Smart Vinyl: The First Computerized

Thesis in the Field of

Software Engineering

In Partial Fulfillment of the Requirements

For a Masters of Liberal Arts Degree

Harvard University

Extension School

Submission Date: August 20, 2018

Gerald Trotman

geraldtrotman@g.harvard.edu

Proposed Start Date: June 1st 2017

Anticipated Date of Graduation: November 2018

Thesis Director: Stratos Idreos

Abstract

DJ systems today are over engineered to the point that the record, the stylus, and even the turntable itself have been left out of the design. The problem then becomes that in removing these components from DJing equipment, DJs can no longer execute the techniques that make them turntablists. To solve this, I have fitted the components of a tablet computer and embedded it inside a time coded vinyl record allowing DJs to both source and control audio files on their existing turntable decks. It runs open source Digital Vinyl System (DVS) software from within the time coded vinyl record, providing the same look and feel of performing with vinyl records.

In reading further about my solution to this problem, you will understand the motivation that went into my technological solution for turntablists as well as the granular detail behind its design.

Contents

List of Figures

List of Tables

List of Code

Chapter 1: Thesis Project Description

While advancements in music technology have improved DJing products in some respects, I will demonstrate that there are specific areas where it has fallen short. I will even go as far as to prove that music technology has compromised DJs who subscribe to turntablism. As the technological focus continues to shift away from vinyl records, the stylus, and even the turntable itself, the problem arises where DJ equipment is simply no longer designed for turntablists.

What makes this problem significant is that it sheds light on the disconnect between the music technology industry and the very core user base that created the demand in the first place. Turntablism techniques span as far as the DJs wildest imagination, yet paradoxically, their constrains magnify with every innovation.

Albeit, the music technology industry is posed with the difficult task of generating revenue by solving the problem of how to get DJs to buy new product. However, therein lies a tougher, more encompassing question. How do we build and improve upon the products that DJs already have?

Thus far, technological solutions for DJs have made strides consolidating functionality in an effort to make more standalone systems. Nonetheless, how the music technology industry plans to innovate without turntablism remains an open problem for as long as the record, the stylus, and the turntable are omitted from the design.

Smart Vinyl introduces a new platform to the world of DJing by putting a computer inside a vinyl record. In doing so, it will serve as a standalone solution for sourcing audio playback and visualization specifically designed with the stylus and turntable in mind.

This section goes into the motivation of this project, some background, hardware, and software of the *Smart Vinyl* system.

1.1. Background of the Project

1.1.1 A Brief History of Turntablism, DJing Systems and their Evolution

The word *turntablist* was originated by Luis “DJ Disk” Quintanilla (Primus, Herbie Hancock, Invisibl Skratch Piklz). After a phone conversation with Disk, it was later “coined” and used in 1995 by DJ Babu to describe the difference between a DJ who simply plays and mixes records and one who performs by physically manipulating the records, stylus, turntables, turntable speed controls and mixer to produce new sounds (Turntablism (nd)).

The emergence of turntablism DJing systems corresponded with the rise of hip hop culture in New York City during the early 1970s. At that time and for the years that followed, being a professional DJ meant competing with two turntables and traveling with crates of vinyl records. DJs would put together countless routines that involved placing physical stickers on records to mark where in the song they wanted to “scratch” a particular sound as it was being blended into another song.

Then, in 2001, vinyl emulation software and hardware packages began to be publicly sold, making it such that DJs no longer needed to bring crates of records around to perform. They simply needed a computer which housed a music library and provided a way of visualizing a linear audio wave file. This would prove to be a huge technological step forward for both DJs and turntablists alike. However, since this evolution in DJ technology from crates of records to control vinyl synchronized to music libraries on computers, I have observed DJ technology advancements slowly remove both the vinyl record and the computer, replacing them with knobs, buttons, switches, touch screens and CDs. Despite these “advancements,” professional DJs still used control vinyl on the same signature Technics SL-1200 turntables from the 1970s along with their laptops (White, D (l 27)).

1.1.2 The Return of Turntables, Vinyl, and the need for 21st Century Turntablism Solutions

Meanwhile, millennials and their purchasing power and habits have given rise to a return to some classic products that were once considered obsolete. Vinyl records sales outpaced the revenue of digital downloads for the first time ever in 2016 (Heath, T. (er 7)). The Technics SL-1200 also went back into production in 2016 after being discontinued in 2010 (Technics SL-1200 (nd)). The Berklee School of Music began teaching turntablism as a formal instrument. All of these changes speak to the fact that the classic art of turntablism is also ready to make its return and that now is the time for a new DJing platform aimed at DJs who are or who want to be classic turntablists. As an engineer and an avid DJ enthusiast, I believed there was a way for technology to reflect the needs of the DJing culture rather than succumb to the technology driving the “innovations” and imposing their technological “advancements” on the culture. The market was ripe for an updated “old school” product like a traditional turntablist system that incorporates new technology without comprising the artistry of a classic turntablist DJ.

1.1.3 *Smart Vinyl* - The DJ System for 21st Century Turntablists

There had been some attempts to fill this market need by companies such as Hewlett Packard, as well as researchers at Stanford University. The current trend of DVS solutions pushed touch screens displaying audio wave files built directly into the turntable and mixer housing onto their users. However, in spite of the steady increase in vinyl enthusiast nostalgia, all of these newer technological solutions source audio playback directly from digital storage options, completely sidestepping vinyl. As audio files would no longer be sourced from turntables decks, DJs, turntablists especially, would be forced to purchase entirely new equipment.

This project introduced a new platform – *Smart Vinyl* - to the world of turntablist DJing that bridged these two trends – touchscreens and vinyl. *Smart Vinyl* is a tablet computer running Digital Vinyl System (DVS) software embedded inside a time coded vinyl record operating autonomously on a turntable. My solution allows DJs to both source and control audio files on their existing turntable decks, providing the same look and feel of a vinyl record, eliminating the need for laptops and other unnecessary equipment, and marking a return to the purist form of DJing using only a turntable and vinyl record. Many DJs who care about the history, artistry, and culture of turntablist DJing had expressed their desire for this product as it allows them to further consolidate their equipment without compromising turntablism execution and redirect their entire focus back on the turntable and crowd engagement, which they are currently unable to do with existing DJing equipment.

With the primary objective refined, the secondary objective will be to add to the current repertoire of turntablist techniques and instrumentation. One example would be to include the installation of music creation applications on the Smart Vinyl platform that would introduce the ability to play the notes of a musical instrument digitally while performing turntablist techniques on those notes on the fly. I expect the use case scenarios for this platform to be expansive.

1.1.4 Related Work

There was some early research regarding the ongoing evolution of digital vinyl systems. Two graduate students from Stanford University wrote a similar research paper called "Two Turntables and a Mobile Phone," a DVS setup which used mobile phones in place of time coded vinyl records to both view and manipulate audio wave files leveraging gyroscope and accelerometer data (Bryan, N.J and Wang, G. (2011). Hewlett Packard, the popular computer company, tried their hand in the evolution of DJing with the product called Djammer (Slayden et al. (2005)).

There are a variety of DVS products on the market today that have all incorporated touch screens, which are now being built into decks and mixers. In many products, touch screens are now the entire DVS device itself. DJing equipment has been shifting to a more autonomous approach evolving from the dependency of both laptops and turntables to a single standalone product. Figure 1.1 is an example of a DVS deck with a LCDs built into their designed by Denon. However, there are various comparable products from manufacturers such as the XDJ-1000 MK2 by Pioneer and the NS7III by Numark. But it isn't just the decks themselves which are now coming equipped with LCDs built in. DJ mixers are also starting to become manufactured with built in LCDs. Figure 1.2 is an example of a relatively newer manufacturer called Thud Rumbler that introduced this product to the market. Inevitably, brands like Traktor and Pioneer ensued with similar products such as the S5 and the DJM-Tour1. Eventually, technology escalated the DVS design even further such that there was no need for turntable decks or mixers by putting out an entirely tablet based solution that runs the DVS software in place of vinyl. Figure 1.3 is a product built by Kontrol Surface. However, iOS apps such as djay Pro and DJ Player Pro are comparable for those who think of DJ equipment as simply an iPad or an iPhone.

Conversely, Smarty Vinyl was predicated from the *Do-It-Yourself* school of thought and the turntablists who used reverse engineering and micro controllers to do so. It came from this need to create simplified DVS systems using open source DVS software instead of the proprietary DVS software and computers for that matter. Figure 1.4 and Figure 1.5 demonstrates a turntable hack that pushes back on their replacement with LCDs by building the audio playback functionality into the button controls on the turntable.

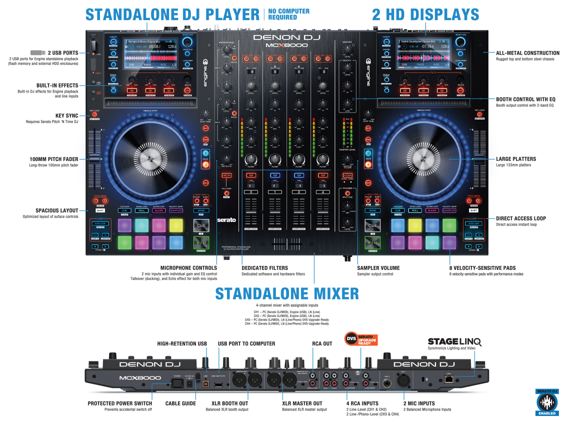

STANDALONE DJ PLAYER
NO COMPUTER REQUIRED
2 HD DISPLAYS
DENON DJ
MCX8000
2 USB PORTS
2 USB ports for Engine standalone playback (flash memory and external HDD enclosures)
BUILT-IN EFFECTS
Built-in DJ effects for Engine playback and line inputs
KEY SYNC
Requires Serato Pitch 'N Time DJ
100MM PITCH FADER
Long-throw 100mm pitch fader
SPACIOUS LAYOUT
Optimized layout of suface controls
ALL-METAL CONSTRUCTION
Rugged top and bottom steel chassis
BOOTH CONTROL WITH EQ
Booth output control with 2-band EQ
LARGE PLATTERS
Large 133mm platters
DIRECT ACCESS LOOP
Direct access instant loop
MICROPHONE CONTROLS
2 mic inputs with individual gain and EQ control
Talkover (ducking), and Echo effect for both mic inputs
DEDICATED FILTERS
Dedicated software and hardware filters
SAMPLER VOLUME
Sampler output control
8 VELOCITY-SENSITIVE PADS
8 velocity-sensitive pads with performance modes
STANDALONE MIXER
4-channel mixer with assignable inputs
CH1 – PC (Serato DJ/MIDI), Engine (USB), LN (Line)
CH2 – PC (Serato DJ/MIDI), Engine (USB), LN (Line)
CH3 – PC (Serato DJ/MIDI), LN (Line/Phono) DVS-Upgrade-Ready
CH4 – PC (Serato DJ/MIDI), LN (Line/Phono) DVS-Upgrade-Ready
HIGH-RETENTION USB
USB PORT TO COMPUTER
RCA OUT
DVS SERATO UPGRADE READY
STAGELINQ
Synchronize Lighting and Video
PROTECTED POWER SWITCH
Prevents accidental switch off
CABLE GUIDE
XLR BOOTH OUT
Balanced XLR booth output
XLR MASTER OUT
Balanced XLR master output
4 RCA INPUTS
2 Line-Level (CH1 & CH2)
2 Line-/Phono-Level (CH3 & CH4)
2 MIC INPUTS
2 Balanced Microphone Inputs
SERATO DJ ENABLED

Figure 1.1: The Denon MCX8000

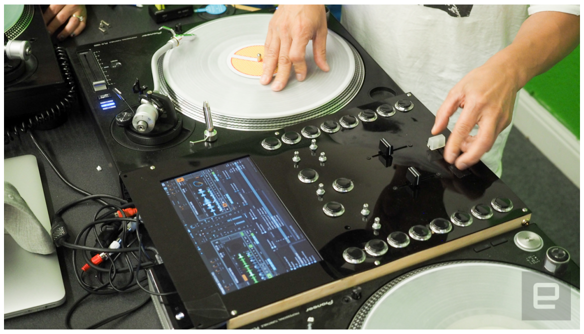


Figure 1.2: The Thud Rumble Invader Mixer

Figure 1.3: The Kontrol Surface 32

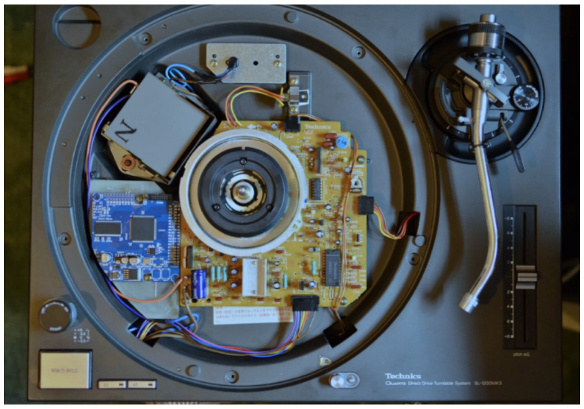


Figure 1.4: An XILINK FPGA board embedded audio playback system for Technics 1200 Mod

Figure 1.5: The button control for various modes and menus of Technics 1200 Mod

Most of these systems provided a means to select from an audio library and display audio wave files. Some of these options however decoupled displaying the audio from the source of playback which requires more cabling (in the case of DVS mixers with built in screens). Some options have eliminated displaying audio files entirely in spite of being wireless solutions (in the case of DJammer from HP). One thing these devices all seem to share is that they source audio playback directly from digital storage options, completely sidestepping vinyl and thereby compromise the look and feel of a spinning vinyl record. On the contrary, *Smart Vinyl* sources the audio playback from the timecode of a vinyl record, wirelessly synchronizing what the DJ hears audibly to that of what the DJ sees visibly from an audio wave file. The entire attention of the DJ is focused on the turntable, just like with classic turntablism.

Chapter 2: Description of the Hardware and Software

In order to support the added feature of sourcing the audio playback from a time code, one needs a physical timecode layer from which to read the audio file. *Smart Vinyl* is comprised of a clear timecode control vinyl overlay. Furthermore, in order to represent the exact position of the needle placed within the grooves of that timecode layer as a representation of the exact position within an audio wave file, it requires an analog position to be converted into a digital one. To do so, I equipped *Smart Vinyl* with a single board computer running the Linux Operating System. There is an LCD to provide visual feedback for selecting and viewing audio wave file as they otherwise would on a computer, two other key features. *Smart Vinyl* is also outfitted with sensors and radios for detecting rotation providing wireless communication to other devices and databases. Then, with the hardware modifications made and the sensors added the next step in the process was to implement the software solution. The idea was that by using non proprietary software solutions, I could set a foundation for other developers to contribute solutions and to future proof Smart Vinyl. Conceptually, there are two main components to the software system. The first was the lower-level sensor interface. The second was the higher-level system, responsible for the DVS.

2.1. Time Coded Vinyl Overlay

The time-coded vinyl overlay is the heart and soul of this project. It allows users to “physically manipulate the playback of digital audio files using the turntables as the interface, preserving the hands-on control and feel of scratching with vinyl” (Vinyl Emulation Software (nd)). It is the singlehanded reason why turntablists still participate in the advancements of music technology to this day.

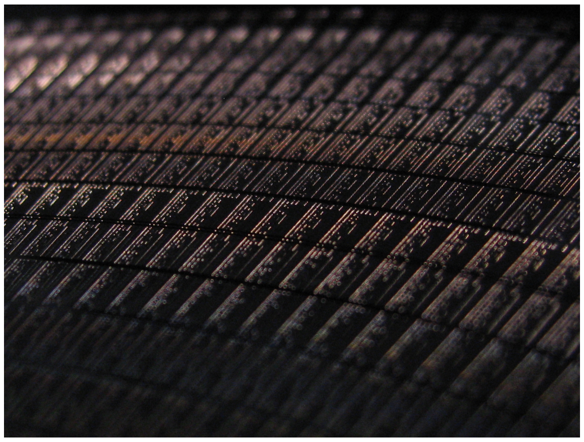


Figure 2.1: Timecode etched on vinyl record

Because of vinyl enthusiast nostalgia, revenue from sale of vinyl records have far surpassed that of even digital downloads. Yet, the current trend of DVS solutions today are pushing touch screens displaying audio wave files built directly into the turntable and mixer housing onto their new users. One of the main objectives of *Smart Vinyl* was to bridge these two trends.

The process of how vinyl records, particularly master records, have been made for decades, is as follows: take a large aluminum disc, coat it with a veneer of lacquer, and then dry it. Once dried, it is taken to a machine that etches the recording of the analog audio signals often referred to as timecode. The Discovery Channel did a two part segment of How It’s Made on vinyl records that explores the complete process

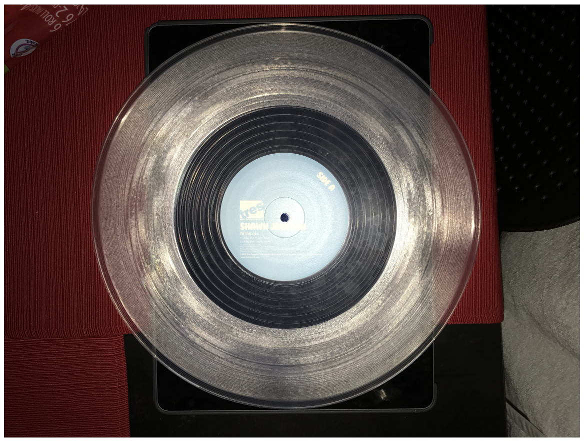


Figure 2.2: Clear vinyl record overlaying 12” iPad Pro for size comparison

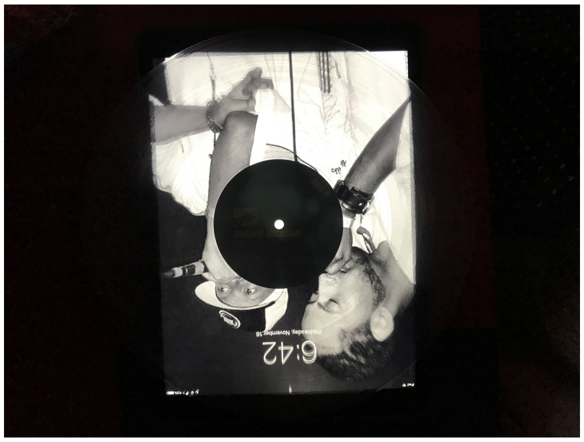


Figure 2.3: Clear vinyl record overlaying 12” iPad Pro for transparency

in depth (Wallack, J (Writer) (2005)).

Much of the display assemblies of todays smartphone and tablet computer screens, particularly the digitizer which is simply the overlay of the LCD display are entirely made out of glass. Users are often encouraged to protect their digitizers with plastic films which vary in thickness yet don't compromise on capacitive input.

Smart Vinyl adopts the same vinyl record manufacturing practices in applying clear veneer lacquer to an LCD display assembly, drying it, and then machining the timecode such that it integrates like a screen film cover on an iPad.

Other alternatives to lathe cutting directly onto the lacquered LCD were 3D printing a thin layer of the time code onto the touch screen, laser cutting the time code into thin pieces of materials such as acrylic, or simply lathe cutting time code into thin films of plastic. This will be one of the few areas of experimentation for this project.

2.2. Power

Smart Vinyl uses the Raspberry Pi Zero W which runs off of a 5 V 1-2A power adaptor. Though I am still in the testing phase, the Power-boost 1000C power supply is a good example of a board needed to interface with the right lithium polymer batter powerful enough to run the Smart Vinyl system for as many hours as possible.

Smart Vinyl required an interface to supply power to the Raspberry Pi, the controller board for the LCD, and the converter board for HDMI to DisplayPort. Many power supplies these days have several smart features such as low battery indication and the ability to prevent brown outs from high currency draw.

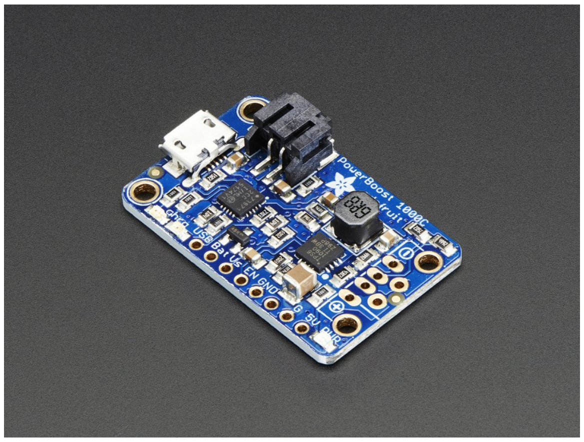


Figure 2.4: the Power-boost 1000C power supply

2.3. DVS/Turntable Control

The approach to building the DVS setup was initially based on the project PiDeck, an open source hardware and Free Software project retrofitting the Raspberry Pi onto a turntable. It's a simple DIY project that requires a Raspberry Pi, a screen, a sound card to connect the audio input from the turntables to the single board computer, an SD card for the software, a USB stick containing a music playlist, and of course control vinyl.

Because *Smart Vinyl* is an autonomous DVS solution, it required a wireless solution that transmitted the digital audio signal from the Raspberry Pi Zero W while simultaneously having the analog stereo signal coming from the turntables fed back. I chose to experiment with configuring the Raspberry Pi Zero W bluetooth interface within the ALSA framework and a bluetooth adaptor connected to an ALSA approved USB to Stereo Audio Interface. Figure 2.5 is an example of a DIY (Do It Yourself) controller board that converts the USB signal to Bluetooth from the ALSA supported Audio Interface.

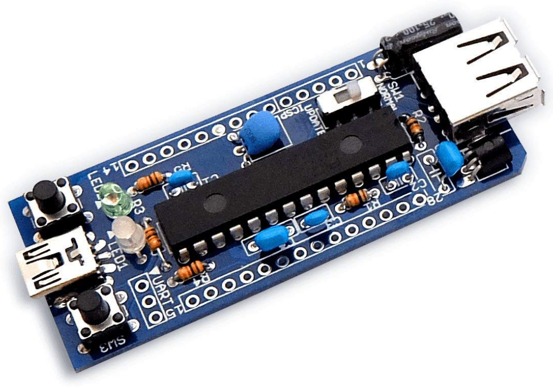


Figure 2.5: USB to Bluetooth Convert Adapter ”USB2BT” Assembly Kit Japan Import

2.4. Sensors

Sensors were added to the control vinyl, allowing the single board computer to assess the orientation on the turntable dynamically. Table 2.1 lists the sensors included in *Smart Vinyl*.

The main plan for the sensors was to use the combination of the gyroscope data (yaw, roll, and pitch) and a three axis accelerometer to most accurately track the motion of the turntable. The accuracy of the orientation of the turntable using both a three axis accelerometer and gyroscope came from Complementary Filters, statistical filters (Kalman Filters), or other methods commonly referred to as sensor fusion algorithms. As there is a single axis of movement, a circular rotation implemented by centripetal force, *Smart Vinyl* exploits this to fix the position of the UI in its “default mode” where a DJ can do things like select audio files they want to play. In “DJ mode” where the DJ is focused on techniques such as scratching and beat juggling, you can disengage from the fixed UI, which allows the DJ to “bookmark” points in the audio wave file of interest.

The responsibility of the display assembly was to simply display the Graphical

Table 2.1: List of Sensor Components

Sensor	Main Computer	Description
Gyroscope	Direct pin connection/Pi or T cobbler	For tracking the yaw, roll, and pitch of the record as it rotates
3 Axis Accelerometer	Direct pin connection/Pi or T cobbler	Aiding in tracking the motion of the record as it rotates to lock the UI in one orientation
Display Panel	LVDS to DispalyPort/HDMI	For user input and video output

User Interface (GUI) as well as handle registered input from touch appropriately. Though circular display assemblies begun appearing in consumer electronics such as Android wear and Nest Smart Thermostats, circular digitizers of a 12" circumference are still unheard of. For this project, I chose to prioritize the function of the LCD and the best way to handle connecting it to the Raspberry Pi Zero W first. I continued to experiment with alternatives to form and leveraged my academic resources at MIT and Harvard as well as my industry resources at E-Ink.

2.5. Main Computer

There were numerous important factors in choosing a single board computer for this system. The board needed to be powerful enough to run the hardware and software, and needed to fit into the dimensions of something as arbitrarily thin as control vinyl. Because of the scope of the project, another important factor to consider was whether there was any prior feedback on the single board computer and the projects it had been used for potential leveraging. These two factors helped persuade me to select the Raspberry Pi Zero W.

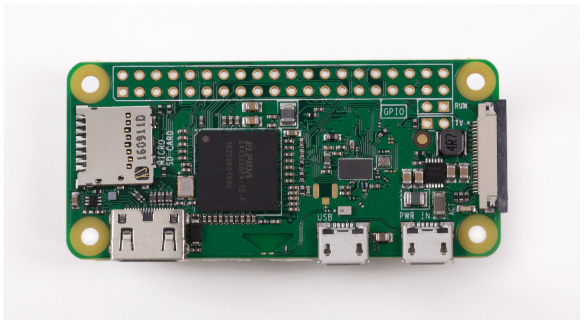


Figure 2.6: The Raspberry Pi-Zero W

The Raspberry Pi Zero W uses an ARM11 running at 1GHz and 512 MB of RAM. It is equipped with a Broadcom BCM2835 SoC which handles component integration. What made this evolution of the Raspberry Pi so desirable even after drastically shrinking its dimensions to 65 mm x 30 mm x 5 mm and rebranding it the Raspberry Pi Zero is that they introduced a new iteration called the Raspberry Pi Zero W (February 2017) and added a 2.4GHz 802.11n wireless LAN as well as Bluetooth Classic 4.1 and Bluetooth LE. It's still powered by 5 V supplied via micro USB connector. The storage is still implemented via MicroSD card and it has a 40-pin general purpose input/output (GPIO) for connecting sensors.

Lastly, *Smart Vinyl* has a bottom enclosure that houses the single board computer as well as the sensors and battery.

2.6. Lower-Level System (OS)

The lower-level system in this project was simply the Linux Kernel. Because *Smart Vinyl* applied the same approach as the PiDeck project, the intent was to use the same armhf (ARM Hard Float) port of Debian GNU/Linux stable (jessie) option before I learned the inherent problem with this approach. Regardless, It was from this point that configuring low level components, particularly the gyroscope and accelerometer was possible.

Having made use of the gyroscope and accelerometer was only possible after I gained access to the Inertial Measurement Unit (IMU), to programmatically convert the raw data into useable angles. The Inner Integrated Circuit (I2C) framework and GPIO pin interface were needed to get access to the IMU raw data.

The low level driver tools for IMU based Open Graphics Library (OpenGL) were used in order to interpret the readings from the gyroscope and accelerometer data to detect the rotation of the record on the turntable, which in turn, programmatically locks the position of the UI in place.

2.7. Higher-Level System (DVS Application)

The higher-level system in this project is defined as the DVS application. Again, because this project applied the same approach as the *PiDeck* project, it used the same *xwax* option for an open source DVS software solution. Though it might be worth noting that *terminatorX*, *Digital-Scratch*, and *Mixxx* are all legitimate open source options as well. They each come with their own set of strengths and weaknesses.

Rather than install the OS on the Raspberry Pi Zero W and then install the open source application and its dependencies in several steps, I simply followed the exact *PiDeck* process of installation using the same single-purpose distribution. Later you will learn the limitations of this process and how I worked around them.

Chapter 3: Build Process

In this section the focus is to go into granular detail about how the software and hardware explicitly come together. Some hardware and software were added or modified since the process began. But the principles and the inherent design have not. From setting up the Raspberry Pi Zero W to lathe cutting over a lacquered LCD, each stage comes with an explicit set of instructions to be executed and the lessons learned from those decisions.

3.1. Configure the Raspberry Pi Zero W (Getting Started)

In order for the Raspberry Pi Zero W to become fully operational, I began by obtaining the necessary peripherals. I managed to purchase a startup kit comprised of a Micro USB power supply, a Mini HDMI to HDMI adaptor, and a Micro USB to USB type B adaptor. However, no Raspberry Pi Zero W kit would be complete without the most important peripheral of all, the SD card.

The Micro SD card was first formatted and imaged (using a program called Etcher) with the latest version of the GNU/Linux stable distribution.

Since the initial approach in creating the digital vinyl software set up was to simply follow the PiDeck steps for installation, copying the PiDeck image, an armhf (ARM Hard Float) port of Debian GNU/Linux stable (jessie) combined with xwax, onto my 16 GB SD card was the first step. However, despite following the

PiDeck guidelines exactly, I was faced with one fundamental problem. Both of my Raspberry Pi Zero Ws simply failed to POST (power on self test) after server attempts at installing what I believed to be a known good PiDeck image on two completely separate known good SD cards. It was at this point that I began to suspect the issue had to do with the PiDeck image itself. I soon then discovered that the PiDeck project only successfully tested their set up with a Raspberry Pi 3 model B and the first generation Raspberry Pi that they explicitly stated does not work at all. The question now was not only whether or not the PiDeck image would work on the Raspberry Pi Zero W. Rather, the question was whether or not xwax itself would work on the Raspberry Pi Zero W. In reaching out to the developers of PiDeck whom I befriended through correspondences, Daniel James and Christopher Obbard confirmed my suspicions by reminding me that the Raspberry Pi Zero used the same chip as the Raspberry Pi 1, which was compatible with Debian armel. [We] targeted only the Pi 3, which was compatible with Debian armhf. They both needed separate packages and kernels built so we only built the kernels and packages for armhf. The Pideck distro simply won't work on the Pi Zero so I didn't bother building anything for the Zero/Pi1 as it was double the code to maintain. (Daniel James and Christopher Obbard, personal communication, October 19 (2017)) Their response gave me the means for a solid approach: I used component isolation to verify the SD Card was known and good by installing GNU/Linux stable first on what I believed to be two known good Raspberry Pi Zero Ws. Then, I tried to installing a standalone image of xwax on top of the GNU/Linux image.

Specific to the Debian image for the Raspberry Pi micro controllers were firmware instructions that allowed the Raspberry Pi to complete a POST after the peripherals were properly connected since the Raspberry Pi Zero W had no means of powering on via buttons or switches. Incidentally, when the Pi issued a solid green

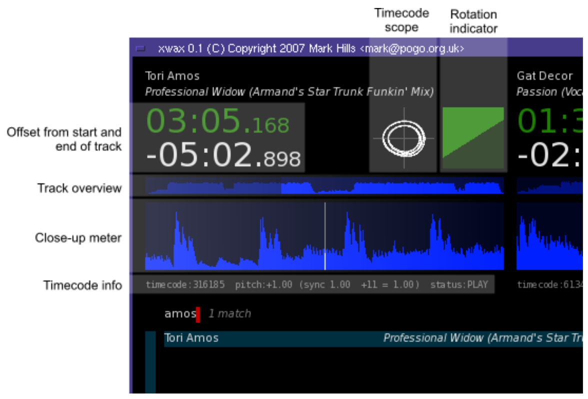
LED (indication that the POST was successful) before ultimately outputting signal to the display after being imaged with the latest version of Debian (Stretch at the time of building and writing this), I was able to validate both a working OS running on a working microcontroller.

Upon successful completion of the Raspberry Pi Zero W booting to the desktop, I felt it necessary to implement the same basic configurations one would typically do to any GNU/Linux environment. I granting myself the necessary admin permissions, performed the necessary updates and installed my preferred editor of choice before I proceeded to the next step: Installing xwax.

3.2. Configure xwax (to reenact Serato setup)

The xwax developers define their software as “an open-source DVS for Linux. It allows DJs and turntablists to playback digital audio files (MP3, Ogg-Vorbis, FLACC, ACC and more), controlled using a normal pair of turntables via timecoded vinyls.” Both xwax and the Smart Vinyl Project share the same vision to provide a solution emphasizing the needs of both Djs and Turntablists. The xwax project is “designed for both beat mixing and scratch mixing. Needle drops, pitch changes, scratching, spinbacks, and rewinds are all supported, and feel just like the audio is pressed on the vinyl itself.” (Hills , Mark (2007))

Xwax requires “modest CPU, Pentium III 600 or faster” and “lots of RAM, typically 512 MB or more.” The hardware specifications for the Raspberry Pi Zero W just about met those requirements. When it came to the actual install of xwax itself, there were two schools of thought. One could either download the 80 KB image for their site which contains the source code, written in the C programming language, to be uncompressed, configured for whichever audio interface of ones choosing (i.e. Open Sounds System (OSS), JACK, or ALSA), and ultimately compiled. Or, one could simply install the xwax package from the corresponding universe package repository. Since I was able to experiment with two separate Raspberry Pi Zero Ws using known good 16GB SD Cards, I was able to execute both methods of installation.



Timecode scope
Rotation indicator
xwax 0.1 (C) Copyright 2007 Mark Hills <mark@pogo.org.uk>
Tori Amos
Professional Widow (Armand's Star Trunk Funkin' Mix)
Gat Decor
Offset from start and end of track
03:05.168
-05:02.898
Track overview
Close-up meter
Timecode info
timecode:316185 pitch:+1.00 (sync 1.00 +11 = 1.00) status:PLAY
amos 1 match
Tori Amos

Figure 3.1: Anatomy of the xwax User Interface

Since xwax runs from the command line, I called xwax like one would any other Unix utility or executable. When I ran xwax from the command line, there were arguments that needed to be called as well for things like calling the directory where the music files are, or specifying which type of hardware is being connected such as turntable decks. For a more extensive list of command line options, one consults the xwax manual page or use the help option. From there it gave you even more granular levels of functionality. One could do things like specify the speed of the vinyl record rotating at either 33 or 45 rpms. You could specify which side of the timecoded vinyl record you are using, side a or side b. You could even specify the type of connection you are using for input be it phono or line in. And of course, without specifying these configurations, it would simply run in a default mode, assuming certain configurations for you. Here is an example of a set up where you can specify extra devices for more decks and other options:

```
$ xwax -t serrato_2b -l ~/music -l cdtracks -a hw:1 -a hw:2
```

Table 3.1: Short List of xwax Commands

`xwax`	Running the xwax command
`-t serrato_2b`	Using Serato 2nd Edition, side B timecode (see xwax -h for available timecodes)
`-l ~/music`	A path to a directory containing music tracks
`-l cdtracks`	A path to a directory containing the virtual music tracks (which instruct the importer to extract from audio CD)
`-a hw:1`	Audio device for first deck
`-a hw:2`	Another audio device for second deck

What you were actually doing in the command above is summarized in the following on Table 3.1:

At a bare minimum, in order to start running xwax, you would call the program followed by the directory where the audio file(s) live and at least one audio device specifying at least one deck. A thorough list of the flags and their explanations can be found on Table 3.2 in order to give a more granular sense of what xwax is capable of.

The Advanced Linux Sound Architecture is the default software framework and part of the Linux kernel that provides an application programming interface (API) for sound card device drivers. The flexibility of this architecture allows sound

Table 3.2: An Explanation of the Flags

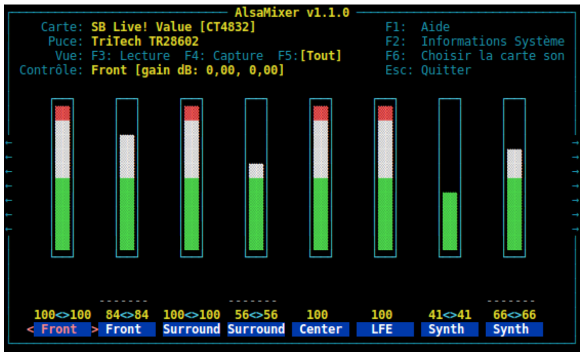
Runtime Flag	**Flag Function**
`-l path`	Scan the music directory or playlist at the given path. The directory/-playlist specified here will show up in the list of "crates? to the left of the xwax interface. See below for a script you may find useful for passing music directories to xwax.
`-t name`	Use the named timecode for subsequent decks. Multiple timecodes can be specified for a deck or decks and can be switched between live using the shift + F3/F7/F11 key combo for decks 0, 1 and 2 respectively. The available timecodes are: serato_2a (the default if no timecode is specified), serato_2b, serato_cd, traktor_a, traktor_b, mixvibes_v2, mixvibes_7inch.
`-33`	Set the reference playback speed for subsequent decks to 33 and one third revolutions per minute. This is the default which xwax will use for decks with neither -33 or -45 set.
`-45`	Set the reference playback speed for subsequent decks to 45 revolutions per minute.
`-c`	Protect subsequent decks against certain operations (such as track loading) during playback. This will stop you from accidentally loading a track onto a playing deck!
`-u`	Allow all operations on a deck during playback. This is the inverse of the -c option, and is the default.
`-i path`	Use the given importer executable for subsequent decks. Not specifying an import executable will cause xwax to use the default import executable.
`-s path`	Use the given scanner executable to scan subsequent music directories or playlists specified with the -l flag.
`-k`	Lock into RAM any memory required for real-time use. This includes audio tracks held in memory which can be large. Use ulimit -l to raise the kernel?s memory limit to allow this.
`-q n`	Change the real-time priority of the process. A priority of 0 gives the process no priority, and is used for testing only. The default value is 80.
`-g nxn[+n+n]`	Change the geometry of the display. This size and position is passed to SDL, which may use it to set the display mode, or the size of an X window. Typically this will dictate the starting size of the xwax window.
`--no-decor`	Request the window manager to create a ?frameless? window which does not have the regular controls such as title bars and buttons. This can be useful in conjunction with the -g flag for dedicated xwax installations.
`-h`	Display the help message and default values. More info can be found on the xwax man page, accessed by entering man xwax into the terminal.

Table 3.3: ALSA Device Options

Runtime Flag	**Flag Function**
`-a device`	Create a deck which uses the given ALSA device (eg. plughw:0).
`-r n`	Set the sample rate in hertz for subsequent decks. If left blank xwax will use the default value.
`-m n`	Set the ALSA buffer time in milliseconds for subsequent decks. If left blank xwax will use the default value of 8ms.

servers like JACK to work on top of ALSA. Alternatively, the Open Sound System (OSS) architecture can be configured as well to support the narrow margin of hardware that ALSA does not seeing as how their compatibility is no longer supported depending on which version of Linux is installed. “Additional to the software framework internal to the Linux kernel, the ALSA project also provides the command-line utilities `alsactl`, `amixer`, `arecord/aplay`, and `alsamixer`, an ncurses-based TUI.” The `arecord/aplay` utility was particularly helpful when it came to troubleshooting and verifying that audio was in fact successfully being decoded and that the connected output actually worked. The `alsamixer` was also pretty helpful because it allowed for more precise control of the audio settings by prompting a GUI that displayed audio levels which is demonstrated in Figure 3.2. (Advanced Linux Sound Architecture (nd)) Table 3.3 is a brief list of ALSA specific flags and their definitions when running xwax from the command line.

Since most devices are detected on a USB device tree automatically, having to explicitly point xwax to the corresponding audio interface typically was not necessary especially since there were no other devices connected to the serial bus that could be interpreted as an audio interface. However, in the event that there were devices connected that could be interpreted by the ALSA framework as audio interfaces, there was a way to explicitly do so by way of a configuration file. Previous versions of Linux



```
                         AlsaMixer v1.1.0
   Carte: SB Live! Value [CT4832]                F1:  Aide
    Puce: TriTech TR28602                        F2:  Informations Système
     Vue: F3: Lecture  F4: Capture  F5:[Tout]    F6:  Choisir la carte son
Contrôle: Front [gain dB: 0,00, 0,00]            Esc: Quitter

           -------            -------                                -------
 100<>100  84<>84  100<>100  56<>56    100      100      41<>41   66<>66
< Front  > Front   Surround  Surround  Center   LFE      Synth    Synth
```

Figure 3.2: The ALSA Mixer interface

(depending on the version), would provide the file in both a system level director as well as the inside the user directory. However, in the latest version of Debian, there is no config file and so this means you explicitly have to create a hidden file called ".asoundrc" in your user directory. (Asoundrc (nd)) Modifying this file gave me the ability to control just about every aspect of how the audio interface profile was viewed by my linux system. Since the Raspberry Pi Zero W has the option of carrying audio signals through HDMI, bluetooth and USB, the .asoundrc file was helpful in its ability to explicitly designate the desired audio interface I needed to carry to and from xwax. Here is an example config file that does just that:

(How to change the default audio interface - https://learn.adafruit.com/usb-audio-cards-with-a-raspberry-pi/updating-alsa-config)

With the ALSA framework in place to standardize audio interfacing and the .asoundrc configuration file in place to manipulate the audio signal itself, the only thing remaining to get xwax up and running was picking the right audio interface itself.

3.3. Pairing and Implementing Audio (going from wired to wireless)

In order to get to the ultimate goal of a completely wireless audio solution, I recreated what we already known to be a working solution using wired audio interfaces. The idea was that once the wired audio interface was connected and running, study the audio signal flow of control first before ultimately implementing a way to wirelessly carry the audio signal from the turntable to xwax and then from xwax back to the turntable.

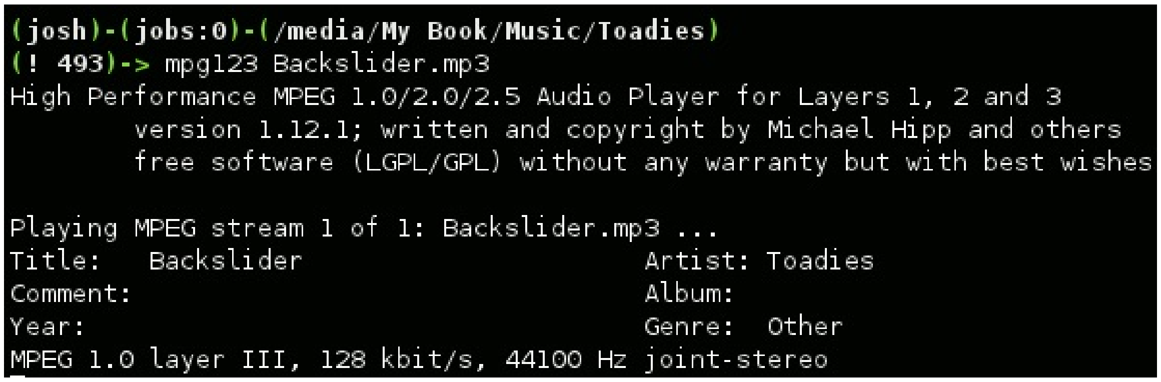
USB sound cards are typically the most conventional means of audio interfacing. Aside from the fact that it uses the standard peripheral inherent in its name "Universal Serial Bus", it means that frameworks such as ALSA can easily talk to the low level firmware of the USB controller on the sound card which then ultimately lets the ALSA framework know that it is an audio interface. The Kingwin USB-3DSA was a cheap and easy example of a plug and play sound card. Its specifications were comprised of being a USB Type A connector with a mono microphone input jack and a stereo audio output jack. It is USB bus powered so it did not require any external power and it did not require the installation of drivers for reasons stated earlier about the ALSA framework.

For the purposes of feeding the audio into the USB sound card for xwax, I first connected the RCA Stereo male connections coming directly out of the turntable into an RCA Stereo female to 3.5 mm stereo audio auxiliary before I connected that into the microphone input jack. Next, I connected another 3.5 mm stereo audio auxiliary coming from the line out of the USB sound card which adapted to RCA Stereo male prongs. The corresponding left and right audio prongs were plugged into a mixer which was ultimately connected to stereo speakers.

Xwax imports digital audio files using an external decoder and holds them in memory as uncompressed raw audio to ensure fast and reliable seeking during playback. Since using an external decoder meant you could configure to use your codec and file format of ones choosing, decoding audio in a separate process meant there was no risk of xwax crashing because of corrupt audio files. (http://xwax.org/overview.html). The only thing left now was to validate the installation of the audio decoder as well as validating the connection of the audio interface itself. I called the corresponding audio decoder binary (maybe a flag argument) and passed the binary an audio file argument from the command-line:

```
$ mpg123 MySong.mp3
```

that played the single track from start to finish with output identical to the example from Figure 3.3.



```
(josh)-(jobs:0)-(/media/My Book/Music/Toadies)
(! 493)-> mpg123 Backslider.mp3
High Performance MPEG 1.0/2.0/2.5 Audio Player for Layers 1, 2 and 3
        version 1.12.1; written and copyright by Michael Hipp and others
        free software (LGPL/GPL) without any warranty but with best wishes

Playing MPEG stream 1 of 1: Backslider.mp3 ...
Title:   Backslider                       Artist: Toadies
Comment:                                  Album:
Year:                                     Genre:  Other
MPEG 1.0 layer III, 128 kbit/s, 44100 Hz joint-stereo
```

Figure 3.3: Example of Single Track Output running mpg123 from

This validated both the audio decoder and the audio interface. Additionally, there was an audio import script file that allowed for the flexibility of installing different audio decoders:

Listing 3.1: **Default xwax Import Script**

```
#!/bin/sh
```

```
#
# Audio import handler for xwax
#
# This script takes an output sample rate and filename as arguments,
# and outputs signed, little-endian, 16-bit, 2 channel audio on
# standard output. Errors to standard error.
#
# You can adjust this script yourself to customise the support for
# different file formats and codecs.
#
FILE="$1"
RATE="$2"
case "$FILE" in
*.cdaudio)
        echo "Calling CD extract..." >&2
        exec cdparanoia -r ‘cat "$FILE"‘ -
        ;;
*.mp3)
        echo "Calling MP3 decoder..." >&2
        exec mpg123 -q -s --rate "$RATE" --stereo "$FILE"
        ;;
*)
        echo "Calling fallback decoder..." >&2
        FFMPEG=$(which ffmpeg 2> /dev/null || which avconv 2> /dev/null)
        if [ -z "$FFMPEG" ]; then
                echo "$0: no ffmpeg or avconv available to decode file" >&2
                exit 1
        fi
        exec "$FFMPEG" -v 0 -i "$FILE" -f s16le -ar "$RATE" -
        ;;
esac
```

The decoders in the default script were the basis for why I chose my particular decoder. That being said, the script could be modified to handle other audio decoders that were not already in the script.

With the framework for the flow of audio control in place and validated, the next logical progression was to play the audio within the xwax program. I called the xwax program with the following:

```
$ xwax -l /̃home/pi/Music -a plughw:1
```

Upon successful execution, the interface window opened and I then loaded the audio to be played by highlighting the song in the library followed by an F1 keystroke. The audio wave file loaded, then what is supposed to happen is one simply hits play on the turntable to watch the audio wave file scroll from left to right across a black vertical marker. What was actually happening however was that the audio wave file was feeding into the xwax user interface backwards. Why was this happening I asked? After extensive troubleshooting, I was able to narrow down the problem to the audio being fed into the Raspberry Pi Zero W by way of the microphone port. However, at this point, my faith in both my understanding of the ALSA Framework config file and the audio interface were razor thin. After talking to a close colleague and Audio Engineer Alvin Carter, he confirmed my suspicions of the audio interface, particularly in how it is carrying the audio signal from the interface to the Raspberry Pi (Alvin Benjamin Carter III, personal communication, February 5 (2018)). I figured that this was a good opportunity to use an audio interface that has already been vetted by the xwax community instead of using an audio interface that was not. I purchased the Behringer U-Control UCA202 Ultra Low-Latency 2 In/2 Out Usb/Audio Interface With Digital Output, shown in Figure 3.4 and continued to troubleshoot. What was it that made the Behringer audio interface guaranteed to work that the Kingwin unsuccessful? The answer, it turns out, was in the technical specifications of the Kingwin device that I simply missed. Evidently, the fact that the Kingwin USB-3DSA was a Mono Microphone-Input Jack and Stereo Output Jack(http://www.kingwin.com/adapters/usb-sound-adapter/) explained exactly why the audio signal was outputting backwards. Only half of the audio signal data was being sent from the audio interface to the Raspberry Pi and consequently the xwax program, confirming Alvin and my own suspicions. Furthermore, once the Behringer audio interface came, I was able to confirm that worked as well.

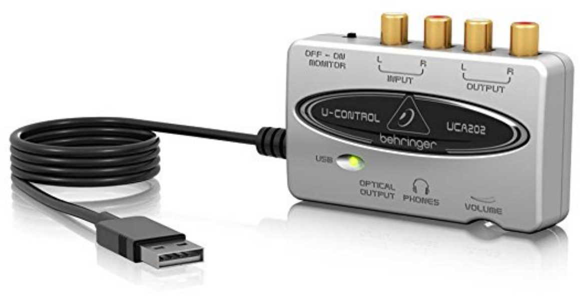


Figure 3.4: Behringer U-Control Ultra Low-Latency 2 In/2 Out USB/Audio Interface with Digital Outputs

Now that the audio flow of control was validated and fully vetted, the next step was to implement a way to communicate an ALSA approved bluetooth stereo adaptor that could carry the full digital stereo signal to the Raspberry Pi which then pushed that signal back out as a full analog stereo signal. This happens to the be the only remaining feature that I do not yet have a working solution for as of this write up.

3.4. Implementing and Connecting Display to Raspberry Pi

Again, the objective of this thesis project was to house a tablet computer running Digital Vinyl Software inside of a vinyl record. Therefore, much of the early research efforts leading up to the development of this project were predicated under the assumption that because vinyl records were traditionally circular, if a display were to be housed inside the record, the display itself would also need to be circular. I began by focusing on the state of circular digitizers and looked for commercialized products such as The Nest Thermostat, Samsungś Galaxy Gear Watch, or the round ePaper display by Eink that I could reference and even reverse engineer into my design. The one glaringly obvious issue I had to figure out which all of these products shared in common was that these digitizers didnt́ scale anywhere near the size of a 12" Vinyl LP. The further down this path I went, it seemed as though there simply were no circular digitizers that scaled to the size I needed at all let alone for commercial use. If they did scale, cost would be the primary limiting factor. A colleague and friend of mine, Alain Bouchard, who just so happens to be a senior engineer at E-ink came up with the brilliant idea of simply taking the smaller ePaper displays and tiling them together (Alain Bouchard, personal communication, August 30 (2017)). Ultimately, we came to the forgone conclusion that ePaper refresh rate performance would be too underwhelming for this application. Solutions for a circular LCD were seemingly slim to nonexistent and was proving to be quite the bottleneck for the project overall. If I was going figure out a way to get this to work, I needed to reassess the design entirely all while thinking about resources that were commercially available and easily accessible.

In thinking more critically about the form factor of the device as a whole, I first had to concede to the fact that all of the LCDs readily available to me in the dimension I needed would be rectangular. More importantly, it reminded me of the fact that the dimensions of the device were purely dictated by the dimensions of the turntable itself. Hence, figuring out the dimensions necessary to clear rotation on a turntable, i.e. making sure the device didnt́ hit the tone arm, coupled with a basic understanding of the dimensions of a 12" LP were the keys to circumventing the need for a circular LCD entirely. Knowing this provided two primary schools of thought in reassessing the design: The LCD could be inscribed inside a circular 12" vinyl record. Or, The LCD could be larger than a 12" LP so long as it cleared rotation on the turntable.

The process for lacquering and lathe cutting over a digitizer was virtually identical to the existing process of cutting a master record when the LCD fits a 12" diagonal and is simply inscribed inside of a circle. However, fitting an LCD around the dimensions of a 12" LP, despite being valid geometrically, was entirely contrary to my original assumption that the form factor of a vinyl record was traditionally circular. I would soon learned that my assumption was wrong. Some tradition were meant to be broken. It first came to my attention in the form of a David Bowie single entitled "The Next Day". The record was white, with a 7" diagonal and square.

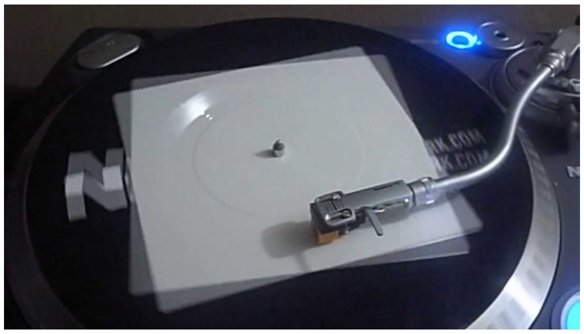


Figure 3.5: The 7" single entitled "The Next Day" by David Bowie that inspired my design change

Similar to David Bowie and his music, it defied convention, it was ahead of its time, and provided the insight I needed. It inspired the fact that the shape of the vinyl material that housed the grooves did not have to be circular at all. Rather, it was the actual lathe cut that had to be circular for a turntable stylus to properly read. It allowed me to shift the focus away from researching the state of circular digitizers entirely. Rather, I simply used the thinnest LCDs from the technology that were already available, overlayed it with a clear vinyl lacquer and inscribed a lathe cut into that. A close colleague of mine and long time music vendor, Jeremy Sullivan, introduced me to a whole other world of vinyl record products. These products reinforced this idea that not only did vinyl records come in various shapes other than 12" circles, they came in various materials too such as plastic and even cardboard (Jeremy Sullivan, personal communication, September 15 (2017)). Instead of being lathe cut, they used laser etching and 3-D printing and still played on a turntable just the same. Flexi disc was of particular interest because it was extremely thin, came in various transparencies, and sounded crystal clear when played from a turntable (Flexi disc (nd)).

In spite of the fact that just about any LCD of the correct dimensions could work for the purposes of this project, it still begs the question of how I connected the LCD to the our Raspberry Pi. The Raspberry Pi is equipped with a mini HDMI out port. However, most LCD panels connect to what is called an LVDS (Low Voltage Differential Signaling) cable. To make matters worse, depending on the manufacturer or the LCD panel, some LVDS cables might interface differently than others. It turns out, the best way to select the appropriate LCD panel was to select a controller board to interface with the Raspberry Pi first. Upon further research, I finally came across a controller board that seemed promising. It was called a DisplayPort to Apple MacBook Pro Retina LCD interface built by Daniel Rozsnya of Digital Cinema Devices.

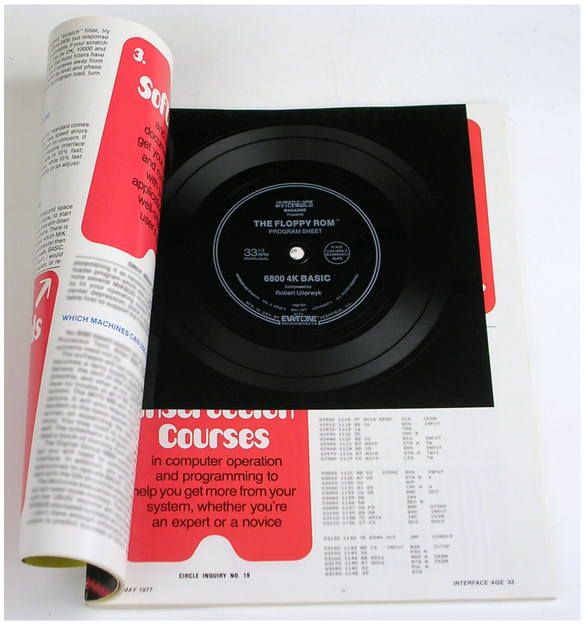

THE FLOPPY ROM
PROGRAM SHEET
6800 4K BASIC
Robert Uiterwyk
Courses
in computer operation and programming to help you get more from your system, whether you're an expert or a novice
CIRCLE INQUIRY NO. 19
MAY 1977
INTERFACE AGE 33

Figure 3.6: Flexi discs like this Interface Age "Floppy ROM" program sheet were occasionally included as inserts in computer hobbyist and video game magazines during the late 1970s and early 1980s.

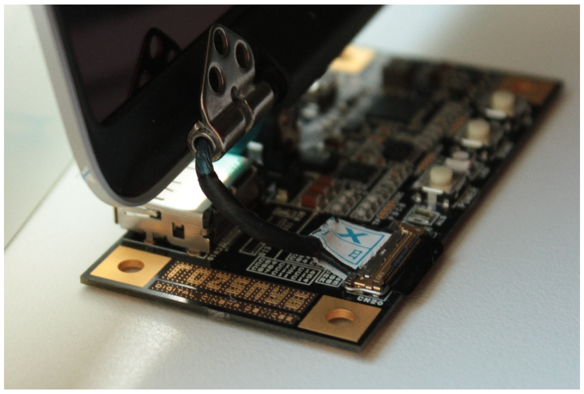


Figure 3.7: The DisplayPort to Apple MacBook Pro Retina LCD controller board

This controller board came in a modest form factor (40mm x 80mm) and connectd the proprietary LVDS cables from a specific range of MacBook Pro Retina Display LCD models. The video input source connected to DisplayPort which added a layer of complexity because conventionally, DisplayPort passively converts to HDMI.

(maybe add some more technical explanation to how the board works and then slip in the block diagram)

There are however some active conversion solutions that will be used to experiment in connecting the HDMI output from the Raspberry Pi to the DisplayPort in. Figure 3.8 was used to carry the signal from the Raspberry Pi Zero W to the DisplayPort to Apple MacBook Pro Retina controller board which ultimately output to the Macbook Pro Retina LCD.

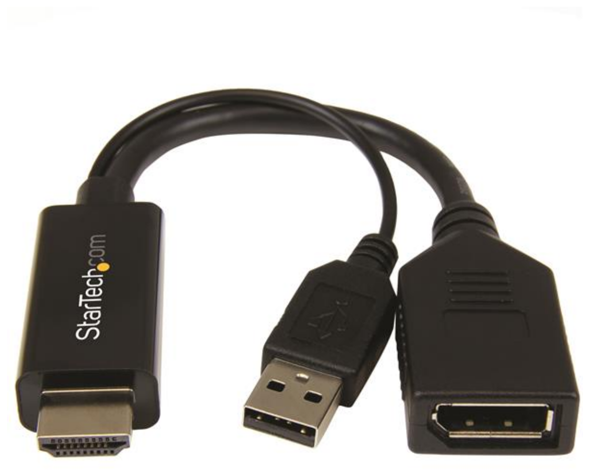


Figure 3.8: Active Conversion HDMI to DisplayPort Adaptor

The video signal, upon being fed from the Rasperry Pi, was handled in one of two ways by the controller board depending on the behavior of the Retina Display. The controller board either detected the video signal automatically and registered the appropriate brightness. Or, it received the signal but could not register the corresponding brightness. It was at this point then that the controller board would be toggled into "forced" mode where the brightness would be adjusted manually. Figure 3.9 represents an operation diagram for the two use cases of the DisplayPort to Apple MacBook Pro Retina LCD controller board.

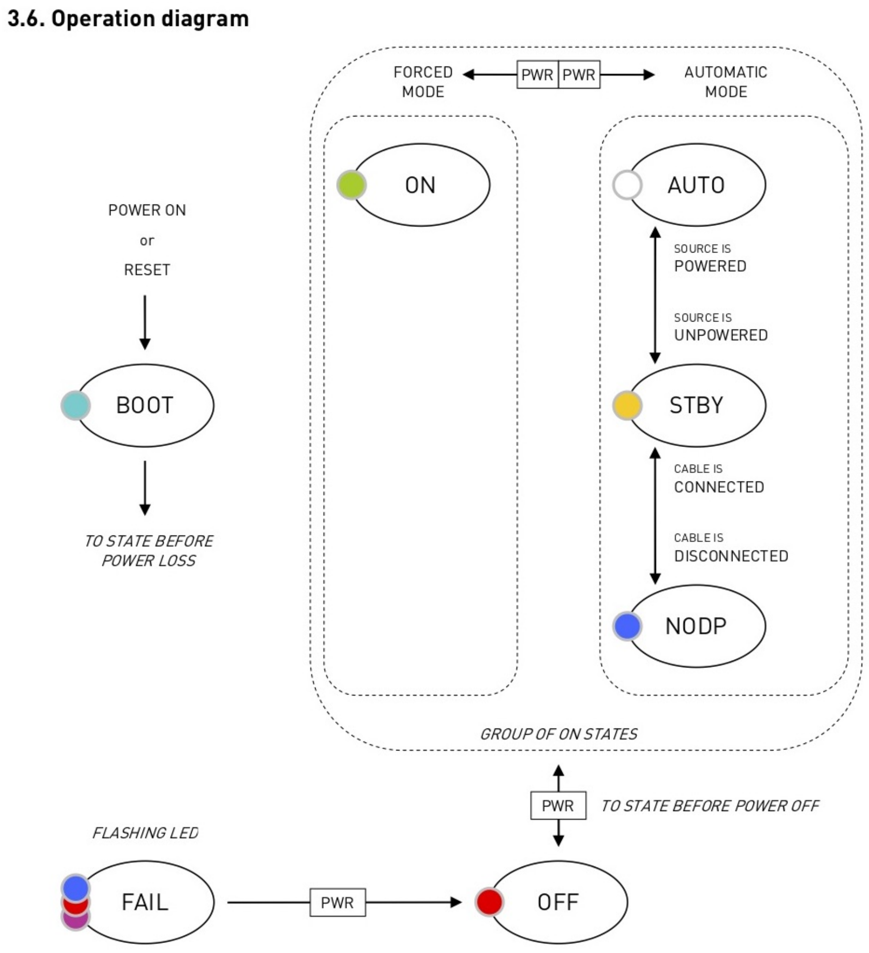
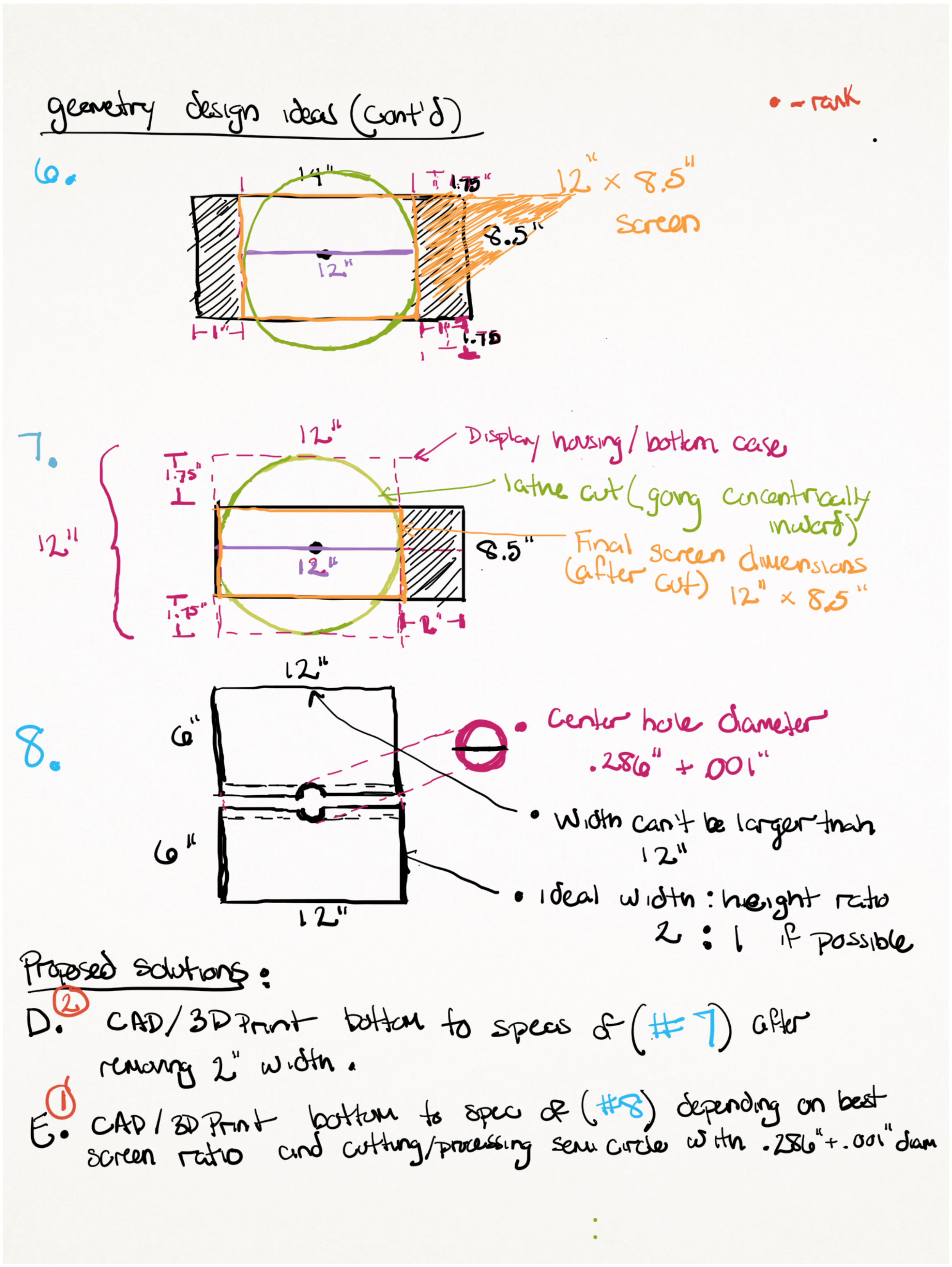


Figure 3.9: Operation Diagram that illustrates the two modes of usage

Everything implemented thus far was done to recreate the look and feel of a vinyl record all while housing the electronics necessary to run digital vinyl software from the vinyl record directly. However, one of the attributes inherent to being a vinyl record was its ability to be placed on a turntable which fits around a spindle. And since displays generally tend to not work very well with spindles protruding through them, the original work around was to have DJs purchase thicker slip mats that would interface with a special engineered bottom housing that sat on top of the spindle. It turns out, my assumption about displays not working when spindles are protruding through them was also wrong. After some searching, I came in contact with Jim Kennedy, V.P. of Sales and Marketing at Pixel Scientific which is a company that had patented the process of customizing and resizing LCD displays. After several correspondences back and forth of LCD model numbers and a schematic drawing of my design, he responded with a couple of important take away points.

Where one cuts along the display greatly depends on the OEM specifications. If the vertical or horizontal cut ran along circuits necessary to driving the display, it would no longer function. With regards to my design, specifically section 8 of Figure 3.10, he was eager to point out that it was possible to fabricate such a design and attached an example of a product that they had successfully cut and sewn back together in a precisely similar fashion. He said that in that example, it would be possible to put a hole in the front side gap where the 2 cut edges are put back together. Figure 3.11 represents the example of the two tiled LCDs and the side gap space represented between the two. He did wanted me to keep in mind that the donor LCD model was EOL (end of life) and that it was only meant to serve as an example of what could work.

Once Jim Kennedy was able to verify the candidacy of my MacBook Pro LCD displays, I simply removed the LCD panel from the MacBook Pro display enclosure and sent the panel to Pixel Scientific to be processed and fitted for the .286" diameter aperture.


geometry design ideas (cont'd)
• - rank
6.
14"
1.75"
12" × 8.5" screen
8.5"
12"
1"
1"
.75
7.
12"
Display housing / bottom case
1.75"
lathe cut (going concentrically inward)
12"
8.5"
12"
Final screen dimensions (after cut) 12" × 8.5"
1.75"
2"
8.
12"
6"
• Center hole diameter .286" + .001"
• Width can't be larger than 12"
6"
• ideal width : height ratio 2 : 1 if possible
12"

Proposed solutions:

D. ② CAD/3D Print bottom to specs of (#7) after removing 2" width.

E. ① CAD/3D Print bottom to spec of (#8) depending on best screen ratio and cutting/processing semi circle with .286"+.001" diam

Figure 3.10: Hand drawing the geometric dimensions for viable design ideas

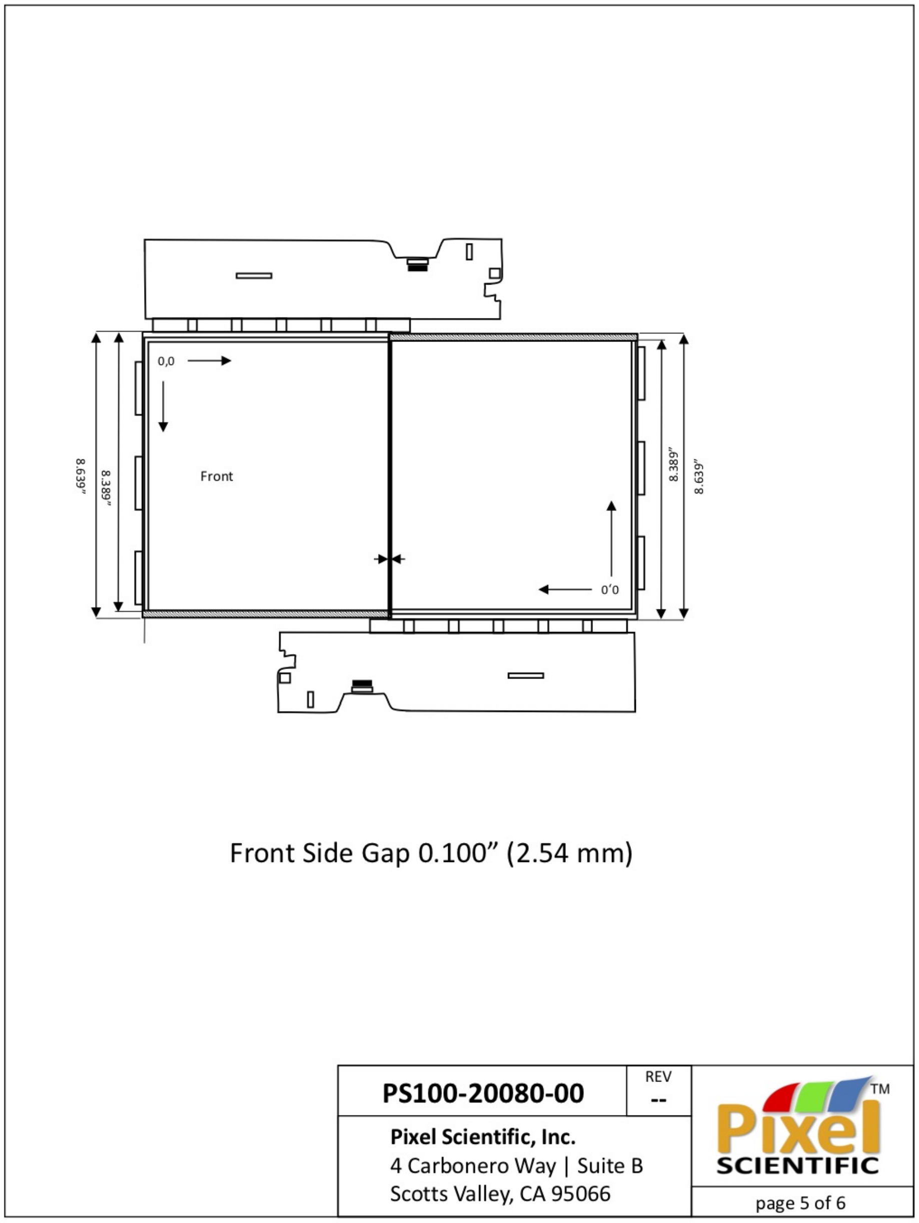


Figure 3.11: Tiled 10 inch x 8 inch Open Display by Pixel Scientific

3.5. Connecting and Configuring Gyroscope and Accelerometer to Raspberry Pi

In order for the DJ to select music in stride as well as visualize where they are in the audio wave file, the DJ needs the ability to interface with Smart Vinyl while it rotates on the turntable at 45 RPMs (Rotations Per Minute). In order to achieve this, the Raspberry Pi needed a way to both track movement with respect to its orientation. Additionally, the Raspberry Pi needed a way to sustain a fixed visual orientation contrary to the centripetal movement of the stylus on a turntable. This was the perfect opportunity to introduce an Inertial measurement unit (IMU) to interface with the Raspberry Pi Zero W. The focus of this section is to give a brief overview of coupling 3-axis gyroscope and accelerometer data, how I stepped through the process of connecting the IMU to the Raspberry Pi, and then finally configured the IMU data to a user interface.

The IMU that I chose to use for this project is called a first generation BerryIMU which uses a LSM9DS0 comprised of a 3-axis gyroscope, a 3-axis accelerometer and a 3-axis magnetometer to measure velocity, orientation and gravitational forces. It is also equipped with a BMP180, which is a barometric sensor which can be used to calculate altitude. There is even a temperature sensor (BerryIMU Technical Specs (nd)). What makes an IMU device so accurate is their ability to combine the readings of individual sensor data. For the purposes of using the IMU to calculate angles, reading from both the gyroscope and accelerometer were needed due to their individual shortcomings. Inherently, a gyroscope is good for quick sharp movements by measures rotation rate but has to do so over a period of time in order to calculate its current angle which can cause the gyroscope to drift. Conversely, accelerometers are good for measuring the current angle at any given time but produce a lot of noise and are only useful for tracking angles over a long period of time. The most important distinction is that an accelerometer can not measure what is called “yaw” or the z-axis. Simply put, if you rotated an accelerometer on a flat surface, the z-axis would read constant (Guide to interfacing a Gyro and Accelerometer with a Raspberry Pi (nd)).

Connecting the BerryIMU to the Raspberry Pi Zero W was fairly straight forward. Depending on the form factor, there were two sets of soldering points to connect the serial data, serial clock data, the 3.3V and ground to the GPIO pins on the Raspberry Pi Zero W. For the purposes of maintaining a low profile the plan was to directly solder jumper pins from the BerrryIMU to the Raspberry Pi Zero W (How to assemble BerryIMU (nd)).

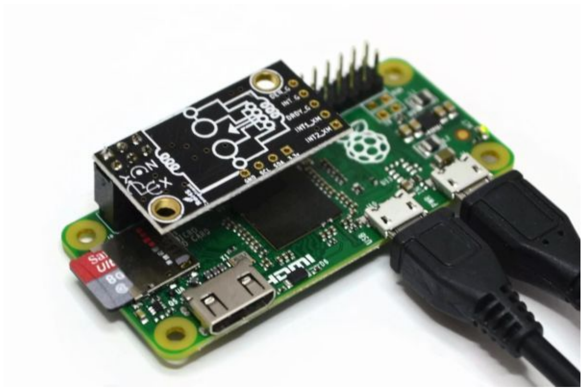


Figure 3.12: An example on how to connect the Raspberry Pi Zero W to the BerryIMU

Once the connections were properly made, the BerryIMU needed the means to communicate with the Raspberry Pi. To do so, there was what is called an I2C interface that must be enabled on the Raspberry Pi by installing the I2C-tools. The I2C is a communication protocol that runs over a two wire bus. The two wires are called SDA (Serial Data) and SCL (Serial Clock). In general, the I2C bus has one or more master devices and one or more slave devices which share the same data and clock serial lines. For this project we only used one master device, the Raspberry Pi, and technically one slave device, the LSM9DS0 on the BerryIMU.

However, the BerryIMU houses our accelerometer, gyroscope and magnetometer. And with I2C, every device has an address that each communication must be prefaced with (http://ozzmaker.com/i2c/). This meant that once the I2C was enabled, I then used the i2cdetect tool to see the three registered addresses. The gyroscope on the LSM9DS0 was using the address of 0x6a, and the accelerometer and magnetometer were using 0x1e. The pressure sensor used address 0x77(site quote).

(insert picture of address output from i2detect)

There of course is an entire data sheet for the LSM9DS0 that goes in more detail of the coupled magnetometer, accelerometer, and gyroscope(http://ozzmaker.com/wp-content/uploads/2014/12/LSM9DS0.pdf). And if you wanted more information on the BM180, the pressure sensor also configured on the BerryIMU, there is a data sheet for that as well (http://ozzmaker.com/wp-content/uploads/2015/01/BMP180-DS000-09.pdf).

With the I2C communication protocol enabled and the connection to the BerryIMU verified, the next step was to read and write to our accelerometer and gyroscope programmatically. In order to do so, I first had to obtain the addresses of both these sensors so that data could be written to them. The act of writing to the accelerometer and gyroscope simply turned on their axes as well as set their sensitivity levels by values all of which can be configured from the data sheet. Now that the addresses that correspond to the sensors were turned on, I then had the means to read raw values from the sensors. However, in order to be of use, the raw data needed to be converted to human-readable measurements such as degrees in order to be properly interpreted. Degrees per second are typically used to measure how fast a gyroscope rotates. And since the gyroscope references sensitivity as an attribute from the data sheet, there were the means to construct an algorithm that extracted the angle of the gyroscope in realtime where that was equal to the raw angle which is directly proportional to set sensitivity value over a given loop period. Similarly, converting the accelerometer raw data to human readable measurements was simply a matter of using trigonometry from the raw data and then converting the radians to degrees. Then, I still had to take into consideration the fact that gyroscopes drifted and accelerometers created noise. This introduced the implementation of filters. The basic idea was that the inherent accuracy of the IMU device was derived from the combined accelerometer data over long periods of time with gyroscope data over short periods of time. For the sake of being less CPU intensive, using Complementary Filters was more optimal for the use of this project. The equation for Complementary Filters was the following:

Current angle = 98% x (current angle + gyro rotation rate) + (2% * Accelerometer angle).

Programmatically, this was implemented by the following:

Listing 3.2: **C Code Sample for Combining Angles from the Accelerometer and Gyro**

```
CFangleX=AA*(CFangleX+rate\_gyr\_x*DT) +(1 - AA) * AccXangle;
CFangleY=AA*(CFangleY+rate\_gyr\_y*DT) +(1 - AA) * AccYangle;
```

A nice last finishing touch for configuring the BerryIMU device was to zero out the angle reading of the angles when the device is upright to give the human-readable measurements better context and for troubleshooting(http://ozzmaker.com/berryimu/).

With our BerryIMU connected and configured programmatically, the last thing left to do was to connect our user interface. Though I had not fully vetted the solution for this particular functionality, I narrowed down my options to two example projects.

The first project example was a "Digital Compass with the Raspberry Pi - Smartphone Replica". It builds on top of the very same gyroscope and accelerometer tutorial code and added the Simple DirectMedia Layer (SDL) library to produce graphics. Since we rely on xwax to produce our graphics, I didnt́ necessarily need to pay attention to how the SDL made graphics so much as how the gyroscope and accelerometer interface communicated with the SDL. To start, we simply needed to install the SDL:

```
$ sudo apt-get install libsdl1.2-dev libsdl-image1.2-dev
libsdl-gfx1.2-dev libsdl-ttf2.0-dev
```

Instead of stepping through the process of creating the SDL layer and having to load an image from scratch, xwax already implements an SDL layer. The task then becomes a matter of having to figure out how to collectively assign the entire UI instead of a single png file to be passed into a function that rotates the UI based on the angle from the IMU very similar to the function called rotozoomSurface() does in the Digital Compass example. (http://ozzmaker.com/compass04/)

Alternatively, there is an Instructables project that auto rotates the entire display output using an Arduino micro controller connected to a different IMU model. The difference in this example is that it interfaced a different IMU device using an Arduino written in python. However, once that is understood, the implementation of the auto rotating portion should be almost identical.(http://www.instructables.com/id/Desktop-Display-Auto-Rotate/)

3.6. 3D Print Bottom Case Housing

Everything leading up to this point had been done to recreate and highlight the best attributes of current Digital Vinyl System setups. What set Smarty Vinyl apart, however, was the ability to consolidate those attributes into one device. The basic anatomy of Smart Vinyl was comprised of our Raspberry Pi Zero W, the BerryIMU, the DisplayPort to MacBook Pro Retina controller board, and various interconnected cables. Since the turntable was the basis for our size constraint of 13.5" and the LCD already provided a nice guideline of a form factor, an easy and practical design for a bottom case was to simply build a rectangular bottom case around the dimensions of our LCD. It turned out, the 12.9" iPad Pro happened to be a perfect design point of reference for the Smart Vinyl form factor. Similarities in functionality nonewithstanding, the 12.9̈diagonal of the iPad Pro provided a dimension of 12" (305.7 mm) x 8.68" (220.6 mm) x .27" (6.9 mm). This meant that approximately 104 square inches (67000 square millimeters) of surface area were available to fit our electronics inside of it. According to the mechanical drawings, the Raspberry Pi Zero W dimension were 65 mm x 30 mm with a surface area of 1950 square millimeters (Zwetsloot , Rob (2017))

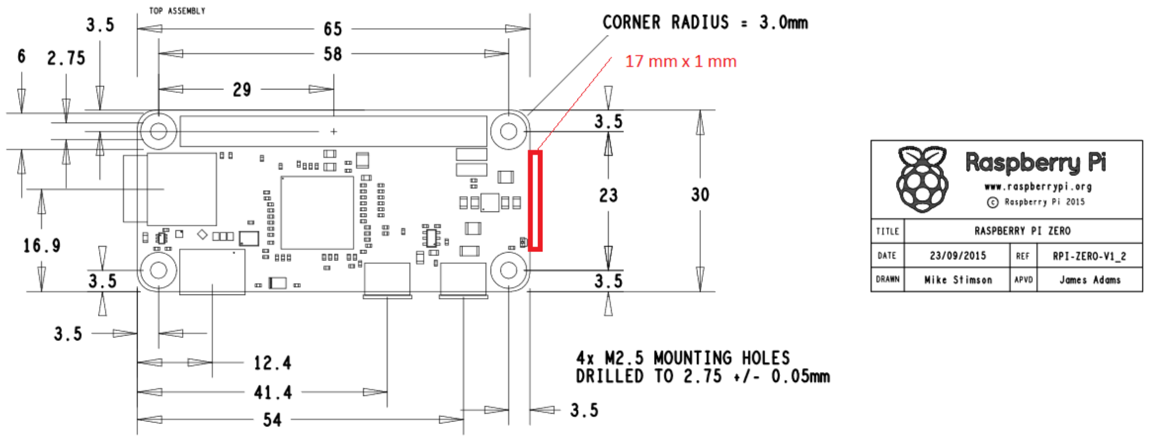


Figure 3.13: Mechanical Drawing of Raspberry Pi Zero

The dimensions of the BerryIMU are 34.25 mm x 17.4 mm with a surface area of approximately 596 square millimeters. (How to assemble BerryIMU (nd))

The dimension of the DisplayPort to MacBook Pro Retina display controller board are 80 mm x 40 mm with a surface area of 3200 square millimeters (Rozsnyo, Daniel (nd)).

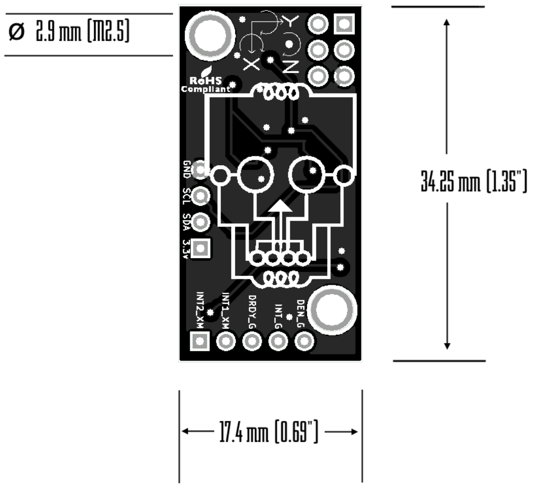


Figure 3.14: Mechanical Drawing of the BerryIMU

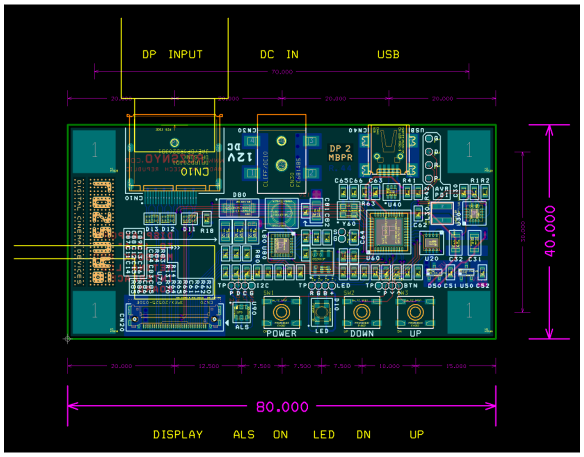


Figure 3.15: Mechanical Drawing of the DisplayPort to Macbook Pro Retina display controller board

There was also an interconnect that converted the HDMI signal to DisplayPort carried from the Raspberry Pi Zero to the DisplayPort controller board. As a consumer, it came in the form of cabling 255 mm x 23 mm, a surface area of approximately 5900 square millimeters (HDMI or DVI to DisplayPort Active Converter (nd)). So, once every controller board has been placed inside the bottom case housing, this still leaves just under 58000 square millimeters of available surface area for cables and a lithium polymer battery.

Of the various components needed to run Smart Vinyl, there are very few components that really contribute to its thickness. In fact, the only thing that contributed to thickness at all before any modifications were the EMI shielding that went around the DisplayPort, the DisplayPort itself, and the barrel jack adaptor. Of these connection points, It was the HDMI to DisplayPort adaptor that contributed to the thickness the most at 13 mm in height. Therefore, the total volume of the bottom case at the minimum needed to be 305.7 mm x 220.6 mm x 13 mm in order to comfortably fit all of the electronics necessary to run Smart Vinyl. Figure 3.16 demonstrated the connected components on the left-hand side necessary to fit inside the bottom enclosure dimensions specified before it ran as an autonomous system.

3.7. Miscellaneous Components for Project

Until connections can be soldered directly on board, much of the interconnected components had to rely on current peripheral connection standards like HDMI, DisplayPort, and micro USB. And since the previous section brought to light the undesired increase in thickness the peripheral connections contributed to the form factor, picking the interconnects themselves with the lowest profile was even more important. For example, a company called Aerialpixels customizes, builds, and sells flat ribbon HDMI cables which help to maintain a low profile(https://aerialpixels.com/product-

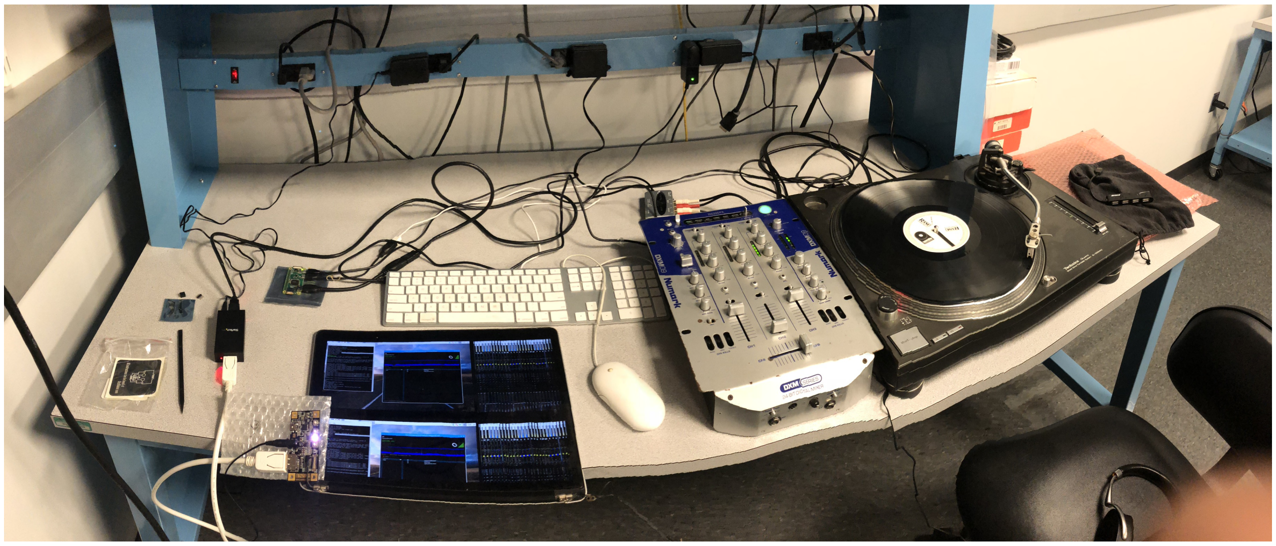


Figure 3.16: Working Phase One of Connected Open Source DVS

category/audio-video-cables-accessories/power-cables-and-adapters/). Figure 3.17 demonstrates the aerial pixel flat ribbon cable along with a variation of different HDMI interface options. Another option would be to carefully remove all of the casing from the cabling in attempts of reducing the profile of the interconnection cables.

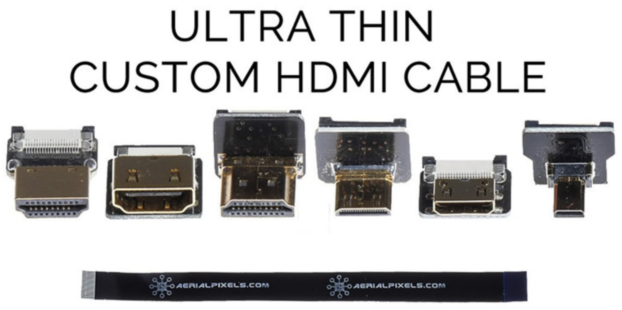


Figure 3.17: Custom HDMI Cable Builder

The barrel jack needed to adapt to the interface for a battery to power the DisplayPort converter as well as the Macbook Pro controller board. Another battery needed to interface with the Raspberry Pi Zero W by way of the micro USB.

3.8. Process Vinyl Overlay

Once the controllers were placed in the bottom case housing and the LCD was placed on top, the last remaining thing left to do in order for Smart Vinyl to interface with the turntable was to fabricate a surface for the stylus of a turntable to interpret from. In processing the top layer of the LCD, similar to that of a lacquered master record, the surface was lacquered with liquid vinyl, dried, then run through an audio fed lathe cutter applied to the dried surface. Since the machinery involved in performing the lathe cut is highly specialized, once assembled, Smart Vinyl needed to be sent to a company that made master records to perform the lathe cut. However, this wasnt́ simply lathe cutting regular audio. Rather, I had to supply the master record company the mp3 of the actual time code for them to feed to their lathe cutting machine. I have been in correspondence with Kim Gutzke, owner of Custom Records , who specializes in making one off lathe cut vinyl records. Having understood the nature of what I was providing him to be lathe cut, he explained that within a given thickness to maintain transparency and audio fidelity, a clear lacquer could easily be applied to overlay Smart Vinyl that would interface with the stylus of a turntable like any other record (One-off Lathe-Cut Vinyl Records (nd)).

References

Advanced Linux Sound Architecture (n.d.). In Wikipedia. Retrieved April 12, 2018, from https://en.wikipedia.org/wiki/Advanced_Linux_Sound_Architecture.

Alain Bouchard, personal communication, August 30 (2017).

Alvin Benjamin Carter III, personal communication, February 5 (2018).

Asoundrc (n.d.). Alsa project. Retrieved June 28, 2018, from https://en.wikipedia.org/wiki/Advanced_Linux_Sound_Architecture.

BerryIMU Technical Specs (n.d.). Ozzmaker. Retrieved July 9, 2018, from http://ozzmaker.com/product/berryimu-accelerometer-gyroscope-magnetometer-barometricaltitude-sensor/.

Bryan, N.J and Wang, G. (2011). Two turntables and a mobile phone (doctoral dissertation). Retrieved from http://www.gewang.com/publish/files/2011-nime-turntable.pdf.

Daniel James and Christopher Obbard, personal communication, October 19 (2017).

Flexi disc (n.d.). In Wikipedia. Retrieved June 6, 2018, from https://en.wikipedia.org/wiki/Flexi_disc.

Guide to interfacing a Gyro and Accelerometer with a Raspberry Pi (n.d.). Ozzmaker. Retrieved July 9, 2018, from http://ozzmaker.com/berryimu/.

HDMI or DVI to DisplayPort Active Converter (n.d.). Startech. Retrieved June 28, 2018, from https://www.startech.com/ca/AV/Converters/Video/Active-HDMI-or-DVI-to-DisplayPort-Converter~HDMI2DP.

Heath, T. (2016, December 7). Vinyl surpasses digital sales in some music markets. time to ask santa for a turntable? *The Washington Post*. Retrieved from https://www.washingtonpost.com/news/business/wp/2016/12/07/vinyl-surpasses-digital-sales-in-some-music-markets-time-to-ask-santa-for-a-tur ?utm_term=.ed6d56dae9be.

Hills , Mark (2007). xwax. *xwax*. Retrieved from http://xwax.org.

How to assemble BerryIMU (n.d.). Ozzmaker. Retrieved July 9, 2018, from http://ozzmaker.com/berryimu-quick-start-guide/#BerryIMU%20wiring.

Jeremy Sullivan, personal communication, September 15 (2017).

One-off Lathe-Cut Vinyl Records (n.d.). Custom records. Retrieved June 28, 2018, from http://www.customrecords.com/custom_cut_records_one-off_vinyl_mastering.html.

Rozsnyo, Daniel (n.d.). Digital cinema devices. Retrieved June 28, 2018, from http://dp2mbpr.rozsnyo.com.

Slayden, A., Spasojevic, M., Hans, M., & Smith, M. (2005). The djammer: "air-scratching" and freeing the dj to join the party. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05 (pp. 1789–1792). New York, NY, USA: ACM.

Technics SL-1200 (n.d.). In Wikipedia. Retrieved July 11, 2017, from https://en.wikipedia.org/wiki/Technics_SL-1200#cite_note-dmc_world_ceased_statement-1.

Turntablism (n.d.). In Wikipedia. Retrieved July 11, 2017, from https://en.wikipedia.org/wiki/Turntablism.

Vinyl Emulation Software (n.d.). In Wikipedia. Retrieved June 26, 2016, from https://en.wikipedia.org/wiki/Vinyl_emulation_software.

Wallack, J (Writer) (2005). Vinyl records (part 1 and 2) [television series]. In G. Hoss (Producer), *How It's Made*. Quebec Canada: Productions MAJ, Inc. and Productions MAJ 2.

White, D (2016, Jul 27). Are dj turntables outselling cdjs? *Dj TechTools*. Retrieved from http://djtechtools.com/2016/07/27/are-dj-turntables-outselling-cdjs/.

Zwetsloot , Rob (2017). Introducing raspberry pi zero w. *The MagPi Magazine*. Retrieved from https://www.raspberrypi.org/magpi/pi-zero-w/.