Novel Tools and Methods

Sharing an Open Stimulation System for Auditory EEG Experiments Using Python, Raspberry Pi, and HifiBerry

Alexandra Corneyllie, Fabien Perrin,* and Lizette Heine*

https://doi.org/10.1523/ENEURO.0524-20.2021

Auditory Cognition and Psychoacoustics Team, Lyon Neuroscience Research Center, CNRS-UMR 5292, Institut National de la Santé et de la Recherche Médicale U1028, Université Claude Bernard Lyon 1, 69675 Lyon, France

Abstract

In auditory behavioral and EEG experiments, the variability of stimulation solutions, for both software and hardware, adds unnecessary technical constraints. Currently, there is no easy to use, inexpensive, and shareable solution that could improve collaborations and data comparisons across different sites and contexts. This article outlines a system composed by a Raspberry Pi coupled with Python programming and associated with a HifiBerry sound card. We compare its sound performances with those of a wide variety of materials and configurations. This solution achieves the high timing accuracy and sound quality important in auditory cognition experiments, while being simple to use and open source. The present system shows high performances and results along with excellent feedback from users. It is inexpensive, easy to build, share, and improve on. Working with such low-cost, powerful, and collaborative hardware and software tools allows people to create their own specific, adapted, and shareable system that can be standardized across different collaborative sites, while being extremely simple and robust in use.

Key words: auditory stimulation; stimulus synchronization; EEG; low-cost I/O device; timing accuracy; OpenHardware; OpenScience; Raspberry Pi

Significance Statement

The present system shows the sustainability of using low-cost creation tools to solve recurrent constraints in neuroscience experiments. It contributes to the Open Science movement from hardware making to Open Source software and includes all the content necessary for the readers to build and develop their own system.

Introduction

Collaboration between centers is not always easy with regard to hardware and software compatibility. Currently, software for stimulation protocols in behavioral and electrophysiological assessments (e.g., having subjects hear a specific auditory stimuli) can be expensive and proprietary, and often requires technical support (whether open source or not). Even when using fixed software, there is still a wide variability because of computer hardware and installations. These are likely to vary from one laboratory to another because of differences in operating system, drivers, or updates between studies. Even small differences in terms of hardware or software might have repercussions, which are not always tested and reported in studies and can lead to important sources of error impacting the replication in studies (Plant and Quinlan, 2013). For example, hardware and software variations are likely to lead to deviations in the presentation and synchronization timing (i.e., onset and jitter). This will therefore influence analysis and results, particularly in research areas in which a strong temporal accuracy is required, as for

Received December 2, 2020; accepted July 16, 2021; First published July 22, 2021.

The authors declare no competing financial interests.

Author contributions: A.C., F.P., and L.H. designed research; A.C. performed research; A.C. analyzed data; A.C., F.P., and L.H. wrote the paper.

example in time-locked behavior or brain activity analysis (e.g., reaction time, evoked related potentials, phase coherence). The temporal accuracy of the analysis is directly dependent on the temporal accuracy of the stimulation system. It is necessary, for example, to have a jitter of <1 ms to be able to interpret the cortical responses with a temporal precision of ~ 10 ms.

A wide range of software is available to present stimulations to human participants in behavioral and electrophysiological (including EEG) experiments, such as famous programs like Presentation (https://www.neurobs.com/), OpenSesame (Mathôt et al., 2012), Eprime (Taylor and Marsh, 2017), Psychopy (Peirce, 2007), and Psychtoolbox (Borgo et al., 2012), the latter in MATLAB. All of the software allows for use of experimental control and management, resulting in good time performances for experiments. They vary both in terms of specialization of specific functionality and research area. However, all of the software is still dependent on computer hardware, drivers, and operating systems (OS), resulting in variability of configuration settings and installations (Plant et al., 2004; Garaizar et al., 2014).

In addition, the trigger information from the stimulation computer (or other devices, such as response boxes) to the acquisition computer (e.g., in EEG recording) is generally sent through the parallel port (PP; also named DB25 or LPT [line printing terminal] port, from the "D-Sub" connector family), which is reliable, easy to use, and has excellent timing accuracy (Stewart, 2006; Voss et al., 2007). Most conventional EEG systems have a PP as entry for triggers (e.g., BrainAmp, BrainProducts; EGI, Philips) or use similar single-bit technologies (37 pin D-Sub connector, BioSemi). However, the PP is nowadays mostly replaced by other ports such as the universal serial bus (USB; Canto et al., 2011). The lack of these PPs in newer hardware and associated support in recent OS and programming environments is a major problem as this often leads in practice to the use of old material without warranty or the newest updates, or inappropriate systems [e.g., PCMCIA (Personal Computer Memory Card International Association) parallel port adapters as cited in Psychology software tools information (https://support.pstnet.com/hc/en-us/articles/229359707-INFO-Recommended-Parallel-Port-Adapters-for-Machines-without-a-Parallel-Port-18031-); and risk of chipset, BIOS, or OS incompatibilities with extension card, as cited in the Brain Product press release (https://pressrelease.brainproducts.com/triggerbox-tips/)].

Another possible solution is to add an external PP hard PCI (peripheral component interconnect) card, which could be plugged to the motherboard of tower computers, but this does not solve the drivers and material variability. These solutions could lead to heavy technical work at every update and the need to maintain old versions of protocols, which is not efficient in terms of fragmenting the maintenance efforts for such systems.

In addition to these issues, the current stimulation systems are often not easily transportable and shareable. There is thus a strong need to use more standardized materials that are accessible (i.e., inexpensive, open source, easy to use, and easy to adapt for a specific context) as well as durable (e.g., choices in updating, tracking library versions, cross-platform), powerful, portable, shareable, and, in the best scenario, sustained by a community of users.

Several studies have made great strides in this direction and have shown that advances in computer portability through open-source technology can contribute to (neuro-)scientific studies. For example, Kuziek et al. (2017) showed that a Raspberry Pi2 can be used to present auditory (beeps) oddball paradigms. Another example of a Raspberry Pi use in multisite neuroscientific studies results from multisensory studies on rodent behavior (Buscher et al., 2020). Others have shown that an Arduino could be easily used for steady-state visual evoked potential (Mouli et al., 2015). However, these systems have been developed for very simple protocols and materials (beeps) and could be improved both in timing accuracy and audio quality. The need to develop a more generalized protocol player, a highly sound and timing efficient, shareable, and transportable system is part of a general move in research toward more open science. More and more possibilities are being created for sharing of data, analysis [e.g., Open Science Framework (https://osf.io)], and protocols such as the OpenBehavior Project (White et al., 2019).

Here we show that a Raspberry Pi 3, together with a HiFiBerry (additional sound card to Raspberry Pi), allows for high stereo sound quality and good timing performances (in terms of latency and jitter) while being powerful, easy to use, inexpensive, and in an open source ecosystem (software and hardware). We developed a Python library allowing the presentation of any auditory material for behavioral or electrophysiological studies, from simple to sophisticated paradigms. We also provided specifics for a state-of-the-art stimulation box for auditory experimental protocols. This box includes a parallel port for EEG synchronization of triggers, a Raspberry Pi 3, HiFiBerry, and battery. This portable system box can be used in fundamental and translational research settings. It can significantly reduce variation between studies and/or centers as well as simplify the setup of this kind of protocol. It has been developed in a multisite collaboration involving laboratory and clinical environments, thanks to the CogniComa project, and the box is actually used with four protocols in Lyon and Toulouse (e.g., mismatch negativity protocols, language and music protocols, math protocols).

Ultimately, we designed this stimulation system as a shareable project and encouraged its dissemination, use, and future improvements.

This work was supported by CogniComa Grant ANR-14-CE-15-0013.

*F.P. and L.H. are co-authors.

Acknowledgements: We thank Stein Silva, Fabrice Ferré, Edouard Naboulsi, and William Buffières at the Purpan University Hospital and Toulouse NeuroImaging Center (ToNIC) for participation in testing and using the first version of the box.

Correspondence should be addressed to Alexandra Corneyllie at alexandra.corneyllie@cnrs.fr.

https://doi.org/10.1523/ENEURO.0524-20.2021

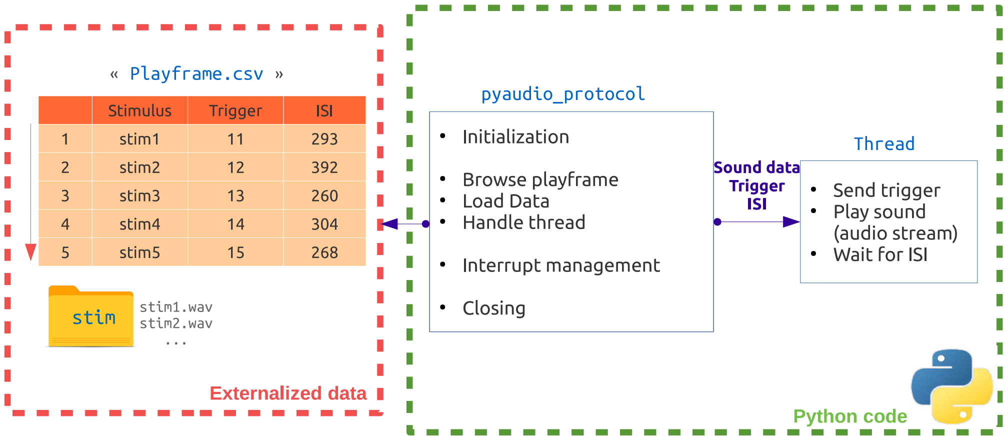


Figure 1. Playframe and software process. Left, An example of a playframe file consisting of columns for stimuli, associated triggers, and ISI (i.e., the time between stimuli). Right, The PyAudioProtocol core, which uses the playframe to send information to a dedicated thread. This thread plays the sound while sending the associated triggers.

Materials and Methods

We designed the system as follows: a "playframe" (simple organized comma-separated file document) carries the information about the protocol (ordered sounds to play and their associated timing and triggers). A Python library allows the playing process (either on a PC or in combination with the proposed hardware) and a new stimulation box based on a Raspberry Pi as hardware to propose a possible open standard.

Playframe

The playframe is the input of the reading process and consists of a simple table with the following three columns: the name of the .wav file to be played, the associated trigger value, and the associated timing (Fig. 1). The latter consists of time in milliseconds from the end of the current sound to the beginning of the next sound [i.e., the interstimulus interval (ISI)]. This table should be named "Playframe," saved as a ".csv" or ".xls" file, and constitutes a protocol playlist in a pandas (https://pandas.pydata.org/) dataframe style. It is protocol specific and generated independently of the player. An example of playframe generation with pandas is available in https://github.com/PyOpenProto/PyOpenProto/tree/e-neuro2021/examples/playframe_generation_example. The playframe could then be set on an external USB key, with relative audio stimuli and plugged in to the stimulation box to play the corresponding protocol. This separation between protocol-specific content (USB key) and general reading purpose (stimulation box) makes it possible to switch very easily between different protocols (or different subjects orders) without any change in the playing process, while maintaining the exact same system efficiency across protocols (or subjects). Associated auditory stimuli should be in standard stereo .wav audio format with a 44,100 Hz sampling rate in a "stim" dedicated folder.

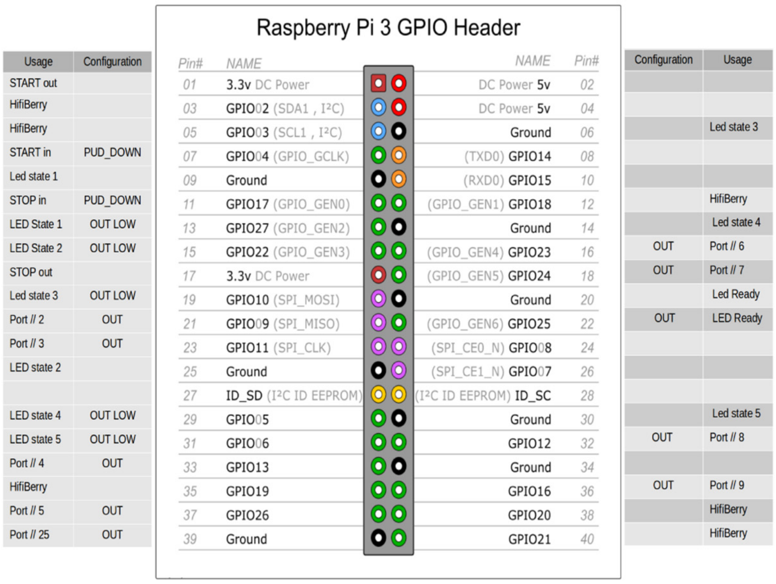
Software

The dissociation of protocol content and playing process allows the player implementation to focus on general core accuracy in timing and sound quality while bypassing specific ordering complexity (Fig. 1). The use of such external session list imports is generally provided in most presentation software.

The Python programming language for the playing process was the obvious coding solution. It allows for interoperability (coupled with a wide range of available powerful libraries), readability, efficiency, and is associated with a large collaborative community (Muller et al., 2015). The *Python in neurosciences* ecosystem provides an excellent unifying solution addressing the needs of both scientists and engineers, from data acquisition to treatment and analysis.

We designed a Python library named PyAudioProtocol, which is a part of the more general project 'PyOpenProto' available on GitHub (https://github.com/PyOpenProto/PyOpenProto/tree/e-neuro2021) PyAudioProtocol relates to pure audio stimuli presentation coupled with parallel port triggering, subject to strong constraints in sound quality and timing accuracy. This library hosts two versions for auditory stimulus presentation. One version is intended for classical computers ('core_gui.py' file), which expects the presence of a parallel port to send the triggers to the EEG acquisition system. The second version allows the use of the Raspberry Pi3-based hardware described in the next section ('core_rpi_nogui.py' file). This version simulates a parallel output via GPIO (general purpose input/output). Both versions were tested, compared, and used, as described in the Results section and follow the same simple code architecture described below.

The PyaudioProtocol reads the playframe (Fig. 1) and loads the data, after which it initializes a thread using the python-sounddevice (https://python-sounddevice.readthedocs.io) module, which provides the bindings to the PortAudio (http://www.portaudio.com/) library to benefit from precise audio-streaming management. At the same time, a trigger is sent using pyparralel (https://github.com/pyserial/pyparallel) or Rpi.GPIO (https://pypi.org/project/RPi.GPIO/) from PC or Raspberry Pi3 versions, respectively. The next trigger and sound will be sent to the output device after the given ISI. In parallel to this, interruption management allows the protocol



Raspberry Pi 3 GPIO Header

Usage	Configuration	Pin#	NAME	NAME	Pin#	Configuration	Usage
START out		01	3.3v DC Power	DC Power 5v	02		
HifiBerry		03	GPIO02 (SDA1 , I²C)	DC Power 5v	04		
HifiBerry		05	GPIO03 (SCL1 , I²C)	Ground	06		Led state 3
START in	PUD_DOWN	07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08		
Led state 1		09	Ground	(RXD0) GPIO15	10		
STOP in	PUD_DOWN	11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12		HifiBerry
LED State 1	OUT LOW	13	GPIO27 (GPIO_GEN2)	Ground	14		Led state 4
LED State 2	OUT LOW	15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16	OUT	Port // 6
STOP out		17	3.3v DC Power	(GPIO_GEN5) GPIO24	18	OUT	Port // 7
Led state 3	OUT LOW	19	GPIO10 (SPI_MOSI)	Ground	20		Led Ready
Port // 2	OUT	21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22	OUT	LED Ready
Port // 3	OUT	23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24		
LED state 2		25	Ground	(SPI_CE1_N) GPIO07	26		
		27	ID_SD (I²C ID EEPROM)	(I²C ID EEPROM) ID_SC	28		
LED state 4	OUT LOW	29	GPIO05	Ground	30		Led state 5
LED state 5	OUT LOW	31	GPIO06	GPIO12	32	OUT	Port // 8
Port // 4	OUT	33	GPIO13	Ground	34		
HifiBerry		35	GPIO19	GPIO16	36	OUT	Port // 9
Port // 5	OUT	37	GPIO26	GPIO20	38		HifiBerry
Port // 25	OUT	39	Ground	GPIO21	40		HifiBerry

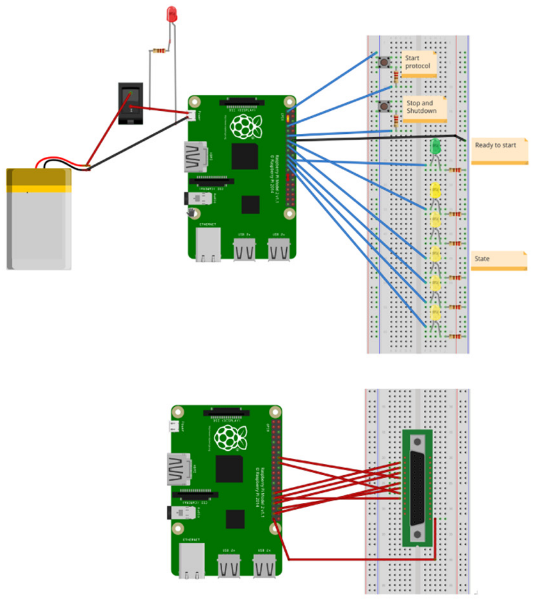


Figure 2. GPIO usage. GPIO pins (3, 5, 35, 38, and 40) are used by the HiFiBerry (not represented in the right diagram); GPIO OUT pins (16, 18, 21, 23, 32, 33, 36, 37, and 39) for parallel port communication (giving $2^8 = 256$ possibilities for markers values); GPIO PUD_DOWN pins (7 and 11) are used for start stop buttons; and GPIO OUT LOW pins (13, 15, 19, 29, and 31) for LED lights indicating the protocol progression to the user.

to be stopped at any given time. Finally, PyaudioProtocol allows the correct initialization and closing down of material and programming objects used.

Hardware

The hardware can be a standard computer with a parallel port (internal or external, PCI card recommended in this case), to be used with the computer code version available on GitHub (https://github.com/PyOpenProto/PyOpenProto/blob/e-neuro2021/pyaudio_protocol/core_gui.py).

A standardized, portable, and accessible system using a Raspberry Pi is also proposed. The Raspberry Pi3 B+ board was used, which natively has a low sound quality and a variable trigger-tone latency, as previously described by Kuziek et al. (2017). To overcome those limitations, the HiFiBerry DAC+ pro [digital analog converter (DAC)] was used as an additional sound card. It has a dedicated 192 kHz/24 bit high-quality Burr Brown DAC coupled to an ultra-low-noise voltage regulator for best sound quality and a dual-domain clock circuit to produce low-jitter performance. [THD+N (total harmonic distortion plus noise): −92 dB, signal-to-noise ratio = 112 dB]. This card, with its low-jitter clock generator and optimal audio performances, ensures access to most of the Raspberry Pi GPIO that could be used for other purposes.

The Raspberry Pi3 itself uses the Raspbian OS (https://www.raspberrypi.org/software/operating-systems/). This open source system is installed on a classic secure digital card and is easily replicable. A small number of specific configurations was adapted. Precisely, we included the HiFiBerry audio output, activated SSH (secure shell protocol for cryptographic networking) and automatic logging. In addition, we used an ARM I2C (microprocessor interintegrated circuit) interface and allowed for an automatic run of PyAudioProtocol starting script when the system switches on. A ready-to-use operating system with all those configurations is available for download on https://osf.io/3muqk/.

We used the GPIO to simulate a parallel port as described in Figure 2. It was further used to manually start and stop the systems (using buttons) and to allow visual feedback on protocol progression using LEDs.

A battery (10,000 milliampere hours; Solo 5, ROMOSS) was used to supply the system with energy while avoiding a connection to a general power outlet. An analog stereo RCA adapter to jack 3.5 female was added for audio output. One USB port was dedicated to a USB key, which allows the deliverance of one playframe and multiple audio files for one participant and one test. Figure 3 shows the finalized box. The overall material cost was <300 €.

Testing procedure methods

We measured the latencies and jitters (i.e., latency variability) between audio stimuli and associated triggers for 12 configurations (see below). To realize accurate measurements while avoiding additional timing treatment, we used a direct simple cable physically joining the jack and the parallel outputs from the stimulation system (computer or stimulation box) to a two-paired jack input of a recording computer, as described in Figure 4.

The test (Fig. 4) consisted of playing a 1000 Hz sound a thousand times (sample frequency = 44,100 Hz, duration = 200 ms) coupled with a triggering marker (value = 255). Output from the jack and parallel port were recorded at the exact same moment. A simple threshold (moment where signal is >0) was then used on the recording to detect the exact moment of the sound onset. This was compared with the timing of the

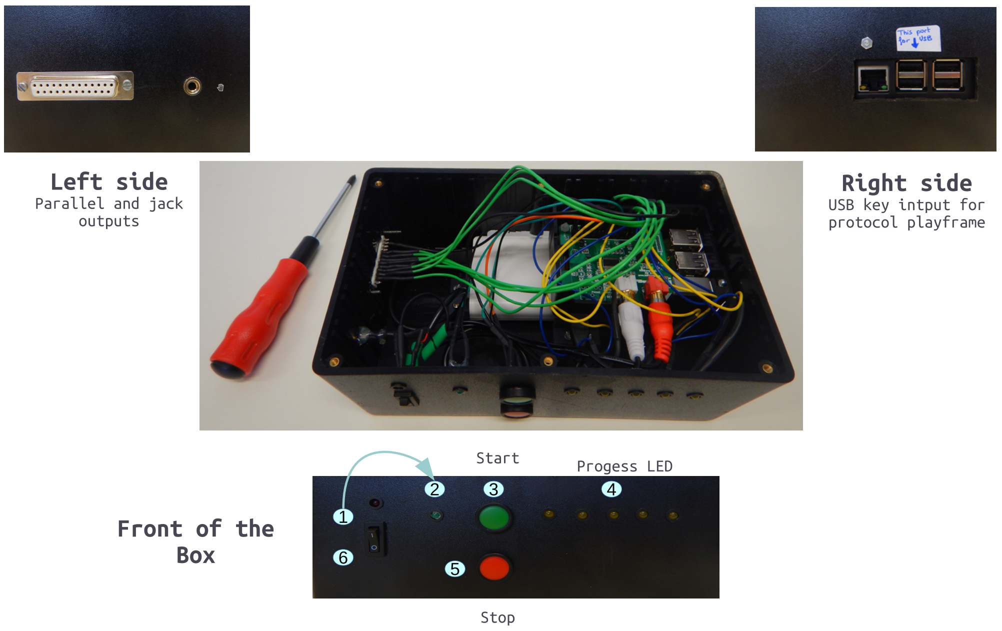

Left side
Parallel and jack outputs
Right side
USB key intput for protocol playframe
Front of the Box
Start
Progess LED
Stop
1
2
3
4
5
6

Figure 3. Stimulation box. Top, Output via a parallel port for triggers and auditory jack for sounds (left side), and input via USB keys for the auditory files and playframe (right side). Middle, Top view of the box without its lid. Bottom, Setup procedure (front of the box): (1) switch on the box; (2) wait for the green LED to indicate complete startup; (3) press start to start protocol reading; (4) yellow LEDs indicate progress; (5) end of protocol or urgency stop button; (6) turn box off.

trigger occurrence. The difference between each sound onset and the corresponding trigger onset was kept as a latency value, whose distribution across repetitions gives an idea of the jitter (latency standard deviation SD) and its variability.

Comparison with other systems

To compare these measures with other systems, it is very important to keep in mind that sound performances depend on the wide variety of materials and configurations. First, performance is based on the capacity of the used sound card (e.g., sound fidelity, low jitter performances) and how the sound card interacts with the computer. Indeed, the OS (Windows, Mac, Linux), the selected driver, the software (e.g., Presentation, MATLAB, Opensesame, Psychopy) used and the version used for each one will influence sound control performance.

Moreover, hardware and software interactions are based on an API [application programming interface (also called the "back-end")] which defines how to manipulate

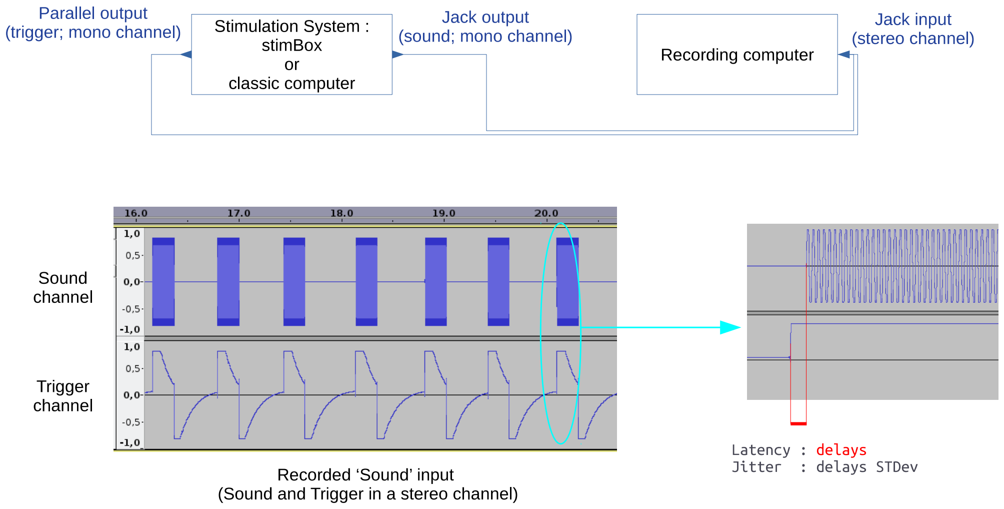


Figure 4. Latency and jitter testing procedure. A recording computer is used to receive and record both outputs of the tested system into one input. One channel is used to record the audio channels as well as the triggers. The difference in onset between the trigger channel and the sound channel are compared. The delay between the two constitutes the latency, while the variance in this latency establishes the jitter (SD).

Name and resume	Material	Operating System	Software (or running environment)	Configuration (and/or library) used to play sounds	Time performances (mean values, in ms)
Stimulation Box New software (PyAudioProtocol) and hardware	Raspbery Pi Hifi Berry Sound Card GPIO for LPT port	Raspbian (Linux adapted for RaspBerry Pi)	Python 3.8	SoundDevice (PyAudioProtocol sound library)	latency : 1.4 jitter : 0.03
Dell390_L16_OS_ sounddevice Old PC on Linux with: - native parallel port - external sound card	Dell Precision 390 Native LPT port PCI sound card X-Fi Sound Blaster	Linux 16.04 LTS	Python (2.7) OpenSesame with PyAudioProtocol core	SoundDevice	latency : 2.06 jitter : 0.10
Dell390_L16_OS_ legBuff128 Old PC on Linux with: - native parallel port - external sound card	Dell Precision 390 Native LPT port PCI sound card X-Fi Sound Blaster	Linux 16.04 LTS	Python (2.7) OpenSesame	Legacy back-end sound buffer size = 128	latency : 8.52 jitter : 1.24
Dell390_WXP_Pres_ DirectX Old PC on Windows with: - native parallel port - external sound card	Dell Precision 390 Native LPT port PCI sound card X-Fi Sound Blaster	Windows XP	Presentation® v5.12	Buffer: DirectX	latency : 11.26 jitter : 0.04
Dell3630_W10_Pres_ exclusiv Recent PC on Windows with: - external parallel port	Dell Precision 3630 PCI card for LPT port Native sound card Realteck	Windows 10	Presentation® v22	Buffer : exclusive mode	latency : 1.56 jitter : 0.011
Dell3630_W10_Pres_ DirectX Recent PC on Windows with: - external parallel port	Dell Precision 3630 PCI card for LPT port Native sound card Realteck	Windows 10	Presentation® v22	Buffer : DirectX	latency : 49.98 jitter : 0.63
Dell3630_W10_Pres_ Exclusiv_XFi Recent PC on Windows with: - external parallel port - external sound card	Dell Precision 3630 PCI card for LPT port PCI sound card X-Fi Sound Blaster	Windows 10	Presentation® v22	Buffer : exclusive mode	latency : 11.78 jitter : 0.01
Dell3630_L16_OS_ legBuff128 Recent PC on Linux with: - external parallel port	Dell Precision 3630 PCI card for LPT port Native sound card Realteck	Linux 16.04 LTS	Python 2.7 Opensesame	Legacy back-end sound buffer size = 128	latency : 9.52 jitter :1.01
Dell3630_L16_OS_ psychBuff128 Recent PC on Linux with: - external parallel port	Dell Precision 3630 PCI card for LPT port Native sound card Realteck	Linux 16.04 LTS	Python 2.7 Opensesame	Psychopy back-end sound buffer size = 128	latency : 9.47 jitter :1.08
Dell3630_L16_OS_ sounddevice Recent PC on Linux with: - external parallel port	Dell Precision 3630 PCI card for LPT port Native sound card Realteck	Linux 16.04 LTS	Python 2.7 OpenSesame with PyAudioProtocol core	SoundDevice ALSA sysdefault audio mixer	latency : 14.07 jitter : 1.07
Dell3630_L20_OS_ legBuff128 Recent PC on Linux with: - external parallel port	Dell Precision 3630 PCI card for LPT port Native sound card Realteck	Linux 20.04 LTS	Python 3.8 Opensesame	Legacy back-end sound buffer size = 128	latency : 60.50 jitter : 8.24
Dell3630_L20_OS_ sounddevice_XFi Recent PC on Linux with: - external parallel port - external sound card	Dell Precision 3630 PCI card for LPT port PCI sound card X-Fi Sound Blaster	Linux 20.04 LTS	Python 3.8 Opensesame with PyAudioProtocol core	SoundDevice ALSA sysdefault audio mixer	latency : 0.42 jitter : 0.03

Figure 5. Detailed configurations tested to compare the stimulation box results. It illustrates the latency and jitter variability across different common configurations and variants. Configurations with poor audio quality have been removed.

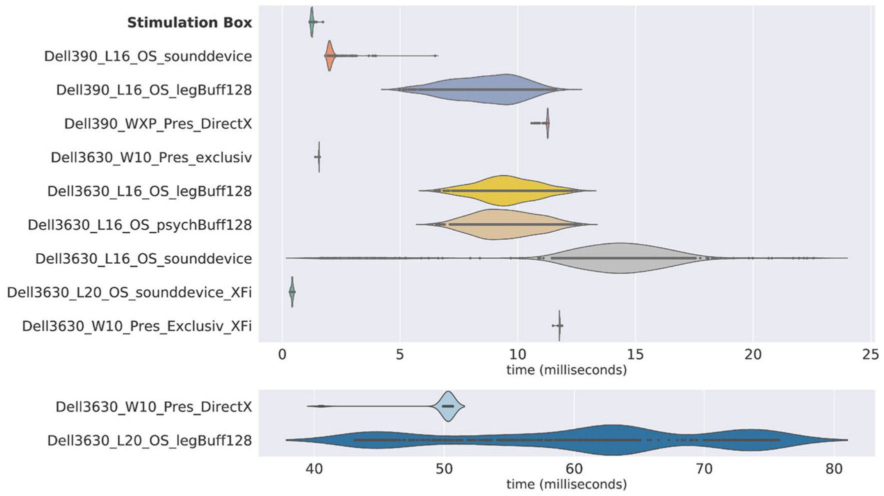


Figure 6. Latencies and jitter results. Results of the stimulation box latency testing are compared with different methods using different computers, operating systems, software, and parameters. Violin plots represent the distribution of the latencies for each of the 1000 sounds and triggers, indicated by points.

sound data and pilot the sound card. There is a wide range of APIs available within each OS (Windows: MME, DirectSound, WASAPI, ASIO; Linux: ALSA, ALSIHPI, JACK; Mac OS X: Core Audio, JACK). Finally, some parameters could be available for a given hardware/software solution (e.g., sound buffer size, reading mode) that could also have an impact on latency performance.

This exponential variability of possibilities [material * driver * OS * API * software * version * configuration] makes overall benchmark results impossible and shows the necessity of testing a setup before running scientific studies that have strong sound constraints.

For these reasons, we compare our system to some common configurations available to the authors, as detailed in Figure 5. Note that those tests are not covering all the possible solutions but illustrate the variability of the material, OS, software, or version result. Figure 6 in the Results section plots the latency distribution for each configuration, based on the names, as proposed in the first column.

Data availability

The code described in this article is freely available online at https://github.com/PyOpenProto/PyOpenProto/tree/e-neuro2021/pyaudio_protocol or as the Extended Data 1.

Results

Figure 6 shows the latency and jitter test results for the 12 tested configurations. Average latencies range from 0.42 to 60 ms, and average jitter ranges between 0.01 and 8.24 ms in our tested setups. Different solutions showed good latency performance, but the results were not consistent in the case of OS or software version changes. The best tested setups were the Opensesame software running the pyaudioprotocol codes on Linux 20.04 with an external sound card on a recent computer (jitter, 0.42 ms; latency, 0.03 ms) and the stimulation box itself (jitter, 1.4 ms; latency, 0.03 ms). Another good scoring setup was shown to be the Presentation software with exclusive mode on Windows 10 in a recent computer (latency, 1.56 ms; jitter, 0.011 ms).

The same testing protocol was used with longer and variable duration audio stimuli (names and sentences) with very similar results. Further tests with the stimulation box were performed using moving interstimulus intervals. This did not change results either.

Creation of your own box

All software, as well as a user manual, can be found on GitHub (https://github.com/PyOpenProto/PyOpenProto) under a BSD (Berkeley Software Distribution) license (CNRS CeCill B). A detailed description of the fabrication of a stimulation box can be found on Hackaday (https://hackaday.io/project/181042-stimbox), containing a step by step guide to create the box. A ready-to-use Raspbian operating secure digital card system is available on OSF (https://osf.io/3muqk/). For those who do not wish to create one themselves, you can contact us so that we may redirect you to our collaborators. Furthermore, a 3D version of the stimulation box container for 3D printing is available on Thingiverse (https://www.thingiverse.com/thing:4592271) or as the Extended Data 2.

Discussion

This article describes a simple, inexpensive, and open source stimulation system for EEG auditory experiments. It combines hardware such as a Raspberry Pi3 and a HiFiBerry audio card with software for stimulus presentation. The software can be used together with the hardware or as a stand-alone software on a common computer. Both auditory stimulation and trigger timing are optimized when the hardware and software stimulation systems are combined. All information about the Python library, the hardware system, and the configuration settings are open source.

The stimulation box combined with PyAudioProtocol software showed short latency and limited jitters, compared with the 11 other methods. In fact, the results of the stimulation box are among the best for a significantly lower cost. Even if the tests performed were not entirely exhaustive and included only a selection of known configurations used in our research network, they underline the huge variability of possible configuration and associated timing performances. This clearly shows the need for proper specification in research publications and for the standardization of materials through multisite collaborations.

The addition of the HiFiBerry has broadened the range of studies that can be performed using a Raspberry Pi. Previous work from Kuziek et al. (2017) and Mouli et al. (2015) showed that open source stimulation systems could be used to present simple stimulation (beeps). Our solution extends their work by allowing all types of (personalized) audio protocols with good sound quality. In addition, such mobile solutions allow for easy combination with available EEG systems, both classic laboratory-based options as well as the newer open source and mobile EEG options (Pietto et al., 2018; Reiser et al., 2019). Together, this might significantly increase the study possibilities, both in terms of research capacity in smaller laboratories and for experiments outside the laboratory. For example, open-source stimulation and acquisition have already shown their utility in rodents (Mukherjee et al., 2017), and EEG-RaspberryPy2 systems have shown brain–computer interface possibilities (Szewczyk et al., 2020). The open software combined with the accessible and open hardware proposed by our current setup helps to solve some of the challenges that exist concerning standardization for mobile EEG technologies (Lau-Zhu et al., 2019). Furthermore, these new technologies open up a whole field of modular solutions that can simplify and customize studies while being easier to maintain as well as more resilient.

Our system goes back to basics and dissociates the core player from the scheduling intelligence. All necessary ordering of auditory stimulation, from basic to very sophisticated, are prepared in advance in an easy-to-read format (a playframe table, equivalent to a session list file). This limits presentation errors at the study level and facilitates stimulation verification, both before recording and during data analysis. Most importantly, having one USB key per protocol makes experiment setup very quick. Switching from one project and/or participant to another requires only a switch of USB keys and a press of the "start" button.

Currently, the system supports only auditory experiments. Next, improvement will allow the use of a screen to indicate protocol progress instead of the current LED lights. Moreover, a future version will allow the possibility for visual and auditory-visual experiments, as well as adaptive protocols where subject responses are recorded by the system and are used to change the stimulation order. Such improvement will not change the actual audio performances, as they will be handled by different threads or processes. Furthermore, simple adaptations can be made to allow trigger output other than the parallel port. We have also currently added a battery for power to improve mobility and limit connections between the participant and the electrical power lines. However, this could be changed to allow additional external devices. This stimulation system was designed as a shareable project, and we encourage its dissemination, adaptation, use, and further development.

In conclusion, we aimed at creating a user-friendly auditory stimulation system to simplify the creation and implementation of future studies. The portable and easy-to-use stimulation box allows sustainable comparisons between studies and/or centers, which is helpful for reproducibility and collaboration.

Thanks to the development of inexpensive creation tools (e.g., Raspberry Pi, 3D Printing, Arduino), open source languages, like R and Python, and the simultaneous movement toward open science, neuroscience research is able to reclaim transparency, mastery of tools, and excellent practices in our work.

References

Borgo M, Soranzo A, Grassi M (2012) Psychtoolbox: sound, keyboard and mouse. In: MATLAB for psychologists (Borgo M, Soranzo A, Grassi M, eds), pp 249–273. New York: Springer.

Buscher N, Ojeda A, Francoeur M, Hulyalkar S, Claros C, Tang T, Terry A, Gupta A, Fakhraei L, Ramanathan DS (2020) Open-source Raspberry Pi-based operant box for translational behavioral testing in rodents. J Neurosci Methods 342:108761.

Canto R, Bufalari I, D'Ausilio A (2011) A convenient and accurate parallel input/output USB device for E-Prime. Behav Res Methods 43:292–296.

Garaizar P, Vadillo MA, López-de-Ipiña D, Matute H (2014) Measuring software timing errors in the presentation of visual stimuli in cognitive neuroscience experiments. PLoS One 9:e85108.

Kuziek JWP, Shienh A, Mathewson KE (2017) Transitioning EEG experiments away from the laboratory using a Raspberry Pi 2. J Neurosci Methods 277:75–82.

Lau-Zhu A, Lau MPH, McLoughlin G (2019) Mobile EEG in research on neurodevelopmental disorders: opportunities and challenges. Dev Cogn Neurosci 36:100635.

Mathôt S, Schreij D, Theeuwes J (2012) OpenSesame: an open-source, graphical experiment builder for the social sciences. Behav Res Methods 44:314–324.

Mouli S, Palaniappan R, Sillitoe IP (2015) pp241–269. A Configurable, inexpensive, portable, multi-channel, multi-frequency, multi-chromatic RGB LED System for SSVEP stimulation. In: Brain-computer interfaces (Hassanien AE, Azar AT, eds). Cham, Switzerland: Springer International.

Mukherjee N, Wachutka J, Katz D (2017) Python meets systems neuroscience: affordable, scalable and open-source electrophysiology in awake, behaving rodents.

Muller E, Bednar JA, Diesmann M, Gewaltig M-O, Hines M, Davison AP (2015) Python in neuroscience. Front Neuroinform 9:11.

Peirce JW (2007) PsychoPy—Psychophysics software in Python. J Neurosci Methods 162:8–13.

Pietto ML, Gatti M, Raimondo F, Lipina SJ, Kamienkowski JE (2018) Electrophysiological approaches in the study of cognitive development outside the lab. PLoS One 13:e0206983.

Plant RR, Quinlan PT (2013) Could millisecond timing errors in commonly used equipment be a cause of replication failure in some neuroscience studies? Cogn Affect Behav Neurosci 13:598–614.

Plant RR, Hammond N, Turner G (2004) Self-validating presentation and response timing in cognitive paradigms: how and why? Behav Res Methods Instrum Comput 36:291–303.

Reiser JE, Wascher E, Arnau S (2019) Recording mobile EEG in an outdoor environment reveals cognitive-motor interference dependent on movement complexity. Sci Rep 9:13086.

Stewart N (2006) A PC parallel port button box provides millisecond response time accuracy under Linux. Behav Res Methods 38:170–173.

Szewczyk R, Zieliński C, Kaliczyńska M (2020) Automation 2020: towards industry of the future. Proceedings of Automation 2020, March 18–20, 2020, Warsaw, advances in intelligent systems and computing. Cham, Switzerland: Springer International.

Taylor PJ, Marsh JE (2017) E-Prime (software) In: The international encyclopedia of communication research methods. Atlanta, GA: American Cancer Society.

Voss A, Leonhart R, Stahl C (2007) How to make your own response boxes: a step-by-step guide for the construction of reliable and inexpensive parallel-port response pads from computer mice. Behav Res Methods 39:797–801.

White SR, Amarante LM, Kravitz AV, Laubach M (2019) The future is open: open-source tools for behavioral neuroscience research. eNeuro 6:ENEURO.0223-19.2019.