



A simple and cheap setup for timing tapping responses synchronized to auditory stimuli

Martin A. Miguel^{1,2} · Pablo Riera² · Diego Fernandez Slezak^{1,2}

Accepted: 15 June 2021 / Published online: 3 August 2021
© The Psychonomic Society, Inc. 2021

Abstract

Measuring human capabilities to synchronize in time, adapt to perturbations to timing sequences, or reproduce time intervals often requires experimental setups that allow recording response times with millisecond precision. Most setups present auditory stimuli using either MIDI devices or specialized hardware such as Arduino and are often expensive or require calibration and advanced programming skills. Here, we present in detail an experimental setup that only requires an external sound card and minor electronic skills, works on a conventional PC, is cheaper than alternatives, and requires almost no programming skills. It is intended for presenting any auditory stimuli and recording tapping response times with within 2-ms precision (up to - 2 ms lag). This paper shows why desired accuracy in recording response times against auditory stimuli is difficult to achieve in conventional computer setups, presents an experimental setup to overcome this, and explains in detail how to set it up and use the provided code. Finally, the code for analyzing the recorded tapping responses was evaluated, showing that no spurious or missing events were found in 94% of the analyzed recordings.

Keywords Timing experiment · Auditory stimuli · Sensorimotor synchronization

Humans have a very distinct ability to synchronize motor movements to regular sound patterns. We can finger-tap or sway along to a metronomic pulse and, moreover, we are able to extract an underlying clock, the beat, from non-isochronous rhythmic patterns (Repp & Su, 2013). Beat perception is a fundamental component for experiencing music, ranked among life's greatest pleasures (Dubé & Le Bel, 2003).

This ability to synchronize movement to an external stimuli—known as sensorimotor synchronization or SMS—has been studied in detail. Studies have revealed slowest and fastest tapping rate limits, what is the most common

spontaneous tapping rate, and how it evolves from faster to slower with age, that age allows us to synchronize to a wider rate range and that musical training improves synchronization accuracy. Several models of how we synchronize to rhythms and perform corrections in our tapping to compensate for changes in the pacing signal have been introduced and tested experimentally. When analyzing this phenomenon from the perspective of music, studies have found that the rhythmic structure of the musical signal affects synchronization precision. For a full review, please refer to Repp (2006) and Repp and Su (2013).

To understand these phenomena, behavioral studies require an experimental setup that allows presenting an auditory stimuli and record participants' responses with great time fidelity. In several cases, it is also important to capture the asynchrony between a participant response and the onset times present in the stimulus. Figure 1 presents the common scheme of a trial in an SMS experiment.

This general scheme can be instantiated in experiments performed in the literature, as presented in Fig. 2. The study in Krause et al. (2010) explored the relationship between sensorimotor synchronization and musical training. One of the tasks consisted of tapping in synchrony to an isochronous stimuli in two modalities: visual and auditory.

✉ Martin A. Miguel
mmiguel@dc.uba.ar

¹ Universidad de Buenos Aires, Facultad de Ciencias Exactas y Naturales, Departamento de Computación, Buenos Aires, Argentina

² CONICET-Universidad de Buenos Aires, Instituto de Investigación en Ciencias de la Computación (ICC), Buenos Aires, Argentina

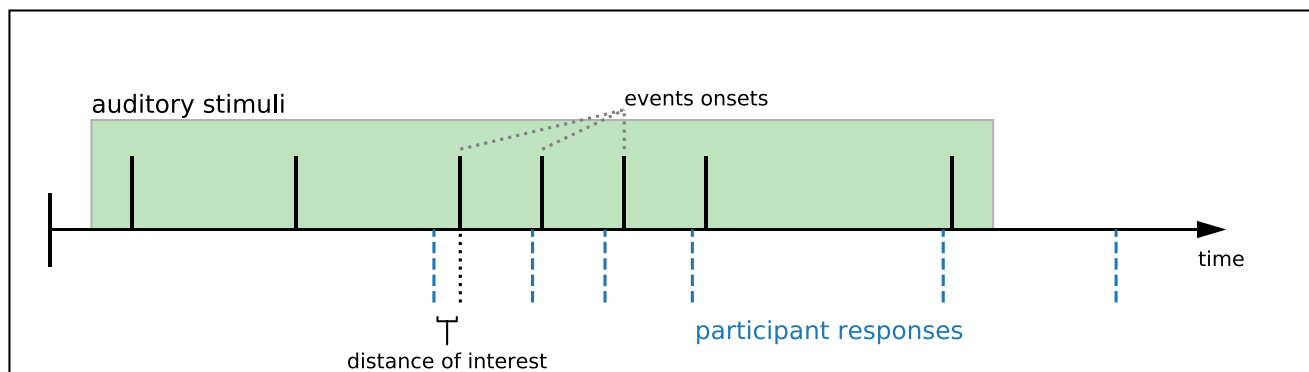


Fig. 1 Schematic of trial in a sensorimotor synchronization (SMS) experiment. A trial consists of an auditory stimulus with identifiable onsets developing in time. A participant has to listen to the stimuli and

produce responses. The measure of interest is the time interval between the participant's response and the stimulus' onset time

In the auditory modality, the trial presented an isochronous tick to which the participants had to synchronize until it stopped (see Fig. 2a). In McAuley et al. (2006), participants of different ages were asked to tap in synchrony to a metronome to study whether age changed the ability to synchronize at different tapping rates. Trials started with an isochronous tick to which the participant had to synchronize, but they were also asked to continue tapping to the metronome's rate after it had stopped. This paradigm is known as synchronization-continuation (see Fig. 2b). The task presented in Repp et al. (2005) asked participants to synchronize to non-isochronous stimulus by reproducing it. It also uses a synchronization-continuation paradigm where the continuation phase may contain a pacing signal instead of the original signal (see Fig. 2c).

Other paradigms that fit into the general experimental setup scheme proposed are auditory Go/No-Go (Barry et al., 2014) tasks and auditory time interval reproduction (Daikoku et al., 2018). The Go/No-Go task presents one of two stimuli, with one designated as *target*. During the experiment, each trial consists of presenting one of the possible stimuli and participants must respond only when the target stimulus is presented. In the auditory mode, stimuli are sounds, and target stimulus may be distinguished, for example, by pitch. In a time interval reproduction task, a time interval is presented by two sounds separated in time. Afterwards, participants must try to reproduce the interval as accurately as possible. Both task schemes are presented in Fig. 2d and e, respectively.

The setup also allows performing tasks with richer auditory stimuli where participants must signal the time location of relevant events. In general, these experiments fit the trial description presented in Fig. 1. For example, the data collection procedure in McKinney et al. (2007) required participants to tap to the beat to 30-s music excerpts. This configuration mimics Fig. 1 with the exception that the stimulus did not have relevant onset

times. Falk and Bella (2016) presented participants with repeated spoken sentences where the second presentation could include a word change. Participants were required to tap as soon as the change was detected and response time was measured. This constitutes a Go/No-Go task (Fig. 2d). In this case, the relevant onset time is the perceptual center of the changed word. A different example of a time interval reproduction task is performed in Noulhiane et al. (2007). There, participants were asked to listen to sounds of different lengths and emotional valences and then reproduce its duration. In contrast to the example in Fig. 2e, the auditory stimulus included the sound, a pause of varying length and a pure tone. The duration was indicated by the participant performing a single tap after the pure tone. In this case, the relevant onset is the beginning of the tone.

Recording stimuli onset times and participants' response times precisely cannot be directly achieved in an experimental setup using only a computer and requires specialized equipment or software. Setups presented in the literature either use specific input devices (mostly MIDI instruments) (Snyder & Krumhansl, 2001; Fitch & Rosenfeld, 2007; Repp et al., 2005; Patel et al., 2005; Babjack et al., 2015), specialized data acquisition devices (Elliott et al., 2014) or low-level programming of a microcontroller (e.g., Arduino) to work as an acquisition device (Schultz & Palmer, 2019; Bavassi et al., 2017a). Drawbacks of these setups come either from the cost of the equipment or the technical skills required. MIDI input devices and data acquisition devices (DAQs) generally used cost over 200 USD. A programmable micro-controller is cheaper (about 30 USD) but requires low-level programming skills and does not include the input device.

In this paper, we present an experimental setup that requires no programming skills, is simple to assemble, and costs under 60 USD (including the input device). Beyond simplicity and affordability, the setup proposed focuses on reliably capturing the time interval between stimulus onset

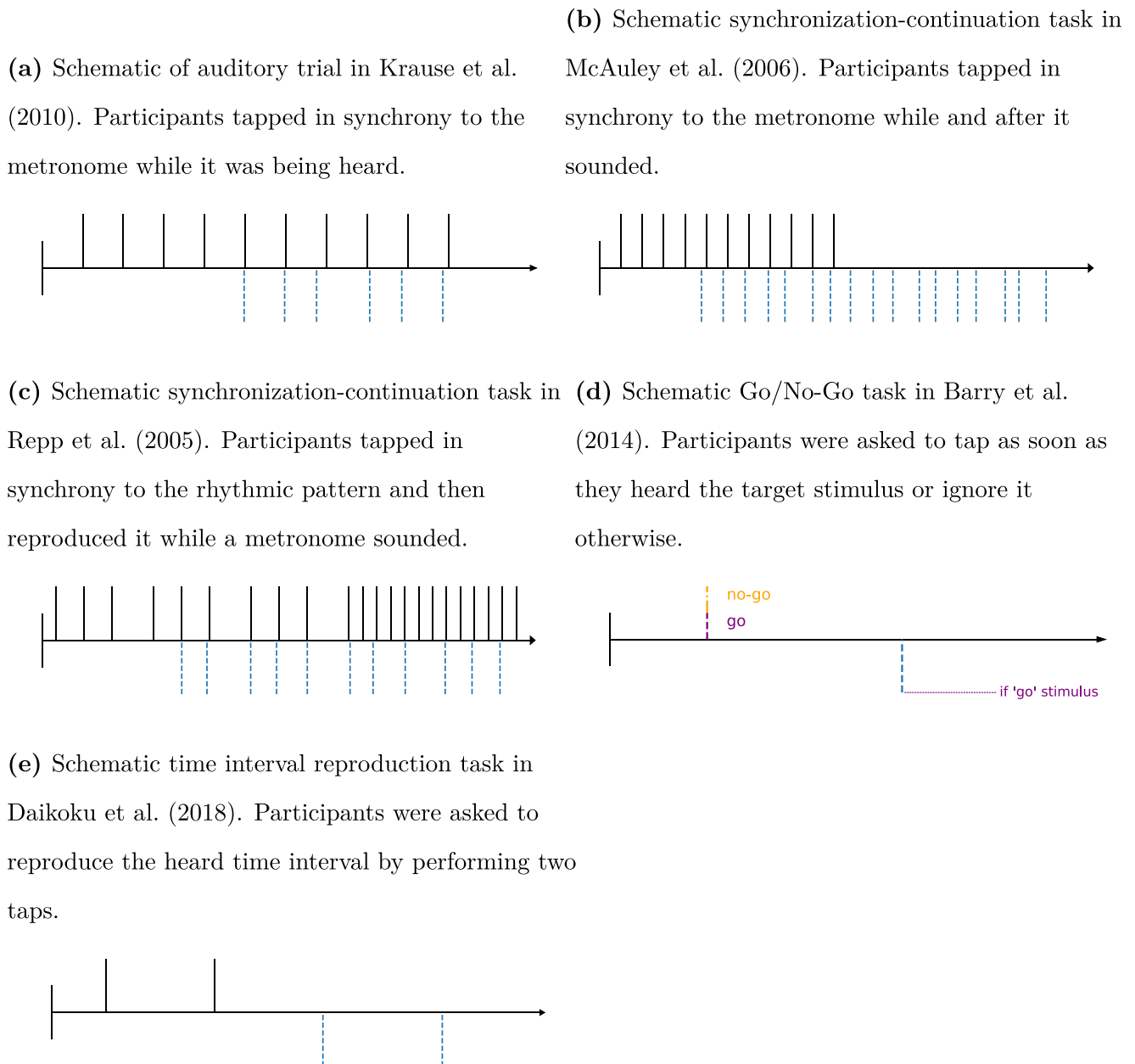


Fig. 2 Instantiations of SMS trials in various experiments

and the participant's response. To manage this using MIDI devices, latency times for both input and output devices must be verified (Finney, 2015). On the other hand, using a micro-controller can provide accurate stimulus timing but cannot easily produce rich sounds (Schultz & van Vugt, 2016). A solution with precise input and output timing using specialized equipment and software is presented in Babjack et al. (2015). More recently, some software setups allow presenting auditory onsets with precision and ease, but still rely on expensive input equipment to gather timely responses (Bridges et al., 2020).

The next section (“[Problem description](#)”) describes in detail why it is not straightforward to record stimuli to response time intervals using only a computer's input and output hardware and more sophisticated solutions are required. This section also reviews previous solutions to the problem. In section “[Setup description and installation](#)”, the setup presented here is described in detail along with assembly instructions. The main features and limitations of the setup are described in detail there. The section “[Tool suit and usage workflow](#)” presents the software tools provided to use the hardware setup and the evaluation performed

on the precision of the method for collecting participants' responses.

Problem description

The commonly expected setup to run the described experiments in a personal computer would be to have the computer present the stimuli (in this case, audio) and simultaneously record participants' responses via standard input devices such as a keyboard or a mouse. In more detail, this situation implies the following steps in time:

1. Computer program produces auditory stimuli and records stimuli onset time (o^c).
2. Sound is produced on the speakers or headphones and heard by the participant (o^p).
3. Participant produces a response by operating the input device (keyboard, mouse, etc.) (r^p).
4. The response is captured by the computer and the time is recorded (r^c).
5. Response time is calculated from the times recorded by the computer ($rt^c = r^c - o^c$).

The situation described above is depicted in Fig. 3. The issue with this simple conception of the experimental setup is that the moment where the auditory stimuli is actually produced may be significantly delayed from the moment the computer program decided to produce the sound ($\Delta o = o^p - o^c$). Additionally, the time the computer learns a keyboard key is pressed can be much later than the time the key was effectively pressed ($\Delta r = r^p - r^c$). As a consequence, the response time captured (rt^c) may be different than the real response time (rt^p).

The delays in stimuli production (Δo) and response capture (Δr) are analyzed in two magnitudes: lag (or accuracy) and jitter (or precision). Lag refers to a constant delay between the two onset or response times. Jitter refers to the unknown variability in the delay. If the delay is thought of as a random variable, the lag or accuracy would be represented by the expected delay and the jitter or precision with

the standard deviation. Depending on the setup and experimental question at task, lag can be cancelled out. In some setups, it may be possible to measure and subtract or, if the question is a comparison between groups, the comparison of response times will inherently ignore such delay. On the contrary, jitter is unknown and may differently affect each trial, being therefore more difficult to disregard.

In common computers, onset delay (Δo) may be caused by several factors. In a standard installation with an operating system, several layers of software drivers exist between the experiment's program and the sound card that translates the digital encoding of sound into electric pulses for the speakers. These layers may cause the message to be delayed between the user's software and the hardware. An exploration of the effects of different sound card configurations, computer hardware, and operating system on onset delay is presented in Babjack et al. (2015), showing that onset delay is generally greater than 20 ms with standard deviation ranging from 1 ms to over 20 ms. Finally, some sound-producing devices, such as MIDI instruments, may also require time to process the onset digital signal to effectively produce sound. Regarding responses, capture delays (Δr) can also be a product of the transit from the driver receiving information from hardware devices and the experiment's software. Some devices may also introduce delays between receiving the participant's pressure action and producing a signal. For example, standard keyboards are known to have a lag larger than 10 ms, with varying jitter of about 5 ms (Segalowitz & Graves, 1990; Shimizu, 2002; Bridges et al., 2020).

Proposed approaches

There are two main approaches to overcome the latency problems described: either reduce the delays (Δo and Δr) below a required value or record the actual onset and response times perceived and provided by the participant in a way that is independent of the computer and does not introduce relevant latencies.

Finney (2001) takes on the first approach and uses MIDI devices for both input and output. The Musical Instrument

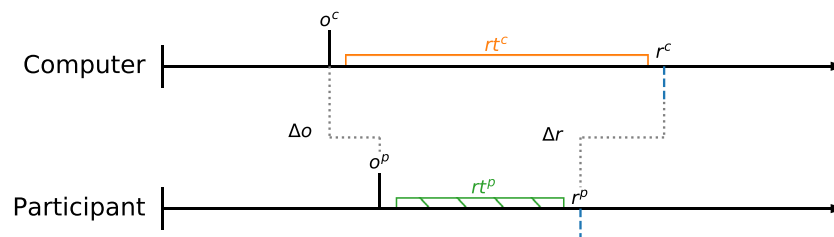


Fig. 3 Depiction of the Delays between computer and participant's stimuli onset and response times. In a common computer setup, response times are calculated from the onset and response times known to the computer (o^c and r^c , respectively). These times may differ from

the actual onset and response times perceived and produced by the participant (o^p and r^p). As a result, the obtained response time (rt^c) is different from the one that is of interest (rt^p)

Digital Interface (MIDI) is a communication protocol for sending and receiving information on how and when to play musical notes through MIDI devices. Examples are physical instruments such as keyboards or drum pads that can produce messages, synthesizers that receive MIDI messages and produce sounds, or computers with MIDI ports that may do both. In his work, Finney presents FTAP, a software tool for running experiments involving auditory stimuli and response time collection. For timing precision, FTAP takes advantage of the MIDI protocol's capacity to exchange messages at approximately one message per millisecond. The software package includes a utility to test whether such an exchange rate is achieved in a specific computer setup (computer, operating system, MIDI drivers, and MIDI card) as not every configuration allows such an optimal rate. Moreover, testing of the input and output hardware used is recommended, as it has been seen that some MIDI input devices can introduce delays of several milliseconds from the moment it is actuated until the MIDI message is sent (Schultz & van Vugt, 2016; Finney, 2015). End-to-end testing of an experimental setup implies measuring the complete time from the participant's input (pressing on the device) to the time the computer captures the message or auditory feedback is produced, depending on the requirements of the experiment. Commercial equipment for this purpose has been presented in Plant et al. (2004) and an alternative using an Arduino controller is described in Schultz (2019).

Schultz and van Vugt (2016) focuses on presenting a setup to provide timely auditory feedback to a participant's response. To do so, they capture the response and provide feedback using a programmable micro-controller (namely Arduino). Micro-controllers often provide input and output pins that allow interfacing with external hardware by measuring and producing changes in the voltage of the electrical current that runs through the pins. The importance of using a microcontroller lies in that it provides the programmer with direct access to the processor and the input and output pins. This allows more precise control over the delays of processing the input and producing the output, in comparison with using a computer with an operating system where the delays of the hardware drivers and the multitasking capabilities are harder to manage or know. In their proposed setup, they capture the participant's tap using a force-sensitive resistor (FSR) (depicted in Fig. 7). An FSR is a device shaped as a flat surface that varies the voltage of an electrical current according to the pressure it receives. Such changes in voltage can be measured in an input pin in the micro-controller to recognize when the device is being pressed. Finally, they test two methods to provide feedback from the Arduino. One method is connecting a headphone directly to an output pin of the controller. This allows the program to produce feedback very quickly (delay of 0.6 ms,

SD of 0.3 ms) with the caveat of it being a simple sound. Another method tested is the Wave Shield for Arduino, an extension hardware that allows reproducing any sound file. The Wave Shield feedback requires more time to emit a sound (2.6 ms on average) but still has low jitter (0.3 ms).

The timing mega-study in Bridges et al. (2020) analyzes onset lag and jitter for auditory and visual stimulus on a variety of existing software packages for designing and running behavioral experiments on PC. These packages provide utilities to generate programs that run experiments, present stimuli, collect answers, randomize trial conditions, among other features commonly used. Moreover, these software packages manage drivers and settings in order to produce onsets, both auditory and visually, with less than a millisecond unknown delay. The results of this work show that common computer hardware can be used to produce auditory stimuli quickly in spite of the stack of drivers and multitasking mentioned. To do so, the right software configuration is required. For example, PsychoPy must be updated to version 3.2+, which recently included the correct software to achieve millisecond auditory onset presentation. The issue still remains on the precision of the input capture, which in Bridges et al. (2020) is solved by using a specialized response button device.

In Babjack et al. (2015), different configurations of computer hardware, sound cards, sound drivers and operating systems are tested for lag and jitter on presenting sounds. In this study, audio onset lag is generally above 20 ms with the exception of some configurations of sound cards and drivers. The work also presents the Chronos response box, which uses specialized hardware and software that interfaces with the E-Prime software allowing presentation of auditory stimuli with less than a millisecond lag and jitter. The response box is also equipped with buttons that allow collecting response data also with millisecond precision. This is an all-in-one solution with the caveat that it requires its own proprietary software.

The MatTAP tool suite, presented in Elliott et al. (2009), uses the second approach to work around the delays and achieve precise recording of stimuli and response times. This approach is based on using a recording device independent of the computer running the experimental procedure in such a way that producing the stimulus and recording responses is not affected by the software stack of the operating system. More specifically, they use a data acquisition device (DAQ). DAQs are devices that can produce and record multiple analogue and digital signals simultaneously with a sampling rate of hundreds of kilo-samples per second, providing sub-millisecond precision. The MatTAP tool suite is a MATLAB toolbox that communicates with the DAQ in order to provide the stimuli onset times and sounds. The DAQ then produces the sounds and simultaneously records the input from the input device. Each

auditory onset is accompanied by a digital onset on a separate channel that is required to be looped back into a recording channel of the DAQ. Although it produces the stimulus without lag relative to the stimuli sequence, there may be a delay introduced by the initial communication between the computer and the DAQ. As a consequence, the loop back is required to be able to synchronize the stimuli onset times with the response times (see the next section and Fig. 4 for a more detailed explanation). Finally, if the DAQ has more than two input channels, MatTAP is capable of recording two input devices. Also, the digital output signal produced with each stimulus onset may be used to drive another output device. The MatTAP tool suit provides software utilities for creating up to two metronomes for synchronization experiments, allows managing settings for multiple trials, and also collects the experiment response data for each trial. The tool suit also provides a customizable utility to analyze the input signals, extract responses, and calculate stimulus to response asynchrony times.

The setup proposed here also follows the second approach. In comparison with MatTAP (Elliott et al., 2009), we use an external sound card as an acquisition device, which works as a less expensive replacement. Moreover, the software tool set provided here does not require proprietary software such as MATLAB. Finally, we present instructions to assemble an input device that can be connected to the sound card and provide accurate response time recording.

Setup description and installation

In “Problem description” we established two main approaches to solve the issues that arise when recording response times to auditory stimuli due to delays in both onset presentation

and response time collection. One approach is to reduce such delays below an accepted value. Another approach is to record the onset times actually perceived and produced by the participant with an independent recording device. Our setup takes the second approach and proposes to do so using either the sound card already present in common desktop computers or an inexpensive external sound card. In this section, we present why our setup addresses the problem, how our setup is assembled, and what the expected workflow is for running experiments with it. To complete the setup, this work presents instructions to assemble a pressable input device using a force-sensitive resistor (FSR) and an open-source software tool suit to produce the stimuli, record the responses, and obtain response times from our proposed input device. The instructions to assemble the input device are introduced in the next subsection. Then we present instructions to connect the input device, the recording device, and test the setup connections. The tool suite is introduced in the next section.

The key component of the approach used in this setup is to be able to record both the participant’s responses and the stimuli simultaneously. This allows having the stimuli synchronized with the responses on one device’s timeline. Because one of the delays is introduced between the moment the experiment code produces a stimulus and when it is played on the output device, producing each stimulus separately would add variability to the inter-stimuli interval. Such behavior can render certain experimental conditions unusable. Our proposal is to package all stimuli onsets into one audio file. This introduces only one delay on when the whole stimuli set is reproduced ($\Delta s = s^c - s^p$) but no variability in inter-stimuli intervals. Finally, stimuli and participant’s responses are recorded by the device. Stimuli onset times can be retrieved by synchronizing the output audio file with the recording. Response times

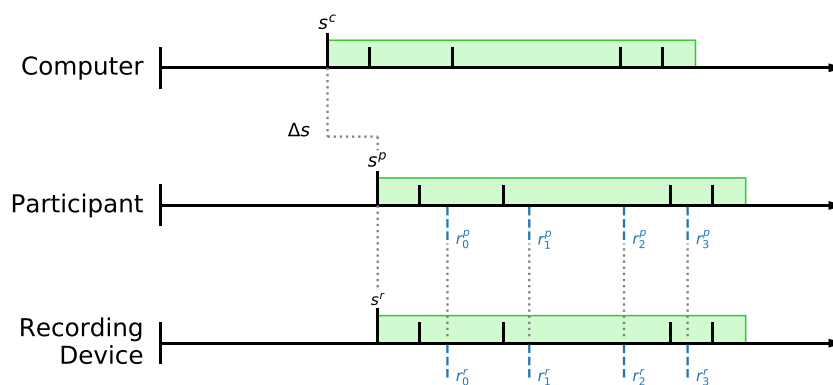


Fig. 4 Representation of stimuli and recording times when using an independent recording device. This approach adds a new device that simultaneously records the stimuli and the responses with high accuracy. Stimuli are presented as one audio with multiple onsets (box

containing lines). The stimulus presentation time may lag from the computer command but is recorded at the same time it is heard by the participant ($s^c \leq s^p$ and $s^p = s^r$). Response times (r_i) are captured in synchrony with stimulus presentation

can be obtained from the recorded signal relative to the beginning of the stimulus. How this procedure is performed is explained in section “[Tool suite and usage workflow](#)”. The setup and new definition of delays is depicted in Fig. 4.

To achieve recording the stimuli and responses simultaneously, the recording device used must have a stereo output and at least two input channels (or one stereo input). With this, one channel of the stereo output can be looped back into one of the input channels. Moreover, to keep the setup simple, our proposed setup requires the recording device to have a secondary output that mirrors the primary. While the primary output signal is looped-back into one input channel, the secondary output is connected to the output device (speakers or headphones). Finally, the signal of the response device is connected to the second input channel of the recording device. With this connection setup, the audio input of the recording device can be collected into a stereo file containing the stimulus in one channel and the response signal on the other. The connections mentioned are depicted in Fig. 5. The audio output and input signals are depicted in Fig. 6.

In the next subsection, we present how to assemble the input device used in the complete version of the setup. Then we show how to connect the input device, the recording device, and the computer, and then test that the connections work correctly.

Input device (FSR)

Our setup uses a pressable input device. The main component of the device is a force-sensitive resistor (FSR), a flat sensor whose electrical resistance is reduced when pressed (Fig. 7). The variance in resistance can be used to create a variation in voltage that can be recorded by the audio input of a sound card. The proposed device uses a 3.5-mm female audio jack that allows connecting the input device with the sound card using standard audio cables.

The circuit allowing this variation requires a voltage source. We propose using a standard USB (type a) cable connected to a computer for this purpose. The voltage variation provided to the sound card can saturate the recording, depending on the device’s sensitivity. This saturation can modify the activation profile captured from the FSR. The proposed circuit adds a voltage divisor that allows limiting the maximum voltage received by the sound card. In our proposal, we use a 10k Ω potentiometer, another adjustable resistor that can be set by trial and error to prevent the signal from saturating the recording.

The proposed circuit is presented in Fig. 7. For the voltage source, we use a standard USB (type a) cable connected to the computer. USB cables have four pins, two for data, one that drives a 5-V signal (VBUS or VCC), and a ground connection (GND). The VBUS pin (Fig. 7, red cable)

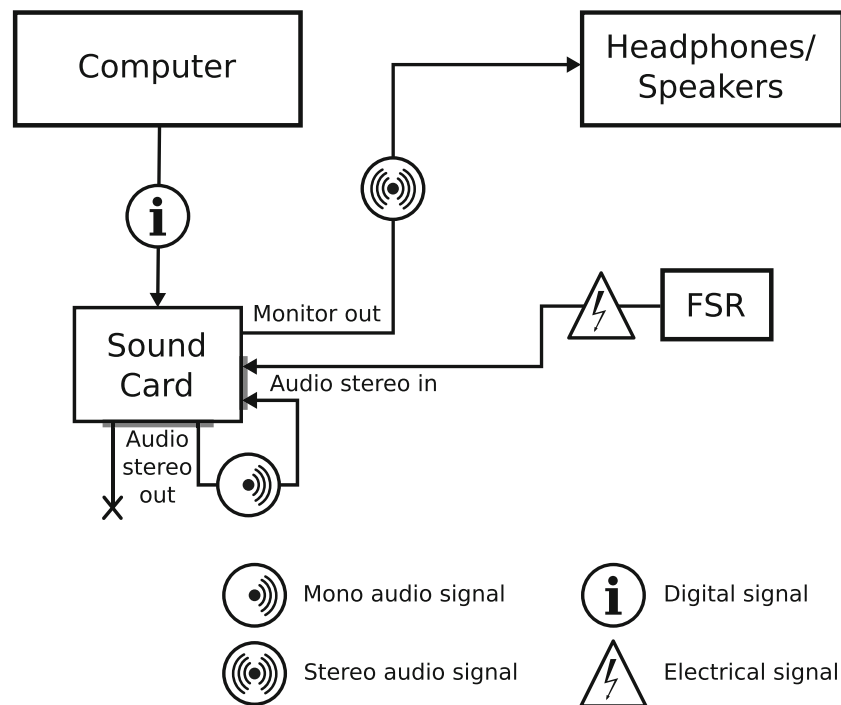
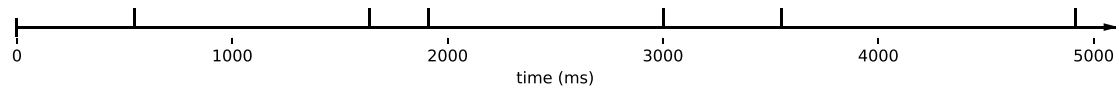
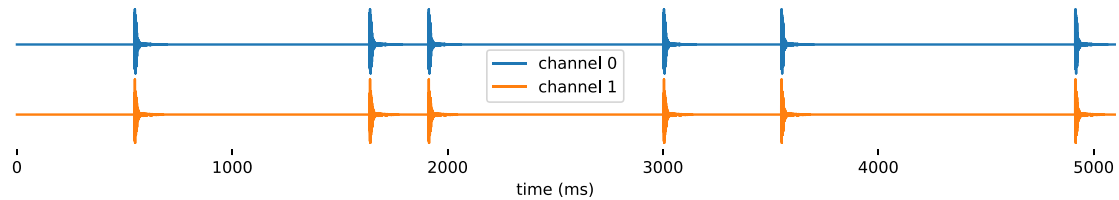


Fig. 5 Schematic of the connections of the setup. The computer exchanges information with the recording device (sound card). One of the device’s sound outputs is sent to the participant and the other has one channel looped-back to one of the recording device’s input channels. The response device is connected to the other input channel

(a) Stimulus onset times



(b) Stimulus audio



(c) Mixed loopback and input device recording

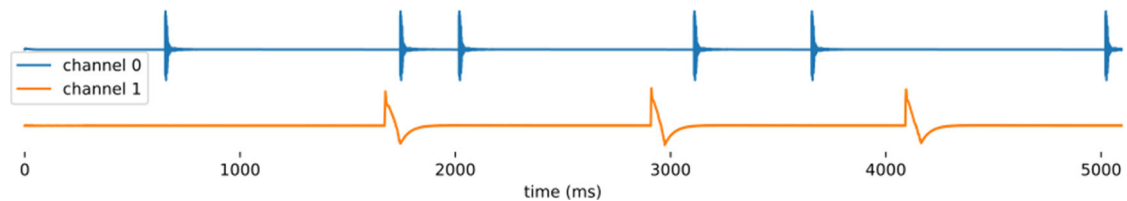


Fig. 6 Example of stimulus audio signal (*middle*) and an input recording (*bottom*). The stimulus signal is an arbitrary stereo audio file (b). In this case, it is an audio signal constructed from designated onset times

(a). The input recording (c) contains on one channel the looped-back signal from the stimulus (0) and on the other the electrical signal from the input device (1)

is connected to the divisor circuit, where the centerpiece is the potentiometer (Fig. 7, purple cable). The division goes to the FSR (orange cable) and back to the ground (black cable). The other end of the FSR (yellow cable) is connected to the 3.5-mm jack (blue) and simultaneously grounded through a 22k Ω resistor (black). More detailed instructions for assembly are presented as a video tutorial, linked in the Open Practices Statements section.

The next subsection explains how the FSR input device is connected with the rest of the setup and how to test the connections. It also explains how to use the potentiometer to adjust the signal amplitude to avoid saturation. The exact circuit used in this work is presented in Fig. 17 and replaces the potentiometer with two resistors selected for the sound card used (Behringer UCA-202). It has a further adjustment to deliver a descending (instead of ascending) voltage change when the FSR is actuated given that this sound card inverts the signal when recording. The tool suite provided here has a setting that allows managing this situation.

Setup assembly

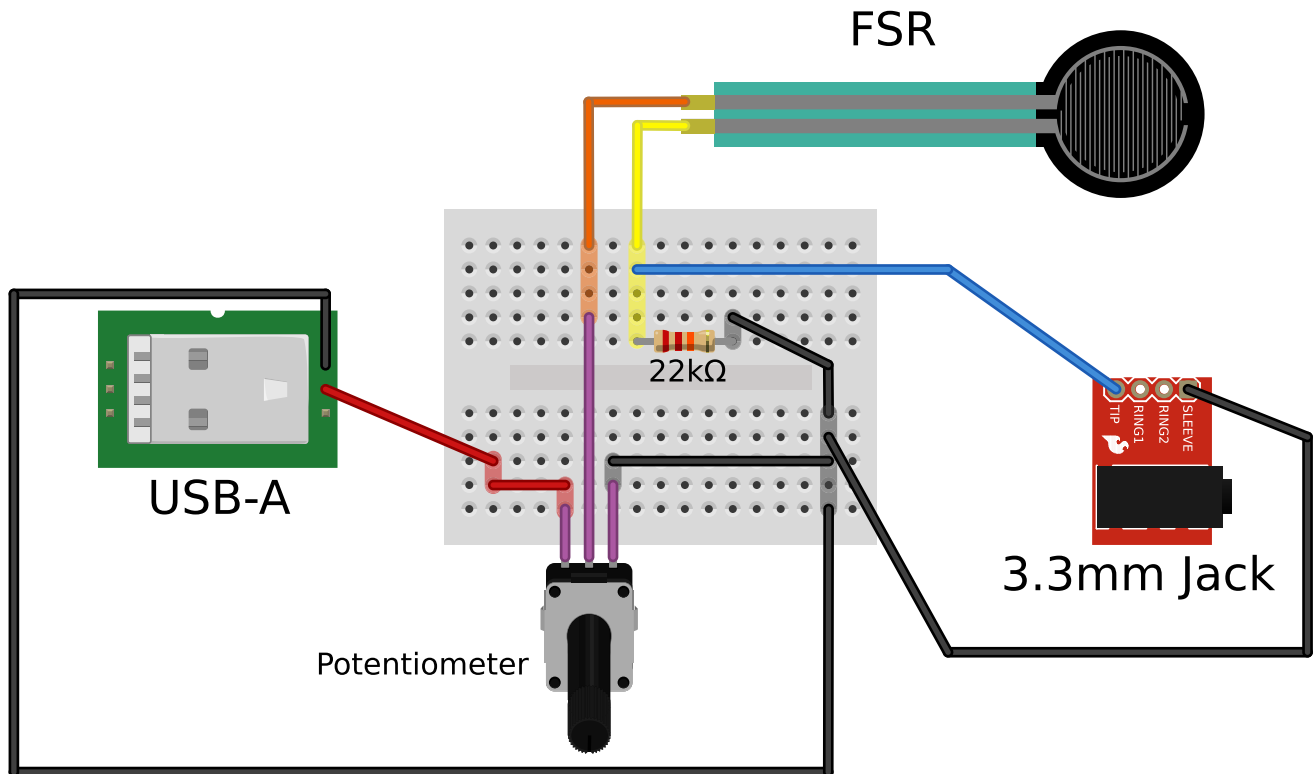
The three key components of the setup are the recording device with two input channels and output channels, the

loopback between output and input, and the mixing of the input device with the loopback in the stereo input. A schematic of these connections is presented in Fig. 5. In Fig. 8 we present the same connection scheme with the picture of the devices connected. Given that our external sound card (Fig. 8a) uses RCA plugs, we use a male-male RCA-RCA cable to perform the loopback (Fig. 8b). Although we used a stereo cable, a mono cable is sufficient. Finally, we connect the FSR setup with the recording device using a 3.5-mm plug to RCA cable (Fig. 8c). Again, we used a stereo cable, but a mono cable is sufficient.

The connections can easily be tested by playing an audio from the computer and recording the audio while tapping on the input device. In Fig. 9, we present part of the interface of an open-source sound recording and editing software (Audacity Team, 2021). By recording audio from the recording device with the loopback connection, an audio track such as the one in Fig. 6c should be produced. The track should contain the stimulus signal on one channel and spikes corresponding to the tapping on the other one.

This setup and software can also be used to inspect the recording of the FSR signal to calibrate the FSR input device. Peaks are expected to look as shown in Fig. 9b. In

(a) Schematic of the component connections



(b) Schematic of the electronic connections

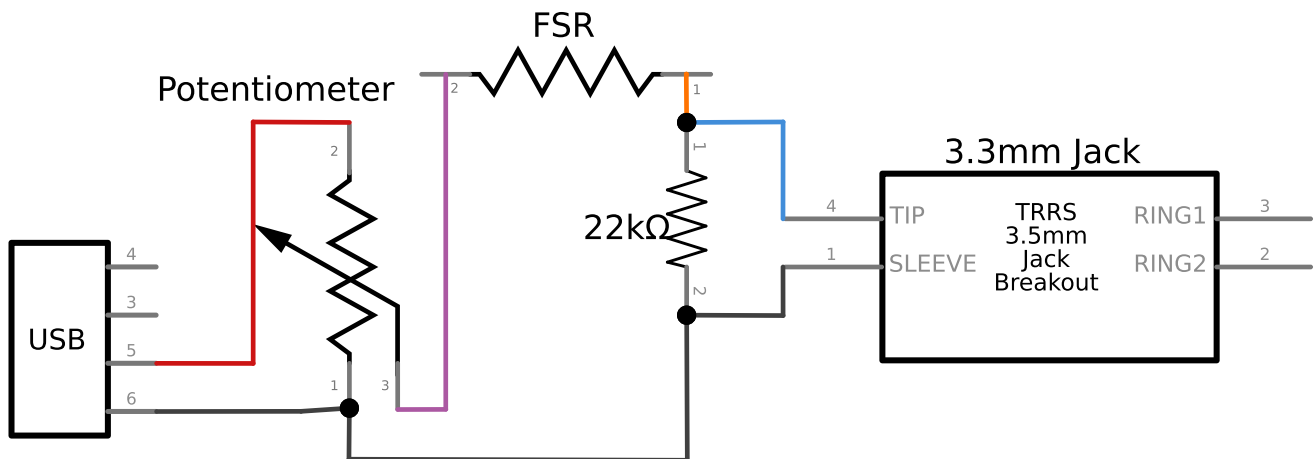


Fig. 7 Setup diagram for a tapping input device using a force-sensitive resistor (FSR). The setup drives current with varying voltage from a VBUS pin of a USB (type a) connector to the pin of a 3.5-mm audio jack. Voltage is limited using a voltage divider circuit (orange and purple). The final output voltage into the audio jack depends on the

resistance provided by the FSR, which drops with pressure. The higher the pressure, the higher the voltage provided in the audio jack and recorded by the sound card. The schematic was drawn using Fritzing (Knörig et al., 2009)

case the FSR signal saturates the audio card, the peak will contain a flat top, as shown in Fig. 9c. This can be solved by adjusting the potentiometer, which regulates the maximum height of the peak. Another issue might come from the sound card inverting the signal as in Fig. 9d. This can be

solved by inverting the audio recording on the recorded channel. An option to manage this is provided in the tool suite described in “[Tool suite and usage workflow](#)”.

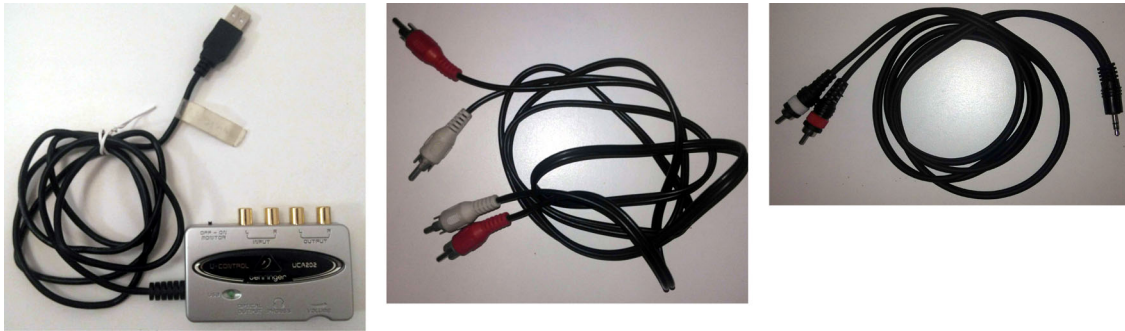
A website where further detail and tutorials are provided is detailed in the Open Practices Statements section.

(a) Sound card with stereo

input and output and monitor

(b) Stereo male-male rca cable.

(c) Stereo male-male rca-3.5mm jack cable.

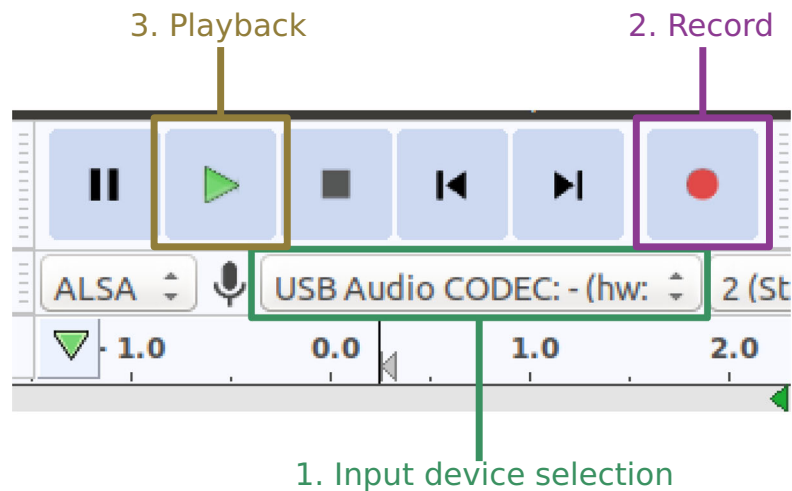


(d) Picture of connections in the setup. It replicates the diagram in figure 5.

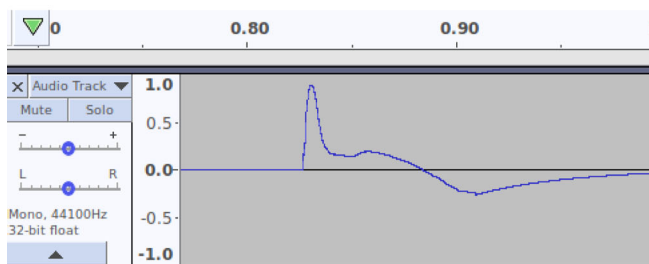


Fig. 8 Reference pictures of the elements and connections of the setup

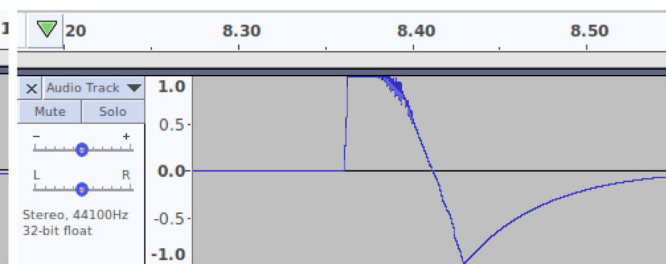
(a) Relevant buttons and configuration



(b) Depiction of a recorded fsr peak



(c) Depiction of a saturated fsr peak



(d) Depiction of an inverted fsr peak

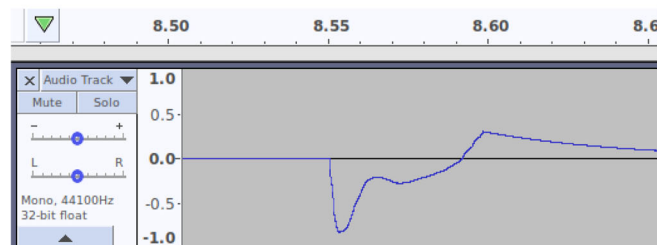


Fig. 9 Interface of Audacity (Audacity Team, 2021), used to test the recording. To test the setup, a recording of the computer's output and inputs on the device must be performed. The input device (in case of an external sound card) must be selected (device selection menu a.1), recording should be started and stopped while the input device is operated (record button a.2), and then the recording should be played and a

mixture of the sound being played and the tapping on the input device should be heard (playback button a.3). A peak is expected to look as presented in (b). (c) presents the case where the peak saturated the recording, resulting in a flat top. (d) presents an inverted peak, where the first peak is negative

Tool suite and usage workflow

To make the use of the proposed setup as convenient as possible, in this work we present a tool suite to produce the stimuli, record responses, and analyze the recordings. The tools are Python programs open-sourced under an MIT license. The tool suit was developed using the UNIX

Philosophy (Raymond, 2003), so each program is independent and dedicated to solving one issue. We also provide code examples for using the setup with the experimental design frameworks OpenSesame (Mathôt et al., 2012), PsychoPy (Peirce et al., 2019), and Psychtoolbox (Kleiner et al., 2007). We provide more details of these examples at the end of the section.

```

546
1638
1911
3003
3549
4914

```

Fig. 10 Example text file indicating onset times in milliseconds, one per line

Next, we outline the workflow considered and what are the tools we provide to address each stage. An expected experiment design workflow would have the following stages:

- **Define the stimuli.** Stimuli can be any arbitrary set of audios. Many sensorimotor synchronization experiments use stimuli conformed by discrete sound onsets at designated times. We provide a utility (`beats2audio`) to transform a text file with onset times into an audio that produces a sound on each onset time.
- **Expose participants to the stimuli and collect responses.** Given a stimuli set, participants should hear each audio and produce responses by tapping on the response device. The utility provided (`runAudioExperiment`) receives a configuration file declaring the audio stimuli set and presents each one while recording the response. Responses for each trial are saved as an audio file (Fig. 6c) on a designated output folder.
- **Extract tap times from the recordings.** Using the original stimulus and the response recording, tap times are extracted relative to the beginning of the stimulus. A different tool is provided for this purpose (`rec2taps`).

In case the stimulus to be used is an audio with simple identical onsets on designated times, the utility `beats2audio` receives a text file (Fig. 10) with a list of onset times in milliseconds and outputs an audio file (Fig. 6b). A click sound is produced on each onset time.

To run the tapping experiments, i.e., producing the stimuli and recording the loopback and input, we provide a simple utility named `runAudioExperiment`. The utility requires three arguments: the path to a configuration file, the path to a trial file, and the path to an output directory where recordings are to be stored. The experiment execution follows the steps depicted in Fig. 11. Trials are run in a succession, each trial consisting of five steps. First, a black screen is presented for a specified duration. Secondly, screen turns to a (possibly) different color and white noise combined with a tone is played. This option is intended to help remove rhythmic biases between trials. Third, the screen goes black again and the trial stimulus is played as recording is enabled. Fourth, the screen stays black in silence. Recording continues in this stage. Finally, another optional colored noise screen is produced. Following this screen, the next trial, starting with the black screen, begins.

The configuration file provided as the first argument of the utility specifies the parameters of the execution of the steps mentioned above (Fig. 12). The trial file is a plain text file containing the stimuli set, one per line as paths to audio files. The output directory defines where the experiment outputs are to be saved. The utility produces as an output one recording per trial, as obtained from the sound device, and a `csv` file containing a table describing the details of the experiment execution (Table 1). The name of the output folder can be used to identify experiment runs either by date, run id number, or participant's initials.

The last utility, `rec2taps`, extracts tap times from the audio recordings produced during the experiment. To do so, it requires two arguments, a trial recording audio file and its original stimulus file. The utility uses the original stimulus audio to find its starting point (Fig. 4) in the recording. It does so by looking for the maximum cross-correlation between the stimulus and the loopback channel of the recording. Then, using the channel where the input signal is recorded, the utility finds peaks in the signal and extracts tap times as the location of the maximum of each peak. Given that the beginning of the original stimulus can be found in the trial's recording, the playback delay can be subtracted from tap times, obtaining tap times relative to the beginning of the stimulus. Tap times are printed out in milliseconds, one per line.

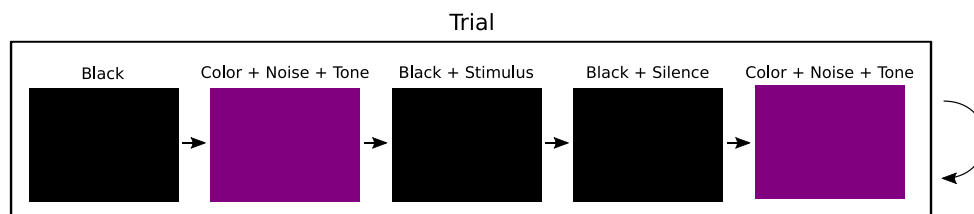


Fig. 11 Depiction of a trial

```

black_duration: 600           # Duration of black screen (in ms)
c1_duration: 3000           # Duration of first noise screen (in ms)
c1_color: "#afd444"         # Color of first noise screen
c2_duration: 000           # Duration of second noise screen (in ms)
c2_color: "#afd444"         # Color of second noise screen
randomize: false           # Whether trial order should be randomized
sound_device: "default"     # String or int identifying the sound device used
silence_duration: 1500     # Duration of silence after stimuli playback
c_volume: 1.0              # Volume of cleaning sound

```

Fig. 12 Configuration file example

Detailed instructions on how to use the utilities described are provided in each code's project readme files. Links to code projects are listed in the Open Practices Statements section.

The setup can also be integrated with other software frameworks for running experiments such as OpenSesame (Mathôt et al., 2012) or Psychtoolbox (Kleiner et al., 2007). In general, all that is required to integrate the setup is to be able to play an audio file and record the input of the sound-card simultaneously, having the loopback connection in place. The recording must then be analyzed to extract the taps and synchronize them to the beginning of the stimuli (e.g., using `rec2taps`). We provided code examples that integrate the setup with other experiment frameworks (OpenSesame, PsychoPy, and Psychtoolbox). These examples can be found in the `example` folder in the repository for the `runAudioExperiment` utility. In Python-based frameworks, we used the same library for playing and recording sounds simultaneously that was used in our tool. In Psychtoolbox there was no such option, but an equivalent result was achieved by starting and ending the recording before and after the audio playback, respectively.

The next subsection informs in more detail how tap times are extracted from the recording and analyzes its accuracy.

Signal analysis

The goal of the setup is to be able to record a participant's tap times with respect to the stimulus with high precision. We achieve this by recording the tap action and the stimulus as it is heard by the participant at the same time. This allows synchronizing the recording with the original stimulus signal and therefore aligning the tap times with respect to the beginning of the stimulus. Because the setup does not

limit the delay between the moment the computer proposes to play the stimulus and the time it is actually heard (Δo in Fig. 3), the only two timing validations required are whether the recording can be correctly aligned back to the original stimulus and whether tap times are captured with precision from the recorded signal of the input device.

First, the proposed loopback connection (Fig. 5) records the audio signal directly from its output. The audio signal is recovered as is, allowing for correct alignment of the signal with its recording. Second, in the presented setup, the signal from the input device is a function over time of the activation of the device, recorded as an audio signal. From this signal, we intend to extract individual time points that represent each actuation of the input device. In electronic input devices as the one presented here (the FSR), the activation signal is not a simple on-off function, but a curve describing the change of pressure on the device through time. To obtain individual tap times for each actuation, the signal must be processed to recognize each activation and then select a point in time within the curve that is representative of the time of actuation.

This process raises two aspects subject to analysis. First, whether the processing misses any individual activation or detects spurious activations. Secondly, how representative the selected time point is within the activation curve of the actuation process. Considering the functionality of the FSR, the signal peak is the moment where the highest pressure is applied to the input device. We decided to select the maximum of the activation curve as a representation of the tap time. We will now focus our attention on the analysis of the performance of the signal processing algorithm used in `rec2taps` when applied to recordings performed with the presented input device.

Table 1 Example of an output table from an experiment run

index	stimulus_path	recording_path	black_duration	c1_duration	c2_duration	silence_duration
0	s1.wav	s1.rec.wav	600	300	0	1000

The proposal of a force-sensitive resistor (FSR) as input device was due to the clarity of the signal provided. The signal is very close to zero when it is not being actuated and then rises rapidly, proportionally to the pressure applied. Figure 13a shows the shape of one FSR activation when aligned to the detected maximum and normalized to the peak's height. Figure 13c shows mean activation over time for 3000 peaks from 68 recordings. The process to detect activations starts by rectifying (setting to 0) the signal below a threshold defined as 1.5 times the standard deviation of the signal amplitude (Fig. 13b). Afterwards, peaks are found as points in the signal that are local

maximums and have a prominence of at least the mentioned threshold and are distanced between each other at least 100 ms. The prominence of a peak measures the height relative to the smallest valleys between a possible peak and any closest greater peak. Our utility uses the function `find_peaks` from the `scipy.signal` package (version 1.2.0) (Virtanen et al., 2020).

To inspect the recall and over-sensitivity of the algorithm used, we inspected its performance over a set of tapping recordings from a beat tapping experiment using the proposed setup (Miguel et al., 2019). The experiment required participants to listen to non-isochronous rhythmic

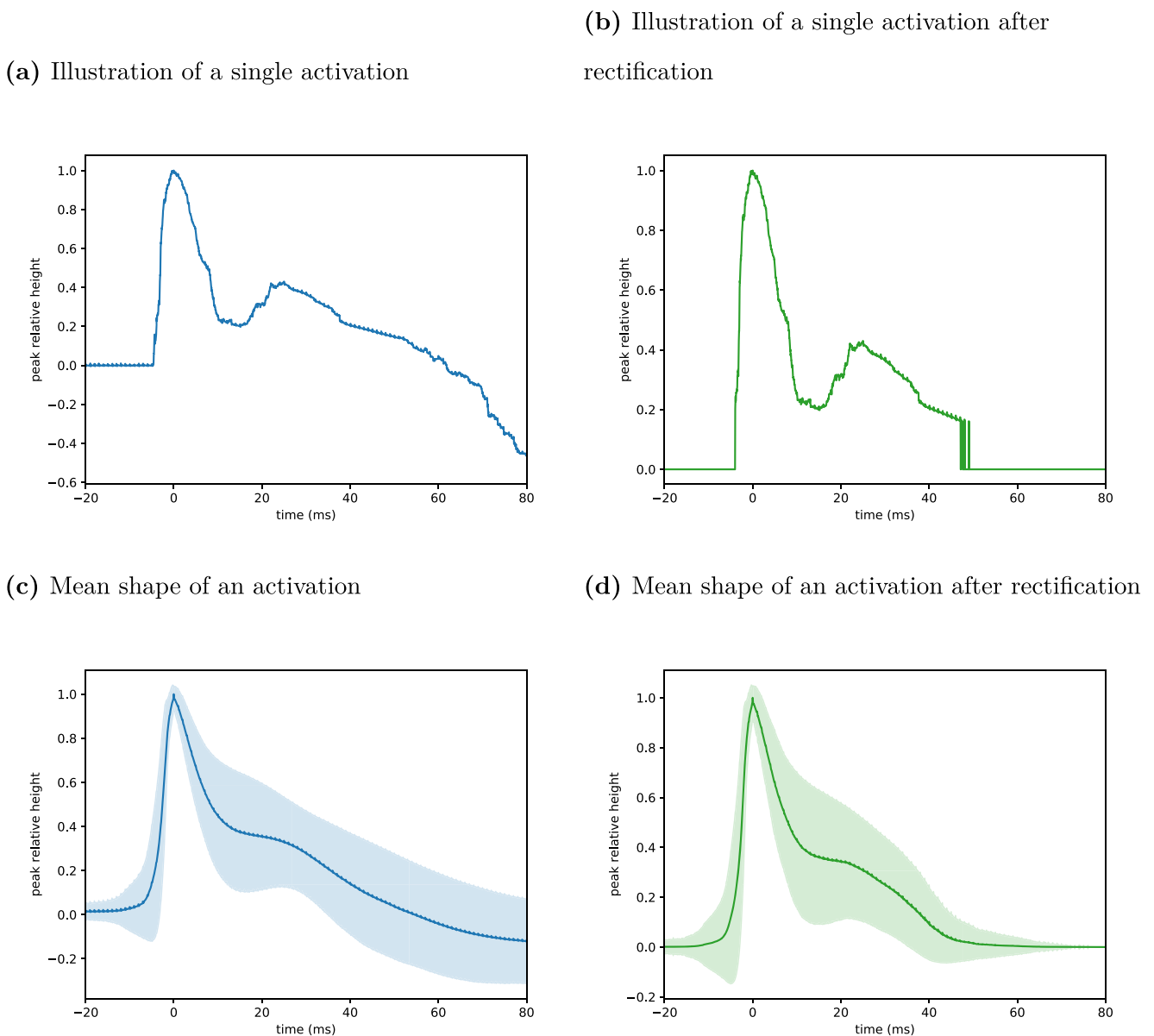


Fig. 13 Shape of an FSR activation peak. The *main line* shows mean relative activation over-time. Time zero represents the maximum of the peak. *Shading* represents standard deviation of the activation

passages performed by identical click sounds and tap to a self-selected beat. Participants were free to choose the beat and were allowed to change the beat mid-rhythm or even pause tapping. The data set comprises 518 recordings from 21 participants. The evaluation of the peak picking process consisted of visually inspecting the recording signal with the detected tap times overlapped and annotating for each recording the number of missing and spurious activations. Inspection was performed by one of the authors. On 491 (94.79%) of the audios, no missing or spurious activations were seen and only on five (0.97%) were more than two missing or spurious activations reported. The experiment contained rhythms of varying complexity, some of which had a very ambiguous beat. As a consequence, some participants produced taps of varying strength, including some weak onsets. Whether these activations are to be considered may depend on the nature of the experiment and can require making the peak picking process more sensitive. The provided utility, `rec2taps`, allows configuring the mentioned detection threshold as a parameter. It also allows producing plots of the FSR signal and detected peaks to calibrate the parameter. In the [Supplementary Material](#), we present plots as the one produced by the utility as examples of tap detections for both general cases and for cases with taps of varying strength to illustrate the workings of the peak picking procedure.

A final caveat of the signal processing regards the usage of a sound card as the recording device. Sound cards generally high-pass filter the signal at about 5 Hz. As a consequence, the shape of the input signal may be modified, especially if the tapping action was too soft. We examined the effect of a high-pass filter on a FSR signal recorded with a digital signal acquisition device at 1000 Hz. To do so, we resampled the original 1000 Hz signal to 48,000 Hz using a linear interpolation, applied a 5-Hz high-pass filter and recalculated the location in time of the peak of the modified signal. We looked into 1508 tap activation profiles from a synchronization experiment. In 67.57% of the cases, the maximum of the filtered signal remained in the same millisecond position. In 26.72% and 5.5% of the cases, the maximum of the filtered signal was 1 or 2 ms ahead of the original signal, respectively. In 0.2%, the shift was greater than 2 ms, up to -25 ms. We hypothesized the negative lag of the maximum of the filtered signal to be related with a soft tapping action. We inspected this hypothesis by looking at the maximum amplitude of the FSR signal with respect to the peak's lag. Effectively, lags greater than 2 ms were seen only in taps three times softer than average. Figure 14 presents the distribution of the amplitude of the peaks for each lag found.

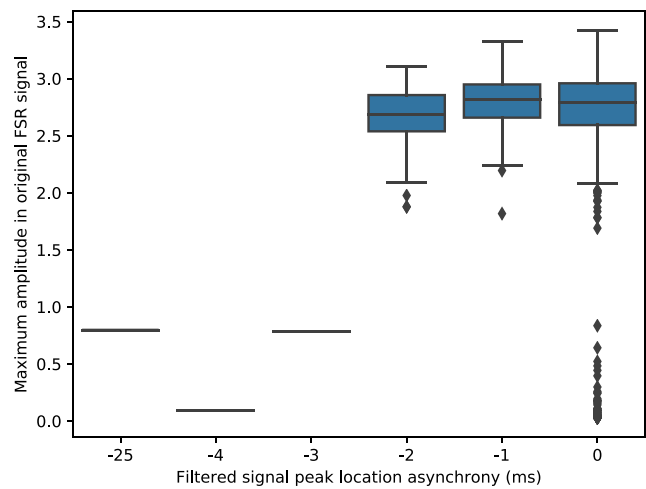


Fig. 14 Distribution of tap strength for delays of filtered signal maximum

Discussion

The current work presents an experimental setup intended to collect timed responses with high precision (less than -3 ms of delay) synchronized to onset times in auditory stimuli. Another main characteristic of the proposed setup is its inexpensiveness and simplicity. The introduction presents the general schematic of experimental trials with auditory stimuli where the quantity of interest is elapsed time from the moment a stimuli is heard until a response is provided. In the [“Problem description”](#) section, we explained why this quantity cannot be measured in a standard experimental setup using a computer. We also review previous approaches to the issue. In the [“Setup description and installation”](#) section, we describe our proposed setup and provide detailed instructions for assembly. Finally, in [“Tool suite and usage workflow”](#), we present an open-source software tool suite to run experiments using the presented setup and references to code examples for integration with other experiment design software (e.g., OpenSesame and Psychtoolbox).

Being able to collect response times to auditory stimuli with precision cannot be easily done using a standard computer with default input devices (keyboard or mouse). This is due to latencies introduced between the input device and the experiment software or between the experiment software and the output device. Approaches to this issue are either using specialized hardware for input and output, running the experiment using a programmable micro-controller or recording auditory output and responses in a separate recording device. Our setup uses the last method.

Although this approach has been presented before in Elliott et al. (2009), we here present a less expensive alternative by using an inexpensive recording device. Moreover, we present assembly instructions for an inexpensive input device that allows high-precision recording. This setup easily allows using any audio as stimulus. In addition, we provide an open-source tool suite for using the setup that does not require proprietary software. We also provide integration examples with other open-source experiment frameworks.

An evaluation of the performance of the software's capability to detect participant's responses is described at the end of the previous section. The evaluation showed no spurious or missing activation detections in 94% of the analyzed recordings and under 1% presented more than two. Miss-detection of activations was seen to relate to situations where the tapping action varied in strength throughout the experiment's trial.

A main limitation of this setup is that it cannot respond to participants responses, either by changing the course of the experiment or providing feedback. In that situation, the most inexpensive approach is given in Schultz and Palmer (2019). In case of access to precise input equipment, more direct setups can be accomplished (Finney, 2001; Bridges et al., 2020; Babjack et al., 2015). Another caveat of the setup presented here is a possible shift in response time in case of soft tapping (see Section "Signal analysis"). Finally, assembly of the input device requires a minimum knowledge of electronics.

The main idea of the proposed setup can be extended to be used in combination with other equipment. For tasks where more than one type of response is necessary, more than one input device can be built and connected to separate recording channels using a sound card with more than two input channels. Some inexpensive options are available if two input devices are required. If more are needed, a data acquisition device (DAQ) can be used in a similar fashion as explained in Elliott et al. (2009). The setup can also be used in combination with EEG equipment. The multi-channel signal of an EEG helmet is recorded in a device with multiple input channels, one per electrode, and extra channels that are used for synchronization with the stimulus computer and other equipment (Bavassi et al., 2017b). In the same fashion as our proposal, the stimulus output can be input into the EEG recording device together with the FSR signal. Later, the recording of the stimulus is aligned with the original audio, aligning the FSR and EEG signal with it. As a general rule, any equipment providing an analog signal can be recorded in synchrony with the auditory stimulus with a sound card with enough input channels. If the signal is digital, a DAQ can be used. For example, common eye-trackers communicate with a computer using a custom protocol that allows synchronization between

the computer and the eye-tracker. To synchronize the eye-tracker information with the auditory stimulus, a setup similar to the previously described for the EEG can be used: the stimulus and FSR is recorded by a DAQ, together with a TTL signal from the eye-tracker computer that is then used to synchronize the recording (Dimigen, 2020; Care et al., 2020).

In summary, we provide an inexpensive setup for recording responses to auditory stimuli with millisecond precision together with a software tool suite for using the setup. The main focus is on getting high-precision response times relative to the auditory stimulus with minimal calibration.

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.3758/s13428-021-01653-y>.

Open Practices Statements The software tool suit is available with an MIT license via GitHub repositories. Data used to evaluate the precision of the tap extraction algorithm are available upon request. None of the experiments were pre-registered.

Links to open-source content:

- Detailed instructions, tutorials and discussions on setup assembly: https://github.com/m2march/tapping_setup
- Code for beats2audio utility: <https://github.com/m2march/beats2audio>
- Code for runAudioExperiment utility and examples for integration with PsychoPy, Psychtoolbox and OpenSesame: <https://github.com/m2march/runAudioExperiment>
- Code for rec2taps utility: <https://github.com/m2march/rec2taps>

Acknowledgements This work was carried out by the corresponding author under a PhD Scholarship provided by CONICET. No conflict of interests are declared.

Declarations

Conflict of Interests We have no known conflicts of interest to disclose.

References

- Audacity Team (2021). Audacity(R): Free audio editor and recorder [computer application]. Version 2.3.3 from <https://audacityteam.org/>.
- Babjack, D. L., Cernicky, B., Sobotka, A. J., Basler, L., Struthers, D., Kiscic, R., . . . , Zuccolotto, A. P. (2015). Reducing audio stimulus presentation latencies across studies, laboratories, and hardware and operating system configurations. *Behavior Research Methods*, 47(3), 649–665. <https://doi.org/10.3758/s13428-015-0608-x>
- Barry, R. J., De Blasio, F. M., & Borchard, J. P. (2014). Sequential processing in the equiprobable auditory Go/NoGo task: Children vs. adults. *Clinical Neurophysiology: Official Journal of the International Federation of Clinical Neurophysiology*, 125(10), 1995–2006. <https://doi.org/10.1016/j.clinph.2014.02.018>
- Bavassi, L., Kamienskowski, J. E., Sigman, M., & Laje, R. (2017a). Sensorimotor synchronization: Neurophysiological markers of the asynchrony in a finger-tapping task. *Psychological Research*, 81(1), 143–156.
- Bavassi, L., Kamienskowski, J. E., Sigman, M., & Laje, R. (2017b). Sensorimotor synchronization: Neurophysiological

- markers of the asynchrony in a finger-tapping task. *Psychological Research Psychologische Forschung*, 81(1), 143–156. <https://doi.org/10.1007/s00426-015-0721-6>
- Bridges, D., Pitiot, A., MacAskill, M. R., & Peirce, J. W. (2020). The timing mega-study: Comparing a range of experiment generators, both lab-based and online. *PeerJ*, 8, e9414. <https://doi.org/10.7717/peerj.9414>
- Care, D., Bianchi, B., Kamienskowski, J. E., & Ison, M. J. (2020). Face processing in free viewing visual search: An investigation using concurrent EEG and eye movement recordings. *Journal of Vision*, 20(11), 1691–1691. <https://doi.org/10.1167/jov.20.11.1691>
- Daikoku, T., Takahashi, Y., Tarumoto, N., & Yasuda, H. (2018). Motor reproduction of time interval depends on internal temporal cues in the brain: Sensorimotor imagery in rhythm. *Frontiers in Psychology*, 9, 1747. <https://doi.org/10.3389/fpsyg.2018.01873>
- Dimigen, O. (2020). Coregistration of eye movements and EEG in natural reading: Analyses and review. OSF. osf.io/cd9am.
- Dubé, L., & Le Bel, J. (2003). The content and structure of laypeople's concept of pleasure. *Cognition and Emotion*, 17(2), 263–295. <https://doi.org/10.1080/02699930302295>
- 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(1), 250–257. <https://doi.org/10.1016/j.jneumeth.2008.10.002>
- Elliott, M. T., Wing, A. M., & Welchman, A. E. (2014). Moving in time: Bayesian causal inference explains movement coordination to auditory beats. *Proceedings of the Royal Society B: Biological Sciences*, 281(1786), 20140751. <https://doi.org/10.1098/rspb.2014.0751>
- Falk, S., & Bella, S. D. (2016). It is better when expected: Aligning speech and motor rhythms enhances verbal processing. *Language, Cognition and Neuroscience*, 31(5), 699–708. <https://doi.org/10.1080/23273798.2016.1144892>
- Finney, S. A. (2001). FTAP: A Linux-based program for tapping and music experiments. *Behavior Research Methods Instruments & Computers*, 33, 65–72. <https://doi.org/10.3758/BF03195348>
- Finney, S. A. (2015). In defense of Linux, USB, and MIDI systems for sensorimotor experiments: A response to Schultz and van Vugt. Unpublished manuscript. Retrieved from <http://www.sfinney.com/images/pdfs/sf/finney2016a.pdf>.
- Fitch, W. T., & Rosenfeld, A. J. (2007). Perception and production of syncopated rhythms. *Music Perception: An Interdisciplinary Journal*, 25(1), 43–58. <https://doi.org/10.1525/mp.2007.25.1.43>
- Kleiner, M., Brainard, D., & Pelli, D. (2007). What's new in Psychtoolbox-3?.
- Knörrig, A., Wettach, R., & Cohen, J. (2009). Fritzing: A tool for advancing electronic prototyping for designers. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, (pp. 351–358). <https://doi.org/10.1145/1517664.1517735>.
- Krause, V., Pollok, B., & Schnitzler, A. (2010). Perception in action: The impact of sensory information on sensorimotor synchronization in musicians and non-musicians. *Acta Psychologica*, 133(1), 28–37. <https://doi.org/10.1016/j.actpsy.2009.08.003>
- Mathôt, S., Schreij, D., & Theeuwes, J. (2012). Opensesame: An open-source, graphical experiment builder for the social sciences. *Behavior Research Methods*, 44(2), 314–324. <https://doi.org/10.3758/s13428-011-0168-7>
- McAuley, J. D., Jones, M. R., Holub, S., Johnston, H. M., & Miller, N. S. (2006). The time of our lives: Life span development of timing and event tracking. *Journal of Experimental Psychology: General*, 135(3), 348–367. <https://doi.org/10.1037/0096-3445.135.3.348>
- McKinney, M. F., Moelants, D., Davies, M. E. P., & Klapuri, A. (2007). Evaluation of audio beat tracking and music tempo extraction algorithms. *Journal of New Music Research*, 36(1), 1–16. <https://doi.org/10.1080/09298210701653252>
- Miguel, M., Sigman, M., & Slezak, D. F. (2019). Tapping to your own beat: Experimental setup for exploring subjective tacti distribution and pulse clarity. OSF. osf.io/7sqaw. <https://doi.org/10.17605/OSF.IO/7SQAW>
- Noulhiane, M., Mella, N., Samson, S., Ragot, R., & Pouthas, V. (2007). How emotional auditory stimuli modulate time perception. *Emotion (Washington, D.C.)*, 7(4), 697–704. <https://doi.org/10.1037/1528-3542.7.4.697>
- Patel, A. D., Iversen, J. R., Chen, Y., & Repp, B. H. (2005). The influence of metricality and modality on synchronization with a beat. *Experimental Brain Research*, 163(2), 226–238. <https://doi.org/10.1007/s00221-004-2159-8>
- Peirce, J., Gray, J. R., Simpson, S., MacAskill, M., Höchenberger, R., Sogo, H., . . . , lindeløv, J. K. (2019). PsychoPy2: Experiments in behavior made easy. *Behavior Research Methods*, 51(1), 195–203. <https://doi.org/10.3758/s13428-018-01193-y>
- Plant, R. R., Hammond, N., & Turner, G. (2004). Self-validating presentation and response timing in cognitive paradigms: How and why? *Behavior Research Methods Instruments & Computers*, 36(2), 291–303. <https://doi.org/10.3758/BF03195575>
- Raymond, E. S. (2003). *The art of Unix programming. The art of Unix programming*. Boston: Addison-Wesley Professional.
- Repp, B. H. (2006). Musical synchronization. In Altenmüller, E., Wiesendanger, M., & Kesselring, J. (Eds.), *Music, Motor Control and the Brain*, (pp. 55–76): Oxford University Press, <https://doi.org/10.1093/acprof:oso/9780199298723.003.0004>.
- Repp, B. H., London, J., & Keller, P. E. (2005). Production and synchronization of uneven rhythms at fast tempi. *Music Perception: An Interdisciplinary Journal*, 23(1), 61–78. <https://doi.org/10.1525/mp.2005.23.1.61>
- Repp, B. H., & Su, Y. H. (2013). Sensorimotor synchronization: A review of recent research (2006–2012). *Psychonomic Bulletin & Review*, 20(3), 403–452. <https://doi.org/10.3758/s13423-012-0371-2>
- Schultz, B. G. (2019). The Schultz MIDI Benchmarking Toolbox for MIDI interfaces, percussion pads, and sound cards. *Behavior Research Methods*, 51(1), 204–234. <https://doi.org/10.3758/s13428-018-1042-7>
- Schultz, B. G., & Palmer, C. (2019). The roles of musical expertise and sensory feedback in beat keeping and joint action. *Psychological Research*, 83(3), 419–431. <https://doi.org/10.1007/s00426-019-01156-8>
- 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(4), 1591–1607. <https://doi.org/10.3758/s13428-015-0671-3>
- Segalowitz, S. J., & Graves, R. E. (1990). Suitability of the IBM XT, AT, and PS/2 keyboard, mouse, and game port as response devices in reaction time paradigms. *Behavior Research Methods Instruments & Computers*, 22(3), 283–289. <https://doi.org/10.3758/BF03209817>
- Shimizu, H. (2002). Measuring keyboard response delays by comparing keyboard and joystick inputs. *Behavior Research Methods Instruments & Computers*, 34(2), 250–256. <https://doi.org/10.3758/BF03195452>
- Snyder, J., & Krumhansl, C. L. (2001). Tapping to ragtime: Cues to pulse finding. *Music Perception: An Interdisciplinary Journal*, 18(4), 455–489. <https://doi.org/10.1525/mp.2001.18.4.455>
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., . . . , SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17, 261–272. <https://doi.org/10.1038/s41592-019-0686-2>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.