The TeensyTap Framework for Sensorimotor Synchronization Experiments

Floris Tijmen van Vugt[1, 2, 3]

[1] McGill Psychology Department, Montreal, Canada

[2] Haskins Laboratories, New Haven, USA

[3] Département de Psychologie, Université de Montréal, Canada

ABSTRACT

Synchronizing movements with an external periodic stimulus, such as tapping your foot along with a metronome, is a remarkable human skill called sensorimotor synchronization. A growing body of literature investigates this process, but experiments require collecting responses with high temporal reliability, which often requires specialized hardware. The current article presents and validates TeensyTap, an inexpensive, highly functional framework with excellent timing performance. The framework uses widely available, low-cost hardware and consists of custom-written open-source software and communication protocols. TeensyTap allows running complete experiments through a graphical user interface and can simultaneously present a pacing signal (metronome), measure movements using a force-sensitive resistor, and deliver auditory feedback, with optional experimenter-specified artificial feedback delays. Movement data is communicated to a computer and saved for offline analysis in a format that allows it to be easily imported into spreadsheet programs. The present work also reports a validation experiment showing that timing performance of TeensyTap is highly accurate, ranking it among the gold standard tools available in the field. Metronome pacing signals are presented with millisecond accuracy, feedback sounds are delivered on average 2 ms following the subjects' taps, and the timing log files produced by the device are unbiased and accurate to within a few milliseconds. The framework allows for a range of experimental questions to be addressed and, since it is open source and transparent, researchers with some technical expertise can easily adapt and extend it to accommodate a host of possible future experiments that have yet to be imagined.

KEYWORDS

sensorimotor synchronization

microcontroller

auditory feedback

metronome

music

INTRODUCTION

Dancing to music or tapping along with the beat of our favourite tune seems effortless, but constitutes a remarkably complex human skill. Aligning our movements with external periodic stimuli is a rare feat in the animal realm (Patel et al., 2009). This capacity is studied in sensorimotor-synchronization (SMS) experiments, where participants are asked to tap their finger along with a periodic signal, for example, a metronome (Repp, 2005; Repp & Su, 2013). There is an extensive body of literature investigating this capacity, but most setups use tools that are expensive and have questionable timing performance (Schultz, 2019). Previously, a platform was introduced based on the Arduino microcontroller that could deliver highly reliable auditory feedback (Schultz & van Vugt, 2016). A microcontroller is a small computer, often only several centimeters in size, that can be programmed to autonomously collect finger tapping data and communicate these to a PC or Mac computer. TapArduino is becoming increasingly popular and has inspired work that has successfully incorporated Arduino hardware into existing testing setups (Scheurich et al., 2020). However, TapArduino is limited in that it cannot deliver a metronome pacing signal and there is no ready-made user interface that allows one to run an entire experiment.

The current article introduces a framework built around the Teensy microcontroller, which is capable of presenting a metronome signal, recording finger taps using a force-sensitive resistor (FSR), and delivering auditory feedback signals. The framework includes a graphical user interface which allows specifying the metronome rate and other parameters on the fly, and allows capturing and saving the data in a

Corresponding author: Floris van Vugt, McGill Psychology Department, Ostry Lab, room 718, 2001 McGill College Avenue, Montreal, QC H3A 1G1.

E-mail: floris.vanvugt@mail.mcgill.ca

format that is easy to import into major software packages, thus making it possible to run entire experiments using TeensyTap and a PC/Mac. Taken together, this forms an inexpensive and user-friendly solution for a host of sensorimotor experiments while requiring only some technical knowledge and relying exclusively on open-source software. While TeensyTap can run a variety of experiments as-is, it is not designed to cover all possible experiments in the field without any adjustments. Rather, simplicity of design and transparency were favored to allow researchers with technical expertise to easily adapt TeensyTap to other experiments, even ones that have not yet been envisaged at this time.

The current article describes the design of the TeensyTap framework and presents the results of an experiment validating the timing accuracy of the device. External recordings of finger taps and sounds delivered by TeensyTap were made in order to establish when the onset of sounds and taps actually happened (ground-truth), so that it could be compared with what TeensyTap reported back to the experimenter. Specifically, we investigated (a) how accurate the metronome signal is in time, (b) how quickly the device produces a feedback sound when the subject taps on the force-sensitive resistor, and (c) how reliable the tap timings are that are logged by the device.

METHODS

TeensyTap Framework

TeensyTap is based on the Teensy 3.2 microcontroller (developed by PJRC, https://pjrc.com/teensy; for another example use of this controller see Romano et al. 2019) combined with the Audio Adapter Board and a force-sensitive resistor (see Figure 1, Panel A). The force-sensitive resistor is a small surface on which the subject taps their finger, causing a change in electrical resistance that is measured by Teensy. The Teensy device connects through a USB cable to a computer (PC or Mac) running a Python user interface that allows the experimenter to choose, among other things, the metronome rate and whether auditory feedback should be delivered for every trial (see Figure 1, Panel B). Subjects wear headphones that are connected to the Teensy audio board through which they hear the metronome as well as a feedback sound from their own taps (if selected). Subjects tap their finger on the FSR, a small pad that is connected to the Teensy, which is able to detect when (and with how much force) the subject taps. The Teensy relays this timing information, as well as the timing of metronome clicks, back to the experimenter's Python interface where it is saved for offline analysis. A demonstration video for the usage of this setup is available here: https://www.youtube.com/watch?v=WwA4infAf5g (this video starts from when the platform has already been built; for details on how it is built, see the Installation and Source Code section below).

The tapping data can be imported into a spreadsheet program. The format is space-separated text (popular spreadsheet programs will offer this option when the file is opened) and results in a table format shown in Table 1. Each row corresponds to an event, which is a click (metronome pacing signal), tap (detected finger tap by the subject) or feedback (presentation of feedback signal). For each event, an onset time is provided (in milliseconds from an arbitrary start time). For finger taps, further details are provided: offset time (the time the finger lifted from the FSR), the time at which the maximum pressure (max_force_t) was attained (in between tap onset and offset) as well as what that maximum force was (max_force, in arbitrary units that can be converted into Newtons).

Installation and Source Code

The TeensyTap source code, the blueprints for how to build it, and Python software for the user interface are made available freely and open-source at http://www.github.com/florisvanvugt/teensytap. Acquiring and building TeensyTap requires the steps listed in Table 2.

HARDWARE

The Teensy 3.2 and the Audio Adapter Board are available for purchase at low cost from electronics stores worldwide. In addition, a 4.2 kOhm resistor is required, as well as some basic electric wires, headphones, and a USB cable that connects the Teensy to the PC/Mac. These can usually be found in a laboratory workshop or purchased for minimal fees. The Teensy and audio board need to be soldered together, which requires basic soldering skills. Beyond this, no technical expertise is required, as the remaining steps involve software installation and are demonstrated step-by-step in the instructional videos (see Table 2).

SOFTWARE

A computer (PC or Mac) with Python installed is required. The software installation is straight-forward and requires less than 15 minutes, starting from scratch as shown in the following video: https://www.youtube.com/watch?v=wIUFWRm3EA0. It involves installing the Arduino IDE and Teensyduino, both available for download for free from the corresponding developers' websites. The experimenter can then download the TapArduino code from the above Github page, upload it to the Teensy device using the automated software provided, and run the Python graphical user interface (GUI) to run the experiment and collect the data. All these steps are shown in the video linked above.

The current interface allows the experimenter to specify the parameters of each trial: the metronome rate (inter-onset-interval between metronome clicks, i.e. IOI, in milliseconds), how many metronome clicks should be presented during a given trial (i.e., the duration of the trial), whether an auditory feedback sound should be presented at each tap (yes/no), whether it should come after a pre-specified delay (in ms), and after how many metronome clicks the feedback delay should come into effect; and whether there should be a continuation phase where the metronome stops but the subject continues tapping (Wing & Kristofferson, 1973). Once the experimenter presses the green "Go" button, all the parameters of the experiment are communicated through USB to the Teensy and it starts running the trial. A beep sound indicates to the subject when the trial ends. The TeensyTap device communicates the time of each metronome click, the onset and

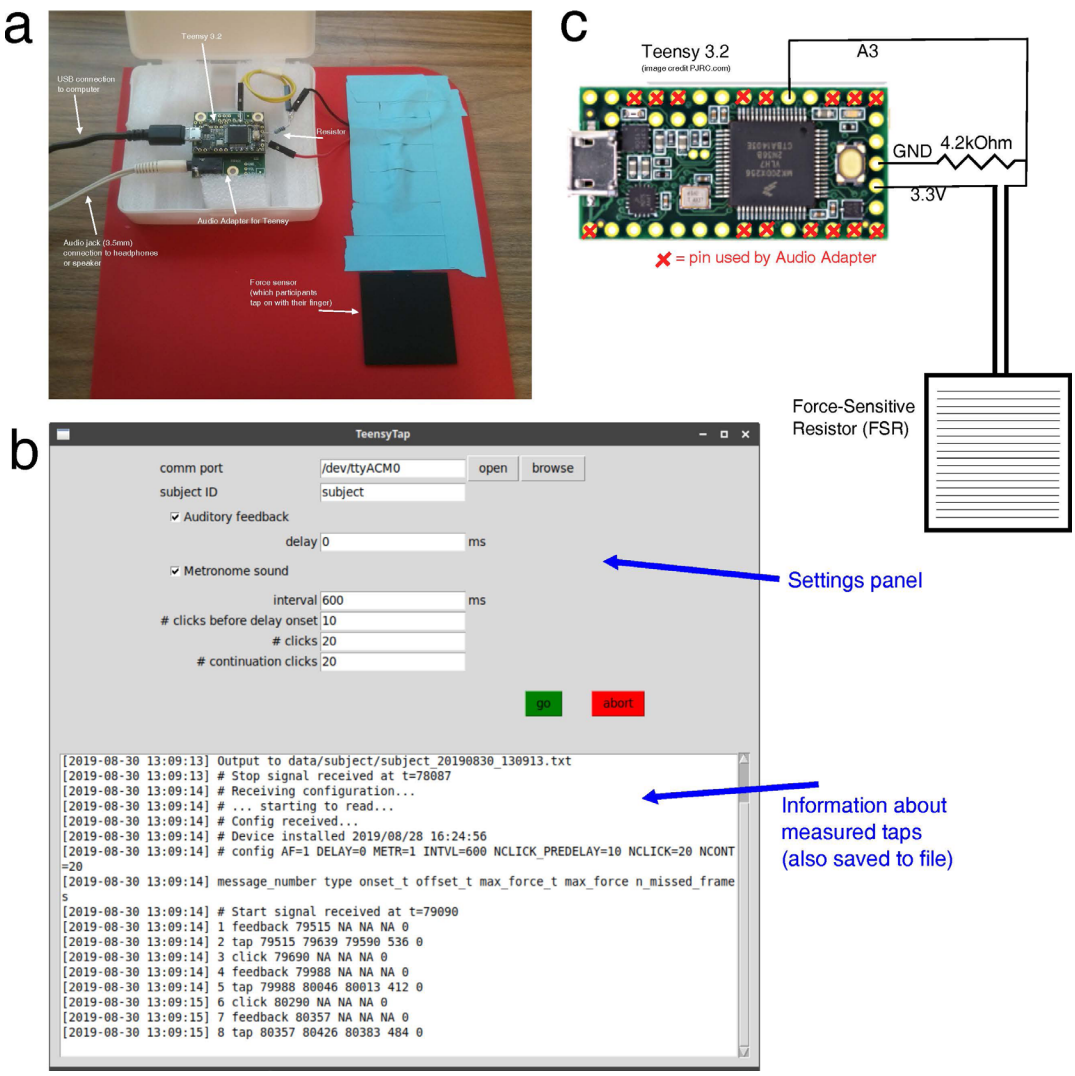

FIGURE 1.

The TeensyTap device. Panel A: Top view of TeensyTap, which consists of a Teensy microcontroller and the Audio Adapter Board, soldered together, connected with headphones and a PC/Mac through a USB cable. The subject taps their finger on a force sensitive resistor (FSR). Panel B: Graphical interface running on the PC/Mac that allows the experimenter to select the desired metronome rate, number of metronome clicks, as well as whether auditory feedback should be presented or not. The green "Go" button launches the trial. During the trial, Teensy reports detected taps as well as metronome clicks and delivered auditory feedback, which is also saved to a file for offline analysis, and can be readily imported in a spreadsheet program to yield a report, as shown in Table 1. Panel C: Schematic of the wiring of the Teensy Tap, which connects to a force-sensitive resistor (FSR) through a resistor.

offset of each finger tap, as well as the timing of the auditory feedback (since it may be experimentally delayed). This data is written in a table format to a space-separated text file by the Python interface and can be imported easily for offline analysis in Python, R, Matlab, or other software. Since Teensy uses its internal clock to determine when events happened (in technical terms, to assign timestamps), possible delays in USB communication or overhead caused by processes inadvertently happening on the host computer do not affect the reliability of the measurements.

Since TeensyTap is fully open-source, it can be easily modified to suit the needs of specific experiments. For example, the experimenter can change the sounds that are played (metronome click, tap feedback) by replacing them with a wave file (.wav) of their choice.

Validation Experiment Setup

A validation experiment was carried out to determine whether the sounds produced by TeensyTap and the data it reports about the onsets of the taps were reliable. In this validation experiment, a CGN521E microphone (AKG Harman, Los Angeles, USA) was placed a few centimeters from the force sensor to capture the sound made by the subject tapping their finger. The microphone signal was captured by a Fireface 800 interface (RME, Haimhausen, Germany) which also simultaneously measured the Teensy stereo audio output. The three tracks (two tracks from Teensy and one mono input from the microphone) were captured on a separate Windows PC running Audition 3.0.1 (Adobe Inc., San Jose, USA) and saved as wave files for offline

TABLE 1.
Sample Data as Delivered By Teensytap. Teensytap Provides Data in Text Format that Can Be Imported Directly into a Spreadsheet Program to Yield a Table as Shown Here

Message number	type	onset_t	offset_t	max_ force_t	max_force
1	click	199561	NA	NA	NA
2	feedback	199754	NA	NA	NA
3	tap	199754	199817	199787	438
4	click	200161	NA	NA	NA
5	feedback	200256	NA	NA	NA
6	tap	200256	200339	200302	470
7	feedback	200735	NA	NA	NA
8	click	200761	NA	NA	NA
9	tap	200735	200848	200773	585
10	feedback	201289	NA	NA	NA
11	click	201361	NA	NA	NA
12	tap	201289	201390	201291	563
13	feedback	201891	NA	NA	NA
14	click	201961	NA	NA	NA
15	tap	201891	202008	201950	580
16	feedback	202493	NA	NA	NA

TABLE 2.
Steps Required To Build And Set Up Teensytap

Step 1: Acquire hardware	Teensy microcontroller, audio adapter, force sensor, resistor and wires https://github.com/florisvanvugt/teensytap#hardware
Step 2: Build hardware (~1 hour)	Solder audio adapter on to Teensy microcontroller, solder wires between force sensor and resistors Solder audio adapter on to Teensy microcontroller, solder wires between force sensor and resistors
Step 3: Install software (15 min)	Install Arduino IDE and Teensyduino (prerequisites), and the TeensyTap source code https://www.youtube.com/watch?v=wIUFWRm3EA0
Step 3a: Enable millisecond resolution (5 min)	If millisecond resolution is required for your application, make a few lines change to the source code. https://www.youtube.com/watch?v=DVS2NKvLXm0
Step 4: Run experiment	Run the TeensyTap source code to collect data https://www.youtube.com/watch?v=WwA4infAf5g

analysis (see Figure 2, Panel A for an excerpt of what was recorded). A separate Linux PC was used to send instructions to TeensyTap through the Python graphical user interface and save the finger tap timing logs that were produced.

First, to validate the timing accuracy of the metronome signal, the Teensy was programmed to produce 1000 metronome clicks for each of the following metronome interval durations: 100, 200, 300, 400, 600, 800, and 1000 ms. During this time, no finger taps were produced. The metronome sound used was the 30ms duration woodblock wave sound included by default in TeensyTap (see Figure 2, Panel A for its waveform).

Second, to validate the timing accuracy of the auditory feedback for the finger taps, the Teensy was programmed to produce 1000 metronome clicks at an interval of 250 ms and the experimenter tapped their finger on the pad as closely as possible in time to the metronome, in the same way that a subject would be instructed to do during a typical sensorimotor synchronization experiment. This procedure was then repeated with the metronome set at 500 ms. The feedback sound used was a 30ms 440 Hz pure sine wave included by default in TeensyTap.

The Teensy code was modified for this experiment so that the feedback and metronome sounds were presented to separate stereo channels so that they could be more easily distinguished offline, instead of both sounds being presented to both ears as is done by default. After the experiment, the audio recordings were imported into Python. In the recording of the Teensy audio output, the sound onsets were established by detecting peaks in the cross-correlation time series between the recorded signal and the wave file of the tap feedback or metronome sound (see Figure 2, Panel A). The physical taps were then searched for in a window of 10 ms preceding each feedback sound (since the feedback sounds could be determined robustly) and were defined as the time when the audio signal exceeded 10 standard deviations away from the mean of the baseline signal just prior to the tap. The onsets identified in this way were verified visually using a display similar to that shown in Figure 2, Panel A.

The offline validation analysis then computed the timing between subsequent metronome onsets and compared it with the metronome interval that TeensyTap was programmed to produce. In the validation analysis, the delay between tap and feedback was computed as the time that elapsed between the onset of the physical tap and the onset of the subsequent auditory feedback sound. It was also investigated whether simultaneously occurring sounds would impair TeensyTap's timing accuracy. For example, if the device is busy playing one sound (such as the metronome sound), would other sounds (such as the tap feedback sound) occur with less reliable timing? Thus, we investigated whether tap feedback sounds that occurred close in time to the metronome sound would be delayed by correlating the absolute tap-to-metronome time with the tap-to-feedback time.

RESULTS

Timing accuracy of the produced metronome signal: The intervals between metronome clicks produced by TeensyTap were within 1ms of the to-be-produced interval across all metronome rates (see Table 3, Figure 2 Panel B). Furthermore, there was no bias, since the difference between the desired and actually produced metronome interval was not statistically significantly different from zero, $t(7034) = 1.30$, $p = .19$).

Timing accuracy of the tap feedback: On average, TeensyTap produced the auditory feedback o 1.79 ms ($SD = .53$) after the physical tap (see Figure 2, Panel C).

Accuracy of the tap timing reported by TeensyTap: The tap times detected from the microphone signal were compared against the timing reported by TeensyTap in the log file saved to the PC. There was a strong correlation between these two intervals ($r > .99$, $p < .00001$, see Figure 2, Panel D). Importantly, there was no bias in interval reported by TeensyTap: The difference between the logged and the true tap in-

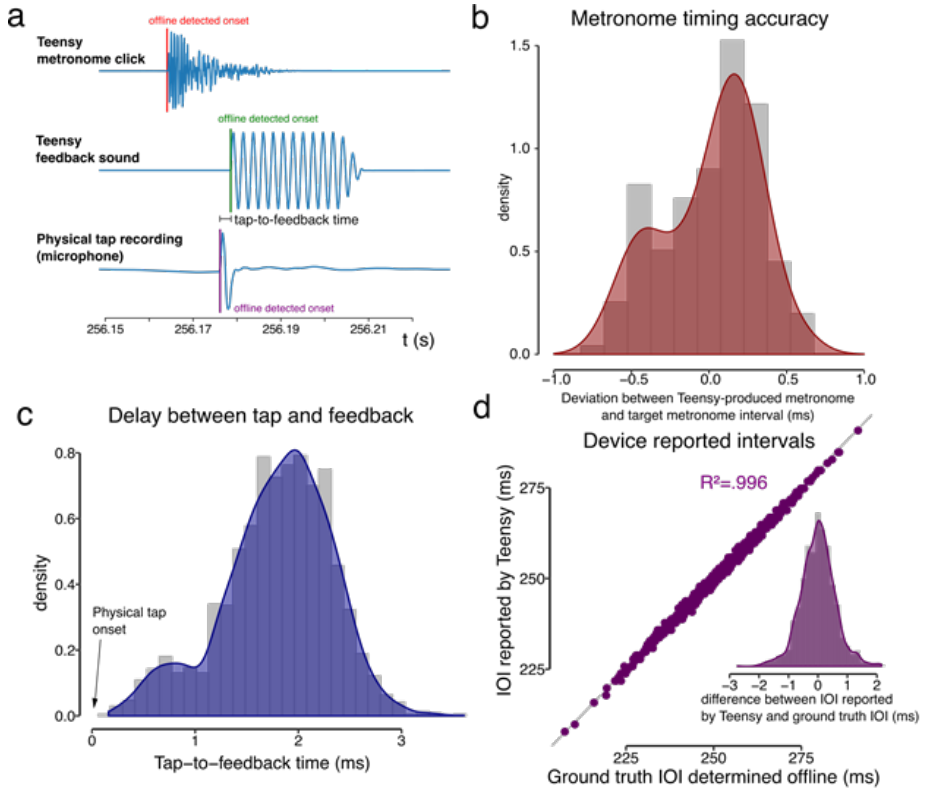

FIGURE 2.

Timing of metronome clicks and feedback sounds by Teensy is highly reliable. Panel A: Simultaneous recordings were acquired from the Teensy audio output (metronome clicks and feedback sounds shown in the top two traces) as well as from a microphone in the vicinity of the force-sensitive resistor (FSR) where the subject tapped their finger (bottom trace). This panel shows a sample of these recordings. An offline script detected the onset of the sounds and the finger tap, which are shown here as vertical lines, and these time points formed the basis for the subsequent analyses. Panel B: The metronome signal produced by the Teensy was accurate to the millisecond. Shown is the distribution of the difference between the desired and actually produced metronome interval. Panel C: TeensyTap delivered the sound within approximately 2 ms after the physical tap. Shown is the distribution of the time difference between the physical tap and the auditory feedback tone, and this difference never amounts to more than a few milliseconds. Panel D: TeensyTap produces an accurate log of the tap times. There is a strong correlation between tap inter-onset-interval (IOI) and the IOI reported by TeensyTap. The difference between the two (shown in inlay) is unbiased and within several milliseconds.

terval (determined from the external audio recording) was not statistically significantly different from zero, $t(2006) = -.34$, $p = .73$. The *SD*, reflecting the TeensyTap measurement error, was 0.60 ms.

It was also investigated whether having to play two sounds simultaneously, such as when the auditory feedback sound was played close in time to the metronome, would cause timing instability or additional delays. For this purpose, for each tap onset the tap-to-metronome time was computed as the absolute value of the timing difference with the closest metronome sound. There was no significant correlation between the tap-to-metronome time and the tap-to-feedback delay, $r = .002$, $p = .94$, indicating that the feedback sound timing was not affected by temporal proximity to the metronome click.

DISCUSSION

Sensorimotor synchronization (SMS) refers to humans' capacity to align their movements with external periodic stimuli. SMS is a remarkable capacity that is noticeably rare in the animal realm. In humans, SMS capacities are studied extensively and deficits in this capacity have been linked with disorders such as attention deficit hyperactivity disorder (ADHD, Noreika et al., 2013) and dyslexia (Colling et al., 2017). Synchronization also has intriguing links with interpersonal relations (Wiltermuth & Heath, 2009) and was shown to bring benefit in neurological conditions such as Parkinson's disease (Dalla Bella, Benoit, et al., 2017). Running sensorimotor synchronization experiments can be technically challenging because it involves generating a precisely timed periodic stimulus (metronome), accurately recording the timing of movements (e.g., finger taps), and possibly delivering auditory feedback, all in synchrony and with minimal delays. The current article introduced a simple and highly functional framework that can do just that and is built exclusively using open-source and free software. We also reported validation data based on ground-truth external observation, and these indicate that the timing performance of this device is reliable, making it suited to be deployed in a variety of experiments in psychology and neuroscience.

TABLE 3.

Results from the Validation Experiment Indicate that Teensytap Produces an Accurately Timed Metronome Pacing Signal. The First Column Indicates the Metronome Interval that was Sent as a Command to Teensytap. The Subsequent Columns Report the Actually Produced Metronome Intervals as Calculated Based on the Audio Recordings (Ground Truth). All Values are Reported in Milliseconds and Show that Timing Errors are Less than 1 ms

IOI instructed to TeensyTap	Min. IOI	Max. IOI	Mean IOI	SD IOI
100	99.37	100.11	100.00	0.25
200	199.46	200.20	200.00	0.32
300	299.55	300.30	300.00	0.36
400	399.66	400.41	400.00	0.36
600	599.84	600.59	600.01	0.29
800	799.32	800.07	800.01	0.17
1000	999.5	1000.25	1000.01	0.34

Sensorimotor synchronization experiments have been performed with a variety of tools and experimental frameworks. Studies measured movements using horizontal-drum kymographs (Stevens, 1886), Morse keys (Klemmer, 1957; Wing & Kristofferson, 1973), percussion pads (Dalla Bella, Farrugia, et al., 2017), or the computer mouse (Steele & Penhune, 2010). Currently, there exist several publicly available hardware, software, or combined packages to facilitate running sensorimotor experiments. For example, MatTAP is a Matlab based toolbox for running SMS experiments (Elliott et al., 2009). The platform allows running a wide range of experiments and its particular strengths are a high degree of customizability as well as built-in analysis code. The platform does not deliver response feedback and requires the proprietary Matlab software (Mathworks, Natick, USA) and potentially costly external hardware (data acquisition card and sensors). Another popular platform is FTAP (Finney, 2001), which has stood the test of time among the most widely used methods for almost two decades. Among its many strengths is that it is highly customizable. Its internal timing can be very precise and is verifiable by the user using a loop test. FTAP requires external hardware for response collection (such as a MIDI percussion pad and a means to connect it with the computer, i.e., a PCI card or USB-MIDI converter, which can cause considerable detriments in timing performance of the overall setup, as observed in Schultz & van Vugt, 2016). FTAP as well as MatTAP cater to a wide variety of users for running sensorimotor experiments. The contribution of the present framework, TeensyTap, is that it is a complete hardware and software setup that does not require any external devices. The benefit of this is that timing performance can be established reliably, as was done in the present validation experiment. Software packages that build on additional hardware or software layers (e.g., MIDI) do not allow straight-forward validation of the final timing performance of all the layers combined, because this often depends on the choice of hardware equipment. Indeed, each layer is shown to add additional, varying timing latencies (Schultz, 2019) which can add up when combined (Schultz & van Vugt, 2016) and thus make it difficult for researchers to estimate the effective timing performance of their system as a whole. Further, many commonly used MIDI-based measurement devices are actually percussion pads, and therefore, are designed to capture relatively forceful stick impacts instead of more subtle finger taps. As a result, subjects are required to tap with greater force than they might naturally do, and a previous study showed these devices often miss softer taps that FSRs (such as used by TeensyTap) can reliably detect (Schultz & van Vugt, 2016). Recent work has elegantly introduced hybrid setups that combine an Arduino microcontroller and an FSR (in the style of TapArduino) with FTAP (Scheurich et al., 2020). The only other complete hardware and software system currently available is Tap-It (Kim et al., 2012), which is an iOS-based application excelling in portability and ease of use. The developers also validated the timing performance indicating tap-to-feedback latencies of about 15 ms. The present paper introduces TeensyTap as a feasible combined hardware and software solution for sensorimotor synchronization experiments with experimentally validated accurate timing performance.

TeensyTap complements existing software packages in that it favors customizability over configurability. The design of software architecture involves a trade-off between configurability and customizability (Dittrich et al., 2009). Configurable systems can perform a range of tasks by letting the user adjust the desired behavior (e.g., through configuration files). The strength of this approach is that, typically, little programming experience is required from the end user. The limitation is that the software code is usually complex and opaque because it needs to handle a wide range of scenarios, making it difficult to customize it to elicit behavior that is not among the options envisaged by the designers. It also makes it difficult to ensure that the system works reliably in all possible cases. However, customizable systems perform a more limited range of tasks at the outset and require some programming experience in order to change the default behavior (e.g., through modifying the source code). The limitation of this approach is that the system is not a turn-key solution for a wide range of scenarios at the outset, but the strength is that the source code is typically more simple, elegant, and transparent, making it relatively easy for those with some programming experience to adapt it to novel situations, even beyond what was envisaged in the original design. In the sensorimotor synchronization field, strong software packages exist that are highly configurable, such as FTAP, which can, for example, change the period and phase of the metronome pacing signal within a signal trial (Dalla Bella, Benoit, et al., 2017; Madison & Merker, 2004; Steele & Penhune, 2010) or produce grouped and metric patterns (Finney & Warren, 2002). The strength of FTAP and similar software packages is that these experiments can be run without requiring modification of the program source code (although they require writing somewhat complex configuration files). TeensyTap does not perform these experiments as-is, but instead offers a framework that is easy to adapt. For example, dyading tapping experiments in which two participants tap simultaneously can be achieved through a relatively straightforward extension of the source code provided with TeensyTap, whereas it remains unclear how existing SMS software packages would accommodate such a scenario. Users with basic programming skills may

thus use the freely available source code of the device as a basis to create future tapping experiments that have not yet been conceived of at present.

One limitation of the TeensyTap framework is that the timing of the communication between the TeensyTap device and the PC/Mac is not controlled or assessed. However, this latency is not critical for the experiments because TeensyTap functions autonomously during the experiment and assigns its own time stamps to events. Another limitation is that TeensyTap currently does not collect triggers from other devices, as is commonly done in EEG measurements performed in conjunction with sensorimotor synchronization. This functionality could be added in future versions. In that case, it will be important to validate the timing accuracy of the processing of these triggers similarly to the validation as was done here for finger taps.

ACKNOWLEDGEMENTS

The author declares no conflict of interest. I am indebted to Dr. Benjamin G. Schultz for fruitful discussions during the conception phase of TeensyTap. Furthermore, the validation experiment relied critically on access to recording equipment kindly provided by Dr. Nicholas Foster and Prof. Simone Dalla Bella.

OPEN PRACTICES STATEMENT

The TeensyTap device blueprints, the C code to run on the device, and the Python code to interface between the TeensyTap and the PC/Mac through USB are all available open-source from https://github.com/florisvanvugt/teensytap

REFERENCES

Colling, L. J., Noble, H. L., & Goswami, U. (2017). Neural entrainment and sensorimotor synchronization to the beat in children with developmental dyslexia: An EEG study. *Frontiers in Neuroscience, 11*, 360. doi: 10.3389/fnins.2017.00360

Dalla Bella, S., Benoit, C.-E., Farrugia, N., Keller, P. E., Obrig, H., Mainka, S., & Kotz, S. A. (2017). Gait improvement via rhythmic stimulation in Parkinson's disease is linked to rhythmic skills. *Scientific Reports, 7*, 42005. doi: 10.1038/srep42005

Dalla Bella, S., Farrugia, N., Benoit, C.-E., Begel, V., Verga, L., Harding, E., & Kotz, S. A. (2017). BAASTA: Battery for the Assessment of Auditory Sensorimotor and Timing Abilities. *Behavior Research Methods, 49*, 1128–1145. doi: 10.3758/s13428-016-0773-6

Dittrich, Y., Vaucouleur, S., & Giff, S. (2009). ERP customization as software engineering: Knowledge sharing and cooperation. *IEEE Software, 26*, 41–47. doi: 10.1109/MS.2009.173

Elliott, M. T., Welchman, A. E., & Wing, A. M. (2009). MatTAP: A MATLAB toolbox for the control and analysis of movement synchronisation experiments. *Journal of Neuroscience Methods, 177*, 250–257. doi: 10.1016/j.jneumeth.2008.10.002

Finney, S. A. (2001). Real-time data collection in Linux: A case study. *Behavior Research Methods, Instruments, & Computers: A Journal of the Psychonomic Society, Inc, 33*, 167–173.

Finney, S. A., & Warren, W. H. (2002). Delayed auditory feedback and rhythmic tapping: Evidence for a critical interval shift. *Perception and Psychophysics, 64*, 896–908. doi: 10.3758/BF03196794

Kim, H.-S., Kaneshiro, B., & Berger, J. (2012). Tap-It: An iOS app for sensori-motor synchronization experiments. 12th International Conference on Music Perception and Cognition, Thessaloniki, Greece.

Klemmer, E. T. (1957). Rhythmic disturbances in a simple visual-motor task. *The American Journal of Psychology, 70*, 56–63. doi: 10.2307/1419229

Madison, G., & Merker, B. (2004). Human sensorimotor tracking of continuous subliminal deviations from isochrony. N*euroscience Letters, 370*, 69–73. doi: 10.1016/j.neulet.2004.07.094

Noreika, V., Falter, C. M., & Rubia, K. (2013). Timing deficits in attention-deficit/hyperactivity disorder (ADHD): Evidence from neurocognitive and neuroimaging studies. *Neuropsychologia, 51*, 235–266. doi: 10.1016/j.neuropsychologia.2012.09.036

Patel, A. D., Iversen, J. R., Bregman, M. R., & Schulz, I. (2009). Experimental evidence for synchronization to a musical beat in a nonhuman animal. *Current Biology, 19*, 827–830. doi: 10.1016/j.cub.2009.03.038

Repp, B. H. (2005). Sensorimotor synchronization: A review of the tapping literature. *Psychonomic Bulletin and Review, 12*, 969–992.

Repp, B. H., & Su, Y.-H. (2013). Sensorimotor synchronization: A review of recent research (2006–2012). *Psychonomic Bulletin and Review*, *20*, 403–452. doi: 10.3758/s13423-012-0371-2

Romano, M., Bucklin, M., Gritton, H., Mehrotra, D., Kessel, R., & Han, X. (2019). A Teensy microcontroller-based interface for optical imaging camera control during behavioral experiments. *Journal of Neuroscience Methods, 320*, 107–115. doi: 10.1016/j.jneumeth.2019.03.019

Scheurich, R., Pfordresher, P. Q., & Palmer, C. (2020). Musical training enhances temporal adaptation of auditory-motor synchronization. *Experimental Brain Research, 238*, 81–92. doi: 10.1007/s00221-019-05692-y

Schultz, B. G. (2019). The Schultz MIDI Benchmarking Toolbox for MIDI interfaces, percussion pads, and sound cards. *Behavior Research Methods, 51*, 204–234. doi: 10.3758/s13428-018-1042-7

Schultz, B. G., & van Vugt, F. T. (2016). Tap Arduino: An Arduino microcontroller for low-latency auditory feedback in sensorimotor synchronization experiments. *Behavior Research Methods, 48*, 1591–1607. doi; 10.3758/s13428-015-0671-3

Steele, C. J., & Penhune, V. B. (2010). Specific increases within global decreases: A functional magnetic resonance imaging investigation of five days of motor sequence learning. *The Journal of Neuroscience, 30*, 8332–8341. doi: 10.1523/jneurosci.5569-09.2010

Stevens, L. T. (1886). On the time-sense. *Mind, 11*, 393–404.

Wiltermuth, S. S., & Heath, C. (2009). Synchrony and cooperation. *Psychological Science, 20*, 1–5. doi: 10.1111/j.1467-9280.2008.02253.x

Wing, A. M., & Kristofferson, A. B. (1973). Response delays and the timing of discrete motor responses. *Perception and Psychophysics, 14*, 5–12. doi: 10.3758/BF03198607

RECEIVED 09.01.2020 | ACCEPTED 17.09.2020