# In Defense of Linux, USB, and MIDI Systems for Sensorimotor Experiments: A Response to Schultz and van Vugt (2015)

Steven A. Finney

Manuscript submitted to *Behavior Research Methods* on Feb 2, 2016; rejected on Apr 6, 2016. You are welcome to circulate this manuscript or include it on a web page.

Abstract

Schultz and van Vugt (2015) compare their Tap Arduino system for finger tapping experiments against a more complex system made up of FTAP software (Finney, 2001a), the Linux operating system, a USB-MIDI interface, and a MIDI percussion pad. They report that the FTAP system showed high latency between tap and sound, with high variability, and suggest that this is due to latency introduced by USB-MIDI and the multi-component complexity of the FTAP system. However, their own data strongly implicate the specific MIDI drum pad they used for input in their FTAP configuration as the primary source of the latency attributed to FTAP. The FTAP software itself, in conjunction with a USB-MIDI interface, runs with demonstrable millisecond accuracy and precision as defined by Plant and Turner (2009). Experimenters who need reliable millisecond-resolution data collection in sophisticated sensorimotor or music experiments can confidently use properly configured Linux MIDI systems such as FTAP with appropriately chosen input and output devices.

*Keywords:* Auditory feedback, millisecond resolution, USB, Musical Instrument Digital Interface (MIDI), Linux

Schultz and van Vugt (2015, henceforth SVV for brevity) describe Tap Arduino, a low-cost system for running simple tapping experiments with auditory feedback. They show that it has accurate input timestamping and low latency between finger tap and associated feedback. In demonstrating the high performance of Tap Arduino, SVV compare it against two different computer MIDI software systems: FTAP (Finney, 2001a) running on Linux and MAX (Winkler, 1988) running on a MacBook Pro. SVV show that their FTAP Linux system using a USB-MIDI interface connected to a Roland Handsonic HPD15 percussion pad had significant higher latency than Tap Arduino, reporting that FTAP's "auditory feedback latencies" had a high mean and variability (M = 14.6 ms, SD = 2.8). They express

The author can be contacted via email at sf@sfinney.com. The current FTAP web page can be accessed at "www.sfinney.com/ftap". Thanks to Pete Pfordresher and Caroline Palmer for assistance and manuscript review, and to Nancy Leeper for editorial assistance.

concern that such “delays observed in FTAP” could compromise sensorimotor experiments, and suggest that USB-MIDI polling, USB-MIDI processing conversion, computer processing of MIDI, and the complex connectivity of the FTAP-MIDI system contribute to unreliable performance.

In this paper, I will demonstrate that both FTAP and MIDI run on a contemporary Linux distribution with full millisecond resolution, as was demonstrated 15 years ago in Finney (2001a, 2001b), and that USB-MIDI does not add significant delay. As SVV’s own data show, the source of the poor latencies is almost certainly the Roland drum pad itself (a component which is not included in their Tap Arduino configuration). Concerns about the performance of FTAP, Linux, or USB-MIDI are therefore unfounded, although experimenters must choose appropriate MIDI input and output devices for their experiments.[1]

It is worth being precise about the software and hardware of the Tap Arduino and FTAP systems being compared. Functionally, both systems can collect timing data from a finger tap, and purport to provide virtually simultaneous auditory feedback to that finger tap. Both systems provide some capability for manipulation of the feedback signal (e.g., delay). FTAP can also produce an isochronous or patterned pacing (synchronization) signal that can be correlated with the tapping data, and can change or perturb both the pacing and feedback signals during the course of an experiment; with these capabilities, FTAP can easily run sensorimotor experiments such as those in Aschersleben and Prinz (1997), Repp (2000), Wing and Kristofferson (1973), and Wing (1977).

For hardware, Tap Arduino uses an Arduino UNO board, which has a 16 MHz microprocessor. A force sensitive resistor (FSR) is connected to the Arduino to register finger taps, and the Arduino can either generate sounds internally or send a signal to an external “Wave Shield” module that can play arbitrary sound files. Generation of a pacing signal (metronome) for synchonization experiments can be done by an external PC playing a .wav file, but SVV acknowledge difficulties in “synchronizing responses and stimuli.” Such an Arduino hardware system is simple and fairly deterministic.

The FTAP system configuration is much more complex, comprising the FTAP application itself, a PC (in 2015, typically with a 6 GHz Intel or AMD processor chip with 2 or 4 processors), some version of the multi-tasking Linux operating system (including drivers for USB and MIDI), and a hardware MIDI interface (a Soundblaster in Finney (2001a, 2001b), or a USB-MIDI [2] Maudio adapter in SVV). I will refer to these components as an “FTAP-MIDI” system. Such a system must be connected to a MIDI input device (a keyboard or drum pad) to collect a subject’s performance data, and to a MIDI output device (a tone generator or sampler) to generate the feedback and pacing sounds; I will refer to such a complete system as an “FTAP data collection” system. It has three independent components, each of which makes some additive contribution to the end-to-end latency of the system.

Finally, a few details about the MIDI protocol and MIDI devices will be useful (see

[1]Although this paper discusses FTAP, which I am familiar with, much of what I write should apply to MAX as well.

[2]A USB-MIDI interface is an external hardware component that connects to a PC’s USB port and (in the simplest case) provides a single MIDI IN connector and a single MIDI OUT connector that can be attached to standard MIDI devices. USB-MIDI must be supported by drivers in the computer OS that implement both generic USB device processing and MIDI-specific processing (ALSA in contemporary Linux); there may also need to be a driver or firmware to support a particular manufacturer’s USB-MIDI hardware interface.

also Finney, 2001a, 2001b). A MIDI keyboard (a commonly used device in both tapping and music experiments) sends a MIDI NOTE ON message when a key is pressed, and a NOTE OFF message when the key is released. When a MIDI output device (a tone generator) receives a NOTE ON message, it starts to play the current selected sound (often called a "voice" or a "patch") on the device; the sound is terminated by a NOTE OFF message. MIDI input and output events have the exact same NOTE ON/OFF message format. The MIDI hardware implementation is such that each event takes about 1 ms to transmit, giving a maximal MIDI wire speed of about 1,000 events/second. Such millisecond resolution is typically considered both necessary and sufficient for sensorimotor experiments.

SVV measure both FTAP and Tap Arduino in an end-to-end configuration: that is, they measure the latency between a subject's physical tap and the resulting "immediate" auditory feedback (the importance of such end-to-end measurements has been repeatedly emphasized by Plant and colleagues, e.g., Plant, Hammond, and Turner, 2004). SVV report that their FTAP system shows mean end-to-end latency of 14 ms, and attribute that to USB-MIDI delays and connection complexity. However, neither of these in fact contributes significantly to the latency.

I will first address SVV's concerns with USB-MIDI, then describe a benchmark that demonstrates that a contemporary FTAP-MIDI system performs with the necessary millisecond resolution, and then provide evidence that the drum pad is the source of SVV's measured FTAP latencies. I will then discuss MIDI input devices in more detail, and provide further arguments in favor of the use of Linux and MIDI in reliable millisecond-resolution experiment systems.

## USB-MIDI and FTAP latency

SVV are concerned that that USB is slow and that USB-MIDI is likely to be unreliable for millisecond resolution data collection. They claim that USB does polling at 8-ms intervals, and that "delays can be introduced by the polling speed"; as a result, there may be issues with "temporal noise" introduced by "USB-MIDI conversions."

However, these claims are based on a misunderstanding of how USB-MIDI is implemented. There is no 8 ms polling with USB-MIDI; the concept of USB polling applies to devices with "Interrupt" type endpoints. Such devices are often USB HID (Human Interface Device) devices such as mice and keyboards, which typically do 8 or 10 ms polling (USBIF, 2001; see also Plant, Hammond, and Whitehouse, 2003). However, USB-MIDI interfaces are *not* HID devices; the USB-MIDI specification defines them as "Bulk" type endpoints, which do not use the interrupt polling mechanism (USBIF, 1999). Thus, such MIDI interfaces are not subject to the delays SVV are concerned about and there is no a priori reason to be concerned about the performance of USB-MIDI.[3]

This can be confirmed in practice by running the FTAP performance test ("looptest") described in Finney (2001a, 2001b); running the test on the experimenter's own data col-

[3]The characteristics of the USB hardware devices (e.g., HID device, Interrupt vs Bulk endpoints) on a user's Linux system can be determined using the "lsusb" command, although the output requires some background to understand. Briefly, run "lsusb -v" as root. Compare the endpoints associated with a USB keyboard or USB mouse with the endpoint for a USB-MIDI device, and look at "Transfer Type" and "bInterval" (polling interval). The latter is typically 0 for a USB-MIDI interface and 8 (ms) for a USB keyboard.

lection system is a prerequisite to any serious research. In this test, the MIDI output from FTAP is physically looped back to FTAP's MIDI input using an appropriate MIDI cable or connector. By doing this, the generated feedback events immediately appear as input keystroke events. For the loop test, FTAP is configured to provide immediate output feedback to input events, and the test is started by sending a single NOTE ON and NOTE OFF event to the MIDI output. FTAP receives these as input keystroke events, and sends out corresponding NOTE ON and NOTE OFF events as feedback. This continues indefinitely until the test stops (in the version provided with the 2.1.05 FTAP distribution, the loop test terminates after 2,000 NOTE ON events). This test exercises the MIDI hardware (both input and output), the core Linux OS, all relevant Linux drivers, all Linux MIDI processing, as well as the FTAP program itself. It does not test any input or output devices, but it is a thorough test of the FTAP-MIDI component. An output text file that records all events (keystroke and feedback) with milliscond precision is available for inspection.

Finney (2001a, 2001b) demonstrated that with a 200 MHz Pentium PC using a Soundblaster 16 MIDI interface card, FTAP processed messages at maximal MIDI speed (that is, the feedback "output" was immediately detected as keystroke input and sent back out as feedback again, all within 1 ms). An input MIDI event was received, on average, every .96 ms (maximum wire speed). In such a successful performance of this test, FTAP is receiving approximately 1,000 MIDI IN ("keystroke") events per second, recording each input event with an accurate and precise millisecond timestamp, while at the same time emitting 1,000 MIDI OUT ("feedback") events per second, synchronized precisely with the corresponding input events. No events are lost or delayed.

I recently ran the loop test on a contemporary Ubuntu Linux system (Ubuntu 14.04.3 with Linux kernel 3.19, on an HP Pavilion computer with a 64-bit 6 GHz AMD Athlon processor), using the FTAP binary in the 2.1.05 FTAP distribution. This adds a USB-MIDI interface to the configuration tested by Finney (2001a, 2001b), as well as exercising USB processing and the ALSA sound software. Two different USB-MIDI interfaces (an Edirol UM-1EX and an M-Audio Midisport UNO) were tested, and a Hosa GMD-108 MIDI Coupler was used to connect the two male MIDI ports on the interface, obviating the need for a custom MIDI cable. A standard, non real-time kernel was used. As with Finney (2001a, 2001b), maximal 1,000 events/second MIDI throughput was achieved with both USB-MIDI interfaces. This test result demonstrates that no significant delay is introduced by the use of MIDI, USB-MIDI, Linux drivers, or FTAP itself.

In fact, SVV report the results of the FTAP loop test on their own system, and confirm that MIDI messages were, on average, sent and received in just over a millisecond. (M = 1.01 ms, SD = 1.03 ms). This establishes that their own FTAP system was properly configured, that they were correctly achieving maximal MIDI throughput (1,000 events/second simultaneously on input and output), and that their USB-MIDI interface did not add any significant latency.

Since FTAP, Linux, and USB-MIDI are not responsible for the large "FTAP" latencies reported by SVV, the MIDI input device (the Roland drum pad) and/or the MIDI output device (a Yamaha TX81Z, using a "noise shot" voice that had a sharp attack envelope) must be the source of the problematic mean 14 ms latencies in their FTAP configuration. Since output devices are purely electronic (that is, they do not have any mechanical sensors), for purposes of discussion I will assume that the TX81Z has a small, constant latency (say, 1-2 ms, although this needs to be verified experimentally), and focus on the input device. SVV note that "common problems in studies using MIDI percussion pads are missing or superfluous responses", and they observe that the HPD15 percussion pad in their own experiment showed both of those problems.

Various aspects of SVV's data indicate problems with the Roland HPD15 drum pad. They report a 9-ms mean latency between hitting the drum pad and audio from the drum pad's own audio output (their "Percussion Pad" condition), indicating that the Roland drum pad itself has a mean 9-ms internal latency even without any external connections or software involved. They also note that the asynchronies in the FTAP percussion pad condition decreased with harder taps, suggesting either that harder taps caused a faster MIDI response, or that harder taps caused fewer missed responses. In fact, SVV at a later point in their paper state that "it appears that the percussion pad is accountable for the majority of the latency" (i.e., not FTAP, not Linux, and not USB-MIDI).

## Discussion

SVV's own data and conclusions thus confirm that the FTAP-MIDI system itself has millisecond resolution, and that the particular drum pad they tested was the source of the measured latency attributed to FTAP. As a result, their data do not provide any evidence that Tap Arduino is more precise or accurate than FTAP when it is connected to a reliable MIDI input device. However, since real-world FTAP data collection requires choosing some specific MIDI input and output device, SVV's paper is useful in focusing attention on end-to-end latency. I will first address the following two MIDI-specific questions (relevant to both FTAP and MAX), and then discuss some more general issues.

1. What sort of issues arise with MIDI input devices, and how have researchers dealt with them?

2. What is known about FTAP's typical end-to-end latency using devices other than the HPD15?

### MIDI Input Devices

SVV conclude that the Roland HPD15 percussion pad is the primary problem in their FTAP system, but it would be useful to understand in more detail exactly how the drum pad contributes to their measured FTAP system latency. There are two obvious concerns with MIDI input devices: what is the latency between the physical tap on the device and the generation of the MIDI NOTE ON message, and does the device miss taps or generate superfluous taps.

SVV's figure 3 suggests that there is a fairly consistent 9 ms +/- latency from the audio generation on the drum pad itself, as well as a fairly consistent 14 ms +/- latency with FTAP plus the drum pad plus the tone generator (perhaps the HPD15 takes a few milliseconds to generate the MIDI signal after the audio is generated). However, it is not entirely clear how the missing taps were handled in their asynchrony analysis. That is, if a tap produced an FSR or piezo signal, but did not generate a MIDI signal (a plausible possibility with their HPD15), how did this contribute to the asynchrony data? How exactly did the FTAP asynchronies differ in the soft- and hard-force conditions?

In practice, the missed and superfluous responses with some MIDI drum pads are perceptually obvious and readily apparent to any researcher who has spent time testing their system. There are various ways of coping with the problem. Pfordresher and Dalla Bella (2011, experiment 1) used a Roland SPD-6 drum pad and noticed occasional missed taps and doubled taps. They filtered outliers from the data in such a way as to remove the flawed data points, and switched to using another device (a Fatar keyboard) for their experiment 2. Repp, London, and Keller (2005) also used a Roland SPD-6 drum pad, and they removed one subjects's data from the analysis because of the number of unregistered (missing) taps due to less force at a fast tempo.

It is also possible that not all drum pads suffer from this problem. Madison (2001) and Madison and Merker (2004) collected MIDI tapping data with two different non-Roland drum pads and did not report issues with missing or duplicated taps. Fischinger (2011) did not report any issues with the Roland PD-8 trigger pad.

Occasionally a researcher has measured and reported the latency of their MIDI input device. Mills, van der Steen, Schultz, and Keller (2015) measured the mean latency between a tap on a Roland Handsonic 10 percussion pad and registration by MAX to be 5 ms. This puts an upper limit on the time between a physical tap and the transmission of a MIDI signal on that device, since the 5 ms also includes MacOS and MAX processing. They assume that this latency is small and constant, although they also observed unregistered taps, which they handled by data filtering.

Users of MIDI keyboards (whether psychological researchers or performing musicians) have not reported the missing and superfluous responses found with some percussion pads. For example, Finney and Warren (2002) used a Fatar-49 musical keyboard for a synchronization tapping study with no obvious issues. Keyboards may be fundamentally more reliable than drum pads because of how the movement is registered: MIDI NOTE ON messages on a keyboard are transmitted while the key is actively moving downward, so issues of debouncing or minimal required pressure to register a tap are of less concern than with a percussion pad. [4]

## FTAP's End-to-end latency

Measuring end-to-end latency in a system such as FTAP requires non-trivial measurement hardware (e.g., the Black Box system of Plant et al. (2004), or the analog input box of SVV), and FTAP users have typically not reported such data. One exception is Maidhof, Kastner, and Makkonen (2014), who did a preliminary latency test of FTAP's performance by comparing the audio output from a Yamaha Clavinova CLP-330 with the audio data generated from FTAP with a MIDI tone generator. They reported a mean latency of 1.4 ms, with a maximum of 3 ms, suggesting that both the input and output devices have low latency. However, this approach did not explicitly measure the asynchrony from the physical tap, and so is not an equivalent end-to-end measurement. Since the FTAP-MIDI system has been shown to add minimal latency (once the loop test has been verified), end-to-end

[4]This can be observed by using the Linux ALSA "amidi" utility, which reads the data transmitted from a MIDI device and prints it to the console. With both a Fatar ALS-49 and a Yamaha DX7-II keyboard, slowly pressing a key shows that MIDI NOTE ON transmission occurs about 1/8 in. above the key stop at the bottom. There is certainly some latency (e.g., the movement from physical top of key to triggering), but it is likely to be small and constant.

measurement of FTAP with a validated low-latency MIDI tone generator would be basically equivalent to measuring the latency of the MIDI input device. A rigorous tap-to-sound measurement of FTAP with multiple different MIDI keyboards or drumpad models would be a valuable contribution.

## Linux and MIDI

There are some useful general lessons about PC MIDI data collection systems such as FTAP and MAX. First, complexity doesn't necessarily contribute to high latency, contrary to the concerns of SVV. Second, although Plant and Turner (2009) and Plant (2015) have raised the concern of potentially unpredictable behavior with multi-tasking operating systems, Linux can be fully adequate for reliable data collection. The same Linux programming techniques for millisecond-resolution programming described and validated in Finney (2001b) still work with modern Linux systems, although FTAP's programming is greatly simplified because it does not have to deal with the complexities of internal sound cards or graphics hardware and software (see Plant et al., 2004). The FTAP-MIDI system has been demonstrated to perform with millisecond resolution on a wide range of hardware and software; the included loop test helps address the replication concerns of Plant (2015). Linux also addresses many of the operating system concerns raised by Plant and Quinlan (2013): lack of visibility into the OS, high levels of OS abstraction, complex and opaque API's, and frequent and massive API change. Linux has multiple advantages here compared with other contemporary OSes: API stability, internal visibility, and readily available low-level tools.

Linux's system call API is fairly stable, even in the face of constant development. The 2.2 Linux kernel tested in Finney (2001a, 2001b) had 2 million lines of code, while the 3.19 kernel used in the test described above has over 15 million lines of code (Wikipedia, 2016). The MIDI hardware and software used in Finney (2001a, 2001b) is completely different from that in current Linux systems. Nonetheless, a *15-year-old* 2.1.05 FTAP application binary executable still runs on a contemporary Linux system, and still shows demonstrable millisecond accuracy and precision. It is almost certain that no such demonstration could be made with a Windows or Apple system.

In addition, the fact that the source code to Linux is publicly available allows a knowledgeable user to investigate, understand, or even modify, compile, and run a modified version of the operating system. For example, Finney (2001b) diagnosed the source of a performance issue with the Linux MIDI drivers by looking at the source code, and switched to an alternate set of drivers that did not have the problem. SVV and Maidhof et al. (2014) installed Linux real-time patches, although FTAP has adequate performance without them. Finally, Linux includes default tools (such as the "lsusb" and "amidi" commands used above) that provide the ability to diagnose low-level issues.

FTAP's performance, portability, and testability is aided by its use of MIDI, a standard low bandwidth data transfer protocol which is nonetheless sufficient for an interesting range of human sensorimotor experiments. MIDI avoids the timing issues with USB HID devices such as mice and keyboards. MIDI supports a wide range of commodity input devices (e.g., both weighted and unweighted musical keyboards, as well as percussion pads and wind controllers). Standard MIDI architecture leads to the use of an offboard tone generator, avoiding issues with internal sound cards; the use of a MIDI-triggered sampler for output would allow presentation (within limits) of arbitrary sound files. The information available from a 3-byte MIDI messages captures the basic data required in music and tapping experiments: what key was pressed, when it was pressed, the velocity (force) of the key press, and when the key was released. A 25 or 49 key MIDI keyboard controller with velocity sensitivity can be purchased for under $100, and thus classifies as a reasonably priced input device for tapping experiments, although the experimenter should confirm that their chosen device is suitable for data collection.

## Arduino Hardware

Tap Arduino, as described by SVV, is not a complete system for sensorimotor experiments (e.g., it cannot generate a synchronization signal). However, a more complete MIDI-oriented Arduino system would be valuable as an input/output device for tapping experiments in MAX or FTAP, removing the need for experimenters to measure the performance of commodity MIDI input and output devices. A small Arduino device that could generate verifiably fast MIDI data in response to taps on a properly debounced input button would be a useful, low-cost, MIDI input device for tapping experiments (i.e., a MIDI equivalent of a Psyscope or E-Prime button box). Ideally, it would register and transmit a calibrated force measurement (encoded as MIDI velocity) as well, and could handle more than one input button (coded as different MIDI note values). If it could also produce a useful range of calibrated sounds in response to MIDI input over a second MIDI connector (perhaps using a Wave Shield module), the result would be a self-contained MIDI input/output device. In conjunction with a Linux laptop running FTAP attached to a USB-MIDI interface, such an Arduino device could be part of a small, portable system for sensorimotor experiments that could be easily carried anywhere. Arduino device design could be based on the existing work of Baath (2011), SVV, or Schubert, D'Ausilio, and Canto (2013).

## General Conclusion

SVV properly emphasize the importance of understanding the end-to-end characteristics of a data collection system including the input and output devices, and demonstrate the latencies that can occur when a poorly performing input device is chosen. However, their results provide no basis for concern about the multi-tasking Linux OS, the FTAP software package, or USB-MIDI for accurate and replicable data collection in sensorimotor experiments.

## References

Aschersleben, G. & Prinz, W. (1997). Delayed auditory feedback in synchronization. *Journal of Motor Behavior*, *29*, 35–46.

Baath, R. (2011). Construction of a low latency tapping board. *LUCS Minor*, *17*. Retrieved from www.sumsar.net.

Finney, S. (2001a). FTAP: A Linux-based program for tapping and music experiments. *Behavior Research Methods, Instruments, and Computers*, *33*, 65–72.

Finney, S. (2001b). Real-time data collection in Linux: A case study. *Behavior Research Methods, Instruments, and Computers*, *33*, 167–173.

Finney, S. & Warren, W. (2002). Delayed auditory feedback and rhythmic tapping: Evidence for a critical interval shift. *Perception and Psychophysics*, *64*, 896–908.

Fischinger, T. (2011). An integrative dual-route model of rhythm perception. *Musicae Scientae*, *11*, 97–105.

Madison, G. (2001). Variability in isochronous tapping: Higher order dependencies as a function of intertap interval. *Journal of Experiment Psychology: Human Perception and Performance*, *27*, 411–422.

Madison, G. & Merker, B. (2004). Human sensorimotor tracking of continuous subliminal deviations from isochrony. *Neuroscience Letters*, *370*, 69–73.

Maidhof, C., Kastner, T., & Makkonen, T. (2014). Combining EEG, MIDI, and motion capture techniques for investigating musical performance. *Behavior Research Methods*, *46*, 185–195.

Mills, P., van der Steen, M., Schultz, B., & Keller, P. (2015). Individual differences in temporal anticipation and adaptation during sensorimotor synchronization. *Timing and Time Perception*, *3*, 13–31.

Pfordresher, P. & Dalla Bella, S. (2011). Delayed auditory feedback and movement. *Journal of Experimental Psychology: Human Perception and Performance*, *37*, 566–579.

Plant, R. (2015). A reminder on millisecond timing accuracy and and potential replication failure in computer-based psychology experiments: an open letter. *Behavior Research Methods*.

Plant, R., Hammond, N., & Turner, G. (2004). Self-validating presentation and response timing in cognitive paradigms: How and why? *Behavior Research Methods, Instruments, and Computers*, *36*, 291–303.

Plant, R., Hammond, N., & Whitehouse, T. (2003). How choice of mouse may affect response timing in psychological studies. *Behavior Research Methods, Instruments, and Computers*, *35*, 276–284.

Plant, R. & Quinlan, P. (2013). Could millisecond timing errors in commonly used equipment be a cause of replication errors in some neuroscience studies? *Cognitive, Affective, and Behavioral Neuroscience*, *13*, 598–614.

Plant, R. & Turner, G. (2009). Millisecond precision psychological research in a world of commodity computers: New hardware, new problems? *Behavior Research Methods*, *41*, 598–614.

Repp, B. (2000). Compensation for subliminal timing perturbations in perceptual-motor synchronization. *Psychological Research*, *63*, 106–128.

Repp, B., London, J., & Keller, P. (2005). Production and synchronization of uneven rhythms at fast tempi. *Music Perception*, *23*, 61–78.

Schubert, T., D'Ausilio, A., & Canto, R. (2013). Using Arduino microcontroller boards to measure response latencies. *Behavior Research Methods*, *45*, 1332–1346.

Schultz, B. & van Vugt, F. (2015). Tap Arduino: An Arduino microcontroller for low-latency auditory feedback in sensorimotor synchronization experiments. *Behavior Research Methods*.

USBIF. (1999). USB Implementers' Forum: Universal Serial Bus device class definition for MIDI devices, v 1.0. Retrieved from www.usb.org/developers/docs/devclass_docs.

USBIF. (2001). USB Implementers' Forum: Universal Serial Bus device class definition for Human Interface Devices, v 1.11. Retrieved from www.usb.org/developers/docs/devclass_docs.

Wikipedia. (2016). Linux kernel. Retrieved from en.wikipedia.org/wiki/Linux_kernel.

Wing, A. (1977). Perturbations of auditory feedback delay and the timing of movement. *Journal of Experimental Psychology: Human Perception and Performance*, *3*, 175–186.

Wing, A. & Kristofferson, A. (1973). The timing of interresponse intervals. *Perception and Psychophysics*, *13*, 455–460.

Winkler, T. (1988). *Composing interactive music: Techniques and ideas using MAX*. Cambridge, MA: MIT Press.