



Silicon Spike: An Arduino-based low-cost and open-access triggerbox to precisely control TMS devices

Giuseppe Ippolito^{1,2} · Thomas Quettier¹ · Sara Borgomaneri¹ · Vincenzo Romei^{1,3}

Accepted: 5 March 2025 / Published online: 15 April 2025
© The Author(s) 2025

Abstract

Transcranial magnetic stimulation (TMS) is a widely used tool in the field of clinical and cognitive neuroscience. To exploit its excellent temporal properties, TMS usually relies on triggerbox devices, which temporize the delivery of magnetic pulses according to the paradigm requirements. However, a main limitation of most of the widely used triggerbox devices is that they rely solely on the experimental computer processor, which might add temporal uncertainty in delivering the TMS pulse when the computer's resources are drained by other experimental devices or by task execution itself, especially during repetitive TMS or dual-coil protocols. We aimed at developing a low-cost and easily reproducible triggerbox device which could overcome these limitations by relying on an external processor to handle the timing precision. We used an Arduino Uno R4 Minima to build Silicon Spike, a low-cost (\$60) triggerbox device. We tested the device's precision in delivering the TMS pulses under different working load conditions, and the impact over time. All of the tests were ecological, delivering real TMS pulses during dual-coil, repetitive, and patterned TMS protocols. We obtained extremely high precision (<0.022 ms) in all of the tests. This means that, for smaller or longer latencies, the error remains negligible for TMS studies. Thus, the Silicon Spike device demonstrated microsecond precision in handling the TMS pulse delivery, establishing itself as a simple and yet precise device. We freely provide the source code and the hardware schematics, allowing anyone to reproduce our work.

Keywords TMS · rTMS · ccPAS · Triggerbox · Arduino · Open source

Introduction

Studying the “when” of neurophysiological processes implies a fine-grained time resolution of the investigation tools employed. Indeed, most cognitive events take place in a matter of milliseconds; thus, a high level of precision is necessary when presenting stimuli or harnessing experimental manipulations. In order to accomplish this, a vast range of software has been developed (e.g., MATLAB Psychtoolbox, E-Prime, PsychoPy, OpenSesame). Most of these

programs allow one to create and present the task stimuli according to the experimental hypothesis with high timing precision while recording the participant's response. However, as usually happens in neurophysiological research, several other devices are typically involved for recording or manipulating the participant's behavior or other physiological variables. Consequently, their proper, coordinated handling is crucial when building a new task, in order to use them without harming the task precision. This is the case with transcranial magnetic stimulation (TMS), in which a brief magnetic pulse is delivered to the participant's scalp to noninvasively modulate their brain functioning (Hallett, 2007). This approach requires close attention to the code structure in order to trigger the device without causing any delay in the experimental task. This is particularly relevant in paired-pulse or dual-coil protocols (e.g., cortico–cortical paired associative stimulation [ccPAS]), or rhythmic and repetitive TMS (rTMS) protocols, in which multiple TMS pulses are delivered to modulate the brain's activity, usually with extremely short intervals—even under 10 ms (Hernandez-Pavon et al., 2023; Di Luzio et al., 2024; Tarasi et al.,

✉ Vincenzo Romei
vincenzo.romei@unibo.it

¹ Center for Studies and Research in Cognitive Neuroscience, Department of Psychology “Renzo Canestrari”, Cesena Campus, Alma Mater Studiorum, Università Di Bologna, 47521 Cesena, Italy

² Laboratory of Cognitive Neuroscience, Department of Languages and Literatures, Communication, Education and Society, University of Udine, Udine, Italy

³ Universidad Antonio de Nebrija, Madrid, Spain

2024; Ippolito et al., 2022; Klomjai et al., 2015; Trajkovic et al., 2022; 2024a, b)—or concomitantly with the stimulus onset (Bertaccini et al., 2023).

In most cases, the communication between TMS devices and the experimental computer is handled through a triggerbox device, an electronic apparatus that allows the TMS to be triggered when a specific prompt is run. However, triggering TMS pulses using the same computer engaged in the task execution—which might involve the presentation of auditory, somatosensory, and/or visual stimuli, response collection, or even communication between several other experimental devices (Borgomaneri et al., 2023; Fotia et al., 2021; Trajkovic et al., 2022)—might result in a delay in the pulse emission. This is particularly relevant for dual-coil TMS and rTMS protocols, in which the sum of small latencies might exceed the computer's resources, leading to an imprecise calculation of distances between the TMS pulses and thus impacting the experimental results. This risk can be avoided by allocating this kind of computation to an external processor, such as that of the triggerbox, allowing for finer timing implementation. Still, most triggerbox devices rely solely on the computer resources. Moreover, the principle behind the triggerbox devices is quite simple, from both a hardware and software point of view. Consequently, it is common to come across handmade reproductions using simple components. Specifically within the electronic and programming community, Arduino products (see: [https://](https://www.arduino.cc/)

www.arduino.cc/) have been distinguished for their accessible price, ease of use, and completely open-access availability (Ismailov & Jo, 2022; Kondaveeti et al., 2021). This has led to a flourishing market and numerous projects shared by enthusiasts and professionals, facilitating their replication across different labs (White et al., 2019). Nonetheless, tests conducted on Arduino's latencies during different tasks with variable code structure and workload (D'Ausilio, 2012) have demonstrated how these cheap machines can provide reliable performance and extremely accurate timing, making them suitable for the lab setting. For this reason, they have been implemented in a multitude of experiments including real-time wireless electrocardiogram (ECG) (Güvenç, 2020), light-emitting diode (LED) stimulators for visual research (Teikari et al., 2012), or even drowsiness sensors based on EEG signals (Mindoro, 2020).

Given this versatility and reliability, it is common to find an Arduino-based triggerbox device in several labs employing TMS. This allows them to run several experiments simultaneously without the need to buy commercially available triggerbox devices, whose cost is estimated to be on average around €1000 each. Nevertheless, custom-made triggerbox devices generally share the same limitations as commercially available systems: they rely on the computer's processor to handle timings (Fig. 1). In other words, they are used merely as a passive transfer to transform a serial input from the experimental computer into a square wave (i.e., transistor–transistor

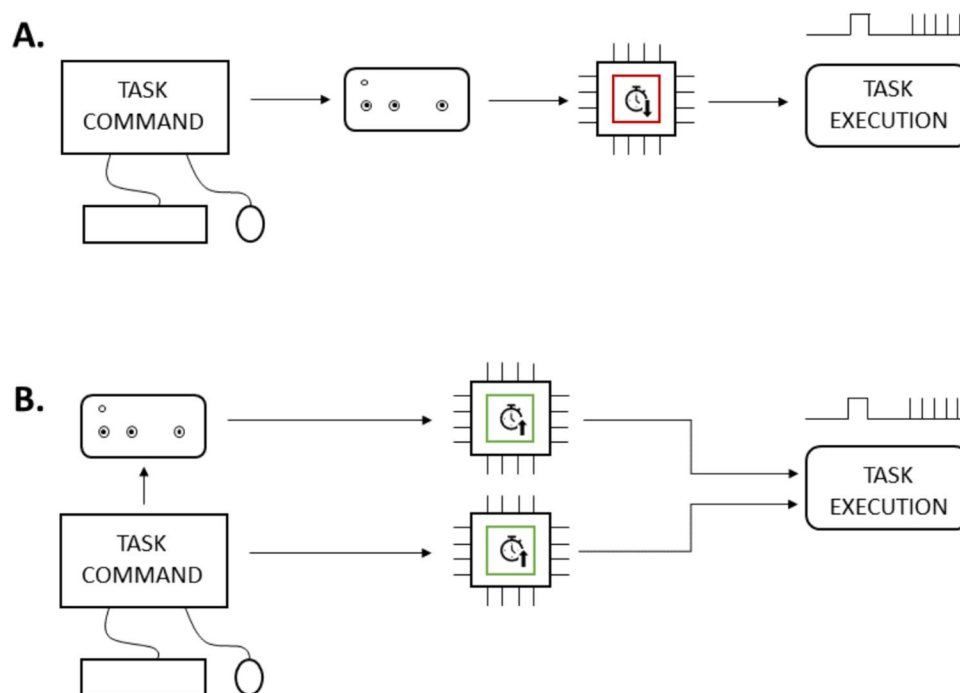


Fig. 1 Visual representation of the difference between a single- (A) and multi-processor (B) approach. Using the former, both the task and TMS timings rely on the computer's processor, potentially adding

latencies. The latter protocol instead leaves the TMS timings to the triggerbox, allowing the computer to allocate its resources solely to the experimental task

logic [TTL]) to trigger the TMS pulses. As stated earlier, such an approach only allows for establishing communication between the experimental computer and the TMS, thus allocating the resources for handling the TMS trigger to the computer itself rather than to the triggerbox. In other words, the triggering precision might be reduced during high-demand protocols (Joao et al., 2012), such as rTMS/dual-coil or even continuous (cTBS) or intermittent (iTBS) theta-burst protocols. Additionally, a few software programs, such as PsychoPy (Peirce et al., 2019), use the screen refresh rate as a standard unit to handle timings. This means that the interval between two actions must necessarily be a multiple of the refresh rate. Considering that most lab computers run at a frequency of 100 Hz—or, occasionally, at 60 Hz—only one action can be performed every 10 ms—or 16.67 ms—making it impossible to stimulate during ccPAS/rTMS protocols with an inter-pulse interval different from the refresh rate multiples.

An alternative approach, as we propose here, involves using the Arduino processor to handle two pulses and the distance between them, given a simple computer prompt. This enables totally independent control of TMS timing, lightening the computer resources. This should ease the task execution and the triggering rate precision. When possible, this approach is preferable (Krauss, 2020).

However, this approach usually requires programming of the Arduino's motherboard for each of the stimulation parameters according to the task demands. This implies that programming skills are required, and that the continuous updating of the code might be more prone to error. This is particularly worrisome when several users harness the lab settings or when numerous experimental setups are needed in the lab, meaning that the device needs on-demand coding updates, even several times a day. Alternatively, it is possible to write more complex and flexible codes which reduce the number of times the script needs to be modified. But this also implies that more exhaustive

tests on its precision are necessary. To the best of our knowledge, there has been no publication to date in which a TMS triggerbox was built using this specific approach.

In order to overcome these limitations, given the higher precision of the multiprocessor versus single-processor approach, we developed a triggerbox device which relies on an additional processor to easily and reliably handle TMS pulse delivery (Fig. 2). Here we test the device's precision in delivering TMS pulses during single-pulse (spTMS), dual-coil, and rTMS protocols, each under different load conditions. The final product allows the user to implement our solution even with little or no programming knowledge. Thus, the aim of the present study is to introduce the Silicon Spike device and working principles to the community of TMS users, to enable the reproduction of the circuit and free use of the device.

Silicon Spike device

Here we present an Arduino-based custom-made device whose production cost is currently around €60. This device consists of a hardware part (Fig. 3) and its proper code (available here: <https://github.com/Ippolz/SiliconSpike>), which needs to be uploaded once on the device motherboard. All of the necessary stimulation parameters (e.g., number of pulses and the distance between them) used for spTMS, rTMS, ccPAS, cTBS, or iTBS can consequently be established through common software used in the research field, such as MATLAB or Python among others. Due to their length and technical specificity, assembly instructions are covered in its user manual (available here: <https://ippoz.gitbook.io/siliconspliketriggerbox/>; also see Supplementary Materials). The Silicon Spike hardware consists of the following:

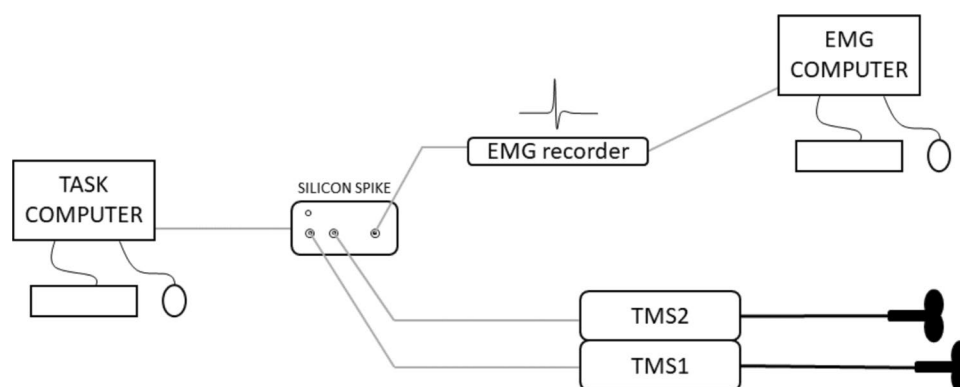


Fig. 2 Schematization of a classical lab configuration. In this case the task computer gives an input to the Silicon Spike triggerbox, which autonomously coordinates TMS and EMG timings and markers,

thus not relying on the computer processor. This resource allocation allows a smoother execution of the experimental task

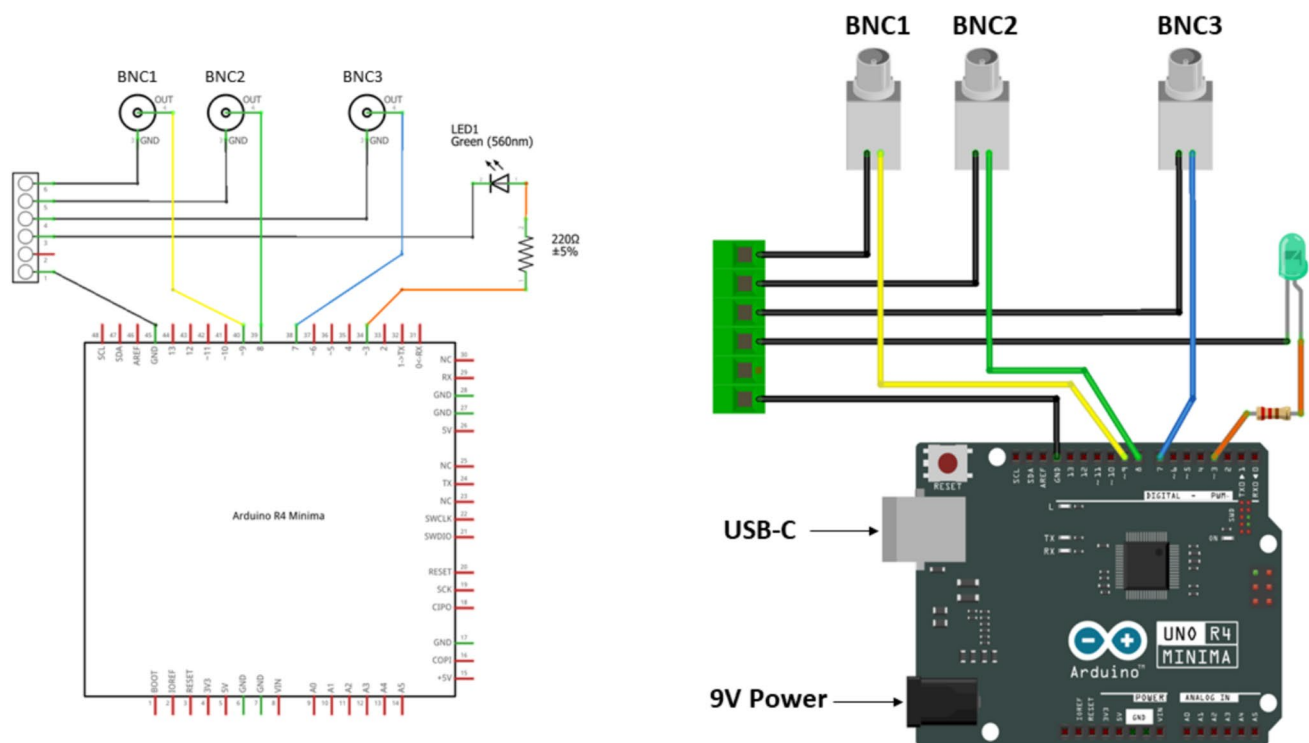


Fig. 3 Silicon Spike hardware in **a** its schematic circuit and **b** graphical representation. Full details in Supplementary Materials or here: <https://ippoz.gitbook.io/siliconspike-triggerbox/>

- Arduino Uno R4 Minima (<https://store.arduino.cc/products/uno-r4-minima>)
- Three BNC pins (TTL outputs)
- One LED and its proper resistor
- One USB type-C (serial communication)
- One 9 V 2.1 mm power jack

The code structure allows for the use of the same software (e.g., MATLAB) running the task to control the stimulation parameters. Once the serial communication is established, the experimenter can define the following:

- The inter-pulse interval (IPI), consisting of the distance between each pulse (only for rTMS and dual-coil protocols)
- The number of pulses within each train (only for rTMS protocols)
- The length of a square wave (5 V), to use it as a marker in the absence of any TMS pulse (e.g., stimulus onset)

All these parameters can be independently declared for nine presets within the same stimulation protocol (e.g., a dual-coil TMS with nine different distances between pulses to call separately in the same task). After that, you need to specify which protocol to use (spTMS, rTMS, dcTMS). Note that iTBS and cTBS protocols are extensions of the

rTMS one, whereas the ccPAS is an extension of the dual-coil protocol.

Materials and methods

In order to measure the precision of the Silicon Spike device timing, we used a Biopac MP-35 (BIOPAC Systems Inc., USA) system and BSL Analysis 4.1 software. Eight tests were conducted to measure the device performance under different load conditions for both rTMS and ccPAS (results also apply to spTMS, being simpler and derived from the previous ones), using a MATLAB (version R2022a) script (available here: <https://github.com/Ippolz/SiliconSpike>) consisting of a loop containing the specific protocol parameters and the firing commands. Each pulse produced a digital square wave of 5 V amplitude on the Biopac recording, running at a 20,000 Hz sampling rate. In order to mimic a realistic scenario, the Silicon Spike device was also connected to the TMS machine, which delivered TMS pulses at 50% of the maximum stimulator output intensity. For the dual-coil protocol, a Magstim® BiStim² model was used, while for the rTMS and iTBS/cTBS protocols, a Magstim® Rapid² was used. The marker (TTL) length (in tests 2, 4, 5) and the interval between each pulse (IPI, in tests 1, 2, 3, 4, 5, 6, 7) were measured (Fig. 4).

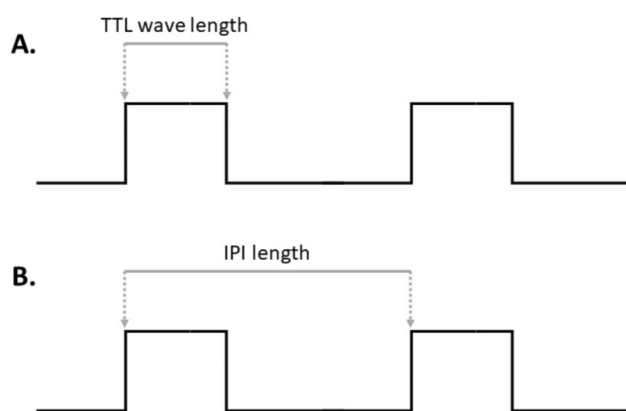


Fig. 4 The two main measures we used are **A** the TTL wavelength, which is the duration of the digital square wave, and **B** the IPI length, which is the distance between the beginning of a digital wave and the consecutive one

Test 1: Low-load rTMS

The first test investigates the rTMS protocol in a low-load condition, delivering 100 trains of seven pulses each, with an IPI of 100 ms (10 Hz) and an interval of 1 s between each train start. Thus, we measured the device precision for each IPI (600 trials).

Test 2: High-load rTMS

The second test repeats the previous one in a high-load condition, creating four different presets for the rTMS protocol. Thus, 100 trains of seven pulses each were delivered for each different IPI (70, 90, 120, 150 ms), randomly called with a distance of 2 s. Each of these is preceded by a random marker between the four declared in the setting phase (7, 9, 12, 15 ms) and a 1 s pause. The TTL length ($100 \times 4 = 400$ trials) and the IPI distance ($600 \times 4 = 2400$ trials) were considered.

Test 3: Low-load ccPAS

This test is aimed at measuring the precision of the ccPAS protocol during a low-load condition. Thus, a series consisting of 100 pairs of pulses with a distance of 150 ms between them is presented with a 1 s interval. The TTL length (100 trials) was measured.

Test 4: High-load ccPAS (long IPIs)

The fourth test repeats the previous one in a high-load condition, creating four different presets for the ccPAS protocol. In this case, long IPIs (150, 500, 700, 1000 ms) were used, with an interval of 4 s between each pair. Each of these pairs was presented randomly, and anticipated by a random

marker of a chosen length (7, 10, 15, 20 ms). The TTL length ($100 \times 4 = 400$ trials) and the IPI distance ($100 \times 4 = 400$ trials) were measured.

Test 5: High-load ccPAS (short IPIs)

The fifth test mimics the fourth one, presenting four pulse pairs preceded by a marker but using short IPIs (7, 10, 15, 20 ms) instead. The TTL length ($100 \times 4 = 400$ trials) and the IPI distance ($100 \times 4 = 400$ trials) were measured.

Test 6: iTBS

The sixth test uses the rTMS parameters to reproduce an iTBS protocol, delivering brief bursts of three pulses each at 50 Hz repeated every 200 ms in short trains of 2 s each repeated every 10 s (20 trains, 600 pulses). The IPI distance (400 trials) was measured.

Test 7: cTBS

The seventh test reproduces a cTBS protocol, delivering brief bursts of three pulses each at 50 Hz repeated every 200 ms for 20 s (300 pulses). The IPI distance (200 trials) was measured.

Test 8: IPI length effect

The eighth and last test is aimed at assessing whether there is an accumulation of latencies at increasing times, by delivering ccPAS pulses whose IPIs increase linearly from short (10 ms) to long (800 ms) intervals. Each interval (10, 100, 200, 300, 400, 500, 600, 700, 800 ms) was repeated 100 times. The IPI distance (800 trials) was measured.

Results

In seven separate tests, we measured the temporal precision for the interval between each pulse (IPI), while in tests 2, 4, and 5 we measured the temporal precision of the marker (TTL) length. For both indices we took into consideration the average variability in the wave duration (standard deviation) and the difference between the expected and the actual duration (delay). Overall, we observed very little difference between the expected and the effective durations (see Table 1). By considering the TTL wavelength, we registered an average delay of 0.008 ms (min = 0.005 ms, max = 0.011 ms). The oscillation around the expected duration was 0.018 ms (min = 0.016 ms, max = 0.021 ms). Results were similar when considering the IPI performance, with an average delay of 0.006 ms (min = -0.011 ms, max = 0.011 ms) between the expected and the effective

Table 1 Average data resulting from the 28 simulations obtained from tests 1 to 7

Expected duration (ms)	Test 1—TTL length				Test 1—IPI length			
	—	—	—	—	100	—	—	—
Avg (ms)	—	—	—	—	100.010	—	—	—
SD (ms)	—	—	—	—	0.020	—	—	—
Delay (ms)	—	—	—	—	0.010	—	—	—
Expected duration (ms)	Test 2—TTL length				Test 2—IPI length			
	7	9	12	15	70	90	120	150
Avg (ms)	7.007	9.008	12.006	15.010	70.010	90.011	120.009	150.009
SD (ms)	0.017	0.019	0.016	0.020	0.020	0.020	0.019	0.019
Delay (ms)	0.007	0.008	0.005	0.010	0.010	0.011	0.009	0.009
Expected duration (ms)	Test 3—TTL length				Test 3—IPI length			
	—	—	—	—	150	—	—	—
Avg (ms)	—	—	—	—	150.005	—	—	—
SD (ms)	—	—	—	—	0.016	—	—	—
Delay (ms)	—	—	—	—	0.004	—	—	—
Expected duration (ms)	Test 4—TTL length				Test 4—IPI length			
	7	10	15	20	150	500	700	1000
Avg (ms)	7.006	10.008	15.007	20.008	150.006	500.000	699.999	999.989
SD (ms)	0.016	0.018	0.017	0.018	0.018	0.011	0.009	0.021
Delay (ms)	0.005	0.007	0.006	0.008	0.006	−0.001	−0.001	−0.011
Expected duration (ms)	Test 5—TTL length				Test 5—IPI length			
	7	10	15	20	7	10	15	20
Avg (ms)	7.009	10.006	15.011	20.007	7.007	10.008	15.008	20.011
SD (ms)	0.019	0.016	0.021	0.017	0.017	0.018	0.018	0.020
Delay (ms)	0.009	0.005	0.011	0.006	0.007	0.007	0.007	0.010
Expected duration (ms)	Test 6—TTL length				Test 6—IPI length			
	—	—	—	—	10	—	—	—
Avg (ms)	—	—	—	—	10.012	—	—	—
SD (ms)	—	—	—	—	0.021	—	—	—
Delay (ms)	—	—	—	—	0.012	—	—	—
Expected duration (ms)	Test 7—TTL length				Test 7—IPI length			
	—	—	—	—	10	—	—	—
Avg (ms)	—	—	—	—	10.012	—	—	—
SD (ms)	—	—	—	—	0.022	—	—	—
Delay (ms)	—	—	—	—	0.012	—	—	—

Values are indicative of the device precision in producing a marker of a fixed duration (TTL length) or a series of TTL waves with a fixed interval (IPI length). Here, for each condition, we can see the expected duration, which is the input value, the average (the average TTL wavelength), SD (standard deviation), and delay (the difference in time between the TTL wave duration and the input value).

interval, with a 0.018 ms (min = 0.009 ms, max = 0.021 ms) oscillation around the expected value.

In the eighth test, we investigated whether using greater IPIs would cause longer delays. Thus, we used a ccPAS paradigm whose IPIs ranged evenly between 10 and 800 ms, measuring its delay. The linear regression analysis (Fig. 5) revealed a significant negative relation ($R=0.988$, $p<0.001$), suggesting that the delay is progressively reduced at increasing times, becoming negative over approximately 300 ms. However, despite being significant, the delay range (from −0.0195 to 0.0095 ms) is over 50 times smaller than a

millisecond, suggesting that this latency does not negatively affect TMS results.

Discussion

TMS is a widely used technique both in clinical settings and in the field of cognitive neuroscience. It makes it possible to selectively interact with one or more brain regions with a neural process by perturbing or even enhancing (Bertaccini et al., 2023; Borgomaneri et al., 2023; Di

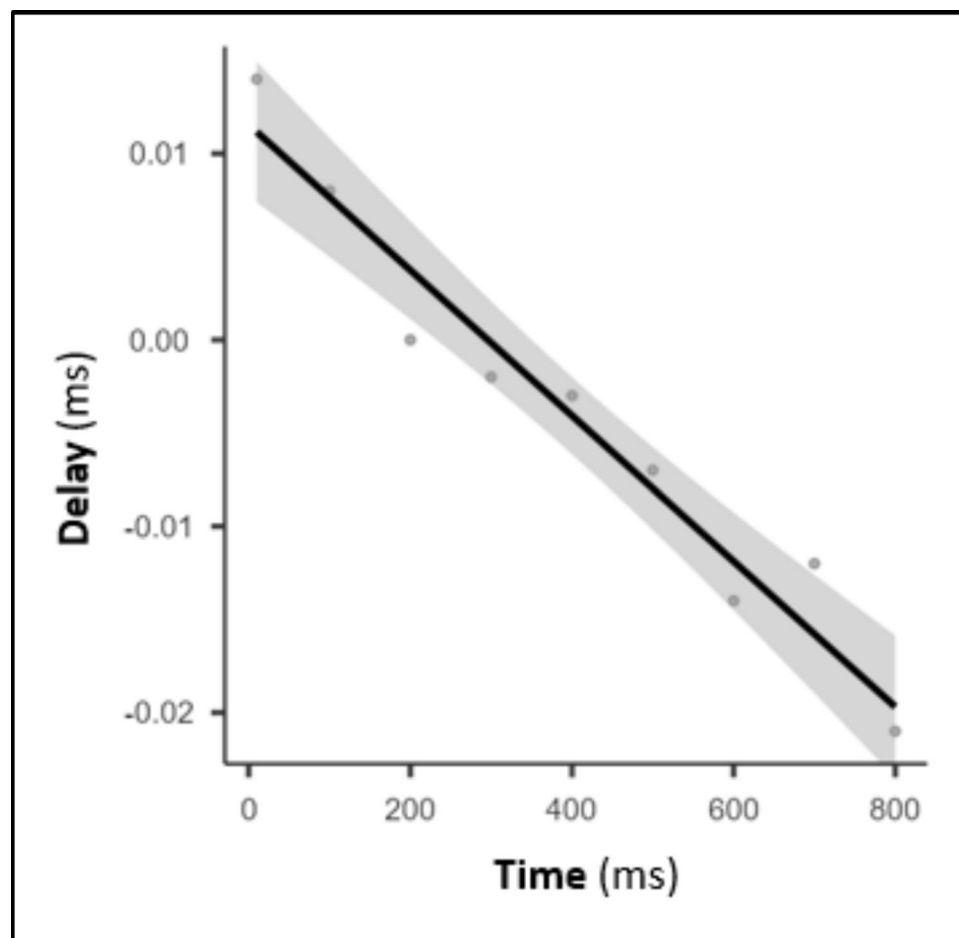


Fig. 5 Linear regression analysis between time and delay. With increasing time, the delay is reduced

Gregorio et al., 2022; Trajkovic et al., 2022; 2024a, b) their function, for example, during an experimental task. Many TMS protocols can significantly manipulate the ongoing brain activity by employing time-sensitive interventions which require precise timing delivery for optimal synchronized stimulus presentation and effective communication between stimulation devices. In this respect, the communication between the experimental computer and the TMS device is usually handled through the use of commercially available triggerbox devices, which simply translate the inputs into triggering instructions without relying on a dedicated internal clock. Indeed, this kind of approach is notoriously prone to bottleneck overflow (Joao et al., 2012), which might end in unwanted stimulus presentation latencies, especially in complex tasks or multi-pulse TMS (such as ccPAS, rTMS, or iTBS/cTBS) protocols, since most of them are not provided with built-in functions for delivering pulses in series. This risk can be avoided by relying on an external processor to handle timings, in order to keep the computations related to the TMS triggering completely independent from those performed by the

experimental computer (i.e., electromyographic recordings, EEG recordings, skin conductance, eye tracker, heart rate monitoring, stimuli presentation, response collection). However, a similar approach requires more complex programming, since several aspects specific to each TMS protocol must be considered. This means that relying on a dedicated processor affords greater precision, but is also potentially more prone to error. Here, we addressed this point by building Silicon Spike, a triggerbox device specifically designed to precisely trigger TMS devices for most of its stimulation protocols (i.e., spTMS, dcTMS, ccPAS, rTMS, rhTMS, iTBS, cTBS). The hardware consists of an Arduino Uno R4 Minima and a few other small components, which makes it a highly precise yet low-cost device (around €60). We freely provide the hardware schematics and the source code (<https://github.com/Ippolz/SiliconSpike>), meaning that the device can be easily assembled, and it does not need to be programmed. Also, its instructions are covered in detail (see Supplementary Materials or <https://ippolz.gitbook.io/siliconspike-triggerbox/>). We are confident that this approach will increase its ease of use

and will encourage colleagues to improve our work by adapting the Silicon Spike device according to the constantly changing needs of the research field.

In this paper we aimed at measuring the precision of the Silicon Spike under a series of realistic TMS protocols with various degrees of computational load. Our results show that the device is highly reliable, with an oscillation of just a few microseconds, across the different loading conditions. Further, we assessed whether these delays, however small, might accumulate at increasing times. Thus, we conducted an additional test in which we delivered pulse pairs at progressively longer distances, measuring their delay. Results indicate that, for small (10 ms) to long (800 ms) distances, the delay is a fraction dozens of times smaller than a millisecond. This means that the Silicon Spike device can be used by anyone for TMS research. It is important to stress the fact that, unlike many commercially available triggerbox devices, Silicon Spike is built on the concept that any temporal computation for stimulus delivery within and between devices relies on the external processor circuitry (the Silicon Spike in particular). This advantage allows the TMS triggering precision to be effectively unaffected by the execution of concurrent experimental task loads, or any other device timestamp requirement involved in the data collection. We believe that this aspect is crucial in the lab setting, where several devices are often involved in the data collection. Also, the Silicon Spike device allows one to set up to nine stimulation protocols at once (e.g., rTMS at 10 or 11 Hz; or ccPAS at 50, 70, or 120 ms). This means that it is possible to shuffle them within a single experimental block, enabling a finer trial randomization.

It might be argued that we do not directly compare our approach, which consists in parallel processing of instructions, to the approach used by most commercially available triggerbox devices, in which inputs are sent in series. However, such a comparison would be beyond the scope of this work. Indeed, commercial devices can be heterogeneous for both the hardware and programming part, often being built for a few specific purposes. In this view, we thought a comparison between devices might not be necessary. What we do know is that the Silicon Spike device has precision far above the millisecond in delivering TMS pulses; thus, we can affirm that it can be safely used in cognitive neuroscience practice.

Embracing an open-science framework, we provide the scripts we used to collect the present data (<https://github.com/Ippolz/SiliconSpike>) so that it is possible to replicate our exact analysis or extend it to include new parameters. As the entire project is freely available online, we are confident that it can be modified and improved by our colleagues in the scientific community. When we built the Silicon Spike device, we aimed at making it as easy as possible to reproduce, so we opted for a compact and simple design. Still, we

think that it is possible, through derivative designs, to build a triggerbox tuned for some specific stimulation paradigms which we do not currently cover (e.g., three-coil TMS), thus overcoming current limitations. Therefore, we highly encourage readers to improve upon our work. Colleagues are also allowed to take advantage of our scripts in order to reproduce our reliability tests on their alternative versions of the Silicon Spike device.

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.3758/s13428-025-02653-y>.

Author contributions Conceptualization: G.I., T.Q.; Data curation: G.I., T.Q.; Formal analysis: G.I.; Funding acquisition: V.R., S.B.; Methodology: G.I., T.Q.; Project administration: V.R., S.B.; Supervision: V.R., S.B.; Writing—original draft: G.I.; and Writing—review & editing: T.Q., V.R., S.B.

Funding Open access funding provided by Alma Mater Studiorum - Università di Bologna within the CRUI-CARE Agreement. V.R. is supported by MUR – Ministry of University and Research, Italy (P2022XAKXL and 2022H4ZRSN), Ministerio de Ciencia, Innovación y Universidades, Spain (PID2019111335GA-I00), and the BIAL Foundation (033/22).

Data availability Data are available at: <https://github.com/Ippolz/SiliconSpike>.

Code availability Code available at: <https://github.com/Ippolz/SiliconSpike>.

Declarations

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

Competing interests The authors report no conflicts of interest.

Open Practices Stimuli, validation data, and analysis code are available at: <https://github.com/Ippolz/SiliconSpike>.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Bertaccini, R., et al. (2023). Rhythmic TMS as a Feasible Tool to Uncover the Oscillatory Signatures of Audiovisual Integration. *Biomedicines*, 11(6), 1746. <https://doi.org/10.3390/biomedicines11061746>
- Borgomaneri, S., et al. (2023). Increasing associative plasticity in temporo-occipital back-projections improves visual perception of emotions. *Nature Communications*, 14(1), 5720. <https://doi.org/10.1038/s41467-023-41058-3>
- D'Ausilio, A. (2012). Arduino: A low-cost multipurpose lab equipment. *Behav Res*, 44(2), 305–313. <https://doi.org/10.3758/s13428-011-0163-z>
- Di Gregorio, F. (2022). Tuning alpha rhythms to shape conscious visual perception. *Current Biology*, 32(5), 988–998.e6. <https://doi.org/10.1016/j.cub.2022.01.003>
- Di Luzio, P. (2024). Investigating the effects of cortico-cortical paired associative stimulation in the human brain: A systematic review and meta-analysis. *Neuroscience & Biobehavioral Reviews*, 167, 105933. <https://doi.org/10.1016/j.neubiorev.2024.105933>
- Fotia, F., Cooke, J., Van Dam, L., Ferri, F., & Romei, V. (2021). The temporal sensitivity to the tactile-induced double flash illusion mediates the impact of beta oscillations on schizotypal personality traits. *Consciousness and Cognition*, 91, 103121. <https://doi.org/10.1016/j.concog.2021.103121>
- Güvenç, H. (2020) Wireless ECG Device with Arduino. In *2020 Medical Technologies Congress (TIPTEKNO)*. 1–4. <https://doi.org/10.1109/TIPTEKNO50054.2020.9299248>.
- Hallett, M. (2007). Transcranial Magnetic Stimulation: A Primer. *Neuron*, 55(2), 187–199. <https://doi.org/10.1016/j.neuron.2007.06.026>
- Hernandez-Pavon, J. C., San Agustín, A., Wang, M. C., Veniero, D. and Pons, J. L., Can we manipulate brain connectivity? A systematic review of cortico-cortical paired associative stimulation effects. *Clinical Neurophysiology*, 154, 169–193. <https://doi.org/10.1016/j.clinph.2023.06.016>
- Ismailov, A. S. and Jo, Z. B. (2022) Study of arduino microcontroller board. *Science and Education*, 3(3).
- Ippolito, G., et al. (2022). The Role of Alpha Oscillations among the Main Neuropsychiatric Disorders in the Adult and Developing Human Brain: Evidence from the Last 10 Years of Research. *Biomedicines*, 10(12), 3189. <https://doi.org/10.3390/biomedicines10123189>
- Klomjai, W., Katz, R., & Lackmy-Vallée, A. (2015). Basic principles of transcranial magnetic stimulation (TMS) and repetitive TMS (rTMS). *Annals of Physical and Rehabilitation Medicine*, 58(4), 208–213. <https://doi.org/10.1016/j.rehab.2015.05.005>
- Kondaveeti, H. K., Kumaravelu, N. K., Vanambathina, S. D., Mathe, S. E., & Vappangi, S. (2021). A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations. *Computer Science Review*, 40, 100364. <https://doi.org/10.1016/j.cosrev.2021.100364>
- Krauss, R. (2020) Real-Time Python: Recent Advances in the Raspberry Pi Plus Arduino Real-Time Control Approach. In: *2020 American Control Conference (ACC)*, Denver, CO, USA: IEEE, pp. 2088–2093. <https://doi.org/10.23919/ACC45564.2020.9147236>.
- José A. Joao, M. Aater Suleman, Onur Mutlu, and Yale N. Patt. (2012). Bottleneck identification and scheduling in multithreaded applications. In Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XVII). *Association for Computing Machinery* (pp. 223–234). <https://doi.org/10.1145/2150976.2151001>
- Mindoro, J. N. (2020) Drowsy or Not? Early Drowsiness Detection utilizing Arduino Based on Electroencephalogram (EEG) Neuro-Signal. *IJETER*, 9(2):2221–2226. <https://doi.org/10.30534/ijatse/2020/200922020>.
- Peirce, J., et al. (2019). PsychoPy2: Experiments in behavior made easy. *Behavior Research Methods*, 51(1), 195–203. <https://doi.org/10.3758/s13428-018-01193-y>
- Tarasi L., Turrini S., Sel A., Avenanti A., Romei V. (2024). Cortico-cortical paired-associative stimulation to investigate the plasticity of cortico-cortical visual networks in humans. *Current Opinion in Behavioral Sciences*, 56, 101359. <https://doi.org/10.1016/j.cobeha.2024.101359>
- Teikari, P., et al. (2012). An inexpensive Arduino-based LED stimulator system for vision research. *Journal of Neuroscience Methods*, 211(2), 227–236. <https://doi.org/10.1016/j.jneumeth.2012.09.012>
- Trajkovic, J., Di Gregorio, F., Marcantoni, E., Thut, G., & Romei, V. (2022). A TMS/EEG protocol for the causal assessment of the functions of the oscillatory brain rhythms in perceptual and cognitive processes. *STAR Protocols*, 3(2), 101435. <https://doi.org/10.1016/j.xpro.2022.101435>
- Trajkovic J., Di Gregorio F., Thut G., Romei V. (2024a). Transcranial magnetic stimulation effects support an oscillatory model of ERP genesis. *Current Biology*, 34(5), 1048–1058.e4. <https://doi.org/10.1016/j.cub.2024.01.069>
- Trajkovic J., Sack A. T., Romei V. (2024b). EEG-based biomarkers predict individual differences in TMS induced entrainment of intrinsic brain rhythms. *Brain Stimulation*, 17(2), 224–232. <https://doi.org/10.1016/j.brs.2024.02.016>
- White, S. R., Amarante, L. M., Kravitz, A. V., and Laubach, M. (2019) The Future Is Open: Open-Source Tools for Behavioral Neuroscience Research. *eNeuro*. 6(4) <https://doi.org/10.1523/NEURO.0223-19.2019>.

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