSL. All libraries, SDKs and dependencies must be updated as soon as possible. We must used cryptographic algorithms and random number generators provided by the Android platform, no custom algorithms.

4.4.1 Android Open Source Project Coding Standards

Android Open Source Project (AOSP) has created a set of standards and rules for Java developers/programmers that want to contribute to the AOSP. These rules aren't necessarily applicable to the average android developer but they are good standards that everyone should abide by to develop an efficient and robust application. These standards help ensure the stability and responsiveness of your application. Future modifications and maintenance of the application would be much easier due to readability and ease of understanding the programmer's intentions when developing the application. The AOSP standards outline the Java language rules, library rules, style rules, and Java test style rules.

4.4.2 AOSP : Java Language Rules

The Java language rules outline standards for dealing with exceptions, finalizers, and imports. When you are writing code that utilizes a "try" and "catch" statement, you should have some way of dealing with the "catch" statement (a.k.a an exception) in case something in your application triggers it. Some acceptable ways of dealing with the exception would be to throw the exception up to the caller of the method. Another way would be to create a new exception that's related to the calling exception. Last but not least, you can handle the exception and have corresponding code to deal with the exception appropriately. One thing you should avoid doing is throw a "RuntimeException" because this will cause the program to crash. You also shouldn't catch generic exceptions because you might catch exceptions that you would never expect like RuntimeExceptions. Generic exceptions also hinder the benefits of exception handling because you aren't able to specifically identify and handle each exception accordingly. Since generic exceptions catch all exceptions, the compiler won't be able to warn the developer. Without any warnings from the compiler, the developer won't know how to handle the exception appropriately or disregard it all together. Generic exceptions should only be used for testing and catching all types of errors for debugging purposes. Alternatives to using generic exceptions would be to use "try" and "catch" statements for each exception, handle IO and parsing exceptions separately with multiple "try" and "catch" statements, or allow the method to throw the specific exception to notify the developer. Since Android doesn't recognize finalizers, the use of finalizers should be avoided. We can achieve the same functionality by using a "close()" method. When importing classes from packages, the developer should explicitly state what Java classes they want to import into the program. This makes it clear to other developers about what Java classes the application is using and makes it easier for maintenance. An exception to this rule can be applied to java standard libraries like "java.util.*" or "java.io.*" and unit test libraries, "junit.framework.*"

4.4.3 AOSP : Java Library Rules

It's ideal to be working with updated libraries and updated programming conventions whenever possible. This increases the longevity of the application and reduces the amount of maintenance in the long run. Creating new software with deprecated libraries is not allowed due to the fact that it would become obsolete in the near future and introduce complications and problems. However, if you are working with older software, old programming conventions or deprecated libraries are okay to use.

4.4.4 AOSP : Java Style Rules

Classes and interface should include copyright information and description of functionality at the top before the package and import statements. Nontrivial public methods should have a description of its use and functionality, intuitive methods don't require comments. You should try to keep methods small and specialized (under 40 lines), anything over should be broken up. Define all of your fields at the top of the file or before the method call. Appropriately assign class access modifiers (private, public, protected, none) to increase readability and maintainability of your code and reduce future errors. Declare variables in the associated method and initialize the variable appropriately. If a variable is initialized with a return value that can throw an exception, you should initialize the variable inside a "try" and "catch" statement. Loop variables should be initialized inside the for loop for readability. Android import statements should be listed first followed by imports from third parties followed by java and javax imports. They should be all alphabetized in their respective groups with a separate line between each group. This standard style increases readability between developers. Naming conventions for fields as follows, non-public and non-static field names should start with m. Any static field names should start with s while other field names should start with lowercase letters. Any public static final fields or constants should be noted in all caps with words separated by underscores. For example Public static final FOO_FIELD = 3.14. Braces shouldn't be placed on their own lines, they should be placed on the same line as the code that uses them. Each line should be limited to 100 characters long. However, there are exceptions to this rule.If a comment line has a command or URL that happens to be longer than 100 characters or import lines. If you are to include any annotations, they should be listed one-per-line in alphabetical order. The three common annotations are @Deprecated, @Override, and @SuppressWarning. @Deprecated discourages the use of old implementation/code and the developer should find an alternative implementation. @Override is used to override methods inherited from superclass. @SuppressWarnings should be used when the implementation is necessary and there is no alternative that would eliminate the warning. They should be followed with a TODO comment stating the condition that cannot be eliminated. Developers should also use ERROR, WARNING, INFORMATIVE, DEBUG, and VERBOSE keywords when logging

information. However, the developer should use these keywords wisely because they can negatively impact the programs performance if too many are used. ERROR should always be used and logged when something fatal happened inside the program. WARNING should always be used and logged to notify the program/user that something serious and unexpected happened. INFORMATIVE is used to note abnormal events in the program that aren't errors and they are always logged as well. DEBUG is used to examine and debug any issues inside the program. You should use these annotations sparingly to avoid any performance issues. VERBOSE should be used for anything else not listed above and will only be logged during debug builds when surrounded in "if (LOCAL_LOGV)" blocks.

4.5.1 Google Style : Android Developer

Developers that follow the Google coding standards are said to be "in Google Style." This means the developer's program strictly follows the Google coding standards (aesthetic standards, conventions, and coding standards) that is universally accepted by other developers and software development tools. These standards ensure that the program written by a specific developer is easily read and understood by other developers.

4.5.2 Google Style: Source File Basics

The source file should have a case-sensitive name that matches with the top-level class with a ".java" extension and encoded in UTF-8. The only special character that is allowed inside a source file is the horizontal space character (0x20) which implies that all other whitespace characters in string and character literals are escape sequences (\b, \t, \n, \f, \r, \", \', and \\). Other non-ASCII characters can be displayed as Unicode characters or Unicode escape (\u221e) but convention discourages the use of Unicode escape outside of string literals or comments.

4.5.3 Google Style: Source File Structure

The structure of a source file should include the License/copyright information followed by package statements followed by import statements followed by one top-level class, all separated by one blank line for each section. Any wildcard imports are discouraged as well as static imports for static nested classes. Static imports are imported with normal imports. Group static imports together in a single block and non-static imports in another block with a blank line separating the two blocks. No blank lines between import statements in the given block and must be in alphabetical order. Methods should be ordered by logic, not by date added, and if there are multiple methods with the same, they should appear sequentially.

4.5.4 Google Style: Formatting

Any “if”, “else”, “for”, “do”, and “while” statements even if the statement has no body or only has one statement. Braces follow the K&R style which means no line break before the opening brace, line break after opening the brace, line break before the closing brace, and include a line break after the closing brace only if the brace terminates a statement or body of a method, constructor, or class. However, empty blocks can choose to follow the K&R style or disregard it by closing the braces immediately except with multi-block statements. Java code must follow the column limit of 100 characters and anything that exceeds the limit must be line-wrapped. However, some exceptions are include long URL or JSNI method reference, package or import statements, or commented command lines. We should include a space between open parenthesis/closing curly brace and any reserved keyword (if, else, catch). A space is required before an open curly brace except if it’s in an annotation or initializing an array. One space should be included before and after an ampersand, vertical pipe, colon, double slash and arrows. This rule doesn’t apply to two colons (::) and dot separator (.) because that’s part of the java syntax. One space should be included after a comma, colon, semicolon, and closing parenthesis. A space needs to be included between the type and the declared variable. You are required to do one variable per declaration except in the header of the for loop. Arrays can be declared as a “block-like construct” but no C-style declarations. Switch block contains multiple statement groups that must include one of the following terminating condition (break, continue, return, or thrown exception). If you don’t use any of these terminating conditions, add a comment to show and explain why the statement group will continue into the next statement group. Each switch block should contain a default statement group (case) but enum types are optional. Annotations that are applied to a class, method, or constructor should appear after the documentation section with one annotation per line. Numeric literals like the “long” use an uppercase suffix to avoid any confusion.

3.5.5 Google Style: Naming

This provides the standards and conventions when creating identifiers (names) in the programs. Common rules that apply to all identifiers are as follows, they should use only ASCII letters, digits, and underscores. However, Google Style discourages the use of prefixes and suffixes like foo_, mFoo, and m_foo. Package names should only be in lower-case and concatenated, no underscores. Class names use UpperCamelCase (first letter of every word is uppercase) and are usually nouns/noun phrases. For example, classes would be “FirstClass” or “MartinClass.” Test classes use the same convention as class names but include “Test” keyword at the end to indicate a test class. Method names are using lowerCamelCase (the first letter of every word starting from the second word needs to be capitalized) and they usually reflect verbs/verb

phrases. For example, methods would be named “sortingTrick” or “calcSum.” Constant variables (immutable values or values that can’t be changed) contain only uppercase letters with each word separated by an underscore and are usually nouns/noun phrases. Non-constant variables (non-immutable values or values that can be changed) are written in the lowerCamelCase format and the name resembles nouns/noun phrases. Parameter names are written using lowerCamelCase and discourages the use of one-character names. Local variables are written in lowerCamelCase and they don’t follow the rules of constants even if they are final and immutable. Variables that are type based (generic) use one of two styles. They can be a single capital letter and possibly be paired with a single number or upperCamelCase name followed by a capital letter. For example, it would be “FooMethodT.”

3.5.6 Google Style: Programming Practices

All methods that override a superclass method, class method that implements an interface method or an interface method that is repurposing a super interface should include the @Override annotation. The only time you shouldn’t have the annotation is when the parent method is marked with an @Deprecated method. A good programming practice would be to handle caught exceptions, don’t avoid. You should either log the exception or rethrow it as an “AssertionError” if it is impossible to handle. The only time a developer could ignore the caught exception if it was handled with a “try” and “catch” block. That way the system would be able to gracefully handle the situation without crashing. When you are referencing a static member (methods, variables, etc.), you should reference it by the class name instead of the class instance variable. For example, if you create an instance variable, FooClass aFooVar = “”, and it has a static method you want to reference, you should always reference the static method by “FooClass.aFooMethod” instead of “aFooVar.aFooMethod.” You shouldn’t override an “Object.finalize” method, leave the job to the java garbage collector to avoid any unforeseen issues.

3.5.7 Google Style: Javadoc

Javadoc can contain information about the program, the developer, what revisions were made, copyright information, and etc. It’s purpose is to give a brief description about the class and the class members. They could be represented in a block format where the first line has “/**” and on the next line “*” and some comment and on the next line ending with “*/”.This all could also be done on one line as well, it’s up to the user to decide. Block tags (@param, @return, @throws, @deprecated) appear with a description that describes the following section. Summary fragments are brief statements (incomplete noun phrase or verb phrase) that detail the method’s return type or warnings. The format for this would be “/**@return the customer ID*/”, not “/** Return the customer ID */”. Javadocs should be used for every public class and public/protected members of

the particular class. You don't have to use javadocs if the class members are self-explanatory. Methods overriding superclass methods don't require javadocs.

5 Project Design

The design of this project is split into two sections: hardware and software. These two following sections will outline how the project will be designed and implemented, both through hardware and software.

5.1 Hardware Design

The hardware designed in this project includes motor drivers, power supplies, PCBs to connect components to the microcontroller, and the actual record player.

5.1.1 Hardware Design Planning

This section details the ideas that were brainstormed for how to implement the actual motor control and two-sided playback of the record. This is particularly difficult because the motors to control the record player arm(s) need to fit inside of the record player's original encasing or additional encasing needed to be made.

5.1.1.1 Design A:

The first design is specified to only play one side of a record. This design uses two different stepper motors to control the pitch and yaw of the record player arm. To clarify, the pitch refers to the angle which corresponds to the vertical position of the arm, while the yaw refers to the angle corresponding to the radial position of the arm in relation to the record. The yaw motor will move the entire arm assembly which will move the arm radially along the record. This is the primary motor that controls which groove will be selected by the device. As such, this motor must be extremely precise and must be able to provide enough torque to move the entire assembly. The pitch motor will control an armrest that the arm lays upon. The armrest is attached to a wheel which rotates up and down slowly vertically. The record player arm lies upon this armrest, allowing the arm to be gently lowered onto the record. Note, the record player arm is not attached to the arm rest. While the armrest is lowered more and more, the armrest continues to spin while the record player arm is stopped by the record. This allows the arm to be lowered without providing excess force to the record, which may ultimately damage it.

It is possible to convert this design in order to play both sides of the record. To do this, another tonearm must be added to play the bottom side of the record. Like the top side, the bottom side tonearm will rest on another motor to

raise and lower it. To ensure adequate pressure is applied to the record, the arm counterweight will be calculated to apply the necessary pressure upward.

5.1.1.2 Design B:

The second design is specified to play both sides of the record. Much like the first design, the second design relies on two different motors. Unlike the first design, the second design moves the cartridge of the record player rather than moving the arm. This is accomplished by attaching the cartridge to a belt which moves it radially along the record. This belt extends beyond the edge of the record and curves under the record. This will allow the cartridge to move, not only along the top of the record, but also along the bottom of the record. This whole assembly is moved up and down vertically by a motor. This motor relies on position controlled feedback to applied the appropriate amount of pressure to reached the desired sound quality. While this design provides the benefit of allowing the user to play both sides of the record, there are several considerable drawbacks.

5.1.1.3 Design C:

Design C is the other option for playing both sides of the record. Rather using a conveyor like Design B, this design uses two separate arms which will play the desired side of the record. Similar to the first design, a stepper motor moving in the yaw direction will move both arms to the desired radial location along the record. Unlike the first design, this will require alteration to the record player base. To play beneath the record, there must be a portion of the record player, beneath the record, hollowed out to allow the arm to move beneath the record. Rather than the armrest mechanism described in Design A, the method to provide the desired amount of pressure to the record will be a position feedback servo. A sensor in either of the heads of the record player arm will detect a precise amount of distance between it and the record. This distance is fed-back into the servo to ensure the distance is consistent. This design has the benefit of playing either side of the record but has significant drawbacks. The biggest drawback is that a large chunk of the record player must be removed to make space for the moving tonearms.

5.1.1.4 Design Comparison:

Design A is the simplest approach, using counterweights to apply the desired pressure to the record. Creating the conveyor for Design B may require too much mechanical experience to be practical for this application given the group members involved. In addition, the conveyor may result in a large decrease in accuracy which is extremely important to meet the accuracy specification requirement. The biggest drawback of Design C is the added

complexity of a position feedback system to apply adequate pressure to the record. Too much pressure will cause scratching to the record and not enough pressure will result in a decrease in audio quality. The position sensors in Design C only measure the distance between the tip of the tone arm and the vinyl so the pressure (the desired value to be held constant) is not directly measured. It is unknown how this could potentially affect the sound quality. These position feedback sensors are also used in Design B so the same argument can be made against that design as well. Should playing the underside of the record substantially negatively impact the sound quality of the audio playback, Design A allows for the simplest design to accomplish one-side-playback. For these reasons, the design chosen for this application will be the simple two-sided approach for Design A.

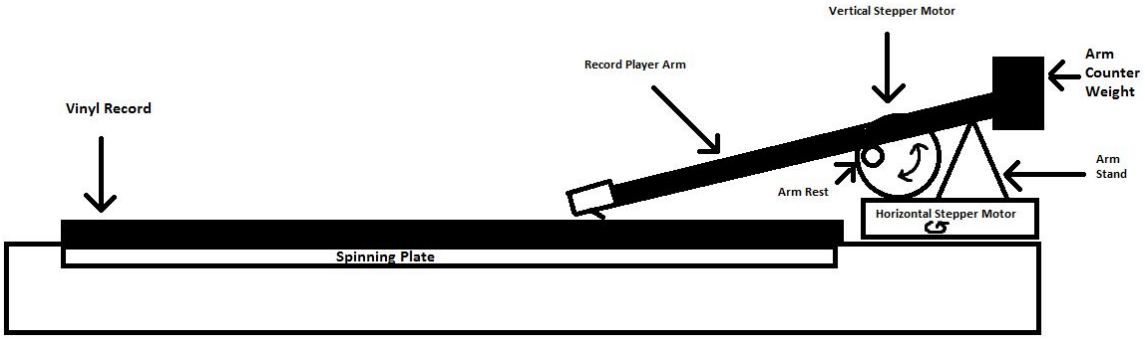


Figure 8. Design A Concept Sketch

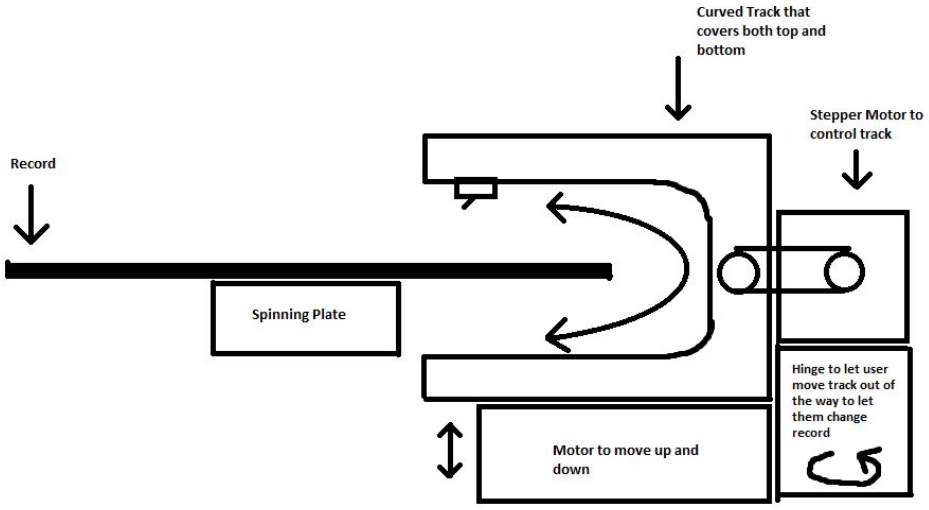


Figure 9. Design B Concept Sketch

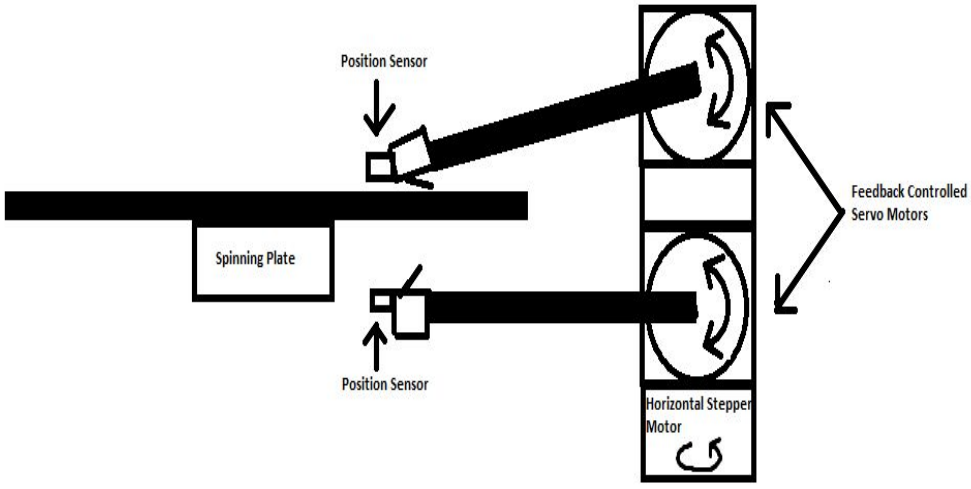


Figure 10. Design C Concept Sketch

5.1.2 Power Supply

One printed circuit board (PCB) will be made to supply power to the entire hardware system. It is important that the power supply PCB is accurate in order to prevent damage to all of the components that will be powered. The following section evaluates the devices that need to be powered and their requirements and what PCB manufacturer will be used for the PCB creation.

5.1.2.1 Power Supply Requirements

The devices that will need to be powered include the record player itself, the microcontroller, three motors (one for horizontal motion and two for vertical motion), motor drivers, and proximity sensors. Each one of these components will have a specific power rating and voltage that they should be run at. Exceeding the power rating of a component could permanently damage the component, and running at a voltage that is two low for the component is likely to affect the component's performance. In designing a power supply, it is also important to understand the initial power input (a wall outlet) and what that source will provide.

	DC Voltage
Microcontroller	6.6 – 12 V
Horizontal Driver	4 V
Vertical Driver	4 V

Vertical Driver	4 V
Horizontal Motor	12 V
Vertical Motor	12 V
Vertical Motor	12 V

Table 15 – Power Supply Requirements

5.1.2.2 Converting AC Power to DC

In the United States, power is supplied from an outlet with a voltage of 120V and a frequency of 60Hz. All of the components that need to be powered are run with DC power, not AC, so it will be necessary to convert the power. Also, the voltage will need to be stepped down to the appropriate levels. A circuit needs to be designed to convert this power. This circuit sections that do the following: transform, rectify, smooth, and regulate.

First, the input signal (from the outlet, 120VAC at 60Hz), needs to be stepped down to a smaller voltage swing. For this, a 5:1 transformer is used first in the circuit. A transformer has two coils, a primary and a secondary. This particular transformer is a 5:1 transformer because the primary coil has 5 rotations for every 1 in the secondary coil. This causes the input voltage to be stepped down by a factor of 5. The output of the transformer stage of the circuit is 24VAC at 60Hz.

Next the signal needs to be rectified. The rectification stage of the circuit essentially ensures that all of the voltage will be positive. This is done using diodes that will prevent the flow of current in a certain direction. For this project, a full wave bridge rectifier is used, which effectively takes the absolute value of the input signal. The full wave bridge rectifier is made up of 4 diodes arranged in a square bridge. The output of the rectification stage has only positive voltages.

After the rectification stage comes smoothing. Though the rectifier keeps the voltage positive, there is still a roughly 15V swing in every cycle of the signal as it moves between 0 and 15 volts. Adding a capacitor after this stage smooths this signal. For this stage, a large capacitor is used. Essentially what happens is as the voltage rises, the capacitor charges up. Then, as the voltage falls, the capacitor slowly discharges. The capacitor has not fully discharged as the input voltage starts to rise again. This causes the dip in the voltage to be much smaller than before the smoothing capacitor. The output of the smoothing phase of the circuit looks nearly like a DC signal, but with small increases and decreases in voltage.

The final stage of converting AC power to DC is to regulate the voltage. A voltage regulator is an integrated circuit component that does just that – regulates voltage. It is rated to output a specific voltage is some voltage that is higher than the output voltage is supplied to the regulator. It is important to choose a regulator and to choose the previous components that will work

together. For example, after the smoothing stage, there could still be a swing in the voltage of about .5 volts. If the smoothing stage had a voltage between 15 and 14.5 volts and the regulator was manufactured to output 14.75 volts, it would fail. The output of this stage is a DC signal.

5.1.2.3 Power Supply Design

The following image shows the schematic diagram of the power supply created. There are three main outputs which, from top to bottom, output 12, 4 and 8 volts.

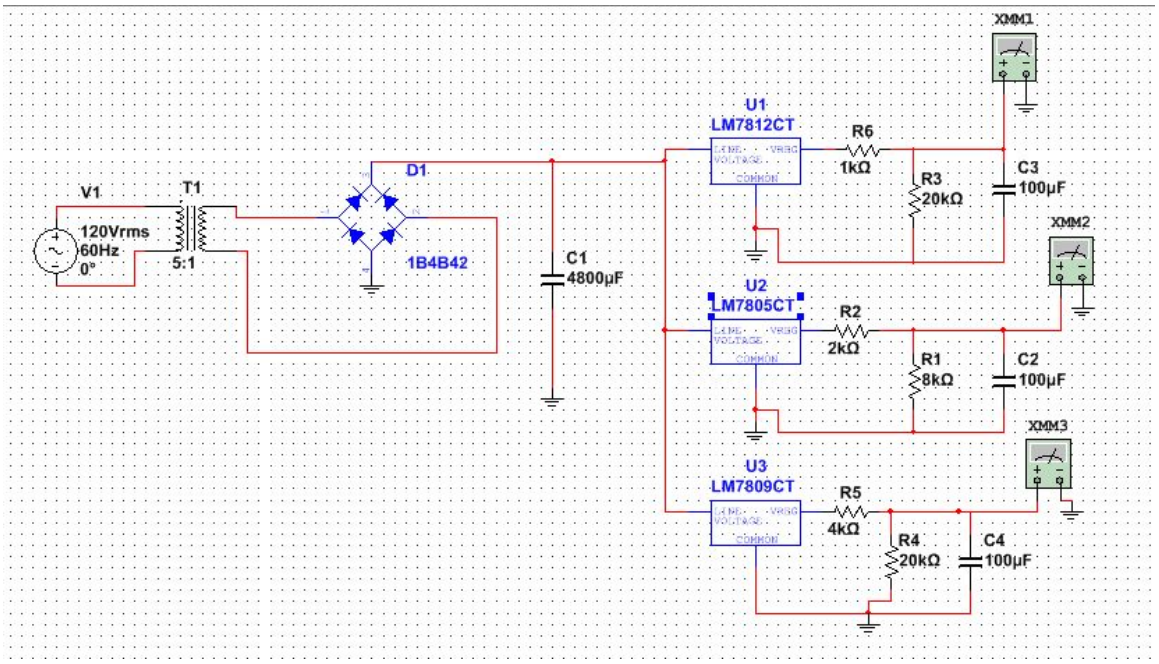


Figure 11 – Power Supply Schematic

5.1.3 Motor Control

The motor is controlled through a motor driver. In the final product, the motor driver would be controlled from one of the GPIO pins on the microcontroller. In a breadboard test however, the driver was DC biased and appropriately and connected to the motor. Then a function generator emulated the output that the microcontroller would product – a pulse signal at about 200Hz. This caused the motor to step at 200 steps per second. Below is a picture of the breadboard setup used to finish phase one of testing.

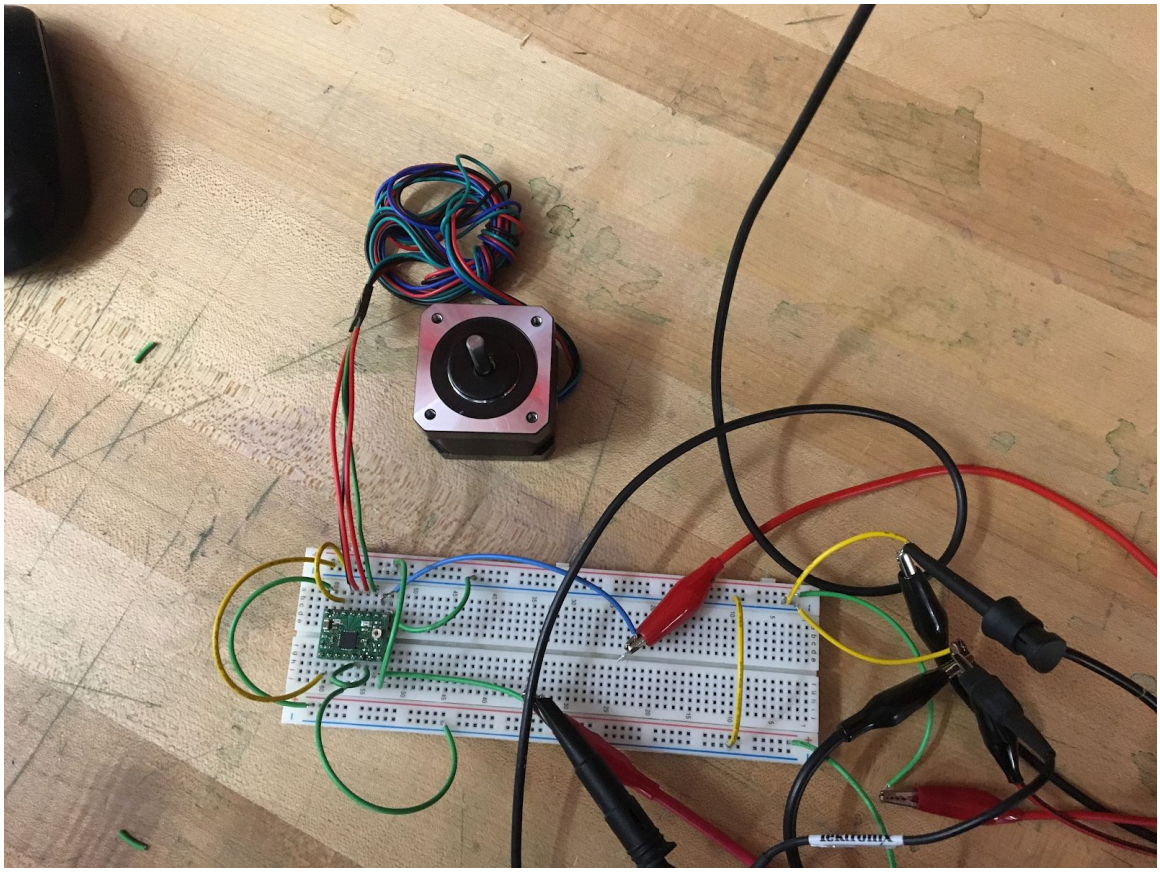


Fig 12. Breadboard Setup Phase 1 Testing

5.1.4 Microprocessor

Once the microprocessor was received, it was necessary to test it to make sure it was functioning as expected. To do this, first it was plugged into a computer through the board's USB port. Upon receipt of the microprocessor, code had already been downloaded onto it to cause some indicator lights built into the board to flash. Once the board was plugged into the computer, it was powered on and the lights flashed as expected. Next, some code was uploaded to the board through the Arduino IDE. Uploading this code was intended to test the boards ability to have code uploaded, to test the output pins on the board, and to test the board's bluetooth capabilities. The code that was uploaded would allow for a bluetooth connection to be made to the board through the board's manufacturer's iPhone app. Once the connection was made and some text was sent to the board, the board would output a voltage to one of its pins. A breadboard with a resistor and an LED was connected to the microprocessor pin to confirm that the board was outputting a voltage as expected once the bluetooth connection was made. Below is an image of the Blend v2 board with the LED circuit connected.

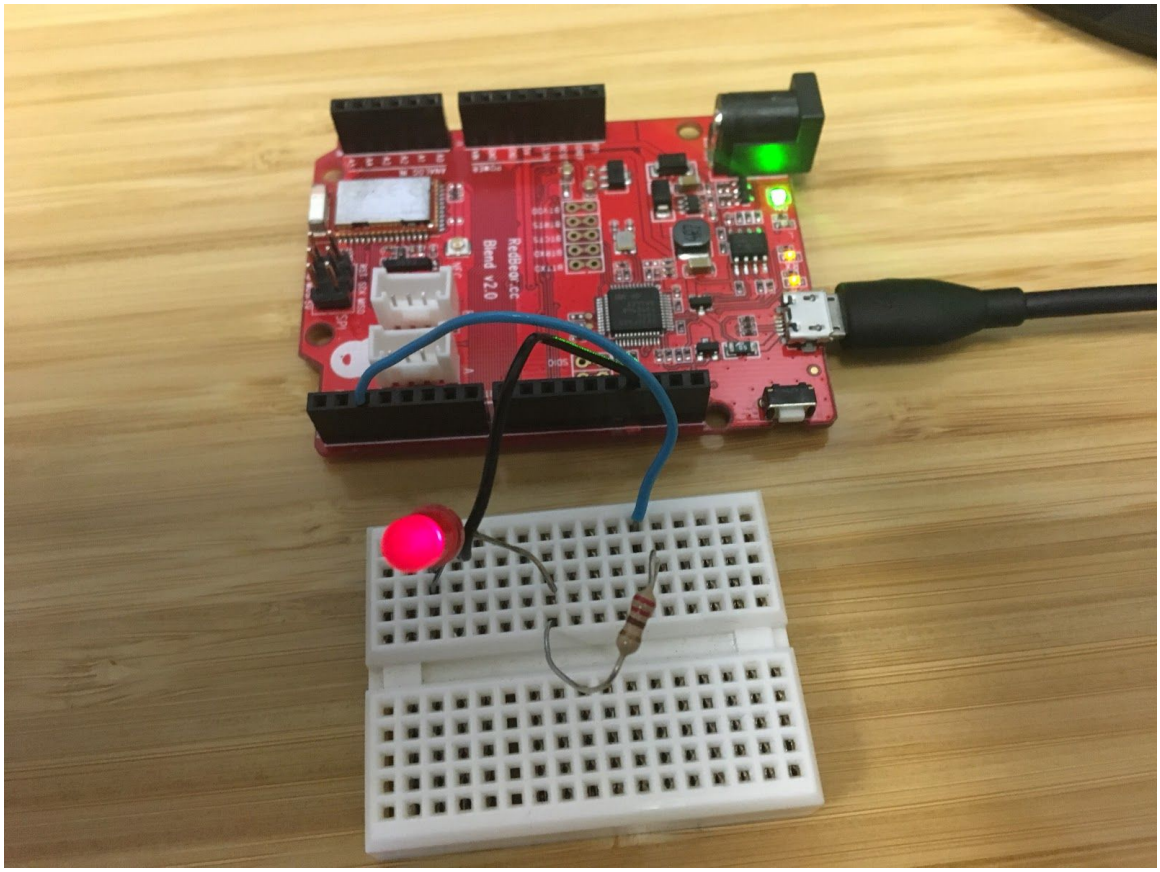


Figure 13 – Blend v2 Board with Attached LED Circuit

5.2 Software Design

This section of the paper will give in-depth descriptions, figures and analyses of the implementation of the various software components contained within the Vinyl Player 2.0 system.

5.2.1 Domain Model

Prior to diving into each individual element of the software, a domain model is constructed through a variety of steps to visualize the conceptual classes in the scope of the problem that the software side deals with. The domain model follows the main success scenario for the system, which is the primary use case that behaves exactly as it is intended.

The main success scenario, with no alternative workflows, of the Vinyl Player 2.0 software system is as follows. The user places a new record on the record player and takes a picture of the record using the phone camera. The system identifies the record and then asks Discogs for more information on the record. The system displays the information that is received from Discogs to the

user. The User selects a song to be played. The system calculates how far to move the needle to play the selected song. The distance is sent from the system to the record player. The record player plays the selected song.

It is important to remember that while the software implementation of the Vinyl Player 2.0 will, at least in part, be object-oriented, the domain model is not a model of the software components itself, rather it provides inspiration for the software components. Another integral aspect of creating a domain model is that it is not a process that is solely done at the inception of the project, but an iterative process that improves the accuracy of the model as the project goes on. That is to say that there will be multiple iterations of the domain model, each becoming slightly more attuned to the real-world problem that the Vinyl Player 2.0 deals with.

5.2.1.1 Initial Domain

This section will express the process of obtaining the initial domain model of the Vinyl Player 2.0 that can then be used as inspiration for the object-oriented implementation of the system. It is important to keep in mind that the domain model is continually reassessed as more knowledge about the problem arises thus improving the representation overtime.

5.2.1.1.1 Visualizing Concepts

The process of identifying an initial list of conceptual classes that effectively illustrate the domain for the problem at hand can be done using two simplistic techniques, a conceptual class category list and identifying noun phrases. Creating a conceptual class category list is the first of the two techniques and is done by using a standard set of conceptual class categories and grouping potential conceptual classes for the problem into these categories. The conceptual class category list for the Vinyl Player 2.0 software system is shown below.

Conceptual Class Category	Examples
physical or tangible objects	*Record, RecordPlayer, Phone*
specifications, designs or descriptions of things	*RecordDescription*
audio	*Album, Song, Artist*
roles of people	*User*
containers of other things	*Phone, RecordPlayer*

things in a container	*Camera, Screen, RecordPlayerController, Motor, Antenna*
abstract noun concepts	*Accuracy, Communication*
events	*Taking picture of Record, Tapping on playback option*
catalog	*RecordCatalog*

Table 16. Conceptual Class Category List

The next useful technique for formulating potential conceptual classes for the domain model is a linguistic analysis of the main success scenario, which consists of identifying nouns and noun phrases in the description of the ideal use case and consider them as candidates for conceptual classes. This process is also known as noun phrase identification. Determining conceptual classes for the domain model using this technique must be done with care because words in natural language are ambiguous, meaning that a multitude of noun phrases can represent the same conceptual class, a noun phrase may incorrectly represent a conceptual class, etc. Ultimately a combination of the two methods, as well as other methods, is ideal to provide conceptual classes for the domain of the problem. The initial list of conceptual classes for the Vinyl Player 2.0 software is: *Record, RecordLabel, RecordDescription, RecordCatalog, RecordPlayer, Phone, Camera, MusicPlayer, Album, AlbumDescription, Song, CVSystem & User.*

5.2.1.1.2 Adding Associations

Now that there is a list of conceptual classes that are encompassed within the domain model, the next step is to specify the important interactions between these classes. This is done using associations, which define the semantic relationship between two or more conceptual classes that involve connections between that instances of those conceptual classes.

The UML notation for the portrayal of associations is a line that connects the conceptual classes with a name for the association and a multiplicity expression at each end of the line. Multiplicity is a characteristic that specifies how many instances of a class can be associated with one instance of the class it is associating with. Associations are by nature bidirectional, but if it helps read the diagram there may be a directional arrow inserted.

Just as in the previous step to construct a domain model, there are a couple of methods for formulating associations between conceptual classes that are important in the domain. The first technique is identifying "need-to-know" associations, which are connections between conceptual classes that express some knowledge that need to be preserved. An example of this would be that the

MusicPlayer class plays and *Album* because it is necessary for the knowledge of which *Album* is being played.

The next method mimics the process of creating a group of categories and using those as inspiration that was seen for creating conceptual classes. This tactic involves generating what is known as a Common Associations List, a list of high-priority associations grouped under common categories. The Common Associations List for the Vinyl Player 2.0 is shown below.

Common Association Category	Examples
A is a physical part of B	*Record — RecordLabel, Phone — Camera,*
A is a logical part of B	
A is physically contained in/on B	*Record — RecordPlayer*
A is logically contained in/on B	*Record — RecordCatalog, MusicPlayer — Phone, CVSystem — Phone*
A is a description of B	*RecordDescription — Record, AlbumDescription — Album*
A uses or manages B	*User — MusicPlayer*
A communicates with B	*Phone — RecordPlayer, MusicPlayer — RecordCatalog*

Table 17. Common Association Category List

5.2.1.1.3 Adding Attributes

The final phase of creating a initial domain model is to identify attributes that go along with each conceptual class. In UML an attribute is defined as a logical data value of an object such as integers or text. The notation for adding attributes to an existing conceptual class is to make a second compartment in the box that house the class and list the names of the attributes with their data type alongside it.

It is at times difficult to decide whether to make an attribute or a separate conceptual class and then create an association between them, but best practices indicate that if an attribute can not be represented as a logical data value then it is preferred to move it out of the class. Seeing that the domain model is purely conceptual, these rules will not necessarily be applied in the actual implementation.

5.2.1.1.4 Initial Domain Model Conclusion

The product that is produced by the previously explained steps is the domain model that will serve as inspiration for the object oriented implementation of the Vinyl Player 2.0 use case. The major conceptual classes that were found through the the first step of creating the model such as *MusicPlayer* and *AlbumDescription* and all of the associations that interconnect the diagram give an abstracted view of how the system works. Through multiple iterations of improving the requirements for the Vinyl Player 2.0 the domain model is also bound to be modified through addition and deletion of conceptual classes, associations and attributes.

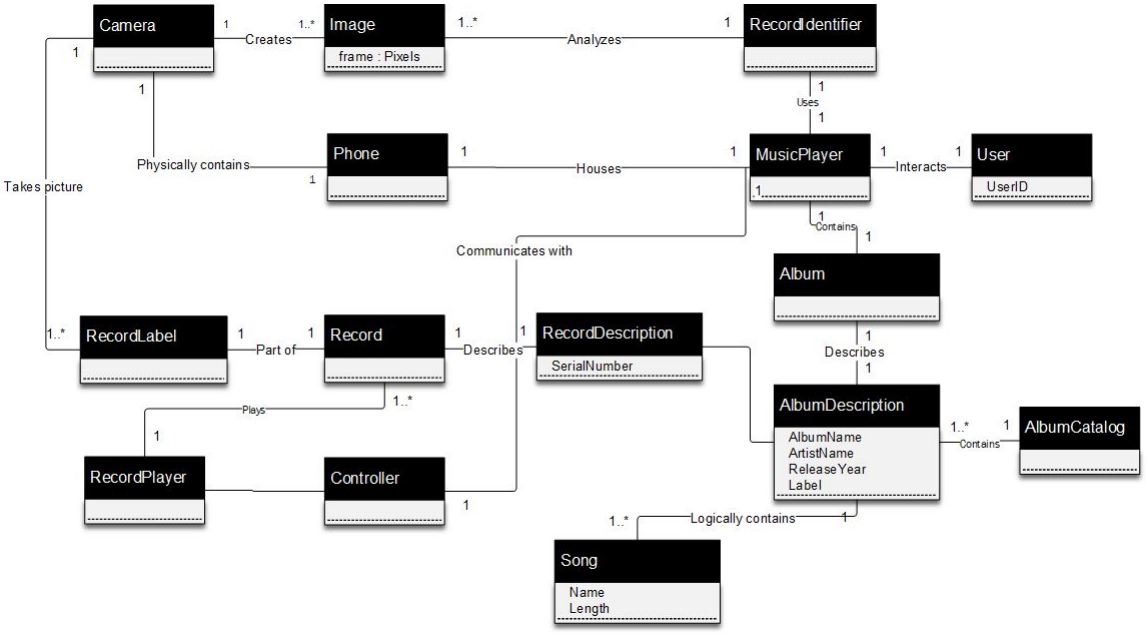


Fig 14. Vinyl Player 2.0 Domain Model

5.2.2 Embedded Programming

The microcontroller will be a crucial portion of this project, as it will be the main relay between the user's device and the record player. It will connect to the user's device and control all of the motors that affect the record player. The programming of this microcontroller is known as *embedded programming*.

5.2.2.1 Embedded Programming Flowchart

The following flowchart outlines how the microcontroller will relay with the user's device and control the motors.

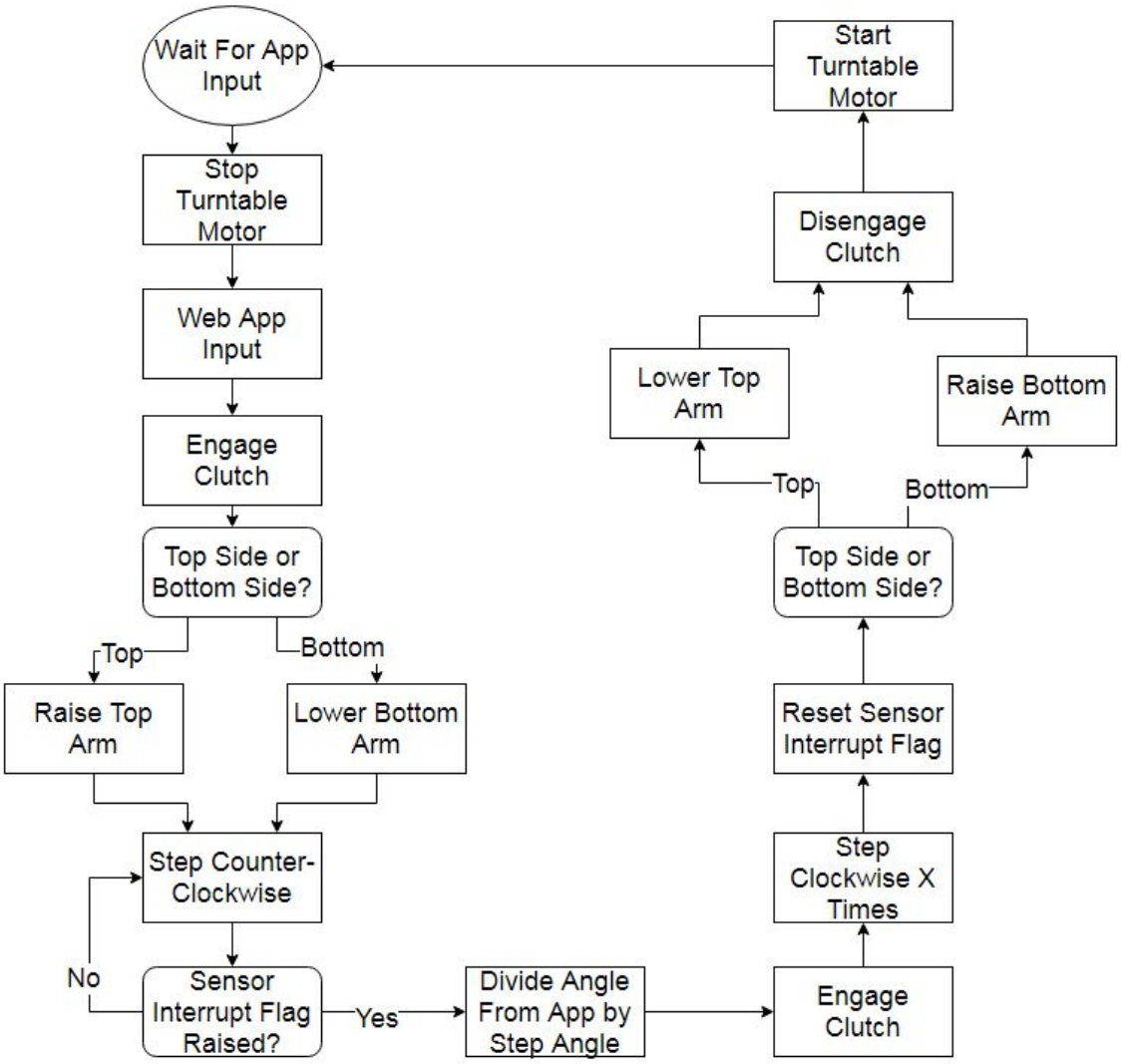


Figure 15. Embedded Motor Control Flowchart

As this flowchart demonstrates, the logic flow for the embedded programming will start with the microcontroller waiting for an input (where to move the arm) from the app. Once an input is received, the turntable will be stopped and the clutch connected to the motors will be engaged. The microcontroller will output on the appropriate pin to raise or lower the appropriate arm. Then, it will step the horizontal motor clockwise until the sensor flag is raised on the microcontroller. At this point, the microcontroller will have a starting point for the horizontal motion. From there, the microcontroller will calculate how many steps need to be made to reach the user's inputted song on the record. Once the horizontal motor has been moved in place, the appropriate arm will be raised or lowered onto the turntable, the clutch will be disengaged, the turntable will be restarted, and the program will wait for user input once again.

5.2.2.2 Embedded Programming Implementation

In implementing the embedded programming, first the programming environment needed to be set up. Once the microcontroller was received, the appropriate software to upload the code had to be downloaded. Arduino IDE 1.8.0 was used to write code and upload it onto the microcontroller. Once it was downloaded, it had to be setup for the specific microcontroller used. To do this, the board used had to be specified and an additional library for the Blend 2 was installed. After that, some of the example code was used and uploaded onto the microcontroller to confirm that it works. The code worked as expected.

The microcontroller package that was downloaded onto the Arduino IDE also included basic code for connecting to bluetooth, and that was also tested. From there, it was necessary to output a voltage to the pins. To do this, in the code, specific pins were declared as outputs and a high signal, delay, low signal, and delay were written in a loop. An LED was connected to the specified pin, and it flashed on and off, confirming that the pins work.

Since the bluetooth and I/O capabilities of the microcontroller were tested, they could be used to implement this project. The user's application will send a location on the record that the arm needs to move to. All of the stepper motor control and clutch engagement/disengagement described in the Embedded Programming Flow section above will be implemented by outputting a voltage to the specific pin that the motor and clutch drivers will be connected to.

The microcontroller will also need to convert the position on the record player to a certain number of steps on the stepper motor. The motor purchased for the horizontal motion of the arm has a step angle of 1.8 degrees. The following formula can be used to calculate the distance that the needle of the arm will travel when the stepper motor steps a single step.

$$d = 2\pi r\left(\frac{a}{365}\right)$$

Equation 1 – Needle Distance Calculation

In this formula, r is the length of the arm and a is the step angle. Because the arm of the record player is 16 cm long, each step of the motor connected at the base of the arm corresponds to .5 centimeters. As each groove is about .013 centimeters apart, this is not accurate enough. Microstepping, however, will decrease the step angle of the motor. If the motor is microstepped to make the angle 1/16th of the original size, each step would correspond to .03 cm of motion on the record. This should be a small enough movement to be within 5 seconds of the selected song. Also, since each step corresponds to .03 cm along the record, the precise number of steps required to reach a specific location on the record player can be calculated.

5.2.x Server Architecture

The application to control the record player will be paired with a server that will store and retrieve information that the application will require. The server will store user and vinyl record information, perform authentication for the user, and fetch information from the Discogs API whenever it is not available in the server database. The server will be developed in JavaScript using Node.js and Express.js to handle server-side web requests, PostgreSQL to store the data, and Heroku to store the data.

5.2.x.1 Initial Development

Node.js makes implementation of a web server very simple and easy. This section will outline the steps taken in development. First, a new repository was created in GitHub and was initialized with a .gitignore file that is specific to a Node.js project. The project was then cloned locally for development to begin. The project now needed to be initialized as a Node project. Once Node was installed, the script ‘npm init’ was run on a command line in the directory of the project. This initializes the project by adding the default Node dependencies and creating a package.json file. This file is a general outline for how the project is run and what dependencies are required. This file allows anyone with Node installed to clone the project and run it with all of the dependencies pulled in automatically. Next, some additional dependencies were added.

Additional dependencies are added in a node project by using the command `npm install [node_module_to_install] --[options]’. The node modules installed included express, pg, pg-query, and eslint. Express is a node module that allows for quick server development. Both pg and pg-query are used to connect to and generate queries to a PostgreSQL database. Eslint is a tool that helps developers keep code formatted correctly by giving warnings when formatting is incorrect.The options used in installing these dependencies were ‘save’ and ‘save-dev’. The save option automatically adds these dependencies to the package.json files so that other developers using this project will be able to install the same dependencies. The save-dev option is similar, but the dependencies added with that option will only be installed in a development environment. Once dependencies were added, an initial commit was made and pushed to the GitHub repository, and web request handling was begun.

5.2.x.2 Web Request Handling

Handling web requests are made very simple through express js. First, a file called ‘server.js’ is created. This is the initial file that is run on startup, as specified by the package.json file. This file, like all of the other server-side files, starts by importing dependencies specific to that file. This is done by calling ‘require(dependency)’ and setting the results of that import to a variable. This variable can then be used and passed as necessary. For this file, the

dependencies imported are express and routes. Routes is a file that will be created next and will contain all of the possible paths that the web request could take. The express app created by calling the express dependency is passed to the routes module in order to handle these routes. Then, any paths that we not accounted for by the routes file are caught in the server.js file, which returns a 404 (not found) response. The app is then told to listen on a specific port.

The routes.js file contains most of the code to handle requests. The only dependency used by this file is ‘db’, which is a file that will be made later to set up and configure a connection to the Postgres database. The db dependency is assigned to a variable called ‘pool’. This variable has a function built in called query, which accepts a query string, an array of parameters to be passed to the query, and a callback function, which is called after the query to the database is completed. The routes file exports a function that, when called by the server.js file, parses which path the web request was sent to and what type of request was sent and calls a function to handle that specific path/type combination. For example, if a POST is made to the route ‘records/search/:name’, the routes file will call the function that searches for a record with the given ‘name’ parameter, and returns the results. A specific route will be set up to authenticate users, and to search for specific artists, records, and songs.

5.2.x.3 Searching For Records

The main purpose of the web server is to give the application record information as it is requested. There will be two main types of searches for information, searches for data that should already be in the server’s database and data that is not available in the database and therefore must be received from elsewhere.

The user will be able to have a saved catalog of their records, so those records will be stored in the database. Whenever the user makes a taps to view their catalog from the application, a POST request will be sent to the server at the path “/getCatalog/:userId”. Once the request is received by the server, the request will be routed to the “getCatalog” function. The request will contain the user’s id, so the function will query the database for all records that are in that user’s catalog. The results of the query should contain all of the records in the user’s catalog, with the record’s name, artist name, and album cover art. Those results should be returned to the user. From there, if the user selects on of those specific records in their catalog, the same process should happen. A POST request is sent to the server from the user’s application (this time to a different path). The request should return all of the song names and durations of songs for all of the songs that were in the record that was requested.

If the user searches for a record that is not yet in their catalog, however, the process gets a bit more complicated. Since the query text entered could be a search for any number of records that have the same name, however, all records with that name should be returned. Since the database only stores records that have been added to the database as users add to their catalog, the results of this

request must be gotten from a different source. From their, an API call is made to Discogs, an open source database that contains information on millions of records, as discussed in the research section of this paper. The text entered into the search bar by the user will be sent in the API call to Discogs, which will return all of the records that have a name similar to the text entered. Once the results of the API call are returned to the server, the server will reformat the data into a format that is more suitable for the application to use and return the data to the user's application, which will display the data to the user.

If the user selects on of the results of the API call, a similar process will be used. The id (from Discogs) of the record that was selected will be used to make another API call to Discogs, this time requesting specific information about that record, such as song titles and times. That data will be returned to the user's application for display. If the user chooses to add that record to their catalog, another request to the Discogs API will be made to request artist, record and song data for the record cataloged. That data will be inserted into the server's database for future use.

5.2.x.4 User Authentication

When storing user information, it is important that the information is safe and secure. Email and password information cannot be stored in plain text, but must be converted to a secure has, which is stored. There are many different algorithms for generating a secure hash. The following section will discuss the process of generating a hash and how it was used to store user information.

A secure hash is generated by using what is called a one-way computation. These are computations that are easy to make but, if given the result of the computation, are very easy to find the original inputs for. So, whenever a password is entered, this one way computation is made on the password, generating a hash for the password entered. Hackers have gotten good at reversing these functions, though, so the result of the function is then fed back into the algorithm for multiple iterations (on the order of 20,000). One thing to note about these functions is that the same input will always produce the same output. Since there are some passwords that are common among users, this would lead to duplications of a single hash in the database. If a hacker was able to gain access to the entire table of data, they would see that a certain password was used many times (because of the duplicates), thus making it easier to find out the passwords. The solution for this problem is using a salt.

A salt is a random hash of digits that is used whenever making a password hash. A salt tends to be 16-256 characters long (180 for this project). Whenever the password is used to generate a password hash, the salt is concatenated to the original password before generating the hash. This makes it so that even if two users use the same password, they will generate a different password hash. Both the salt and password hash are stored in the database, but the original password is not.

When the user logs in from the application, the email and password entered is sent to the web server. This web server receives the email and password and queries the database for the email, password hash, and salt that corresponds to the email address received from the application. The password sent from the application is then concatenated with the salt and the password hashing algorithm is used on the salt/password combination. If the result of this algorithm generates a hash equivalent to the password hash that is stored in the database, the user is successfully authenticated and that result is sent to the application. If the two hashes do not match, a failed response is sent to the application.

Whenever a user is created from the application, the email and password entered is sent to the web server. At the web server, the database is queried to confirm that an account for that email does not already exist. If the user already exists, an appropriate response is then sent back to the application. If the user doesn't exist, a new salt is generated, and is used with the password to generate a password hash. The generated salt, password hash, and email are saved to the database and a session id for the user is returned.

5.2.x.5 Deployment and Database Creation

This server was hosted through Heroku, a company that makes it very simple to host web servers. A new project was created in Heroku, and credentials to that GitHub repository were given. Heroku will automatically deploy, host and (re)build the project every time code is pushed to a specified branch (in this case, master) of the GitHub repository. Also, heroku has an addon to create a PostgreSQL database. This addon was used to create the database. The credentials to access the database (host address, database name, user, and password) are supplied by heroku, and these credentials were used to configure database connections. Now that the project was hosted, the address https://vinyl-player-server.herokuapp.com/ could be used to access the app. The free 'hobby' version of heroku was used for deployment, which sleeps after 30 minutes of inactivity. For $7 a month, the app would never sleep.

5.3 Computer Vision Development

The implementation of the computer vision component of the Vinyl Player 2.0 has the task of taking an image of a vinyl record from the camera of the user and being able to produce the album and artist that will be present on the record label and the center of the analog medium. This objective will be achieved using a combination of the computer vision library OpenCV and the machine learning library TensorFlow as well as a number of popular Python libraries. All of these technologies are used to create a convolutional neural network that will process the images and will independently recognize patterns in images of vinyl records to obtain methods of determining the classification of text on the record label. A hand-written feature, such as the torus nature of a vinyl record, can also be

added as input into the neural network component to simplify the classification of the text in the inputted image.

5.3.1 Image Representation

The images inputted into the deep learning implementation of the computer vision software system of the Vinyl Player 2.0 will need to be represented in a way that that can be easily analyzed by the program. Fortunately an image is simply an array of pixels and each pixel has a value for the intensity of the color, so an image can be expressed as an array of values ranging from 0 to 256. This form of image representation is highly used as it still perfectly characterizes the features of the image. If an image has multiple colors (channels) it can broken up into each of the RGB colors and then a unique pixel intensity array can be obtained.

The process of obtaining an image and then extracting the pixel intensity is relatively simple using the OpenCV library. The image to be analyzed is first loaded as a matrix using the imread(filename) function. Next to obtain the pixel intensity value matrix from the image there needs to be some known information about the image for the parameters of the method, which is whether the image is multi-channel and the image type. Assuming that the image has only one channel and the image type has a pixel coordinate representation of x and y, as opposed to Point representation, accessing the pixel intensity values can be done with the OpenCV method image.at<uchar>(y, x), where image is the matrix that represents the loaded image file and x and y represent the pixel coordinate position. An example of the pixel intensity values that have been normalized to be in between 0 and 1 for an image is shown below.

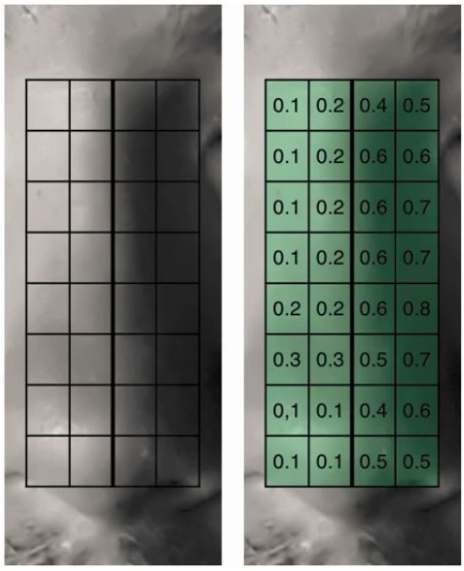


Figure 16 – Pixel Intensity Matrix Example

5.3.2 Feature Extraction

A feature with regards to computer vision is an attribute of the dataset such as whether or not an object has a torus shape which is the convention for vinyl records. When dealing with neural networks, features are effectively the input nodes that are utilized to create a complex representation of the problem through the hidden layers. One technique for obtaining features for input into the neural network is to use the knowledge of the problem and dataset to obtain telling attributes and using a classifier on the image to determine if an image has the feature.

Do to using a convolutional neural network for the Vinyl Player 2.0's implementation the process of feature extraction is simplified as there are multiple layers prior to the deep neural network component that obtain the features of the dataset independently. The functionality and process to create the layers of the CNN that will perform feature extraction will be further articulated in the accompanying sections.

5.3.2.1 Convolution Layer

A convolution is an integral that takes two functions as input and returns a modified version of one of the functions. The mathematical definition is shown below.

$$(f * g)(t) = \int [f(\tau) * g(t - \tau)] d\tau$$

Equation 2. Convolution Integral

In a convolutional neural networks, which derives its name from convolution is done by taking the pixel intensity value matrix that is obtained from a sample in the dataset and multiple it by a 3x3 matrix known as a feature detector or filter to obtain a feature map, a smaller matrix relative to the pixel intensity matrix. The application of this operation is to reduce the size of the image to decrease the time it takes to process the image. An important property of the feature map is that the spatial relationship of the pixels in the image is undisturbed meaning that even though there is a loss of data by performing convolution on an image the information relevant to the feature detector is preserved. There will be a sizeable number, around 30, of feature detectors used for the convolution operation against the input image to give focus to different areas of the image. There can be more than one convolutional layer in the CNN.

The machine learning component of the computer vision software is implemented using TensorFlow and creating convolutional layers for a CNN is effortlessly done using the API. An interface for a 2D convolutional layer is implemented in the tf.layers.conv2d class and added to the CNN simply by calling its constructor which takes a variety of inputs, the most essential being the input layer of the convolutional neural network, the number of filters in the convolution, the dimensions of the filter, and the activation function of the layer. It

was previously stated that the dimensions of the feature detector is 3x3 and while the TensorFlow library allows for this to be variable the convention is 3x3, with 5x5 also being a popular option. While the OpenCV operations described in section x.x involves matrices, TensorFlow of course uses tensors and thus there would be a step for the conversion.

5.3.2.2 ReLU Layer

After the convolution is a layer that is virtually used as a supplement to the convolutional layer. The ReLU layer is designed to take the feature map that is derived from the previous layer and run each value from the matrix through the rectifier activation function. The rectifier functions returns a zero value for non-zero negative input and acts as a linear function for positive input, returning whatever value the input is. The purpose of this layer is to introduce non-linearity into the image, which is useful due to the fact that data that will be found in the vinyl record images and most of the data in the real world will not include negative values.

This step is simply implemented and in TensorFlow it is conveniently include in the constructor of the convolutional layer. The activation function is one of the more important parameters to the tf.layers.conv2d constructor that was described in the previous section and can be chosen to be the rectifier function by passing tf.nn.relu to this parameter. Other activation functions can be used but the rectifier function is the most popular and the most applicable to the classification and recognition tasks required from the Vinyl Player 2.0 implementation.

5.3.2.3 Pooling Layer

Once the vinyl record image has been convoluted and passed through the rectifier activation function the tensor that represents a sample from the dataset of records or a unique image of a record will be downsampled via the pooling layer. Again reducing the size of the tensor will retain the important spatial relations between pixels. Another benefit of the pooling layer of the convolutional neural network will be to account for possible distortion in the feature map.

The type of pooling that will be utilized for this application will be max pooling. Max pooling is implemented by first defining a tensor with 2x2 dimensions, which is the convention but not written in stone, and place this tensor in the top left corner of the feature map. Then the largest number of the feature map that is within the 2x2 tensor is placed in a pooled feature map. This process is repeated after shifting the 2x2 tensor by a certain number of cells known as a stride and repeating the process. The resultant tensor will have a reduced dimensionality from the feature map and will contain the maximum value from each iteration of the masking tensor used on the filter. The pooling process is applied to each individual feature map that was obtained from the convolution layer.

The TensorFlow implementation of adding a pooling layer to a convolutional neural network is similar to the process of adding a convolution layer as it is simply adding calling the constructor of tf.layers.max_pooling2d class. The relevant parameters that go into this constructor is the preceding layer's output (feature map), the dimensions of the tensor that will be used to go over the feature or pool size as TensorFlow documentation coins it and the dimensions of the stride that the pool will take. The pooled feature map that is outputted from the pooling layer is representative of the high-level features of vinyl record database. The string of layers from convolution to pooling can be repeated multiple times where the input of the new convolution layer is the output of the old pooling layer or the pooled feature map, which can allow for a more refined set of high-level features.

5.3.3 Deep Neural Network

The feature extraction process that done via the convolution and pooling layers has been proved to perform much higher at tasks such as text recognition and natural language processing than simply hand-picking features, which is why the convolutional neural network is an optimal choice for the Vinyl Player 2.0 software system.

Once this process is done the final layer, known as the dense layer in TensorFlow's documentation, is a deep feed-forward neural network that takes the values from the pooled feature map tensor as input. The implementation of this neural network can also be broken up into layers, which are virtually sub-layers in the context of the convolutional neural network.

5.3.3.1 Input Layer

This layer of the deep neural network will have a neuron for each individual value in the pooled feature map obtained from the pooling layer. Due to the pooling layer only outputting a 2D tensor, it is necessary to reshape the tensor so it can be inputted into the neural network. This process is called flattening and is done in TensorFlow simply by using the tf.reshape() method and passing the pooled filter as a parameter.

The normalized values of the reshaped pooled feature map will be connected to the first fully-connected layer in the neural network through weights that can be given some default, most likely 0, for each feature.

5.3.3.2 Fully-Connected Layer

An aspect of the deep neural network that appears in a CNN that somewhat sets it apart from any general implementation of the network is that the hidden layer, or layers that are in between the input and output nodes, are fully-connected. This characteristic of the neurons in this form of layer means that each neuron in the layer has a connection to every neuron in the previous layer.

Aside from this aspect the fully-connected layer is the same as a regular hidden layer found in deep neural networks.

The purpose of the neurons in the fully-connected layer is to take the weighted inputs and then pass the result through an activation function, such as the rectifier function that was mentioned in the convolution layer, to create a feature that expresses new knowledge for the problem at hand. The dense layer or fully-connected layer is produced in TensorFlow by using the tf.layers.dense constructor and passing it parameters that will both append it to the existing convolutional neural network and specify characteristics about the layer. Some of these characteristics include the number of neurons is the layer and the activation function that is used in all of the neurons in that layer. Note that tf.layers.dense only adds a single fully-connected layer to the CNN and due to the fact that a deep neural network is defined sa has having multiple hidden layers the constructor for tf.layers.dense will have to be called a number of times with the previous layer being a parameter in each new layer.

5.3.3.3 Output Layer

This is the final layer in the deep neural network component of the CNN is the output layer which is what will classify the words that are recognized from the record label as being the artist, album or neither. The output of each neuron in the output layer will be some value ranging from 0 to 1, which states how sure the system is in classifying each recognized word. Producing the output layer of TensorFlow is done by adding another dense layer, but this time using the string 'dropout' as the inputs parameter and using the unit parameter to specify the number of output nodes that should be present in the layer. The activation function for this layer will often be the softmax function as opposed to the rectifier that is used in many other layers of the network. This activation function is utilized with the tf.nn.softmax class.

5.3.3.4 Backpropagation

The actually learning that occurs in the convolutional neural network is due to the backpropagation process that occurs after the output layer of the deep neural network in the CNN gives values for a sample in the dataset. The accuracy of the outputted value will be checked against the actual value that is given in the dataset. This is done using a cost function, a popular choice being the following equation, where y' is the value outputted by the neural network and y is the actual value.

$$Cost = \sum[1/2 * (y' - y)^2]$$

Equation 3. Cost Function

The goal of the convolutional neural network is to minimizes the cost function because that would mean that the system is correctly classifying with a

high frequency. The minimization is done through gradient descent and from that an update in all of the weights in the convolutional neural network will be performed to tune the model for higher accuracy.

In TensorFlow once the costs are calculated an optimizer that uses that gradient descent technique can be used vias the the tf.train.GradientDescentOptimizer class that takes the learning rate of the system as a parameter to the constructor. Once an instance of this class is created then the optimizer.minimize() method can be used to reduce the cost function by adjusting the weights in the model. This method takes the costs and the model as a parameter. The updated model is then returned.

5.4 System Design & Schematics

This section is dedicated to system design and the various schematics used in the project. This includes the schematics for the boards used in the project that will be edited to create the final pcb for this project.

5.4.1 A4988 Board Schematic

This is the A4988 carrier board schematic. Rather than create the entire board from scratch, the A4988 data sheet gives the Eagle schematic for the board. This board will either be edited and used to be implemented into the pcb or mounted onto the pcb.

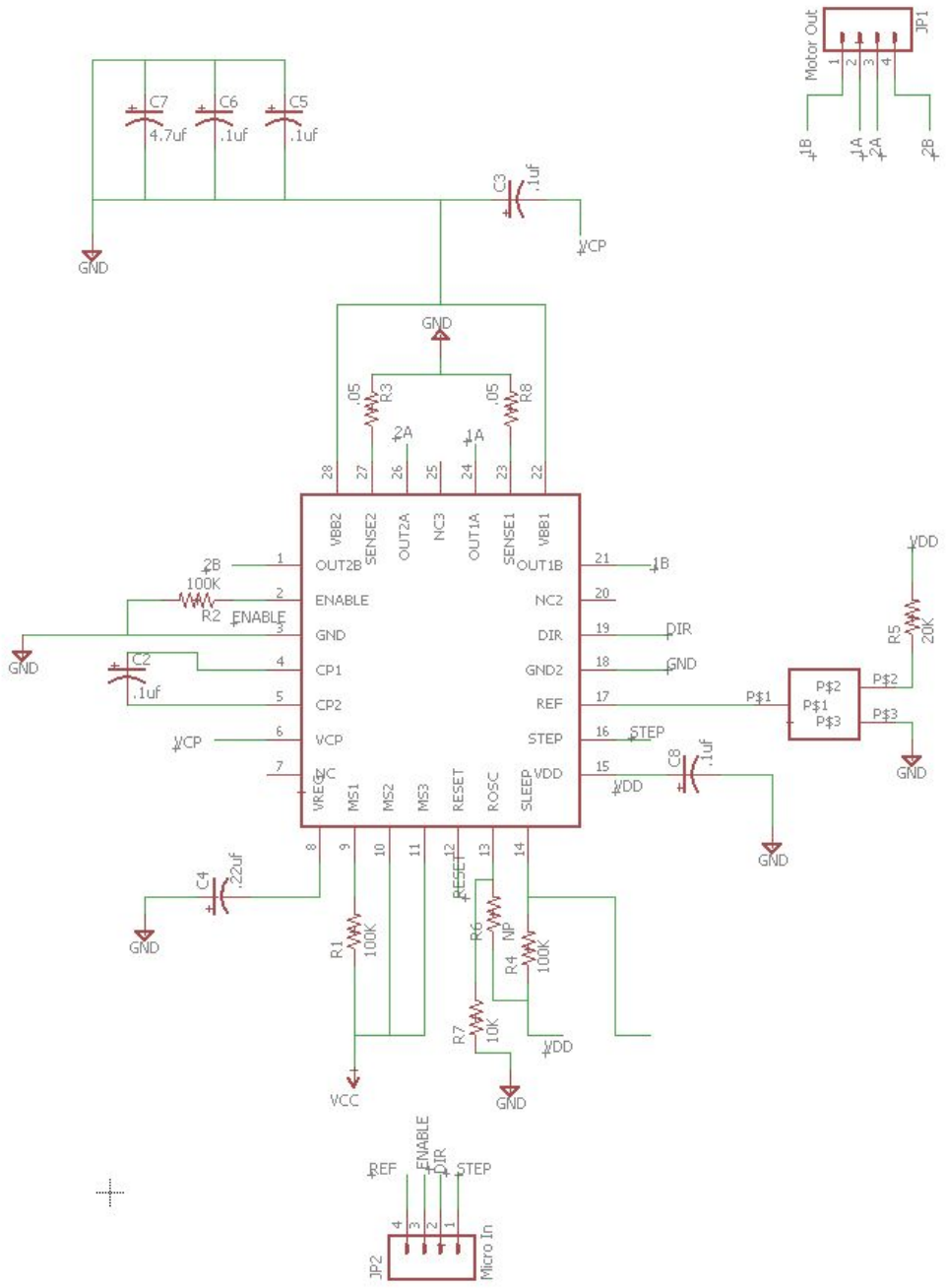


Figure 17. A4988 Board Schematic

5.4.2 BLE Nano 2 Board Schematic

This is the schematic for the BLE Nano 2 Board. This is the board that will be used to control the bluetooth communications and the motor drivers. The I/O pins connected to the MCU will be connected to the various motor drivers and other components that are required to be controlled. Please refer to the testing section for additional information on how these pins will be connected.

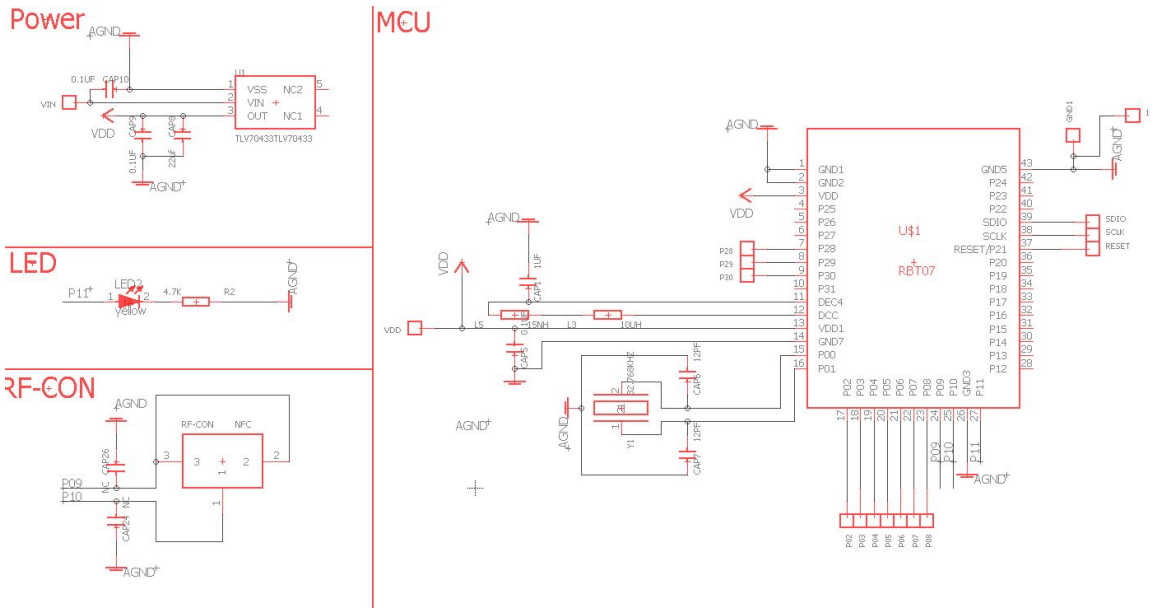


Figure 18. BLE Nano 2 Board Schematic

5.5 Software Flow Chart

Below is a diagram that outlines the process within the mobile application. It models how the user will transverse the mobile application and how the system will respond.

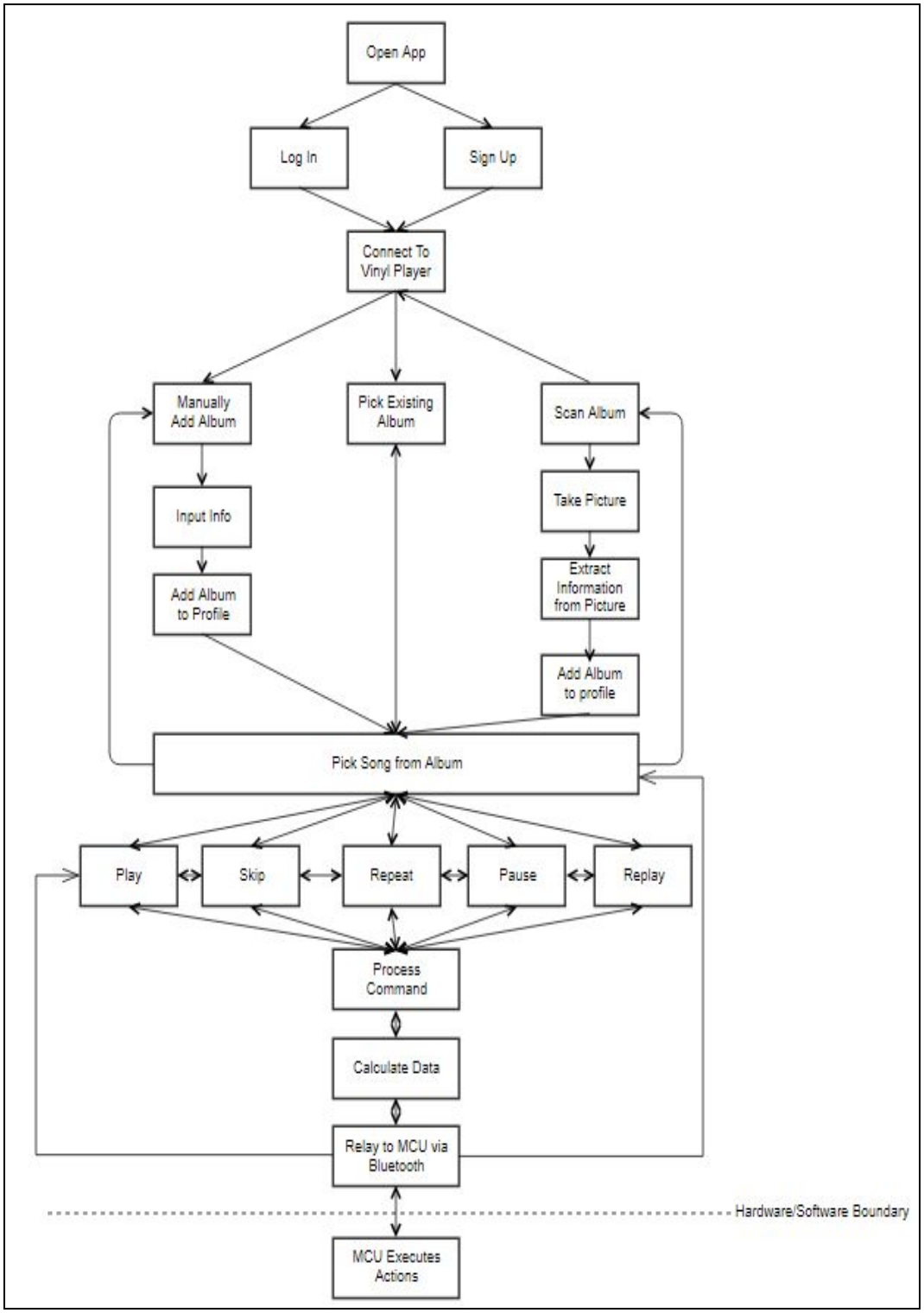


Figure 19. Software Flowchart

3.17.7 GUI Design

The Android application will have multiple GUI interfaces or “activity” as Android likes to call them. Our Android application will have a GUI for the Login page, the home screen, a page to add/modify/delete vinyl albums, and a media player. The following sections will introduce some information about the GUI design and why it was designed.

3.17.7.1 Login Page

The image below is the basic user interface for our home screen. This will be the first page the user will visit when they access our mobile application. The user will be asked to sign up for an account if it’s their first time signing in or to input their previous credentials to log into their account.

3.17.7.2 Home Page

This will be the next screen you will see after you “sign up or login.” This will be known as the user’s home screen where they will find most of the basic features of the mobile application. From initial inspection, the application will greet you and present a profile picture of you, the previous vinyl albums and songs you recently played, and a floating widget that will bring you to your camera. The camera will allow you to take a picture of the vinyl label which will add that vinyl album to your collection. We will identify and pull the information from the vinyl album and display it down below.

3.17.7.3 Add Vinyl Album

The following image below is the “add vinyl album” page. This is where users can add the vinyl album manually if the camera cannot correctly identify the vinyl label or there is no vinyl label. The user will be able to upload a picture of the vinyl label and input the following information about the vinyl album. We need to know the size of the album in inches as well as the number of songs and the duration of each song. Once the user has input this information, the information would be saved in a local database or in the phone’s storage.

3.17.7.4 Music Player

After the user has taken a picture of the vinyl album, the next step would be to play a song from the album. The image below display the basic layout of the media player. You will be shown an image of the vinyl album you have taken a picture of, information about the band, release data, and song list. You will be able to scroll through the song list and pick which specific song you would like to play. You will also have the basic media controls (back, forward, play, repeat and shuffle).

3.17.5 GUI Design Mockups

The following images are the GUI Design mockups. From top to bottom and left to right, the mockups are for the login page, the home page, the add vinyl album page, and the music player page. The GUI designs keep in line with standard Android user interface styling.

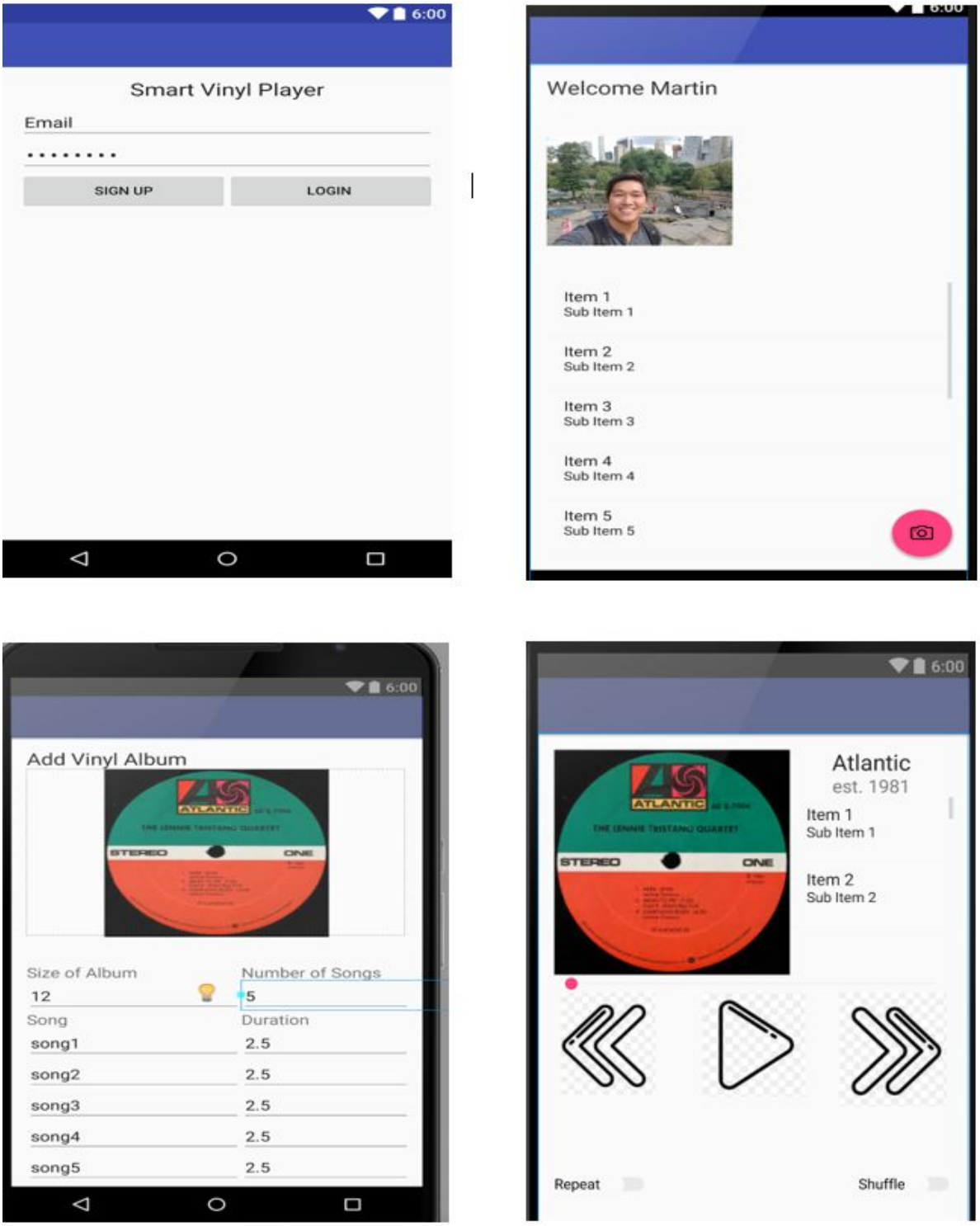


Figure 20 – GUI Mockups

6 Testing

In such a large project, unit testing plays a huge part in successful implementation. As components are ordered and received, they need to be individually tested to ensure that they are working as expected and to confirm their safe operating voltages. As components are put together, regular testing is required to ensure that the components continue to function as expected. The components that need to be tested are the record player, motors, motor drivers, microcontrollers (and their pin outputs), motor accuracy, PCD output, and more. The following section outlines how each of these components were tested, and the final results in testing these components. Additionally, below is a picture of all of the components that have been received for the project. The picture contains the Stanton T.62 record player, three packaged A4988 motor drivers, a breadboard with one unpackaged A4988 motor driver, the Blend v2 development board, two BLE v2s, a USB uploader for the BLE v2s, and three motors (one of which has the planetary gearbox to allow for microstepping).

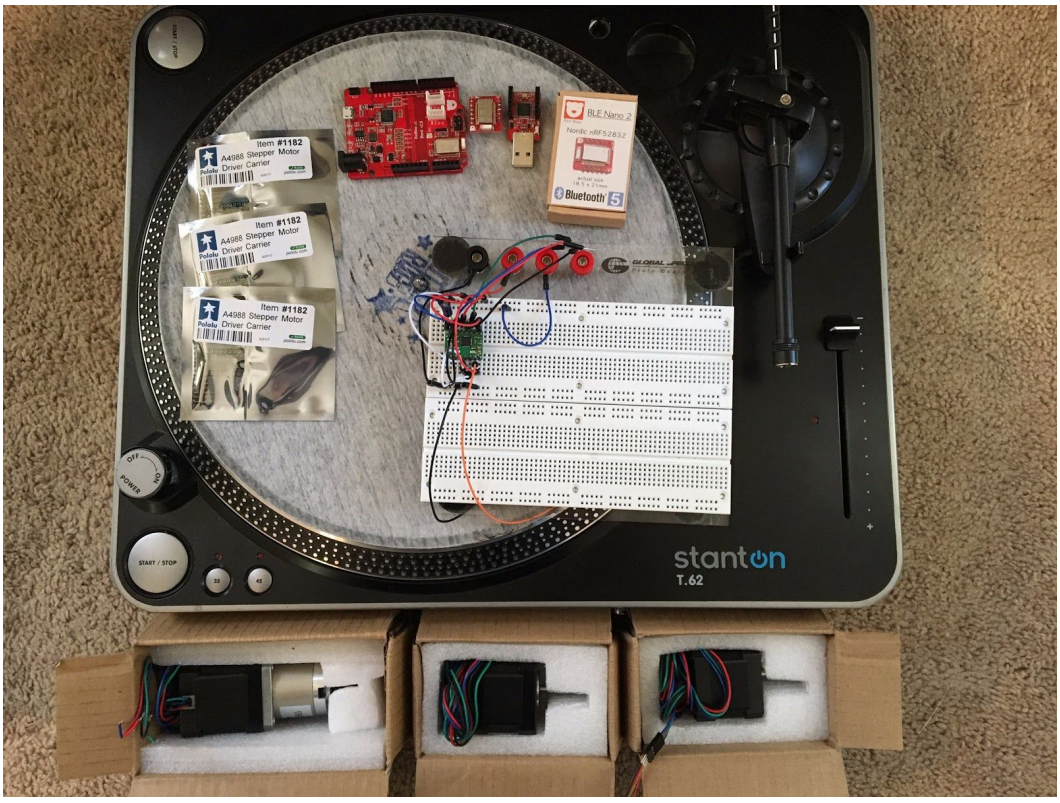


Figure 21 – Project Components

6.1 Record Player Testing

A few factors of the record player needed to be tested as it was received. The factors tested were power, rotation speed, and sound output. To test the power, first the record player was turned on.

To test the rotation speed, the record platter was first marked at a single point. The platter was set to spin at 33 RPMs, and the rotation speeds were measured with a stopwatch. Then, a record was placed on the platter and the needle dropped to make sure the sound outputs as expected.

6.2 Motor Testing

One of the most important aspects of this project is the control of the tonearm movement. Because of this, proper testing of the motors is extremely important. The testing of the motors will be carried out in several phases. This first phase will be spent on trying to achieve rotation of the motors using lab equipment and the A4988 stepper motor driver. The second phase of testing will try to implement the use of a microcontroller to more accurately test the precision of the steps of the motors.

6.2.1 Phase One Motor Testing

This phase of testing will first test if the stepper motor driver A4988 is functioning properly. A successful test of the A4988 driver should consist of successfully starting the rotation of a stepper motor. This phase of testing will not require a microcontroller as the stepper motor driver should be able to be controlled with the lab equipment in the UCF Senior Design Lab.

A4988 Stepper Motor Driver

The A4988 Stepper Motor driver is controlled using 16 pins. The A4988 datasheet contains an example of how the driver is operated.

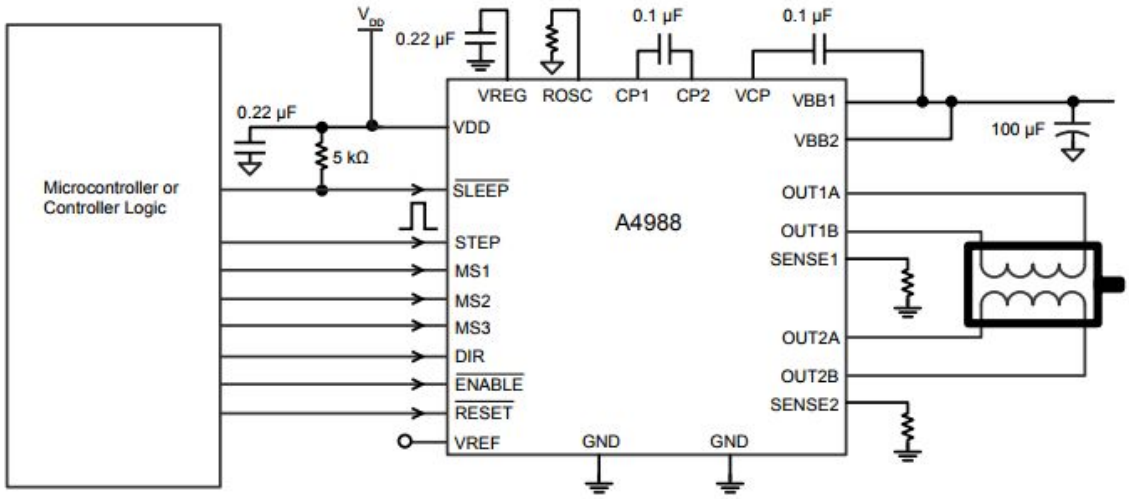


Figure 22. A4988 Datasheet Example

Since this phase of testing will be testing the driver and the motor without the microcontroller, some of the pins which connect the microcontroller to the driver will be connected to a high or low voltage source depending on the requirements of the pin. One of the pins which will not be connected to a constant voltage source is the STEP pin. This pin will be tied to a function generator outputting a square wave. This square wave will control when the motor will step. For this reason, the frequency of the function generator must be extremely low as to not overwhelm the driver and the motor. High speed is not required in this application so high frequency of the square wave will never be required for this application.

Contrary to the diagram figure 22, the sleep and reset pins will be tied together. This is because that when the reset pin is tied low, the inputs of the STEP pin are ignored. For this reason, the reset pin must always be set high if the motor is to be operational. Similar to the SLEEP pin where setting the pin to a low puts the driver into sleep mode. For normal operation, the SLEEP pin must be set to a logical high, similar to the reset pin.

There are several values and results that must be determined during phase 1 of the motor testing. One of the most important values to test is the current output of the stepper motor driver into the motor's inductive coils. It is imperative to not exceed the current rating of the motors, otherwise there is a potential risk of damaging them. The vertical motors have a current rating of 2A. Therefore for the first phase of testing where the vertical motors are being tested, the output current into both terminals should not exceed 2A. This current will be measured using a multimeter supplied in the UCF Senior Design Lab. During this testing, a setup for the drivers should be determined to ensure that the output current does not exceed 1.68A as that is the current rating for the horizontal motor. Other than output current, this phase of testing should determine if the values determined for logical high and logical low voltages are sufficient to drive the driver. In addition, an image of the output of the function generator supplying

the step function for the STEP pin of the driver should be recorded and given. This STEP function should be simulated in phase 2 testing using the microcontroller.

The setup for the board is fairly simple. The diagram Figure 22 shown below gives a simple representation as to how this works. The VMOT pin is the pin that supplies voltage to the motor. VMOT refers to Voltage of the Motor. The ground of the power supply is connected to the Ground pin next to VMOT. A 100uF capacitor is placed between these pins in order to decouple them. The purpose of this capacitor is to deal with voltage spikes which may unintentionally harm the motor and/or the driver. It should be noted that these pins should not be supplied by the microcontroller. There are several reasons for this ranging from safety for the microcontroller and the driver. Another reason is that the microcontroller selected for this project cannot supply sufficient voltage to power the driver. The next four pins are the 1A, 1B, 2A, and 2B pins. These pins refer to the positive and negative terminals of the inductive coils of the motor. Refer to the datasheet for the motor to appropriately connect these pins to the driver. The next two pins are the VDD and GND pin. For the microcontroller testing, these pins will be connected to the voltage supply and ground of the microcontroller. Subsequent pins that require either a voltage high or voltage low will either be connected to the ground or to a value of .7*VDD for a voltage high. For phase one of testing not using a microcontroller, these pins will be connected to a voltage power supply which will supply 5V and 0V respectively. The voltage high and voltage low will be connected to this line as well.

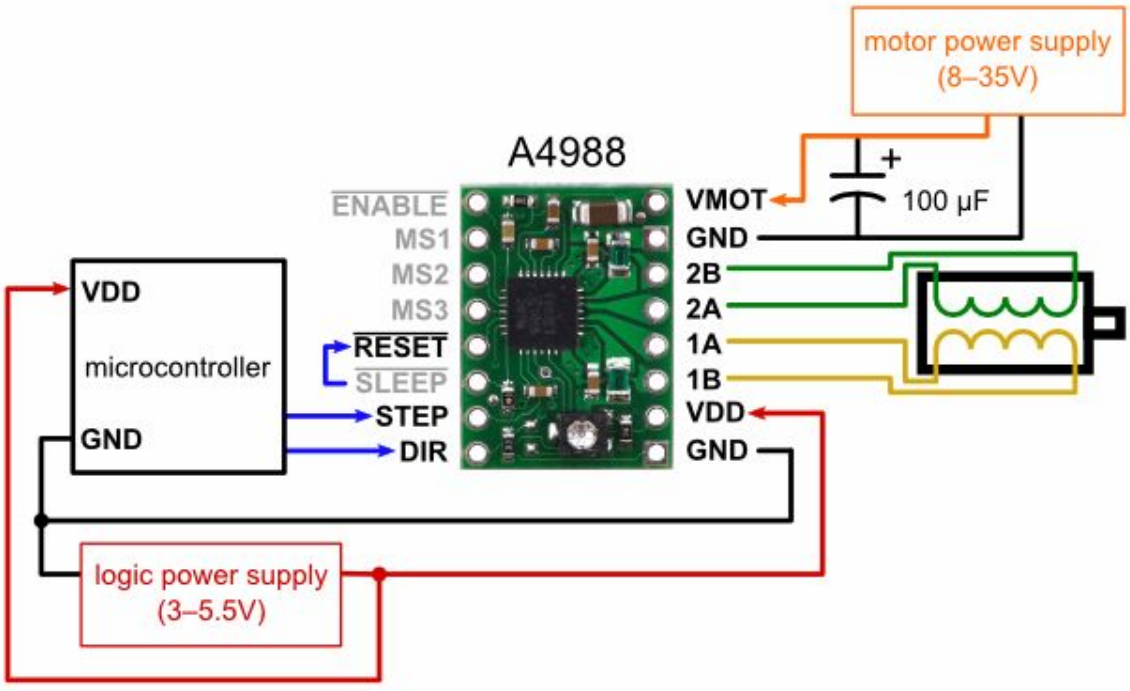


Figure 23. A4988 Minimal Wiring Diagram

The next pin is the ENABLE pin. This pin is active low so for testing this pin should either not be connected or drawn to a voltage low. This pin when not active will ignore all inputs given to the driver. Because of this, this pin must always be active for testing. Ultimately, this pin will be controlled by the microcontroller to allow independent control of each motor. The next three pins are the microstepping pins. These pins will be drawn to voltage low for phase one of testing because microstepping is not required for this testing. These three pins control the microstep setting of the driver. Please refer to Figure 23 for more information regarding the microstepping testing that will occur in phase two of the motor testing. The next two pins are the RESET and SLEEP pins. The RESET pin is a floating pin and should be connect to the SLEEP pin. If this is not done, the driver will ignore the inputs given to the STEP pin and should be avoided. The STEP pin is the pin that controls when the motor will step. In Phase two of testing, this pin will be connected to the microcontroller and drawn high and low to create a step function a specific number of times. In Phase One of testing this will be simulated by connecting the pin to a function generator. The function generator will generate a 50% duty cycle square wave at a very low frequency. If a frequency higher than 10Hz is used then this could potentially either damage the driver or the steps will be ignored. The final pin is the DIR pin. This pin controls the direction of rotation of the stepper motor. When this pin is drawn to a logical high the motor will step clockwise. Inversely, if the pin is drawn to a logical low the motor will step counter-clockwise. For phase one of testing, this pin is not relevant as the only thing being tested is if the motor can be spun. For phase two of testing, this pin is also not relevant as which direction of rotation is not needed for this phase of testing. This pin however must be controlled in the final project as the direction of rotation of the motors must be controlled.

Phase 1 Testing Results

After some minor difficulty with setup and finding a suitable test environment, a successful initial implementation of the vertical stepper motor and the A4988 stepper motor driver was created.

During the first phase of testing, a function generator is used to measure the step input into the A4988 stepper motor driver. In order to ensure proper timing and input, this function generator output was measured with the oscilloscope. A picture of the Oscilloscope output is given in the picture below.

Microstep Testing

As accuracy is extremely important for this application, accuracy must be tested if microstepping is to be used in the final device. Microstepping can have a negative impact on the accuracy of a stepper motor. However to test if the step angle is consistent with the documentation for microstepping, a microcontroller is required to pulse a specific number of times rather than continuously rotating the

motor. For this reason, the microstep testing and evaluation will be covered in the second phase of testing.

The microstep settings of the A4988 stepper motor driver are controlled through 3 pins. Setting these pins high and low correlates to a specific microstep setting for the driver. Setting all three microstep pins to a logical low correlates to a microstep resolution of a full step. Meanwhile, setting all three of the microstep pins to a logical high correlates to sixteenth step microstepping. The table below is the microstepping resolution truth table given in the A4988 stepper motor driver datasheet.

MS1	MS2	MS3	Microstep Resolution	Excitation Mode
L	L	L	Full Step	2 Phase
H	L	L	Half Step	1-2 Phase
L	H	L	Quarter Step	W1-2 Phase
H	H	L	Eighth Step	2W1-2 Phase
H	H	H	Sixteenth Step	4W1-2 Phase

Table 18. A4988 Microstepping Resolution Truth Table

During phase two of the motor testing, the microstep accuracy and setting will be tested. To test the microstepping, the microcontroller will deliver a specific number of pulses in order to step a certain number of times. With the microcontroller programmed with this setup, it will be known what theoretical angle the motor will step to. This is determined by multiplying the number of pulses by the step angle after gear reduction. The first test will be with full step microstepping. According to TABLE 18 this correlates with all three microstepping pins set to a logical low. The resulting theoretical angle will then be compared to the actual angle. Two other microstep setting will be testing along with testing the fullstep. The next setting will be fourth step microstepping which correlates to pins MS1 and MS3 tied low, while pin MS2 is tied high. The theoretical angle will be the full step angle divided by 4 as each step is a fourth of a fullstep. The actual angle will then be compared to the theoretical angle. This process is then going to be repeated using sixteenth step microstepping.

If the theoretical angles match the recorded actual angles closely, higher level microstepping will be considered for this project. Ideally the sixteenth step microstepping does not incur a large accuracy decrease. This is because the lower the step angle, the arm of the of the record player will be able to more accurately be placed.

Phase Two Motor Testing

Phase two of the motor testing contains controlling the motors with the drivers in combination with the microcontroller. The primary task of this phase of

testing is to determine if the motors can be controlled through microcontroller programming. The programming that will be used for this phase of testing will follow the embedded software flowchart in the previous section. Another important thing to test in this phase of testing is to determine if the microstepping function of the A4988 stepper motor driver does not cause a large amount of inaccuracy to the step angle of the motors. This will be determined by finding the theoretical step angle after a specific number of steps and compare that to the measured step angle of the motor.

Phase Two Motor Testing Results

In a previous section, there was discussion about how the microstepping will be tested in phase two of the motor testing. Even though a higher resolution of step angle is not required for the vertical motor, microstepper could help smooth the rotation of the motor and allow less rigid movement of the tonearm. Testing the accuracy of the vertical motor is not extremely important but nevertheless it is good practice to test it. The horizontal motor on the other hand, it is very important to test the accuracy.

	Number of Steps	Theoretical Step Angle	Measured Step Angle
Full Step	100	180 degrees	180 degrees
Quarter Step	100	45 degrees	45 degrees
Sixteenth Step	100	11.25 degrees	11.25 degrees

Table 19. Vertical Motor Microstep Testing

	Number of Steps	Theoretical Step Angle	Measured Step Angle
Full Step	1900	180degrees	180 degrees
Quarter Step	1900	45 degrees	45 degrees
Sixteenth Step	1900	11.25 degrees	11.25 degrees

Table 20. Horizontal Motor Microstep Testing

From the tables above it can be observed that the final angles of rotation in the microcontroller testing have no noticeable difference between the theoretical angle and the actual angle of rotation. This is a fantastic result as this means that the degree desired to be stepped to will be very close to the angle which is required. Also important to note, microstepping does not induce a significant error to the accuracy of the motors. Because of this, it can be

concluded that the project should implement microstepping to smooth the rotation of the motors. This should in theory make smoother motion of the tonearm.

6.3 Computer Vision Testing

Multiple tests of the developed computer vision system will be applied to ensure that the functionality is up to par with the functionality that was described in the requirements and the use case of the system. The results provided from testing will either affirm that the task has been correctly assessed and executed to this point or that there was some error in implementation or design and will provide insight to fix the shortcomings.

6.3.1 Object Recognition

It is critical to ensure that the when a picture of a record label is taken that it is can correctly identified so accurate times and information about the record is pulled from the database. The duty of checking how accurate the computer vision and deep learning system is can be broken down into several assessments. The first and simplest a test of the object detection to confirm that the picture that has been taken and is being analyzed by the deep neural network is indeed a vinyl record. This test will simply involve checking whether an image contains an instance of a vinyl record by passing a training set of images to the neural network and assessing the output of the network to the actual answer. The total percentage of correct answers from the computer vision system is desired to be 85% or above.

6.3.2 Optical Text Recognition (OCR)

A more specialized version of object recognition geared towards words, optical character recognition or OCR is the portion of the computer vision system that will pull all the identifying information on the record label to be analyzed. This means that another major component of this subsystem of the Vinyl Player 2.0 is to check if the text on the record label is being detected and is representing the text accurately. Just as in the object detection there will be a benchmark run on the computer vision system to check what is the accuracy of it, but unlike the previous test this phase of testing will also involve checking that the text that is pulled is the correct text and that it is formatted correctly.

6.3.3 Classification

The final test will be to see if the software can correctly identify the necessary information from record labels to then query the database will be to classify the text that is pulled from the image as artist or album or neither.

7 Project Operation

The following section aims to give clear steps for operating the Vinyl Record 2.0. In creating the project, one goal was to make all user interactions as simple, easy, and intuitive as possible.

7.1 Login & Registration

Logging into the app is as simple as entering in the email and password of the user being logged in as. If the email/password combination doesn't exist in the database, an error will be displayed to the user. It is important to note that the error will not specify whether it was the email or password that were incorrect, as giving this information out could help ill-intentioned users to gain information on how to hack user's accounts. There will also be a button on the login page that allows a first-time user to register. The only information will be collected upon registration is an email and a password. The password will be required to be entered twice to ensure the user correctly entered it. If the account is created successfully, the user will be logged in. Once the user has logged in, a session will be created for the user that will last indefinitely. A session will only be destroyed if the user logs out or logs in on a different device. This will then prompt the user to relogin.

7.2 Cataloging Records

Once the user has logged in, they will be directed to the home screen for that user. The home screen will contain a list of all of the records that have been added to the user's collection. To add a new record to the collection, a record can be searched by pulling the screen down; the search bar will appear at the top of the screen. The search field will allow the user to enter the artist and record name. The results of the search will appear, showing the album's cover art, the artist name, record name, and the year of the release. The user can then tap on one of the listed albums to view more information about the record. Upon tapping on one of the records, the page will be directed to the record detail page. This page will list all of the songs on the record and their durations. Tapping the "Add to Catalog" button at the top will add the record to the user's catalog. Upon pressing the home button, the user would once again be directed to the home screen, which will show their catalog. Their catalog would now include the album that was just added.

7.3 Record Player Setup

To setup the record player, it should simply be place on a flat surface and plugged into the wall. It is important that the record player is placed on a flat,

stable surface in order to prevent the movement of the record from shaking the record player and affecting the sound. Once the record player is plugged in, a record can be placed on the player. To put a record on the record player, the arm of the player should be removed and the record removed from the sleeve. The record should be placed gently such that the hole in the center fits in the silver mounting rod on the record player. Once it has been placed, the arm can be gently lowered onto the desired location on the record and the the record player can be turned on for manual play. The next section will explain how automatic play should be used.

7.4 Queueing a Record Through the App

To start the record player at a specific song on the record, the user should go to the home page and select the record that the user has placed on the record player. From the record's detail page, the song that the user wants to play should be tapped once. The record player arm should then be lifted and calibrated automatically, then moved to the correct spot on the record and lowered gently. The record player will then begin to rotate and start playing. If at any point the user wants to stop the music, they can press the pause button to stop the rotation of the record player platter. If the user wants to play a different song, they can once again tap on the song they would like the play and the arm will once again lift, calibrate, rotate, and lower onto the appropriate spot. If one side of the record has finished playing, the bottom arm will be enacted.

8 Administrative Content

When starting such an extensive project it can be crucial to have predefined guidelines for how much time and money should be spent on each component of the project. This next section will have the initially proposed budget and project milestones, as well as the final cost analysis.

8.1 Initial Budget

The initial breakdown of the items needed for this project, as well as their costs, is outlined below.

Item	Cost Per Item	Quantity	Subtotal
Record Player	**$200**	**1**	**$200**
Arm Motor	**$75 - $150**	**1**	**$75 - $150**
Up & down Motor	**$10 - $30**	**1**	**$10 - $30**
Power Supply	**$20 - $40**	**2**	**$40 - $80**

Controller	**$20 - $40**	**1**	**$20 - $40**
MCU	**$20 - $50**	**1**	**$20 - $50**
Total		**$365 - $550**	

Table 21 – Initial Budget

8.2 Final Cost Analysis

As the project progressed, there was a difference both in cost and the items purchased between the initially proposed budget and what was actually used. Below is a table showing the final items purchased and the total cost of the project.

Item	Cost Per Item	Quantity	Subtotal
Record Player	**$200**	**1**	**$116.41**
Micro – controllers	**$25.57**	**3**	**$76.70**
Motor Driver	**$5.95**	**3**	**$17.85**
Motor Driver (Header Pins Soldered)	**$7.45**	**1**	**$7.45**
Stepper Motor (vertical motion)	**$12.99**	**2**	**$25.98**
Stepper Motor (horizontal motion)	**$47.00**	**1**	**$47.00**
Breadboard and wires	**$10.99**	**1**	**$10.99**
Shipping	**$3.95**	**1**	**$41.95**
Total		**$344.33**	

Table 22 – Final Cost Analysis

8.3 Financing

This project won a grant of $1,000 from Soartech, a company that sponsors one design project each semester. Any costs above $1,000 will be self-financed by the group

8.4 Project Milestones

Below is an outline of all of the milestones to be completed in implementing this project, along with the desired completion date and the actual completion date.

Milestone	Goal Completion Date	Actual Completion Date
Research Project Feasibility	9/15/2017	9/15/2017
Research Project Details	11/3/2017	11/3/2017
Order All Materials	11/15/2017	11/11/2017
Implement Circuit to Power Motors/Controller	11/23/2017	11/13/2017
Finalize Hardware Schematics	1/1/2018	
Finalize Software Design Planning	1/1/2018	
Implement Turntable Motor	1/14/2018	
Implement Motor to Lift Arm	1/14/2018	
Implement Bluetooth Connection with Controller	1/15/2018	
Implement Motor for Radial Motion	1/21/2018	
Implement Calibration and Clutch for Motor	1/21/2018	
Add All Motors to Record Player	2/1/2018	
Complete Application UI	2/1/2018	
Implement Two-Sided Play	3/1/2018	

Finalize Project	3/15/2018	

Table 23 – Project Milestones

9 Conclusion

The conception and design of the Vinyl Player 2.0 has stemmed from the a foreseen issue with the existing record player industry that is not giving a listener the option to enjoy a more casual listening experience and aims to solve this issue through a unique integration of existing analog equipment and forward-thinking technology.

It pursuing this novel solution there was a thorough amount of research to learn about all of the necessary components that goes into a successful implementation of the idea. Using the knowledge gained from research there was a series of design meetings held to discuss topics such as hardware design, software design, part selection, system workflow and supplemental functionality.

With the initial implementation of the Vinyl Player 2.0 agreed upon the team has ordered parts for the hardware portion of the system and begun testing them to assure that they are functional and gain a better understanding of how to control them, which will be critical when integrating the components. On the software side both the initial development of the UI for the mobile application that the user will interface with and the computer vision system that will drive the data that is displayed on the app and used for calculations of movement of the turntable tonearm.

There is an intricate plan in place with goals set at various points in time from now until the showcase that will take place towards the end of the Spring 2018 semester to assure that the development of the product is on time and functioning to specifications. For the software engineers of the team the main objective is have by the end of the year both a fully functional computer vision system that recognizes text on the record label and correctly classifies the artist and album implemented using machine learning and a intuitive mobile app inspired by existing music players that provides users a streamlined interface for utilizing a record player. With the same deadline that the software team has, electrical engineers on the team will have to goal of creating an encasing for the dual needle structure that will be used to play both sides of the record that is on the platter. This is an essential piece of the hardware system to get done early because the usage of the bottom needle is somewhat unprecedented and needs to be analyzed to see what form of adjustments, if any, need to be made.

9.1 Sponsor

SoarTech, an artificial intelligence defense contractor whose Orlando office specializes in intelligent training solutions, is the sponsor of the Vinyl Player 2.0 and has graciously provided the team with 1000 dollars for the expenses of the project. The support given by the Ann Arbor based company has

allowed the team to focus more on the design on the system without the burden of how to source the resources for the components. The physical pieces needed to make up the system were still skillful research and budgeted to avoid overspending and allow leeway in case of error while testing or implementing certain hardware components.

10 Appendices

The following section will contain references to other people's work and diagrams that were discussed, but not included in the report.

10.1 References

Unger, Amy. "Parts of a Record Player." Klipsch, VOXX International, 13 Feb. 2017, www.klipsch.com/blog/parts-of-a-record-player.

Vallejo, Claudio. "Turntable Basics // Belt Drive vs Direct Drive." The Limited Press, The Limited Press, 11 Nov. 2014, http://thelimitedpress.com/turntable-basics-belt-drive-vs-direct-drive.

Weiler, Harold D. "THE WEAR AND CARE OF RECORDS AND STYLI." Shure, Shure, 1954, http://www.shure.com/americas/support/find-an-answer/stylus-wear-and-record-wear.

Euby, et al. "AT-LP60 Fully Automatic Stereo Turntable System." Fully Automatic Stereo Turntable || Audio-Technica US, Audio-Technica U.S., Inc., 10 Mar. 2017, www.audio-technica.com/cms/turntables/9a7f42b88ee1e14b/index.html.

Koolaidwino, et al. "AT-LP3BK Fully Automatic Belt-Drive Stereo Turntable." Fully Automatic Belt-Drive Stereo Turntable || Audio-Technica US, Audio-Technica U.S., Inc., 28 July 2017, www.audio-technica.com/cms/turntables/2e7bb0d91a36a72a/index.html.

Dsg, et al. "AT-LP120BK-USB Direct-Drive Professional Turntable (USB & Analog)." Direct-Drive Professional Turntable (USB & Analog) with AT95E Cartridge || Audio-Technica US, Audio-Technica U.S., Inc., 24 Jan. 2017, www.audio-technica.com/cms/turntables/1468a7a181c8627f/index.html.

"MINIMUM SYSTEM REQUIREMENTS." TT250USB Professional DJ Direct Drive Turntable | Numark, InMusic, Inc., 2017, www.numark.com/product/tt250usb.

"T.62 M2." Stanton T.62 M2 - DJ Equipment, DJ Gear, Phono Cartridges & Needles, DJ Mixer, DJ Turntables, Headphones, CD Players, GIBSON GUITAR CORPORATION, 2012, www.stantondj.com/stanton-turntables/t62.html.

B. Popper, "Google announces over 2 billion monthly active devices on Android," The Verge, 17 May 2017. [Online]. Available: https://www.theverge.com/2017/5/17/15654454/android-reaches-2-billion-monthly-active-users. [Accessed 2 November 2017].

N. Statt, "1 billion Apple devices are in active use around the world," The Verge, 26 January 2016. [Online]. Available: https://www.theverge.com/2016/1/26/10835748/apple-devices-active-1-billion-iphone-ipad-ios. [Accessed 2 November 2017].

Wikipedia, "Windows Phone," Wikipedia, 2 November 2017. [Online]. Available: https://en.wikipedia.org/wiki/Windows_Phone. [Accessed 2017 November 2].

M. Allison, "Microsoft has now sold over 100 million Windows Phones," mspoweruser, 21 July 2015. [Online]. Available: https://mspoweruser.com/microsoft-now-sold-100-million-windows-phones/. [Accessed 2 November 2017].

Wikipedia, "Windows 10 Mobile," Wikipedia, 19 October 2017. [Online]. Available: https://en.wikipedia.org/wiki/Windows_10_Mobile. [Accessed 2 November 2017].

K. Noyes, "10 Reasons Open Source is Good for Business," PCWorld, 5 November 2010. [Online]. Available: https://www.pcworld.com/article/209891/10_reasons_open_source_is_good_for_business.html. [Accessed 2 November 2017].

GitHub, "GitHub Features," GitHub, 2017. [Online]. Available: https://github.com/features. [Accessed 2 November 2017].

Microsoft, "Use Team Foundation Version Control," Microsoft, 29 August 2017. [Online]. Available: https://docs.microsoft.com/en-us/vsts/tfvc/overview. [Accessed 2 November 2017].

Perforce, "All Your Repos In One Space," Perforce, 2017. [Online]. Available: https://www.perforce.com/products/helix-teamhub. [Accessed 2 November 2017].

Amazon, "AWS CodeCommit," Amazon, 2017. [Online]. Available: https://aws.amazon.com/codecommit/. [Accessed 2 November 2017].

Android, "Meet Android Studio," Android, 2017. [Online]. Available: https://developer.android.com/studio/intro/index.html. [Accessed 2 November 2017].

Jet Brains, "IntelliJ IDEA," Jet Brains, 2017. [Online]. Available: https://www.jetbrains.com/idea/. [Accessed 2 November 2017].

Android, "Run Apps on the Android Emulator," Android, 2017. [Online]. Available: https://developer.android.com/studio/run/emulator.html. [Accessed 2 November 2017].

Android, "Test Your App," Android, 2017. [Online]. Available: https://developer.android.com/studio/test/index.html. [Accessed 2 November 2017].

Android, "Build a UI with Layout Editor," Android, 2017. [Online]. Available: https://developer.android.com/studio/write/layout-editor.html. [Accessed 2 November 2017].

Bluetooth, "what is Bluetooth?," Bluetooth, 2017. [Online]. Available: https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works. [Accessed 2 November 2017].
Wikipedia, "Near-field communication," Wikipedia, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Near-field_communication. [Accessed 2 November 2017].

Square, Inc. Property, "Near Field Communication versus Bluetooth," Sqaure, Inc. Property, 2017. [Online]. Available: http://nearfieldcommunication.org/bluetooth.html. [Accessed 2 November 2017].

D. Nosowitz, "Everything You Need to Know About Near Field Communication," Popular Science, 1 March 2011. [Online]. Available: https://www.popsci.com/gadgets/article/2011-02/near-field-communication-helping-your-smartphone-replace-your-wallet-2010. [Accessed 2 November 2017].

C. F. &. J. Layton, "How Bluetooth Works," howstuffworks, 2017. [Online]. Available: https://electronics.howstuffworks.com/bluetooth4.htm. [Accessed 2 November 2017].

J. B. M. B. M. H. R. S. L. C. K. S. John Padgette, "Guide to Bluetooth Security," NIST, May 2017. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-121r2.pdf. [Accessed 2017 November 2017].

"Bluetooth Transmission Technology," [Online]. Available: https://theses.lib.vt.edu/theses/available/etd-11182002-115502/unrestricted/Chapter2.pdf. [Accessed 2 November 2017].

Wikipedia, "Bluetooth," Wikipedia, 2 November 2017. [Online]. Available: https://en.wikipedia.org/wiki/Bluetooth#Bluetooth_5. [Accessed 2 November 2017].

M. Shiekh, "Bluetooth Vs NFC Vs Wifi Direct," Phone Guru Reviews, 20 November 2013. [Online]. Available: http://www.phonegurureviews.com/bluetooth-nfc-wifi-direct/. [Accessed 2 November 2017].

[Online].
Android, "Near Field Communication," Android, 2017. [Online]. Available: https://developer.android.com/guide/topics/connectivity/nfc/index.html. [Accessed 2 November 2017].

Android, "Creating P2P Connections with Wi-Fi," Android, 2017. [Online]. Available: https://developer.android.com/training/connect-devices-wirelessly/wifi-direct.html. [Accessed 2 November 2017].

WiFi Alliance, "Wi-Fi Direct," WiFi Alliance, 2017. [Online]. Available: https://www.wi-fi.org/discover-wi-fi/wi-fi-direct. [Accessed 2 November 2017].

B. Clark, "The Differences Between Bluetooth 4.0 and Wi-Fi Direct You Need to Know," MUO, 13 July 2015. [Online]. Available: http://www.makeuseof.com/tag/the-differences-between-bluetooth-4-0-and-wi-fi-direct-you-need-to-know/. [Accessed 2 November 2017].

Android, "android.bluetooth," Android, 2017. [Online]. Available: https://developer.android.com/reference/android/bluetooth/package-summary.html. [Accessed 2 November 2017].

Android, "Bluetooth Low Energy," Android, 2017. [Online]. Available: https://developer.android.com/guide/topics/connectivity/bluetooth-le.html. [Accessed 2 November 2017].

Android, "Bluetooth," Android, 2017. [Online]. Available: https://developer.android.com/guide/topics/connectivity/bluetooth.html. [Accessed 2 November 2017].

Bluetooth SIG, "Core Specifications," Bluetooth SIG, 2017. [Online]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification. [Accessed 2 November 2017].

J. G. Sponas, "Things you should know about Bluetooth range," GetConnectedBlog, 2 June 2016. [Online]. Available: http://blog.nordicsemi.com/getconnected/things-you-should-know-about-bluetooth-range. [Accessed 2 November 2017].

K. Ren, "Ten Important Differences Between Bluetooth BREDR And Bluetooth Smart," Bluetooth, 6 October 2015. [Online]. Available: https://blog.bluetooth.com/ten-important-differences-between-bluetooth-bredr-and-bluetooth-smart. [Accessed 2 November 2017].

R. Heydon, "Technology Introduction," Bluetooth SIG, 2009. [Online]. Available: https://www.bluetooth.org/OTV/3-TechnologyIntroduction/?_ga=1.90884376.320705699.1462460049. [Accessed 2 November 2017].

R. Heydon, "What is Bluetooth low energy technology," Bluetooth SIG, 2009. [Online]. Available: https://www.bluetooth.org/OTV/1-Whatis/. [Accessed 2 November 2017].

Bluetooth SIG, "develop with Bluetooth," Bluetooth SIG, 2017. [Online]. Available: https://www.bluetooth.com/develop-with-bluetooth/developer-resources-tools/developer-training-videos. [Accessed 2 November 2017].

J. B. M. B. M. H. John Padgette, "Archived NIST Technical Series Publication," NIST, 5 May 2017. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-121r1.pdf. [Accessed 17 November 2017].

D. Nosowitz, "Everything You Need to Know About Near Field Communication," Po
pular Science , 1 May 2011. [Online]. Available:
https://www.popsci.com/g00/gadgets/article/2011-02/near-field-communication-helping-your-smartphone-replace-your-wallet-2010?i10c.encReferrer=#page-3. [Accessed 11 November 2017].

Perforce, "Helix-TeamHub," Perforce Company, [Online]. Available: https://www.perfor
ce.com/products/helix-teamhub/pricing. [Accessed 11 November 2017].

StackShare, "GitHub vs. AWS CodeCommit vs. GitBucket," StackShare, [Online]. Available: https://stackshare.io/stackups/aws-codecommit-vs-gitbucket-vs-github. [Accessed 17 November 2017].

Microsoft, "Choosing the right version control for your project," Microsoft, 12 May 2017. [Online]. Available: https://docs.microsoft.com/en-us/vsts/tfvc/comparison-git-tfvc. [Accessed 17 November 2017].

Wikipedia, "Team Foundation Server," Wikipedia, 17 November 2017. [Online]. Available: https://en.wikipedia.org/wiki/Team_Foundation_Server. [Accessed 17 Novmeber 2017].

Microsoft, "Permissions and groups in VSTS and TFS," Microsoft, 17 October 2017. [Online]. Available: https://docs.microsoft.com/en-us/vsts/security/permissions. [Accessed 17 November 2017].

Microsoft, "Visual Studio Team Services," Microsoft, 2017. [Online]. Available: https://azure.microsoft.com/en-us/services/visual-studio-team-services/. [Accessed 17 November 2017].

Android, "Camera API," Android, 2017. [Online]. Available: https://developer.android.com/guide/topics/media/camera.html. [Accessed 17 November 2017].

Android, "Core App Quality," Android, 2017. [Online]. Available: https://developer.android.com/develop/quality-guidelines/core-app-quality.html. [Accessed 11 November 2017].

Google, "Android 7.0 Nougat," Google, [Online]. Available: https://www.android.com/versions/nougat-7-0/. [Accessed 17 November 2017].

Android, "AOSP Java Code Style for Contributors," Android, 27 November 2017. [Online]. Available: https://source.android.com/setup/code-style. [Accessed 3 December 2017].

Google, "Google Java Style Guide," Google , [Online]. Available: https://google.github.io/styleguide/javaguide.html. [Accessed 3 December 2017].

Android, "Add C and C++ Code to Your Project," Android, 2017. [Online]. Available: https://developer.android.com/studio/projects/add-native-code.html. [Accessed 3 December 2017].

Android, "Connect to Firebase," Android, 2017. [Online]. Available: https://developer.android.com/studio/write/firebase.html. [Accessed 3 December 2017].

Firebase, "Firebase by platform," Firebase, 2017. [Online]. Available: https://firebase.google.com/docs/. [Accessed 3 December 2017].

Android Studio, "Android Lint," Android, 15 November 2011. [Online]. Available: http://tools.android.com/tips/lint. [Accessed 3 December 2017].

Android, "Improve Your Code with Lint," Android, 2017. [Online]. Available: https://developer.android.com/studio/write/lint.html. [Accessed 3 December 2017].

Android, "Add Multi-Density Vector Graphics," Android, 2017. [Online]. Available: https://developer.android.com/studio/write/vector-asset-studio.html. [Accessed 3 December 2017].

Android, "Inspect Network Traffic with Network Profiler," Android, 2017. [Online]. Available: https://developer.android.com/studio/profile/network-profiler.html. [Accessed 3 December 2017].

Android, "Inspect GPU Rendering Speed and Overdraw," Android, 2017. [Online]. Available: https://developer.android.com/studio/profile/inspect-gpu-rendering.html. [Accessed 3 December 2017].

Android, "Profile Battery Usage with Batterystats and Battery Historian," Android, 2017. [Online]. Available: https://developer.android.com/studio/profile/battery-historian.html. [Accessed 3 December 2017].

Android, "View the Java Heap and Memory Allocations with Memory Profiler," Android, 2017. [Online]. Available: https://developer.android.com/studio/profile/memory-profiler.html. [Accessed 3 December 2017].

Android, "Inspect CPU Activity and Method Traces with CPU Profiler," Android, 2017. [Online]. Available: https://developer.android.com/studio/profile/memory-profiler.html. [Accessed 3 December 2017].

QualCommm, "Snapdragon 835 Mobile Platform," Qualcommm, 2017. [Online]. Available: https://www.qualcomm.com/products/snapdragon/processors/835. [Accessed 3 December 2017].

Christ Smith, "Galaxy S8 and Galaxy S8+: The full specs," BGR, 17 March 2017. [Online]. Available: http://bgr.com/2017/03/29/galaxy-s8-plus-specs-official/. [Accessed 3 December 2017].

Android, "JNI Tips," Android, 2017. [Online]. Available:

https://developer.android.com/training/articles/perf-jni.html. [Accessed 3 December 2017].

Android, "Legal Notice," Android, 2017. [Online]. Available:
https://developer.android.com/legal.html. [Accessed 3 December 2017].

“CC2540F128RHAT - TI store,” Texas Instruments | TI Store. [Online]. Available:
https://store.ti.com/CC2540F128RHAT.aspx. [Accessed: 03-Dec-2017].

“MSP430F4618 (ACTIVE),” MSP430F4618 16-Bit Ultra-Low-Power MCU, 116KB Flash, 8KB RAM, 12-Bit ADC, DMA, 160 Seg LCD | TI.com. [Online]. Available:
http://www.ti.com/product/MSP430F4618. [Accessed: 03-Dec-2017].

eZ. Systems, “nRF52832,” nRF52832 / Bluetooth low energy / Products / Home - Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR. [Online]. Available:
https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52832. [Accessed: 03-Dec-2017].

“CC2564 (ACTIVE),” CC2564 Dual-mode Bluetooth® Controller | TI.com. [Online]. Available:
http://www.ti.com/product/CC2564/description. [Accessed: 03-Dec-2017].
“Applications,” ESP8266EX Overview | Espressif Systems. [Online]. Available:
http://espressif.com/en/products/hardware/esp8266ex/overview. [Accessed: 03-Dec-2017].

“PAGINA PRINCIPAL,” Electrnica Estudio | Ingeniera Electrnica y Proyectos PICmicro®. [Online].
Available: http://www.electronicaestudio.com/docs/istd016A.pdf. [Accessed: 03-Dec-2017].

“Arduion Uno Rev3,” Arduino Uno Rev3, 2007. [Online]. Available:
https://store.arduino.cc/usa/arduino-uno-rev3. [Accessed: 03-Dec-2017].

Thomas A. Kinney • Application Engineer • Baumer Electric | Sep 01, 2001, “Proximity Sensors Compared: Inductive, Capacitive, Photoelectric, and Ultrasonic,” Machine Design, 13-Mar-2017. [Online]. Available:
http://www.machinedesign.com/sensors/proximity-sensors-compared-inductive-capacitive-photoelectric-and-ultrasonic. [Accessed: 03-Dec-2017].

www.cycon.de S. cy:con, “OT 18 M 1000 P4-B4 Diffuse Sensor - di-soric,” Sensors, Lighting and Accessories - di-soric. [Online]. Available:
https://www.di-soric.com/en/OT-18-M-1000-P4-B4-31856.html?pdb_kategorie=1523. [Accessed: 03-Dec-2017].

Pepperl Fuchs, “Diffuse Mode Sensors,” Pepperl Fuchs, 01-Dec-2017. [Online]. Available:
https://www.pepperl-fuchs.us/usa/en/classid_47.htm?view=productdetails&prodid=1996. [Accessed: 03-Dec-2017].

E. Huet, “Resurgence In Vinyl Records Means Booming Business -- And Growing Pains -- For Factories,” Forbes, 10-Jul-2015. [Online]. Available: https://www.forbes.com/sites/ellenhuet/2015/07/08/resurgence-in-vinyl-records-means-booming-business-and-growing-pains-for-factories/#40f206d5f380. [Accessed: 03-Dec-2017].

“RokBlok - A Different Spin on Vinyl,” RokBlok - A Different Spin on Vinyl. [Online]. Available: http://rokblok.co/. [Accessed: 03-Dec-2017].

“Home • LOVE TURNTABLE,” LOVE TURNTABLE. [Online]. Available: http://www.loveturntable.com/. [Accessed: 03-Dec-2017].

Redbear, “redbear/nRF5x,” GitHub, 27-Nov-2017. [Online]. Available: https://github.com/redbear/nRF5x/tree/master/nRF52832. [Accessed: 03-Dec-2017].

“Server-side web frameworks,” Mozilla Developer Network. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks. [Accessed: 03-Dec-2017].

“Ruby on Rails: What It Is and Why We Use It For Web Applications | Bit Zesty | London UK,” Bit Zesty, 09-Nov-2017. [Online]. Available: https://bitzesty.com/2014/03/03/ruby-on-rails-what-it-is-and-why-we-use-it-for-web-applications/. [Accessed: 03-Dec-2017].

“Standard for Audio, Video and Similar Electronic Apparatus - Safety Requirements,” Standards Catalog. [Online]. Available: https://standardscatalog.ul.com/standards/en/standard_60065_7. [Accessed: 03-Dec-2017].

“C Coding Standard,” C Coding Standard. [Online]. Available: https://users.ece.cmu.edu/~eno/coding/CCodingStandard.html#documentation. [Accessed: 03-Dec-2017].

“C Programming Standards,” C programming standards. [Online]. Available: http://emboss.sourceforge.net/developers/c-standards_new.html#ANSIStandardC. [Accessed: 03-Dec-2017].

SFUptownMaker, “PCB Basics,” PCB Basics. [Online]. Available: https://learn.sparkfun.com/tutorials/pcb-basics. [Accessed: 03-Dec-2017].

E. Engineering, “Electrosoft Engineering,” Concepts and Terminology used in Printed Circuit Boards (PCB) -Electrosoft Engineering, 2010. [Online]. Available: http://www.pcb.electrosoft-engineering.com/04-articles-custom-system-design-and-pcb/01-printed-circuit-board-concepts/printed-circuit-board-pcb-concepts.html. [Accessed: 03-Dec-2017].

“Surface Mount vs Through Hole Printed Circuit Boards,” pcboardrework, 25-Aug-2016. [Online]. Available: http://www.pcboardrework.com/through-hole-vs-surface-mount/. [Accessed: 03-Dec-2017].

V. Marshall, “Build a Simple DC Power Supply,” Popular Science, 02-Feb-2010. [Online]. Available: https://www.popsci.com/diy/article/2009-12/build-simple-dc-power-supply#page-4. [Accessed: 03-Dec-2017].

“A Guide to TF Layers: Building a Convolutional Neural Network | TensorFlow.” TensorFlow, www.tensorflow.org/tutorials/layers. [Accessed: 03-Dec-2017].

Ujjwalkarn. “An Intuitive Explanation of Convolutional Neural Networks.” The data science blog, 28 May 2017, ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/.

Viola, P., and M. Jones. “Rapid object detection using a boosted cascade of simple features.” Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 2001, doi:10.1109/cvpr.2001.990517

“About.” About - Point Cloud Library (PCL), pointclouds.org/about/.

Jackel, L. D., et al. “Optical Character Recogntion For Automatic Teller Machines.” Industrial Applications of Neural Networks, 1998, pp. 375–378., doi:10.1142/9789812816955_0044

“Deep Learning.” "CBLL, Research Projects, Computational and Biological Learning Lab, Courant Institute, NYU", cs.nyu.edu/~yann/research/deep/.

“List of online music databases.” Wikipedia, Wikimedia Foundation, 23 Nov. 2017, en.wikipedia.org/wiki/List_of_online_music_databases.

“About Discogs.” About Us, www.discogs.com/about.

“SimpleCV Tutorial.” SimpleCV Tutorial — Tutorial, tutorial.simplecv.org/en/latest/.

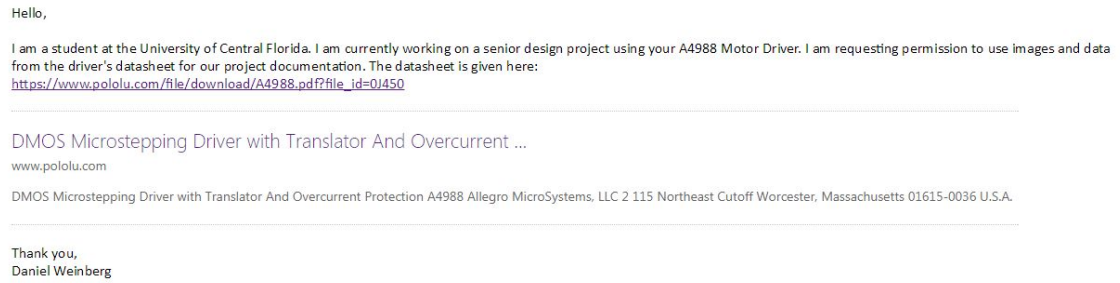
“Welcome.” Welcome — Theano 1.0.0 documentation, deeplearning.net/software/theano/.

“OpenCV.” Wikipedia, Wikimedia Foundation, 21 Nov. 2017, en.wikipedia.org/wiki/OpenCV.

“MATLAB.” MathWorks, www.mathworks.com/products/matlab.html.

10.2 Copyright Permissions

This is the email sent to Allegro MicroSystems requesting use of datasheet items for project report. No response was given.



Hello,

I am a student at the University of Central Florida. I am currently working on a senior design project using your A4988 Motor Driver. I am requesting permission to use images and data from the driver's datasheet for our project documentation. The datasheet is given here:
https://www.pololu.com/file/download/A4988.pdf?file_id=0J450

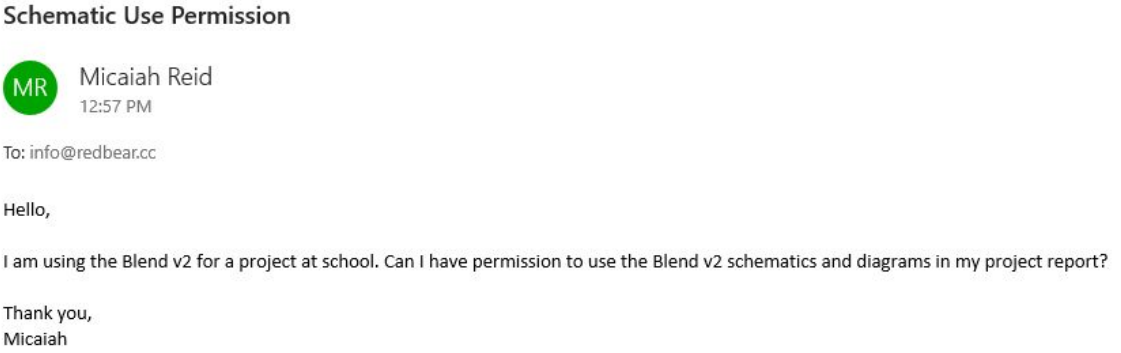
DMOS Microstepping Driver with Translator And Overcurrent ...
www.pololu.com

DMOS Microstepping Driver with Translator And Overcurrent Protection A4988 Allegro MicroSystems, LLC 2 115 Northeast Cutoff Worcester, Massachusetts 01615-0036 U.S.A.

Thank you,
Daniel Weinberg

Figure 24. A4988 Permission Request

This is the email sent to Redbear Labs requesting the use of datasheet items for the project report. No response was given.



Schematic Use Permission

MR Micaiah Reid
12:57 PM

To: info@redbear.cc

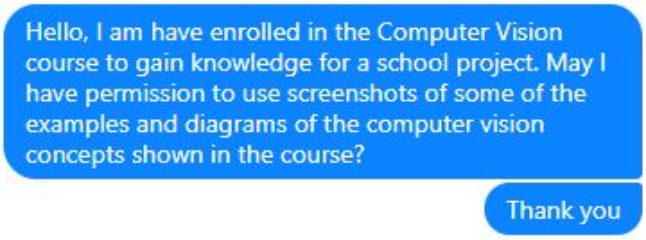
Hello,

I am using the Blend v2 for a project at school. Can I have permission to use the Blend v2 schematics and diagrams in my project report?

Thank you,
Micaiah

Figure 25 – Redbear Labs Permission Request

This is a message sent to the SuperDataScience team, that are the instructors for the Computer Vision A-Z udemy course, asking for permission to utilize some a screenshot from the course. This is not an email because the website of the company did not provide an email on their website and link their 'Contact Us' page to Facebook Messenger. There is no response as of yet.



Hello, I am have enrolled in the Computer Vision course to gain knowledge for a school project. May I have permission to use screenshots of some of the examples and diagrams of the computer vision concepts shown in the course?

Thank you

Figure 26 – SuperDataScience Permission Request