Traktor Kontrol S4 Mk3 Is Not a Turntable No Matter How Bad I Want it To Be: But That's OK Because I Accept It For Who It Is

Dr. Zeph Thibodeau
joseph.thibodeau@gmail.com
Independent Scholar
Kuala Lumpur, Malaysia
Chronogenica Arts & Technologies
Montréal, Canada

Figure 1: Performer's-eye view of an S4Mk3 Controller

Abstract

This paper documents the process of socializing a DJ controller by reverse-engineering its jogwheel motor control system. Motivated by a desire to emancipate the hardware from its proprietary software dependency, the author joins a community of developers for the open-source DJ software *Mixxx*. What begins as a straightforward implementation of a turntable emulator becomes an exercise in balancing contradictory design goals. DJ interfaces and their associated performance techniques differ from their traditional roots, most importantly in the physical dynamics of the interface. Without knowing the details of the proprietary implementation, the author and their collaborators encounter these contradictions and observe the tradeoffs in their resolution. This collective "socializing [of] the engineers' creations" [13] is an act of co-design, where the new-ness of an "old" musical interface is continually refreshed. As the motor controller becomes usable, the DJ interface becomes capable of realizing novel and alternative musical expression. Inspired by mechanical systems and fictional control interfaces, the author suggests creative applications for the DJ controller outside the scope of its traditional use.

Keywords

Machine-Human Social Relationships, DJing, Haptics, Reverse-engineering, Co-design, Control Systems

ACM Reference Format:
Dr. Zeph Thibodeau. 2026. Traktor Kontrol S4 Mk3 Is Not a Turntable No Matter How Bad I Want it To Be: But That's OK Because I Accept It For Who It Is. In *Proceedings of International Conference on New Interfaces for Musical Expression (NIME '26).* ACM, New York, NY, USA, 10 pages.

NIME '26, June 23–26, 2026, London, UK

1 Introduction

Machines and humans are socially inseparable. They grow together within a social context and form interdependencies. Like flowers rely on bees, machines rely on humans as a necessary element in their reproductive and evolutionary cycles. Humans rely on machines to facilitate or automate material interactions throughout our life cycles. It is hardly a surprise that humans form strong social bonds with machine-families and machine-individuals like trains, bicycles, typewriters, and of course, interfaces for musical expression.

Communities of practitioners and technicians[1] grow around a given class of machines in a variety of social contexts. For instance, Julian Orr conducted an ethnography of Xerox field repair technicians, documenting their community knowledge-sharing practices and nuanced relationship with their copiers [12]. To generalize on Waters' notion of *performance ecosystems*, the situations in which machines and humans find themselves are inseparable from the social processes that produce machine and human identities: "configurations of mutual interpenetrations between physical, social and historical/temporal context, and individual capacity"[21]. On his discussion of the ephemerality of musical instruments, Gouard observes that musicians' "feelings and desires, musical skills and awareness, projects and physical capacities, all evolve during their existence", which is "reflected in the instrumental device, by the addition or removal of features, or the development of new instruments related to a new musical project"[5].

[1] among other roles

I myself experienced a musical co-evolution with my trumpets through interface augmentation, which occurred along with evolutions in the social context of our musicking [15, 18][2]. As *DAAT*, electronic music duo, we explore the blurred boundary between machine and human as both an aesthetic and a reality as we constructed science-fictional machine music while constructing the machine-producing machine of which we are components[8]. My recent work has moved towards a radical acceptance of machines as social participants, which I explore primarly through the medium of social robotics [14, 16, 17, 19, 20, 22].

NIME's tendency towards novelty is the subject of ongoing discussion, whether celebrating instrumental ephemerality [5] or attempting to maintain longstanding projects [11]. There are corresponding worries about sustainability [4] and the larger political economy in which NIME exists [10]. Whatever the results of these discussions on the idea of novelty, and however the tendencies within the community change in time, NIME remains a rather durable group of diverse practitioners who gather to discuss a peculiar family of machines, not all of whom are destined to live long. Our repertoire of technical, philosophical and artistic knowledge is by this point extensive. This year's theme reminds us not to take this knowledge for granted. How can we learn and participate in music communities outside the auspices of the conference?

I offer a project with participation-first approach that begins "outside" of NIME, joining and collaborating with an existing community on its collective goals. In the project described in this paper, it was an organic process borne out of my artistic needs, a specific DJ controller, and an open-source DJ software project.

1.1 Personal/Artistic Motivation

Material circumstances and my near-pathological DIY attitude brought me into this project. I had moved internationally, and my beloved turntables couldn't join me. I had long practiced DJing alongside music production, and until the move I didn't fully appreciate the value of having a tactile relationship with the music. The sense of connection that a turntable brings to a track—like the connection to a road through a bike—brings an intimacy of attention as it combines active listening of a composition with performance of the composition. There is plenty of literature about the peculiar history of the turntable as a novel interface [6], so I won't wax too long on the kind of creativity that the instrument facilitates. But it's remarkable to me how the manipulation of a disc, mechanically coupled to a recording, changed my relationship to sound.

In my new situation I needed something to fill the void that my turntables had left, and I wasn't ready to abandon the tactility to which I was so accustomed. Furthermore, I became intrigued by the possibilities that a digital controller would afford. What artistic transformations would it precipitate if it could be more than just a turntable-replacement—and indeed, more than a DJ controller?

And what if, bear with me, the Traktor S4 Mk3 could be more than just an instrument or a machine? What if it could be... a friend? Someone who inspires me, plays with me, someone with whom to share special moments. When we design new instruments, we are not designing "new things" so much as we are creating social connections that in turn produce relationships [14]. If we want our creations to have a lasting impact, no matter how long they live, we must begin by assuming the role of caretaker [17]. Would the Traktor S4 Mk3 fill the void that my turntables had left? Maybe—and if it didn't, then what? Either way, the space between present and future would be filled with mostly dry and technical steps for a long while before any answer would be known.

These everyday and sometimes repetitive aspects of being with machines are all part of getting to know them. So without further ado, allow me to tell you all about the Traktor S4 Mk3, and what makes it tick.

2 Finding the Instrument

2.1 DJ Controllers with Motorized Jogwheels

There are several such controllers on the market. Some, like the *Rane Twelve MkII*, replicate the form and feel of a turntable as closely as possible, with 12" platters and physical slipmats. However, full replication is a large and expensive option. Tradeoffs are necessary between size, complexity, and price point. In my budget and for my uses, the *Native Instruments Traktor S4 Mk3* seemed to have everything I needed, despite having rather small jog wheels. It was usable as a 4-channel mixer even without tethering to a computer. It had four auxiliary inputs, two with mic preamps and two with phono preamps. It seemed to have a good reputation online and I was intrigued by its so-called *Haptic Drive™* technology:

> [the motors can] perform new tricks that weren't possible before, such as giving your fingers a little bump when scrolling past a cue point [...] There's turntable-style torque when beatmatching, and recognition of certain gestures such as backspins, where resistance is instantly dropped so the decks act like vinyl on a slipmat.[9]

The promise of novel modes of interaction was like catnip to me, and I wondered at the possibilities of this DJ controller not just to replace my turntables but to explore the interactive potential of force-feedback jogwheels. I bought the controller, and found it shockingly intuitive to mix out-of-the-box with their turntable emulation in *Traktor Pro 3*. It was an absolute delight.

2.1.1 *Aside: Could the Haptic Drive help revive the D'Groove?*

The idea behind the Haptic Drive was not new even at that time. More than a decade prior to the release of the Traktor S4 Mk3, Beamish designed a haptic DJ controller[2]. Several of the "haptic effect modes" resemble those of Native Instruments: their "haptic cue" operates on the same principle of Beamish's "beat hills". The D'Groove went further with gestures such as "plucking" the platter to induce a prolonged oscillation. Theoretically, Beamish could revive the D'Groove project using the Traktor S4 Mk3 hardware and a custom motor controller!

2.1.2 *The Hardware/Software Duality.*

Digital DJing with a DJ controller is quite different than using timecode vinyl. Realistically this was my first time contending with cue points, loops, sampling and FX. I realized that my performance practice would be tightly bound to the software as much as the hardware. This worried me from the standpoint of instrumental identity and my intended use for the controller. By instrumental identity, I mean the embodied or tacit comprehension of the instrument as an instrument through the performance of musical interactions[7]. I wanted trust and influence in the continuation and mutation of the instrument's form—and my ability to grow with it. If I was

[2] my publications on the subject really only discuss the technical aspects of the relationship. Sadly the musical evidence is lost in old hard drives and I was not sensitive to the sociological aspects at the time.

going to commit myself to develping techniques for this new practice, with this interface, I needed a software whose form I could adapt as well.

To be clear, I have no issue with *Traktor* or Native Instruments. But trusting the software-half of my instrument to a cloud-licensed product brings me existential fear. I have had enough experiences with proprietary software and feeling trapped in a commercially-oriented design space. My 2012 MacBook has been abandoned by Apple, and kept alive with Linux. Sadly, Traktor doesn't work on Linux. And what if Native Instruments goes bankrupt?[3]

I had experienced something like this for another interface, the *Livid Instruments OHM RGB*. The manufacturer is gone, but a devoted community has kept their instruments alive. On Facebook the *Livid Instruments Resources* group trades repair information, firmware and software links, and shares stories of spotting or finding controllers. It's not terribly uncommon, even for companies that merely abandon their products. Vintage gaming consoles have a massive grassroots restoration, modification and customization scene, and for 21st century consoles with defunct cloud-based distribution services this necessitates a degree of piracy and hacking[4]. It seems as though machines stay alive by a process of re-design, which includes maintenance, modification and repair. Although it is sometimes characterized as "the antithesis of design", repair is in fact an "expression of care, and therefore a way of making ethical decisions about design" [3]. Therefore the collectivisation of repair and maintenance is at once a democratic exercise and an expression of social care. It calls into question obsolescence as an unavoidable fate, and creates opportunities for social and technical creativity. Therefore by participating in a community initiative to fix the Traktor Kontrol S4 Mk3's compatibility with the open-source DJ software *Mixxx*, I would be helping create opportunities for this interface beyond the scope of its nominal purpose.

3 Mixxx: an Old Interface for Musical Experimentation

As it happens, *Mixxx* was presented by Tue Haste Andersen at NIME 2003, with the stated goal "to make it possible to conduct interaction studies of novel interfaces in relation to the DJ situation"[1]. Commercial offerings of the day, particularly Stanton's *Final Scratch*, catered to existing turntable-based DJ practices but did little to "provide any further novel additions to the user interface, apart from a standard scrolling waveform display"[1]. Furthermore, Andersen warned that "the purpose of *Mixxx* and the research conducted with *Mixxx*, is not to replace the turntable as instrument, but to find new and better ways of navigating and giving new inspiration to the artists. In this sense, staying completely true to the turntable metaphor is likely to limit more than open new design possibilities"[1]. What an ironic twist, given present company! Here I was, unaware of *Mixxx*'s founding goals, attempting to reinstate a turntable metaphor within the software—using a contemporary interface decked out (so to speak) to faciliate all manner of contemporary DJ practices which would have been novel at the time. All the same, *Mixxx* itself has evolved since then, as have its goals. Among a competitive digital DJ software market, *Mixxx* currently emphasizes its powerful feature set, open source codebase, and friendly community [5].

Over the course of the past year I have collaborated closely with several (volunteer) developers of *Mixxx* to analyze and re-implement the S4 Mk3's software motor controller as a first step to a more general-purpose open-source driver for use in experimental and new interface projects. Before going anywhere near the code, however, I felt it was necessary to work through the physics of a turntable and the model that would have to underly any turntable emulator.

4 Physical Modeling of Turntable and Slipmat

4.1 Problem

Simulate the physical system comprising a vinyl disc, slipmat, and motorized platter. Calculate the outgoing motor torque necessary to mimic the behaviour of given a platter of radius r and slipmat coefficient of friction μ .

4.2 Parameters

- Disc/platter radius (m) and mass (kg). For a 12" disc, say 0.15m and 0.15–0.2kg
- Slipmat coefficients of static and kinetic friction (no unit). Not sure of exact coefficient for a given slipmat material, as people have their own preferences. This will have to be left to trial and error.
- Maximum output torque (Nm) of the motor. For a Technics 1200 this is around 0.15Nm

4.3 Functional States

There are two primary states of the simulation: STICK and SLIP. The disc and platter STICK together unless the performer touches the disc AND alters the angular velocity of the jogwheel enough for the disc to SLIP. The disc remains in the SLIP state until its angular velocity return to within a margin of the target velocity, at which point it STICKs again.

4.3.1 STICK state. The vinyl disc, slipmat, and platter are stuck to each other and move together. In this state, the jogwheel represents the entire assembly. The crown/edge of the jogwheel represents the crown/edge of the platter. The performer can hinder or help the rotation of the jogwheel, and the motor works to correct any deviation from the target angular velocity.

$$\alpha_{motor} = \frac{2}{m \cdot r^2} \cdot \beta \cdot \tau_{max} \tag{1}$$

In equation 1, the rotational acceleration α_{motor} depends on the motor's maximum output torque τ_{max} and a scaling coefficient β determined by the control code.

$$\tau_{T_x} = \frac{2}{3} \cdot m \cdot \overrightarrow{g} \cdot r \cdot \mu \tag{2}$$

Equation 2 describes the maximum torque transfer across the contact plane of the slipmat. Assuming the slipmat is always "glued" to the motor platter, the maximum torque that the motor can transmit to the vinyl disc depends on the disc's mass m, the force of gravity $\overrightarrow{g}$, the disc radius r and the slipmat coefficient of friction μ.

$$\alpha_s = \frac{2}{m \cdot r^w} \cdot \tau_{T_x} \tag{3}$$

[3] as it seems they might, based on recent news: https://cdm.link/ni-insolvency/
[4] see for example https://3ds.hacks.guide/
[5] https://mixxx.org/

Mirroring equation 1, the maximum acceleration that the slipmat surface can tolerate before slipping occurs. The error function driving the system is the difference between the motor platter rotational velocity V_m and the vinyl disc rotational velocity V_D

$$\epsilon_v = V_m - V_d \tag{4}$$

If the performer perturbs the rotation of the jogwheel while touching the disc surface, the same behaviour applies UNLESS the perturbation is large enough to defeat the force of friction that keeps everything stuck together. In other words, the target and measured velocity are too far apart. At this point, the simulated rotation of disc and platter decouple from each other and we enter the SLIP state.

4.3.2 SLIP state. The vinyl disc moves at a different rate than the motor, and only (kinetic) friction works to bring them back in sync (equation 5). In this state, **the jogwheel only represents the rotation of the disc**. The platter is abstracted and we assume that it rotates at the target velocity.

$$\alpha_{slip} = \frac{\mu \cdot \overrightarrow{g}}{3 \cdot r} \tag{5}$$

The torque applied to the jogwheel motor represents the force of friction, which pulls in the direction of the target angular velocity. This means that when the performer is scratching, the motor is sometimes helping, if the scratch is moving the disc closer to synchrony with the platter.

Anytime the disc and platter velocity are nearly matched, the simulated platter regains its influence—in other words, within a certain margin the system returns to the STICK state, even if only for a fraction of a second.

Finally, whether by manually bringing the disc to the target velocity or allowing the simulated friction force to do it, the system syncs up and re-enters the STICK state. We don't care if the performer is still touching the disc, as long as they aren't bringing it out of sync again.

5 Analysis of Existing Motor Controller

5.1 S4 Mk3 Compatibility with *Mixxx*, April 2025

When I began my involvement, *Mixxx* had partial compatibility with the S4 Mk3. It was usable, but lacked implementation of the S4's most sophisticated features: the motorized jogwheels and the built-in LED displays. Both were the subject of ongoing development, but the motor code hadn't been touched in a while. The S4 communicates over USB with an HID protocol instead of the MIDI used by its predecessors (and most other controllers), presumably to better handle the timing resolution and data density needed to control the motor. Thanks to prior community development, the content for the S4's HID reports were already mapped. We could talk to the motors, but the control logic was at that time underdeveloped, and turntable-mode was realistically unusable.

5.2 Procedure for Profiling and Implementing Control Code

The HID mapping had been previously accomplished by sniffing USB traffic between the S4 and Traktor. I recreated and verified this work using *Wireshark* along with *usbpcap* in Windows 10 to analyze Traktor, and *Wireshark* with *usbmon* in Linux for analyzing HID communications in *Mixxx* (see figure 2).

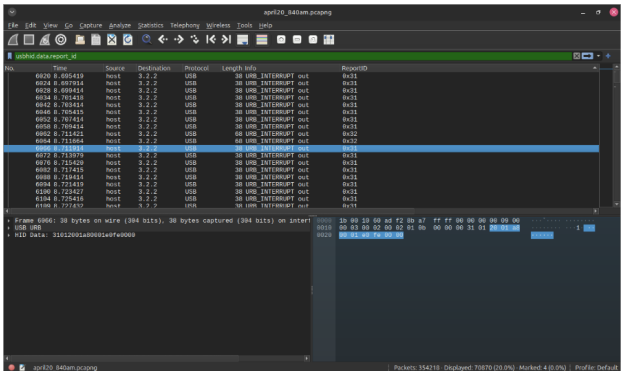

Figure 2: WireShark UI Showing USB data

I recorded various use-cases with the following test procedure:

- With Traktor already running, run USBpcap thru Wireshark
- Turn on the S4
- Adjust some controller settings in Traktor:
- motor "tightness"
- relative vs absolute pitch fader
- 33 vs 45 rotation speed
- Load a track on both decks and press play
- For each deck:
  - Slow the rotation by dragging on the crown
  - Speed the rotation by pushing on the crown
  - Slow the rotation by dragging on the disc
  - Speed up the rotation by pushing on the disc
  - Baby-scratching on the disc
  - Baby-scratching using the crown
  - Play in reverse

The captured traffic was exported as a CSV file to be analyzed in a custom Python script using the csv, numpy, and matplotlib modules. By iterating through rounds of test captures, analyses, consultation and implementation, I converged on a control system that closely matched the response of Traktor.

5.3 Positional Sensing

The S4 jogwheels are sensed with a two-track wheel encoder disc and five infrared reflectance sensors: four aligned with the outer track and one with the inner (see figure 3). The outer track comprises 360 windows alternating between black (no reflectance) and silver (full reflectance). With four sensors positioned out of phase with each other, the resolution of the pattern can be multiplied by a factor of eight. Indeed, this seems to be the case because one of the two positional values reported by the device wraps around at $360 \cdot 8 = 2880$, representing one full turn with a resolution of 0.0125 degrees per pulse. The inner track of the wheel encoder disc has only 10 windows and a single sensor. It is reasonable to assume this provides some degree of robustness to the outer ring measurement, particularly in the case of high-speed rotation which would otherwise cause aliasing.

5.3.1 Redundant Positional Data. Some sensor signal cleanup is evidently performed on the hardware. In the motor reports, which are sent at a rate of 500Hz, there are two values representing wheel's position. A second positional data stream varies in parallel to the first, but unwrapped to use the full range of an 8-bit unsigned integer. It also seems to correct occasional errors

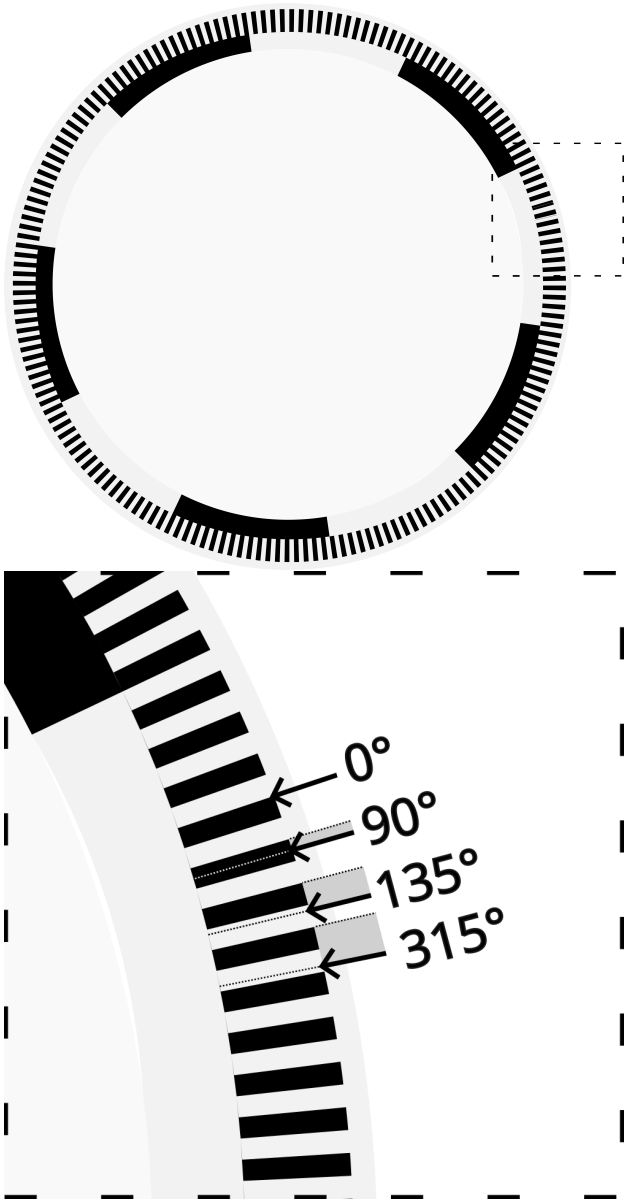

Figure 3: Jogwheel sensing. Top: two-track wheel encoder with 360 outer and 10 inner divisions. Bottom: example of 4-position reflectance sensing to achieve 0.0125-degree precision. In this example, the output will produce a Gray code (differing only by 1 bit for adjacent wheel states).

in the "raw" positional data, as seen in figure 4. The method by which this error correction is accomplished is opaque, but I suspect its ability to correct discontinuities is possible due to the second encoder track on the jogwheel.

5.3.2 Touch Sensor. The metallic surface of the jogwheel is touch-sensitive, represented by a single bit in the HID wheel reports. It appears as though the capacitive sensing is implemented with the electric field sensor itself on the (stationary) jogwheel PCB across from the rotating surface of a ring-shaped antenna that is presumably coupled to the outer jogwheel surface.

Informed by the touch sensor, the motor controller can detect two modes of interaction—disc interactions, which engage

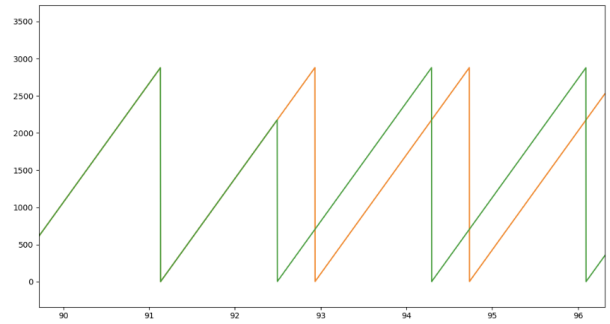

Figure 4: The raw sensor values reset to zero, whereas the processed signal (re-wrapped at 2880 for comparison) does not suffer the same discontinuity.

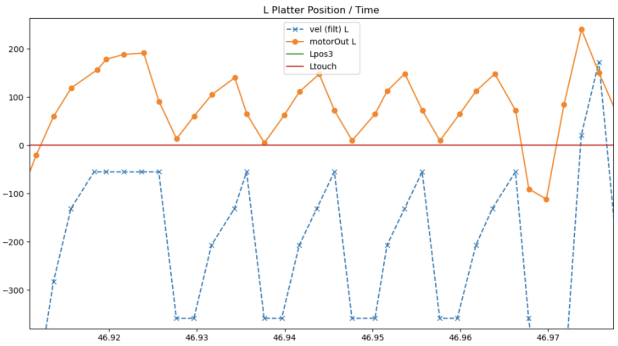

Figure 5: real motor data compared with a filtered version of the reported velocity data — trial and error to make them resemble each other on the premise that the control signal for the motor has to be proportional to the "cooked" input signal.

with slipmat physics, and crown (edge) interactions, which don't engage slip physics.

5.3.3 Prototype signal analysis in Python. I plotted everything using the USB capture clock, which aligned the data nicely. Then, I looked at the motor output compared with the calculated velocity. The velocity is very noisy, and whereas the output is also noisy it appears lowpassed by comparison. My assumption is that the motor output is proportional to the velocity calculated from the position input.

The test data includes a section (during initialization perhaps) where the velocity becomes a 1-sample (alternating sign) pulse wave. This provides a perfect test case for understanding the filtering that produces the motor output. So, by playing with the weights of an FIR filter on the velocity, I was able to produce a contour that approximates that of the motor output signal (figure 5).

I wanted to check the velocity reported by the system, and how it corresponds to 33/100 RPM. I know that 33/100RPM is equal to 200deg/s. I assumed that the optical wheel encoder, which has 360 segments, would give us degrees directly. However, visually analyzing the positional plot and calculating the slope for one full cycle of 2880 ticks produces an average angular velocity of 1600 ticks per second. Since I know this represents 200 deg/s, I

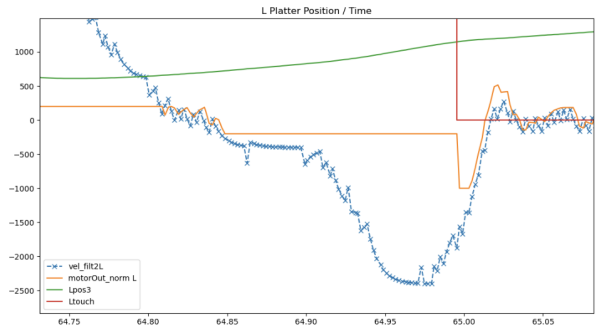

Figure 6: Detail of the motor output as it crosses the target velocity

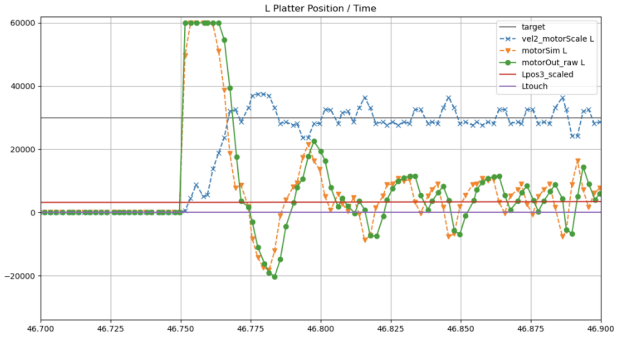

Figure 7: Success! The simulated motor controller trace is a close match to the measured one

can confirm the ratio as 1600/200 = 8 ticks per degree, matching the expected wheel encoder resolution.

With some more playing around with filtering the velocity, adjusting the zero-point to the target velocity, and flipping the sign so that it matches the motor output, I can see some more of what's happening with the model (figure 6).

Here I am able to confirm that the model I've been working on is correct. While scratching, the velocity (blue) passes across the target, causing the motor output (yellow) to switch between its calibrated slip-friction torques. However, as the velocity gets close to the zero-point, the motor starts trying to "snap" it to the target again as the slipmat "sticks" to the platter. This can be modeled as a change in the coefficient of friction accompanied by a "letting through" of the (virtual) motor torque into the user-perceived resistance. This should mean that if you're hand-spinning the disc it will not slip if you're close to the target value. I checked through the traces and this is indeed correct!

I tweaked the velocity filtering to use a proper windowed weighted-average convolution filter, which seemed to reasonably line up with the contours of the motor output. Then, I implemented a PID controller and tuned it to approximate the captured motor output from Traktor. It was almost there but I suspected there was a further smoothing factor at play, so I tried adding a smoothing step and voilà, it's pretty much matching! (see figure 7).

This PID controller and smoothing step are all determined by trial and error, which goes against the physical model implementation I was hoping to use. However, it will still be possible to link the tuned parameters to approximate physical values later.

6 Implementation and Associated Challenges

6.1 Message Timing

The S4 reports its motor data at 500Hz (every 2ms) and expects a control message at the same rate. If no motor commands are received, the jogwheel will brake to a stop. This revealed a problem with the way that input devices are handled by *Mixxx*. Under the hood, the software engine is built in C++ using the QT framework with interface device code in javascript. This imposes a minimum of 20ms on calls to QTimer, which is the only timing mechanism available in the javascript-to-C++ connection.

Consulting with the other developers, it was suggested that I remove the minimum timer value as a temporary measure. This worked on my MacBook running Linux, but others reported they were unable to run the timer at 2ms, so this wouldn't work for a production release [6].

Thankfully, we determined that we could avoid using QTimer for motor output commands by sending them in response to the motor input reports. Since the input reports were processed as soon as they arrived, this meant we could effectively use the S4's message rate as a timing source.

6.2 Playback and Scratching

After porting the experimental Python code to *Mixxx*/javascript, and coupling *Mixxx*'s playback rate to the jogwheel rotation, I noticed that regular playback showed significant flutter—small but rapid pitch variation—during unperturbed playback. It was clear that the system was unable to maintain a steady enough rotation speed to act like a "real" turntable. Traktor didn't have this issue, but upon further testing I observed that activating the touch sensor without affecting the jogwheel rotation would create an identical flutter effect in the official software. Therefore, Native Instruments had encountered this same problem. Their solution was to quantize the playback rate unless the user was touching the platter (presumably to perform scratching). This proved to be the simplest solution for our implementation as well, but touch-sensing alone wasn't sufficient to characterize every instance of slipmat physics. Specifically, if the user spun the wheel quickly and then removed their hand from the disc, the emulator needed to maintain slipmat physics and pull the disc velocity back to the (virtual) motor velocity using a simulated friction force. Only once the velocities were synchronized would we switch back to quantized playback.

6.3 Nudging During Playback

Nudging—making phase adjustments using the crown (edge) of the platter during playbck—presented a different challenge. If the motor output was proportional to the measured velocity, and the user perturbs the rotation *without touching the disc surface*, how can we detect the perturbation?

The answer came thanks again to community discussion. This gesture required comparing the motor output to a known / calibrated value associated with the target rotation speed. It was a reverse mapping of the control model to this point—instead of mapping the measured velocity to playback rate, we now had to map the *motor output error* to playback rate. This required knowledge of the expected rotational velocity for a given output value. Positional measurements were simple because the physical phenomenon was fully known; however, the motor-output to

[6] See https://doc.qt.io/qt-5/qobject.html#startTimer

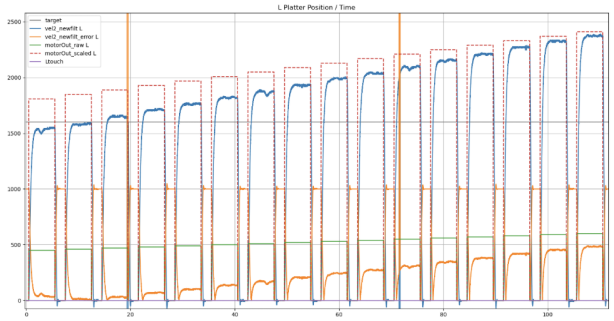

Figure 8: Full motor test shows a roughly linear relationship between motor output value and rotational velocity

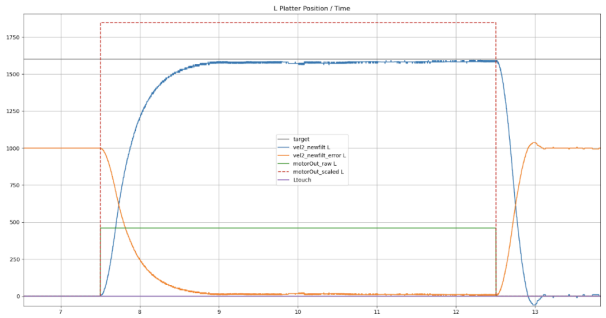

Figure 9: With output value of 460, the jogwheel spins at 1600 pulses per second or 33.3RPM

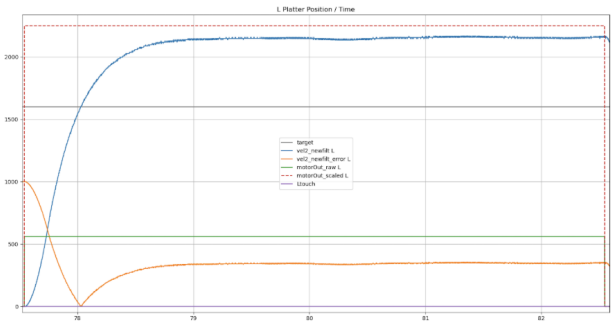

Figure 10: With output value of 560, the jogwheel spins at 2150 pulses per second or 45RPM.

rotational-velocity causal chain depends on elements that we cannot know without deeper investigation of the digital and analog characteristics of the signals involved.

6.3.1 Motor Output Measurements. Unlike the positional data, whose values correspond to physical measures, the motor output level seemed arbitrary. The captured USB traffic showed its maximum value as +/-60000, but other than that we had no clue as to the mapping between this value and the rotation speed. Lacking a theoretical basis to give meaning to this value, I ran another set of tests, this time stepping through a series of output values and measuring the resultant jogwheel velocity. The test procedure was:

- send a steady motor output value to each motor for several seconds
- send zero to each motor and wait a couple of seconds
- increment the output value and repeat the process

This gave me a guideline for where the target output levels would be for 33RPM and 45RPM shown in figures 8 9 and 10. Sure enough, we were able to use these reference values to determine the velocity error due to nudging. With a scaling factor for adjusting the depth of the nudge effect, it worked as expected. This did require an extra layer of state awareness on the part of the software: from a standstill, the jogwheel will ignore velocity error (for the purposes of nudging) until the wheel is fully up to speed (defined by an error threshold).

6.4 Motor Reverse

With the motor controller working and the PID tuned, it was a satisfyingly simple matter to implement reverse-drive. The velocity simply had to be set to a negative value and the control system took care of the rest. Of note is that, when using the 440Hz sine calibration tone, it's obvious that the reverse-drive has a different calibration than forward-drive, requiring higher output to achieve the same rotational speed.

Speaking of, I also note that Traktor never spins the jogwheel backwards in TT mode when REV is applied, whereas we've got the motor spinning backwards which is admittedly very cool. I was playing with a 440Hz Sine tone in *Mixxx* and noticed that the rotation speed is slightly off when we use the REV function (confirmed as frequency-beating when mixed with a 440Hz test tone). I'm not sure if this should be a concern with regards to motor specs and wear&tear but it's worth noting.

6.5 Cueing Behaviour

Cue points (and hot cue point) are markers within a sound file that the performer can jump to at any point. These posed a curious problem when combined with a motorized interface. When hitting the 'cue' button in *Mixxx*, the playhead needs to snap to the cue point and not move at all, even a sample. Otherwise, subsequent presses of the 'cue' button will reset the cue point instead of playing from the cue point.

The difficulty is that it takes time for the motorized platter to slow to a stop. If playback is tied to the rotation of the motor, this results in the playhead moving and the audio slowing down to a stop as well. The spin-down audio effect is actually desirable when we press play/pause during playback, as this emulates a turntable quite well. But it has to be disabled when snapping+stopping to a cue point.

Community discussion about this problem was divided. For some, the cueing behaviour had to meet the same expectations in turntable mode as well as the non-motorized jog mode. However, the simplest way to make this happen would involve decoupling the wheel motion from playback, and re-coupling it once the motor was fully stopped. Easy enough, but this would also remove the spin-down effect that gave turntable mode part of its charm. In other words, we would have to compromise the turntable-ness of the emulation in order to conform to digital DJing performance practices.

Even when this compromise was implemented, it introduced a level of complexity and tricky timing to an otherwise simple interaction. Even a single sample's worth of movement after spinning down the platter would offset the playhead and cause the next button-press to move the cue point instead of starting playback. At the time of writing this is unsolved, as the jogwheel always seems to have some springiness after a hard stop.

To inform our discussion I tested the cueing behaviour in Traktor with turntable mode enabled. Their Cue button behaviour was strange—I couldn't get it to cue-and-stop. It would always continue playback from the cue point without stopping, no matter how the software preferences were set. I was confused by the strange behaviour of the CUE button in Traktor so I kept playing around with it and discovered something amusing and informative.

There is an undocumented 'feature' of the S4Mk3 with Traktor, in which the behaviour of the CUE button changes in TT mode. When the S4Mk3 is in TT mode, it will never cue-and-stop. It will always cue-and-play. This is either because they assume that in TT mode people will be using cue points while scratching, or that people have a different performance expectation, or that they encountered the same problem as us and decided to circumvent it entirely.

Speaking as someone who came from a vinyl system without any experience of CDJs, I'm not bothered at all by this behaviour and didn't notice before—I'm accustomed to moving the needle and then scratching over the cue to 'throw' the platter on the downbeat.

For now we have gone the same route. But as one developer pointed out, as good as it is to match Traktor's behaviour, *we have the chance to make a better system, so why not try?*

7 Can We Keep it Going?

At the time of writing, the turntable emulation is usable for performance, and I have joyfully played several sets with it—small parties—without any technical faults. In comparison to Traktor there are noticeable differences in the haptic feel, but I find that it is becoming less and less helpful to try and tune the jogwheel motors to match Native Instruments' implementation. I've realized this is due to my own adaptation. One might say I'm overtrained now, and it will take a group effort to find the "right" settings for a public release. There are improvements to be made before it can be incorporated into the next public release as a fullblown feature, but it is available if one is willing to compile *Mixxx* for themselves[7]. In fact, the Traktor S4 Mk3's built-in LED screens are also nearly working thanks to the dedicated efforts of the community, which means that the controller will finally be fully functional outside of its proprietary ecosystem! And not a moment too soon, as Native Instruments has began preliminary insolvency proceedings[8].

What does this mean for the future of the S4 Mk3? Well, as long as there is a community that cares about it, there is a future for this remarkable device. We have everything we need to continue our journey together, which in my case will involve opening the device to musical contexts outside the scope of *Mixxx* and DJing.

8 Future work and NIME-ification

With the knowledge of the S4Mk3 and the groundwork in place, possibilities for creativity in its use can be more easily realized. In keeping with *DAAT*'s obsession with vehicles, I draw inspiration from cockpits, ship bridges, space ships and game navigational GUIs. For example, the "navigator" seat from Disney's *Flight of the Navigator* has fascinated me since childhood, and could be adapted to a dual haptic-wheel interface for our interactive track *AC3244*[8] (figure 11). Rotating discs resemble the sweeping of

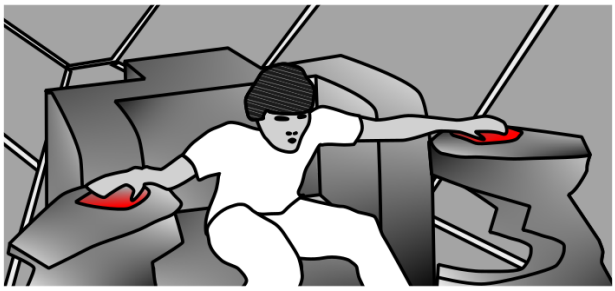

Figure 11: Example of a typical use-case of the flight interface in *Flight of the Navigator*. The spaceship's hand-operated controls are more akin to 3D mice than turntables, but nevertheless fall within a broad category of what I might call open-hand akimbo interfaces

Figure 12: Primary interface element in the video game *In Other Waters* resembling a SONAR display. In gameplay it is used for navigation, object localization/tracking and for "pinging" the environment. Such modes of interaction, while distant from DJ performance techniques, suggest evocative metaphors for navigating a musical space. Indeed, in my work with *DAAT* the concept of vehicular navigation features prominently. Image from promotional material ©2020 Fellow Traveler Games

RADAR or SONAR readouts, for which blips could be represented with tactile "bumps" during rotation. The game *In Other Waters* makes use of such a concept as its central interface element (figure 12)[9]. Audio-only games such as those we described in [8] would lend themselves to nonvisual feedback, and I have prototyped such games before[10] (figure 13).

These are merely a few ideas for where to go, and reflect my own sources of inspiration and approach to music-making. To make the interface available for widespread experimentation, it would require a general-purpose driver or middleware that

[7] current experimental branch at the time of writing can be found at https://github.com/jtMUMT/mixxx/tree/fix/s4-motors-chattering-and-cue
[8] https://cdm.link/ni-insolvency/
[9] https://www.fellowtraveller.games/in-other-waters
[10] audio demo accessible at https://youtu.be/_E441fgJuoU

Figure 13: Screenshot from an audio-only game demo *Attentional Signals*

would take care of basic functions like rotation control while translating the HID data into MIDI and OSC for use in various multimedia production and performance applications.

My early experiments in building such a driver have been fruitful. After all, there isn't much difference between what *Mixxx* already does and what such a program would do. The labour that this community has devoted, towards a shared love for music and for this interface in particular, fills me with determination and a sense of collective responsibility. Whatever new musical roles I might envision for this machine, we are anchored to a community of practitioners and technicians who can both preserve traditional modes of interaction and develop novel ones. As long as there are people who care about the instrument, it will flourish thanks to the diverse expertise and artistic ambition of its humans.

9 Socializing (With?) Machines

Orr's ethnography of Xerox repair technicians describes their role as "both socializing the engineers' creations and dealing with that willful ignorance on the part of both the creators and the society that uses those creations"[13]. I take inspiration from this perspective and offer it as a way of understanding the role of instrument designers and players in the NIME community. My own journey thus far with the Traktor S4 Mk3 has been a process of socialization for both of us: finding our people, finding our place and in doing so finding firm ground on which to build into the future. Whereas the project as described is little more than a compatibility layer connecting an "old" software with an "old" interface, its apparent lack of novelty is misleading. My speculative applications of this device are yet only dreams, but this dreaming is no longer mine alone. At least one other member of the *Mixxx* community has expressed their excitement at how they can expand their DJ practice using the new motor controller. A few others have wandered into the forums to ask about motor controller development. The truth is that I cannot say exactly how—or to what degree—an open-source driver for the Traktor S4 Mk3 will transform our individual and collective musicking. What I can say is that, as someone who cares about this instrument, who has found others to share the work of caring, we can more powerfully evolve alongside our beloved machine-collaborators to realize creative futures together.

10 Ethical Standards

This research was conducted independently with no external funding. In general the members of the Mixxx development community are referenced for their historical contributions to public repositories. Forum discussions over the course of the project are referenced anonymously, with the consent of the specific participants to whom authorly credit was also extended. No machines were harmed or subjected to unnecessary disassembly during the course of this project.

Acknowledgments

Deep appreciation to the *Mixxx* community. Thank you!

References

[1] Tue H. Andersen. 2003. Mixxx: Towards Novel DJ Interfaces. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Montreal, Canada, 30–35. https://doi.org/10.5281/zenodo.1176484

[2] Timothy Mark Edward Beamish. 2003. *D'Groove - a novel digital haptic turntable for music control.* Ph. D. Dissertation. University of British Columbia. https://doi.org/10.14288/1.0051459

[3] Alexandra Crosby and Jesse Adams Stein. 2020. Repair. *Environmental Humanities* 12, 1 (May 2020), 179–185. https://doi.org/10.1215/22011919-8142275

[4] Enrico Dorigatti and Raul Masu. 2022. Circuit Bending and Environmental Sustainability: Current Situation and Steps Forward. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Andrew McPherson and Emma Frid (Eds.). Auckland, New Zealand. https://doi.org/10.21428/92fbeb44.18502d1d

[5] Vincent Goudard. 2019. Ephemeral instruments. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Marcelo Queiroz and Anna Xambó Sedó (Eds.). UFRGS, Porto Alegre, Brazil, 349–354. https://doi.org/10.5281/zenodo.3672990

[6] Kjetil Falkenberg Hansen. 2010. *The acoustics and performance of DJ scratching*. Doctoral. KTH, Stockholm, Sweden.

[7] Sarah Hardjowirogo. 2017. *Instrumentality. On the Construction of Instrumental Identity*. 9–24. https://doi.org/10.1007/978-981-10-2951-6_2 journalAbbreviation: Musical Instruments in the 21st Century: Identities, Configurations, Practices.

[8] Jason Hockman and Joseph Thibodeau. 2018. Games Without Frontiers: Audio Games for Music Production and Performance. *Music Technology with Swing* (2018), 171–183.

[9] Native Instruments. 2018. How we made the Haptic Drive™ – Native Instruments Blog. https://blog.native-instruments.com/how-we-made-the-haptic-drive/

[10] Fabio Morreale, S. M. Astrid Bin, Andrew McPherson, Paul Stapleton, and Marcelo Wanderley. 2020. A NIME Of The Times: Developing an Outward-Looking Political Agenda For This Community. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Romain Michon and Franziska Schroeder (Eds.). Birmingham City University, Birmingham, UK, 160–165. https://doi.org/10.5281/zenodo.4813294

[11] Albert-Ngabo Niyonsenga and Marcelo Wanderley. 2023. Tools and Techniques for the Maintenance and Support of Digital Musical Instruments. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Miguel Ortiz and Adnan Marquez-Borbon (Eds.). Mexico City, Mexico, 212–218. https://doi.org/10.5281/zenodo.11189159

[12] Julian E. Orr. 1996. *Talking about machines: an ethnography of a modern job.* ILR Press, Ithaca, N.Y.

[13] Julian E. Orr. 2006. Ten Years of Talking About Machines. *Organization Studies* 27, 12 (2006), 1805–1820. https://doi.org/10.1177/0170840606071933

[14] Joseph Thibodeau. 2025. *The Real Friends Are The Machines We Made Along The Way.* phd. Concordia University. https://spectrum.library.concordia.ca/id/eprint/995285/

[15] Joseph Thibodeau and Marcelo M. Wanderley. 2013. Trumpet augmentation and technological symbiosis. *Computer Music Journal* 37, 3 (2013), 12–25. http://www.mitpressjournals.org/doi/abs/10.1162/COMJ_a_00185

[16] Joseph Thibodeau, Ceyda Yolgormez, and Patil Tchilinguirian. 2023. ROBO BREAK. https://www.youtube.com/playlist?list=PLjMcjoZxzlegg9KHjry_UvxCrcQqcBJiQ

[17] Joseph Thibodeau and Ceyda Yolgörmez. 2020. Open-source Sentience: the Proof is in the Performance. In *Proceedings of the 2021 International Symposium of Electronic Art.* Montreal, Canada, 393–399.

[18] Joseph L. N. Thibodeau. 2011. *Trumpet Augmentation: Rebirth and Symbiosis of an Acoustic Instrument.* Ph. D. Dissertation. McGill University.

[19] Zeph Thibodeau. 2024. La réserve naturelle des robots inutiles. https://quaidessavoirs.toulouse-metropole.fr/2024/09/16/la-reserve-naturelle-des-robots-inutiles/

[20] Zeph Thibodeau. 2024. Social Machines and Human Values. *Gegenüber Digital Magazine* Issue 2: Synthetic Truth (2024). https://www.goethe.de/prj/geg/en/thm/tru/25348067.html

[21] Simon Waters. 2021. The entanglements which make instruments musical: Rediscovering sociality. *Journal of New Music Research* 50, 2 (March 2021), 133–146. https://doi.org/10.1080/09298215.2021.1899247

[22] Ceyda Yolgormez and Joseph Thibodeau. 2021. Socially robotic: making useless machines. *AI & SOCIETY* (Aug. 2021). https://doi.org/10.1007/s00146-021-01213-0