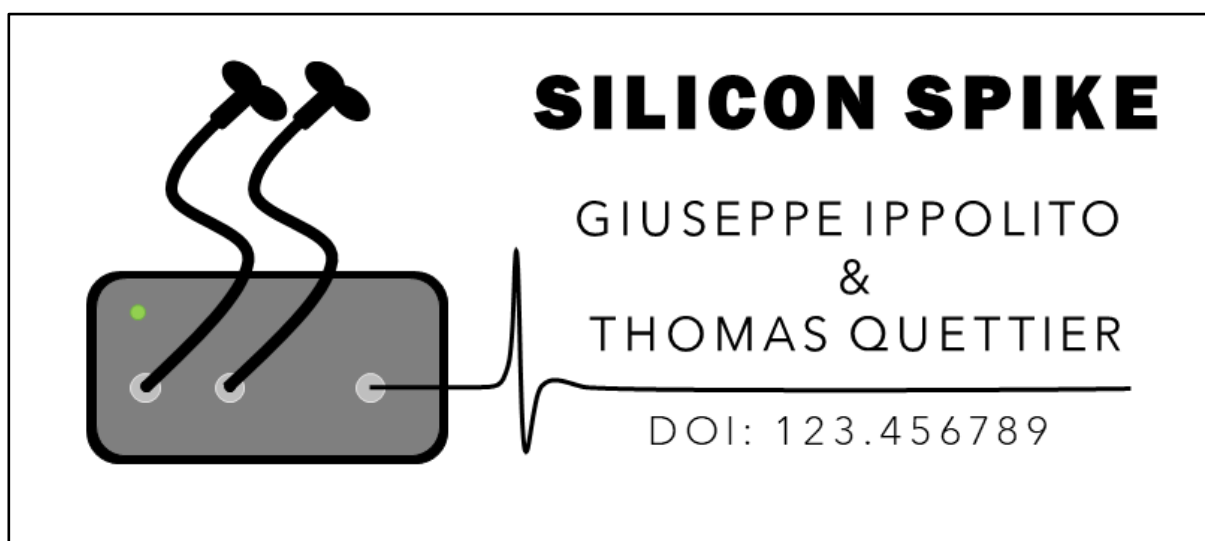


SILICON SPIKE

USER MANUAL



Original work

Please, for any question about performance and timing reliability refer to the original publication:

Ippolito G. Quettier T. Borgomaneri S. Romei V. Silicon Spike: an Arduino-based low-cost and open-access triggerbox to precisely control TMS devices, [XXXXX](#)

DOI: [XXXXX](#)

Contacts:

Introduction

Here we report a novel and reliable tool to trigger transcranial magnetic stimulation (TMS) devices with almost null latencies, easing the task execution during lab experiments. This goal has been achieved with exceptionally good results. Hence, we decided to make the Silicon Spike triggerbox a freely accessible device for anyone to reproduce, implement, and share. If you are using the Silicon Spike triggerbox in your experiment, please acknowledge our work (DOI).

Relative to the most commonly available triggerbox devices, the advantage of Silicon Spike consists in leaving all of the computations necessary to trigger the TMS to its internal motherboard, without interfering with the computer executing the experimental task. It allows control of all the stimulation parameters for the single pulse (spTMS), repetitive/rhythmic (rTMS), and dual coil (dcTMS/ccPAS) protocols with few lines of code, also making it accessible for those without any programming knowledge. It is also possible to set the rTMS parameters to obtain a continuous (cTBS) or intermittent (iTBS) theta-burst stimulation. The stimulation parameters can be declared using any software allowing serial communication; here we will cover in detail this procedure using MATLAB and Python.

Hardware

The Silicon Spike device is composed of the following elements:

- Arduino Uno R4 Minima (product details here: <https://store.arduino.cc/products/uno-r4-minima>);
- Three BNC pins;
- One LED and its proper resistor;
- One USB type-C;
- One 9 Volt 2.1mm power jack

These components need to be assembled following this scheme:

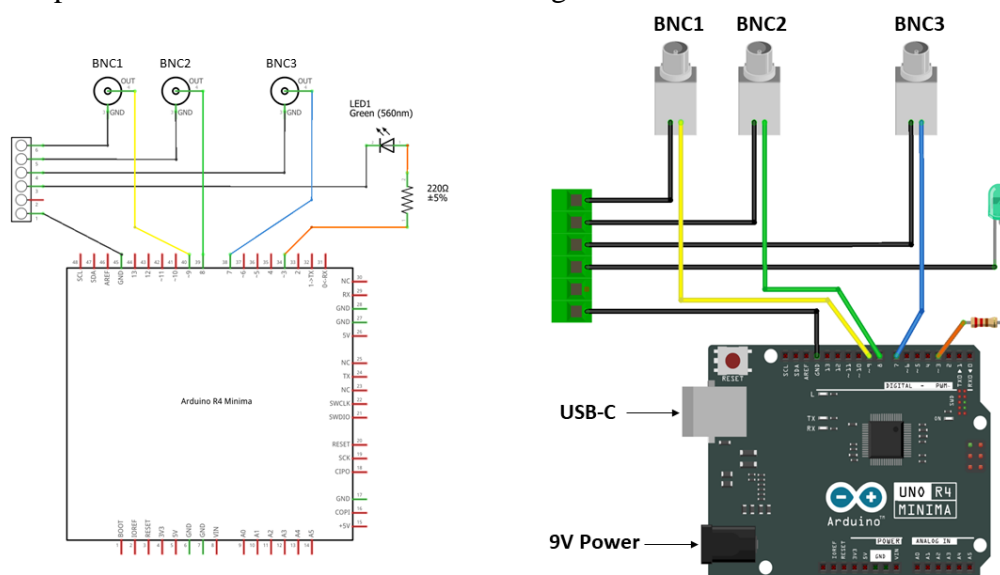


Figure 1 – Silicon Spike’s hardware in a.) its schematic circuit, and b.) graphical representation. Full details here: <https://ippoz.gitbook.io/silicon-spike-triggerbox/>.

Eventually, you can use a female BNC instead of a male one, according to your circumstances. Also, you can use the LED color you prefer – and, consequently, the proper resistance. Importantly, in order to grant both data transmission and the square digital wave on the recording, you need to plug the motherboard with both USB-C and 9 Volt 2.1mm power jack.

Once soldered and assembled the above components the device will work properly. However, we suggest building a case (Fig. 2) for the device, in order to avoid any potential damage which might hamper its functioning.



Figure 2 - A few examples of how the circuit can be arranged into a case. Note that their design can be different, since you can flexibly add features according to your needs.

Software

Silicon Spike is composed by two main codes:

- *Communication code* (available here: <https://github.com/Ippolz/SiliconSpike/tree/main/Codes/Communication%20Code/MATLAB>): is the brief code you use to send information – the stimulation parameters – to the Silicon Spike device. You can copy-paste them, and then just adjust the protocol parameters (e.g., number of TMS pulses; pause between each TMS pulse, etc.) according to your requirements.
- *Main code* (available here: https://github.com/Ippolz/SiliconSpike/tree/main/Codes/Main%20Code/SiliconSpike_MainCode): must be uploaded on the Arduino motherboard, and doesn't need any change. It contains all the instructions to read the stimulation parameters sent through the *communication code*. If the Main code is properly uploaded in the motherboard, the LED will light when the device is powered.

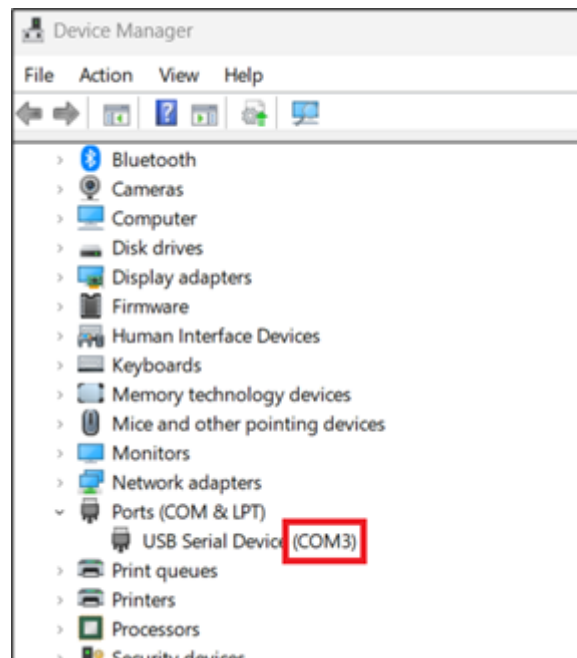
Main code

Once downloaded you just need to open it through the Arduino IDE (see here: <https://www.arduino.cc/en/software>), and then upload it as it is. This code only needs to be uploaded once.

Communication code

This is the code you will use during each TMS stimulation. You can copy-paste it within your task script to trigger the TMS device without relying on the computer resources. Before running the script, you need to declare the stimulation parameters.

Please note, before starting the code explanation, it is important to find the name of the USB port ("COM") that you are using with the Silicon Spike device. Here, opening the Device Manager we can see that the Silicon Spike device is connected to port 3, so we declared "COM3".



A) COMMUNICATION CODE: MATLAB

● single pulse TMS (spTMS)

```
1. % Initialize the serial communication
2. s = serialport("COM3",115200);
3. fopen(s);
4. pause(2);
5.
6. % Mandatory signature
7. fwrite(s,"Triggerbox developed by Giuseppe Ippolito. DOI: 123.456789");
8. pause(0.01);
9.
10. % Declaring marker duration
11. fwrite(s,"SET,MRK1,3");
12. pause(0.01);
13. fwrite(s,"SET,MRK2,5");
```

```

96 14. pause(0.01);
97 15.
98 16. % Protocol type (spTMS, dcTMS, rTMS)
99 17. fwrite(s,"spTMS");
100 18. pause(0.01);
101 19.
102 20. % Commands
103 21. fwrite(s,"1");
104 22. fwrite(s,"2");
105 23.
106 24. fwrite(s,"A");
107 25. fwrite(s,"B");
108 26.
109 27. % Close the serial communication
110 28. fwrite(s,"Z");
111 29. fclose(s);
112 30. delete(s);
113 31. clear s;

```

114 Lines 2-3 initialize the **serial communication**. In older MATLAB versions (prior to 2019b)
 115 you can use:

```

116 % Initialize the serial communication
117 s = serial ("COM3","BaudRate",115200);
118 fopen(s);
119 pause(2);

```

120 From version 2019b onward is necessary to use:

```

121 % Initialize the serial communication
122 s = serialport("COM3",115200);
123 fopen(s);
124 pause(2);

```

125 It is important to change the USB port ("COM") name according to your case as explained
 126 above. Also, it is good practice, during the setting phase, to put a *pause* command between
 127 each serial string. A 10 ms duration is sufficient in any case, except for the first command, in
 128 which the communication between devices is established. In this case, the optimal duration
 129 may vary depending on your computer performance. A 2 sec interval should be enough for
 130 older devices, too. This pause is not necessary after the setting phase, when delivering TMS
 131 pulses.

132 Then, copy-paste the **mandatory signature**. Without this the following lines won't work.

```

133 fwrite(s,"Triggerbox developed by Giuseppe Ippolito. DOI: 123.456789");

```

134 After these mandatory steps, you can declare optional stimulation parameters (e.g., IPI, number
 135 of pulses, markers) in any order. In this case, since we're going to use a spTMS protocol, the
 136 only parameter we may be interested in is the **voluntary marker** duration. This option allows

137 you to place digital markers of a chosen duration (max 9 different lengths) at any moment. This
138 may be useful in case you need to discriminate between different stimuli's onset (e.g., neural,
139 joy, or fearful faces). Remember that voluntary markers always get delivered through BNC3.
140 Its syntax is:

```
141 fwrite(s, "SET,MRKN,X");
```

142 In this case N is the preset number (between 1 and 9), and X is the marker length. Recalling the
143 two stimuli example, we may want two voluntary markers of different duration (e.g., 3 and 5
144 ms) to ease our analysis, later. We will write then:

```
145 fwrite(s, "SET,MRK1,3"); % Preset 1: 3ms marker  
146 fwrite(s, "SET,MRK2,5"); % Preset 2: 5ms marker
```

147 Our last step before calling the pulse command is declaring which TMS protocol we are going
148 to use. We can choose between spTMS, dcTMS, or rTMS. Since we're using the single pulse
149 one, we will write:

```
150 fwrite(s, "spTMS");
```

151 Now that we have declared all of the necessary stimulation parameters, we can finally deliver
152 the TMS pulse. You can independently trigger the BNC1 ("1") or BNC2 ("2"). These lines
153 must be copy-pasted within your code according to your stimulation and task requirements (es.
154 prior to a certain stimulus onset).

```
155 fwrite(s, "1"); % Trigger BNC1  
156 fwrite(s, "2"); % Trigger BNC2
```

157 We can also individually call for voluntary markers at any time using letters. Preset 1 (MRK1)
158 is paired with letter "A" – Consequently, preset 2 with "B", 3 with "C", 4 with "D", 5 with
159 "E", 6 with "F", 7 with "G", 8 with "H", and 9 with "I". Since we declared two different
160 markers ("MRK1,3", "MRK2,5") we can trigger them using the letters "A" and "B".

```
161 fwrite(s, "A"); % Deliver a voluntary marker from preset 1 (3 ms)  
162 fwrite(s, "B"); % Deliver a voluntary marker from preset 2 (5 ms)
```

163 The command code also includes a line to run in case you may reset any setting and return to
164 the mandatory signature. You can then declare once again the parameter settings. The
165 command is:

```
166 fwrite(s, "Z"); % Return to the beginning of the setting phase
```

167 Once the task is completed you can definitely close the serial communication between the
168 Silicon Spike device and the experimental computer, thus avoiding any potential bug.

```
169 fclose(s);  
170 delete(s);  
171 clear s;
```

172 • Dual coil TMS (dcTMS)

173 This protocol allows you to trigger pairs of TMS pulses, adding the option to choose the
174 distance in ms between the two pulses. If you've read the previous section (spTMS) most of
175 this information will be redundant, aside from the "SET,IPI" parameter.

```
176 1. % Initialize the serial communication
177 2. s = serialport("COM3",115200);
178 3. fopen(s);
179 4. pause(2);
180 5.
181 6. % Mandatory signature
182 7. fwrite(s,"Triggerbox developed by Giuseppe Ippolito. DOI: 123.456789");
183 8. pause(0.01);
184 9.
185 10. % Declaring the distance between the two TMS pulses
186 11. fwrite(s,"SET,IPI1,30");
187 12. pause(0.01);
188 13. fwrite(s,"SET,IPI2,50");
189 14. pause(0.01);
190 15. fwrite(s,"SET,IPI3,70");
191 16. pause(0.01);
192 17.
193 18. % Declaring marker duration
194 19. fwrite(s,"SET,MRK1,3");
195 20. pause(0.01);
196 21. fwrite(s,"SET,MRK2,5");
197 22. pause(0.01);
198 23. fwrite(s,"SET,MRK3,7");
199 24. pause(0.01);
200 25.
201 26. % Protocol type (spTMS, dcTMS, rTMS)
202 27. fwrite(s,"dcTMS");
203 28. pause(0.01);
204 29.
205 30. % Commands
206 31. fwrite(s,"1");
207 32. fwrite(s,"2");
208 33. fwrite(s,"3");
209 34.
210 35. fwrite(s,"A");
211 36. fwrite(s,"B");
212 37. fwrite(s,"C");
213 38.
214 39. % Close the serial communication
215 40. fwrite(s,"Z");
216 41. fclose(s);
217 42. delete(s);
218 43. clear s;
```

219 Lines 2-3 initialize the **serial communication**. In older MATLAB versions (prior to 2019b)
220 you can use:

```
221 % Initialize the serial communication
222 s = serial ("COM3", "BaudRate", 115200);
223 fopen(s);
224 pause(2);
```

225 From version 2019b onward is necessary to use:

```
226 % Initialize the serial communication
227 s = serialport("COM3", 115200);
228 fopen(s);
229 pause(2);
```

230 It is important to change the USB port ("COM") name according to your case as explained
231 above. Also, it is good practice, during the setting phase, to put a *pause* command between
232 each serial string. A 10 ms duration is sufficient in any case, except for the first command, in
233 which the communication between devices is established. In this case, the optimal duration
234 may vary depending on your computer performance. A 2 sec interval should be enough for
235 older devices, too. This pause is not necessary after the setting phase, when delivering TMS
236 pulses.

237 Then, copy-paste the **mandatory signature**. Without this the following lines won't work.

```
238 fwrite(s, "Triggerbox developed by Giuseppe Ippolito. DOI: 123.456789");
```

239 After these mandatory steps, you can declare optional stimulation parameters (e.g., IPI, number
240 of pulses, markers) in any order. In this case, since we're going to use a spTMS protocol, the
241 only parameter we may be interested in is the **voluntary marker** duration. This option allows
242 you to place digital markers of a chosen duration (max 9 different lengths) at any moment. This
243 may be useful in case you need to discriminate between different stimuli's onset (e.g., neural,
244 joy, or fearful faces). Remember that voluntary markers always get delivered through BNC3.
245 Its syntax is:

```
246 fwrite(s, "SET,MRKN,X");
```

247 In this case *N* is the preset number (between 1 and 9), and *X* is the marker length. Recalling the
248 three stimuli example, we may want three voluntary markers of different duration (e.g., 3, 5,
249 and 7 ms) ease our analysis, later. We will write then:

```
250 fwrite(s, "SET,MRK1,3"); % Preset 1: 3ms marker
251 fwrite(s, "SET,MRK2,5"); % Preset 2: 5ms marker
252 fwrite(s, "SET,MRK3,7"); % Preset 3: 7ms marker
```

253 For dcTMS paradigms it is also important to declare the distance in ms between the two pulses
254 (IPI), delivered from BNC1 and then BNC2. Its syntax is:

```
255 fwrite(s, "SET,IPIN,X");
```


256 In this case N is the preset number (between 1 and 9), and X is the distance between pulses in
257 ms. For example, we may want to test the difference in delivering pulse pairs with different
258 latencies when the experimental stimulus occurs. We will write then:

```
259 fwrite(s,"SET,IPI1,30"); % Preset 1: 30ms distance  
260 fwrite(s,"SET,IPI2,50"); % Preset 2: 50ms distance  
261 fwrite(s,"SET,IPI3,70"); % Preset 3: 70ms distance
```

262 Our last step before calling the pulse command is declaring which TMS protocol we are going
263 to use. We can choose between spTMS, dcTMS, or rTMS. Since we're using the dual coil one,
264 we will write:

```
265 fwrite(s,"dcTMS");
```

266 Now that we have declared all of the necessary stimulation parameters, we can finally trigger
267 the TMS device. You can independently trigger each of the declared protocols. These lines
268 must be copy-pasted within your code according to your stimulation and task requirements (es.
269 prior to a certain stimulus onset).

```
270 fwrite(s,"1"); % Trigger protocol 1 (two pulses with a 30 ms distance)  
271 fwrite(s,"2"); % Trigger protocol 2 (two pulses with a 50 ms distance)  
272 fwrite(s,"3"); % Trigger protocol 3 (two pulses with a 70 ms distance)
```

273 We can also individually call for voluntary markers at any time using letters. Preset 1 (MRK1)
274 is paired with letter "A" – Consequently, preset 2 with "B", 3 with "C", 4 with "D", 5 with
275 "E", 6 with "F", 7 with "G", 8 with "H", and 9 with "I". Since we declared three different
276 markers ("MRK1,3", "MRK2,5", "MRK3,7") we can trigger them using the letters "A", "B",
277 and "C".

```
278 fwrite(s,"A"); % Deliver a voluntary marker from preset 1 (3 ms)  
279 fwrite(s,"B"); % Deliver a voluntary marker from preset 2 (5 ms)  
280 fwrite(s,"C"); % Deliver a voluntary marker from preset 3 (7 ms)
```

281 The command code also includes a line to run in case you may reset any setting and return to
282 the mandatory signature. You can then declare once again the parameter settings. The
283 command is:

```
284 fwrite(s,"Z"); % Return to the beginning of the setting phase
```

285 Once the task is completed you can definitely close the serial communication between the
286 Silicon Spike device and the experimental computer, thus avoiding any potential bug.

```
287 fclose(s);  
288 delete(s);  
289 clear s;
```

290

291

292 ● repetitive TMS (rTMS)

293 This protocol allows you to trigger trains of TMS pulses, adding the option to choose the
294 distance in ms between the two pulses and the number of pulses within each train. If you've
295 read the previous section (dcTMS) most of this information will be redundant, aside from the
296 "SET,nPULS" parameter.

```
297 1. % Initialize the serial communication
298 2. s = serialport("COM3",115200);
299 3. fopen(s);
300 4. pause(2);
301 5.
302 6. % Mandatory signature
303 7. fwrite(s,"Triggerbox developed by Giuseppe Ippolito. DOI: 123.456789");
304 8. pause(0.01);
305 9.
306 10. % Declaring the distance between each TMS pulse
307 11. fwrite(s,"SET,IPI1,80");
308 12. pause(0.01);
309 13. fwrite(s,"SET,IPI2,100");
310 14. pause(0.01);
311 15. fwrite(s,"SET,IPI3,120");
312 16. pause(0.01);
313 17.
314 18. % Declaring the number of pulses within each train
315 19. fwrite(s,"SET,nPULS1,4");
316 20. pause(0.01);
317 21. fwrite(s,"SET,nPULS2,5");
318 22. pause(0.01);
319 23. fwrite(s,"SET,nPULS3,6");
320 24. pause(0.01);
321 25.
322 26. % Declaring marker duration
323 27. fwrite(s,"SET,MRK1,3");
324 28. pause(0.01);
325 29. fwrite(s,"SET,MRK2,5");
326 30. pause(0.01);
327 31. fwrite(s,"SET,MRK3,7");
328 32. pause(0.01);
329 33.
330 34. % Protocol type (spTMS, dcTMS, rTMS)
331 35. fwrite(s,"rTMS");
332 36. pause(0.01);
333 37.
334 38. % Commands
335 39. fwrite(s,"1");
336 40. fwrite(s,"2");
337 41. fwrite(s,"3");
338 42.
```

```

339 43. fwrite(s,"A");
340 44. fwrite(s,"B");
341 45. fwrite(s,"C");
342 46.
343 47. % Close the serial communication
344 48. fwrite(s,"Z");
345 49. fclose(s);
346 50. delete(s);
347 51. clear s;

```

348 Lines 2-3 initialize the **serial communication**. In older MATLAB versions (prior to 2019b)
 349 you can use:

```

350 % Initialize the serial communication
351 s = serial ("COM3","BaudRate",115200);
352 fopen(s);
353 pause(2);

```

354 From version 2019b onward is necessary to use:

```

355 % Initialize the serial communication
356 s = serialport("COM3",115200);
357 fopen(s);
358 pause(2);

```

359 It is important to change the USB port ("COM") name according to your case as explained
 360 above. Also, it is good practice, during the setting phase, to put a *pause* command between
 361 each serial string. A 10 ms duration is sufficient in any case, except for the first command, in
 362 which the communication between devices is established. In this case, the optimal duration
 363 may vary depending on your computer performance. A 2 sec interval should be enough for
 364 older devices, too. This pause is not necessary after the setting phase, when delivering TMS
 365 pulses.

366 Then, copy-paste the **mandatory signature**. Without this the following lines won't work.

```

367 fwrite(s,"Triggerbox developed by Giuseppe Ippolito. DOI: 123.456789");

```

368 After these mandatory steps, you can declare optional stimulation parameters (e.g., IPI, number
 369 of pulses, markers) in any order. In this case, since we're going to use a spTMS protocol, the
 370 only parameter we may be interested in is the **voluntary marker** duration. This option allows
 371 you to place digital markers of a chosen duration (max 9 different lengths) at any moment. This
 372 may be useful in case you need to discriminate between different stimuli's onset (e.g., neural,
 373 joy, or fearful faces). Remember that voluntary markers always get delivered through BNC3.
 374 Its syntax is:

```

375 fwrite(s,"SET,MRKN,X");

```

376 In this case N is the preset number (between 1 and 9), and X is the marker length. Recalling the
377 three stimuli example, we may want three voluntary markers of different duration (e.g., 3, 5,
378 and 7 ms) ease our analysis, later. We will write then:

```
379 fwrite(s,"SET,MRK1,3"); % Preset 1: 3ms marker  
380 fwrite(s,"SET,MRK2,5"); % Preset 2: 5ms marker  
381 fwrite(s,"SET,MRK3,7"); % Preset 3: 7ms marker
```

382 For rTMS paradigms it is important to declare the number of pulses constituting each train,
383 triggered simultaneously from BNC1 and BNC2. Its syntax is:

```
384 fwrite(s,"SET,nPULSN,X");
```

385 In this case N is the preset number (between 1 and 9), and X is the number of pulses within the
386 train. For example, we may want to test the difference in delivering trains of 4, 5, or 6 pulses
387 prior to the stimulus occurrence. We will write then:

```
388 fwrite(s,"SET,nPULS1,4"); % Preset 1: 4 pulses in each train  
389 fwrite(s,"SET,nPULS2,5"); % Preset 2: 5 pulses in each train  
390 fwrite(s,"SET,nPULS3,6"); % Preset 3: 6 pulses in each train
```

391 Additionally, in rTMS paradigms it is also important to declare the distance in ms between the
392 two pulses (IPI), delivered from BNC1 and then BNC2. Its syntax is:

```
393 fwrite(s,"SET,IPIN,X");
```

394 In this case N is the preset number (between 1 and 9), and X is the distance between pulses in
395 ms. For example, we may want an 80 ms interval in the first preset (4 pulses), a 100 ms interval
396 for the second one (5 pulses), and a 120 ms interval for the third one (6 pulses). We will write
397 then:

```
398 fwrite(s,"SET,IPI1,80"); % Preset 1: 80ms distance  
399 fwrite(s,"SET,IPI2,100"); % Preset 2: 100ms distance  
400 fwrite(s,"SET,IPI3,120"); % Preset 3: 120ms distance
```

401 Our last step before calling the pulse command is declaring which TMS protocol we are going
402 to use. We can choose between spTMS, dcTMS, or rTMS. Since we're using the double pulse
403 one, we will write:

```
404 fwrite(s,"rTMS");
```

405 Now that we have declared all of the necessary stimulation parameters, we can finally trigger
406 the TMS device. You can independently trigger each of the declared protocols. These lines
407 must be copy-pasted within your code according to your stimulation and task requirements (es.
408 prior to a certain stimulus onset).

```
409 fwrite(s,"1"); % Protocol 1 (one train of 4 pulses with a 80ms distance)  
410 fwrite(s,"2"); % Protocol 2 (one train of 5 pulses with a 100ms distance)  
411 fwrite(s,"3"); % Protocol 3 (one train of 6 pulses with a 120ms distance)
```

412 We can also individually call for voluntary markers at any time using letters. Preset 1 (MRK1)
413 is paired with letter “A” – Consequently, preset 2 with “B”, 3 with “C”, 4 with “D”, 5 with
414 “E”, 6 with “F”, 7 with “G”, 8 with “H”, and 9 with “I”. Since we declared three different
415 markers (“MRK1,3”, “MRK2,5”, “MRK3,7”) we can trigger them using the letters “A”, “B”,
416 and “C”.

```
417 fwrite(s,"A"); % Deliver a voluntary marker from preset 1 (3 ms)
418 fwrite(s,"B"); % Deliver a voluntary marker from preset 2 (5 ms)
419 fwrite(s,"C"); % Deliver a voluntary marker from preset 3 (7 ms)
```

420 The command code also includes a line to run in case you may reset any setting and return to
421 the mandatory signature. You can then declare once again the parameter settings. The
422 command is:

```
423 fwrite(s,"Z"); % Return to the beginning of the setting phase
```

424 Once the task is completed you can definitely close the serial communication between the
425 Silicon Spike device and the experimental computer, thus avoiding any potential bug.

```
426 fclose(s);
427 delete(s);
428 clear s;
429
```

430 B) *COMMUNICATION CODE: PYTHON*

431 ● single pulse TMS (spTMS)

```
432 1. # Import the required packages
433 2. from serial import Serial, SerialException
434
435 3. # Initialize the serial communication
436 4. s = Serial()
437 5. s.port = "COM3"
438 6. s.baudrate = 115200
439 7. s.open()
440 8.
441 9. # Mandatory signature
442 10. s.write(b"Triggerbox developed by Giuseppe Ippolito. DOI:
443     123.456789\n")
444
445 11. # For declaring marker duration
446 12. s.write(b"SET,MRK1,3\n")
447 13. s.write(b"SET,MRK2,5\n")
448 14. s.write(b"SET,MRK3,7\n")
449
450 15. # Protocol type (rTMS, dcTMS, spTMS)
451 16. s.write(b"spTMS\n")
452
453 17. # Commands
454 18. s.write(b"1\n")
```

```

455 19. s.write(b"2\n")
456 20. s.write(b"3\n")
457
458 21. s.write(b"A\n")
459 22. s.write(b"B\n")
460 23. s.write(b"C\n")
461
462 24. # Close the serial communication
463 25. s.write(b"Z\n")
464 26. s.close()
465 27. del s

```

466 Lines 4-7 initialize the **serial communication**. Python can use:

```

467 s = Serial()
468 s.port = "COM3"
469 s.baudrate = 115200
470 s.open()

```

471 as the same as:

```

472 s = serial.Serial(port = "COM3", baudrate = 115200)
473 s.open()

```

474 It is important to change the USB port ("COM") name according to your case as explained
475 above.

476 Then, copy-paste the **mandatory signature**. Without this the following lines won't work.

```

477 s.write(b"Triggerbox developed by Giuseppe Ippolito. DOI: 123.456789\n")

```

478 After these mandatory steps, you can declare optional stimulation parameters (e.g., IPI, number
479 of pulses, markers) in any order. In this case, since we're going to use a spTMS protocol, the
480 only parameter we may be interested in is the **voluntary marker** duration. This option allows
481 you to place digital markers of a chosen duration (max 9 different lengths) at any moment. This
482 may be useful in case you need to discriminate between different stimuli's onset (e.g., neural,
483 joy, or fearful faces). Remember that voluntary markers always get delivered through BNC3.
484 Its syntax is:

```

485 s.write(b"SET,MRK\n")

```

486 In this case N is the preset number (between 1 and 9), and X is the marker length. Recalling the
487 two stimuli example, we may want three voluntary markers of different duration (e.g., 3 and 5
488 ms) to ease our analysis, later. We will write then:

```

489 s.write(b"SET,MRK1,3\n") # Preset 1: 3ms marker
490 s.write(b"SET,MRK2,5\n") # Preset 2: 5ms marker

```

491 Our last step before calling the pulse command is declaring which TMS protocol we are going
492 to use. We can choose between spTMS, dcTMS, or rTMS. Since we're using the single pulse
493 one, we will write:

```
494 s.write(b"spTMS\n")
```

495 Now that we have declared all of the necessary stimulation parameters, we can finally deliver
496 the TMS pulse. You can independently trigger the BNC1 ("1") or BNC2 ("2"). These lines
497 must be copy-pasted within your code according to your stimulation and task requirements (es.
498 prior to a certain stimulus onset).

```
499 s.write(b"1\n") # Trigger BNC1  
500 s.write(b"2\n") # Trigger BNC2
```

501 We can also individually call for voluntary markers at any time using letters. Preset 1 (MRK1)
502 is paired with letter "A" – Consequently, preset 2 with "B", 3 with "C", 4 with "D", 5 with
503 "E", 6 with "F", 7 with "G", 8 with "H", and 9 with "I". Since we declared two different
504 markers ("MRK1,3", "MRK2,5") we can trigger them using the letters "A" and "B".

```
505 s.write(b"A\n") # Deliver a voluntary marker from preset 1 (3 ms)  
506 s.write(b"B\n") # Deliver a voluntary marker from preset 2 (5 ms)
```

507 The command code also includes a line to run in case you may reset any setting and return to
508 the mandatory signature. You can then declare once again the parameter settings. The
509 command is:

```
510 s.write(b"Z\n") # Return to the beginning of the setting phase
```

511 Once the task is completed you can definitely close the serial communication between the
512 Silicon Spike device and the experimental computer, thus avoiding any potential bug.

```
513 s.close()  
514 del s
```

515 • Dual coil TMS (dcTMS)

516 This protocol allows you to trigger pairs of TMS pulses, adding the option to choose the
517 distance in ms between the two pulses. If you've read the previous section (spTMS) most of
518 this information will be redundant, aside from the "SET,IPI" parameter.

```
519 1. # Import the required packages  
520 2. from serial import Serial, SerialException  
521  
522 3. # Initialize the serial communication  
523 4. s = Serial()  
524 5. s.port = "COM3"  
525 6. s.baudrate = 115200  
526 7. s.open()  
527
```

```

528 8. # Mandatory signature
529 9. s.write(b"Triggerbox developed by Giuseppe Ippolito. DOI: 123.456789\n")
530
531 10. # For placing markers
532 11. s.write(b"SET,MRKN\n")
533
534 12. # For declaring marker duration
535 13. s.write(b"SET,IPI1,30\n")
536 14. s.write(b"SET,IPI2,50\n")
537 15. s.write(b"SET,IPI3,70\n")
538
539 16. # For declaring marker duration
540 17. s.write(b"SET,MRK1,3\n")
541 18. s.write(b"SET,MRK2,5\n")
542 19. s.write(b"SET,MRK3,7\n")
543
544 20. # Protocol type (rTMS, dcTMS, spTMS)
545 21. s.write(b"dcTMS\n")
546
547 22. # Commands
548 23. s.write(b"1\n")
549 24. s.write(b"2\n")
550 25. s.write(b"3\n")
551
552 26. s.write(b"A\n")
553 27. s.write(b"B\n")
554 28. s.write(b"C\n")
555
556 29. # Close the serial communication
557 30. s.write(b"Z\n")
558 31. s.close()
559 32. del s

```

560 Lines 4-7 initialize the **serial communication**. Python can use:

```

561 s = Serial()
562 s.port = "COM3"
563 s.baudrate = 115200
564 s.open()

```

565 as the same as:

```

566 s = serial.Serial(port = "COM3", baudrate = 115200)
567 s.open()

```

568 It is important to change the USB port ("COM") name according to your case as explained
569 above.

570 Then, copy-paste the **mandatory signature**. Without this the following lines won't work.

```

571 s.write(b"Triggerbox developed by Giuseppe Ippolito. DOI: 123.456789\n")

```


572 After these mandatory steps, you can declare optional stimulation parameters (e.g., IPI, number
573 of pulses, markers) in any order. In this case, since we're going to use a spTMS protocol, the
574 only parameter we may be interested in is the **voluntary marker** duration. This option allows
575 you to place digital markers of a chosen duration (max 9 different lengths) at any moment. This
576 may be useful in case you need to discriminate between different stimuli's onset (e.g., neural,
577 joy, or fearful faces). Remember that voluntary markers always get delivered through BNC3.
578 Its syntax is:

```
579 s.write(b"SET,MRKN,X\n")
```

580 In this case *N* is the preset number (between 1 and 9), and *X* is the marker length. Recalling the
581 three stimuli example, we may want three voluntary markers of different duration (e.g., 3, 5,
582 and 7 ms) ease our analysis, later. We will write then:

```
583 s.write(b"SET,MRK1,3\n") # Preset 1: 3ms marker  
584 s.write(b"SET,MRK2,5\n") # Preset 2: 5ms marker  
585 s.write(b"SET,MRK3,7\n") # Preset 3: 7ms marker
```

586 For dcTMS paradigms it is also important to declare the distance in ms between the two pulses
587 (IPI), delivered from BNC1 and then BNC2. Its syntax is:

```
588 s.write(b"SET,IPIN,X\n")
```

589 In this case *N* is the preset number (between 1 and 9), and *X* is the distance between pulses in
590 ms. For example, we may want to test the difference in delivering pulse pairs with different
591 latencies when the experimental stimulus occurs. We will write then:

```
592 s.write(b"SET,IPI1,30\n") # Preset 1: 30ms distance  
593 s.write(b"SET,IPI2,50\n") # Preset 2: 50ms distance  
594 s.write(b"SET,IPI3,70\n") # Preset 3: 70ms distance
```

595 Our last step before calling the pulse command is declaring which TMS protocol we are going
596 to use. We can choose between spTMS, dcTMS, or rTMS. Since we're using the double pulse
597 one, we will write:

```
598 s.write(b"dcTMS\n")
```

599 Now that we have declared all of the necessary stimulation parameters, we can finally trigger
600 the TMS device. You can independently trigger each of the declared protocols. These lines
601 must be copy-pasted within your code according to your stimulation and task requirements (es.
602 prior to a certain stimulus onset).

```
603 s.write(b"1\n") # Trigger protocol 1 (two pulses with a 30 ms distance)  
604 s.write(b"2\n") # Trigger protocol 2 (two pulses with a 50 ms distance)  
605 s.write(b"3\n") # Trigger protocol 3 (two pulses with a 70 ms distance)
```

606 We can also individually call for voluntary markers at any time using letters. Preset 1 (MRK1)
607 is paired with letter "A" – Consequently, preset 2 with "B", 3 with "C", 4 with "D", 5 with
608 "E", 6 with "F", 7 with "G", 8 with "H", and 9 with "I". Since we declared three different

609 markers (“MRK1,3”, “MRK2,5”, “MRK3,7”) we can trigger them using the letters “A”, “B”,
610 and “C”.

```
611 s.write(b"A\n") # Deliver a voluntary marker from preset 1 (3 ms)
612 s.write(b"B\n") # Deliver a voluntary marker from preset 2 (5 ms)
613 s.write(b"C\n") # Deliver a voluntary marker from preset 3 (7 ms)
```

614 The command code also includes a line to run in case you may reset any setting and return to
615 the mandatory signature. You can then declare once again the parameter settings. The
616 command is:

```
617 s.write(s,"Z\n") # Return to the beginning of the setting phase
```

618 Once the task is completed you can definitely close the serial communication between the
619 Silicon Spike device and the experimental computer, thus avoiding any potential bug.

```
620 s.close()
621 del s
```

622

623 ● repetitive TMS (rTMS)

624 This protocol allows you to trigger trains of TMS pulses, adding the option to choose the
625 distance in ms between the two pulses and the number of pulses within each train. If you’ve
626 read the previous section (dcTMS) most of this information will be redundant, aside from the
627 “SET,nPULS” parameter.

```
628 1. # Import the required packages
629 2. from serial import Serial, SerialException
630
631 3. # Initialize the serial communication
632 4. s = Serial()
633 5. s.port = "COM3"
634 6. s.baudrate = 115200
635 7. s.open()
636
637 8. # Mandatory signature
638 9. s.write(b"Triggerbox developed by Giuseppe Ippolito. DOI: 123.456789\n")
639
640 10. # For placing markers
641 11. s.write(b"SET,MRK\n")
642
643 12. # For declaring the distance between the two TMS pulses
644 13. s.write(b"SET,IPI1,80\n")
645 14. s.write(b"SET,IPI2,100\n")
646 15. s.write(b"SET,IPI3,120\n")
647
648 16. # For declaring the number of pulses within each train
649 17. s.write(b"SET,nPULS1,5\n")
650 18. s.write(b"SET,nPULS2,5\n")
```

```

651 19. s.write(b"SET,nPULS3,5\n")
652
653 20. # For declaring marker duration
654 21. s.write(b"SET,MRK1,3\n")
655 22. s.write(b"SET,MRK2,5\n")
656 23. s.write(b"SET,MRK3,7\n")
657
658 24. # Protocol type (rTMS, dcTMS, spTMS)
659 25. s.write(b"rTMS\n")
660
661 26. # Commands
662 27. s.write(b"1\n")
663 28. s.write(b"2\n")
664 29. s.write(b"3\n")
665
666 30. s.write(b"A\n")
667 31. s.write(b"B\n")
668 32. s.write(b"C\n")
669
670 33. # Close the serial communication
671 34. s.write(b"Z\n")
672 35. s.close()
673 36. del s

```

674 Lines 4-7 initialize the **serial communication**. Python can use:

```

675 s = Serial()
676 s.port = "COM3"
677 s.baudrate = 115200
678 s.open()

```

679 as the same as:

```

680 s = serial.Serial(port = "COM3", baudrate = 115200)
681 s.open()

```

682 It is important to change the USB port ("COM") name according to your case as explained
683 above.

684 Then, copy-paste the **mandatory signature**. Without this the following lines won't work.

```

685 s.write(b"Triggerbox developed by Giuseppe Ippolito. DOI: 123.456789\n")

```

686 After these mandatory steps, you can declare optional stimulation parameters (e.g., IPI, number
687 of pulses, markers) in any order. In this case, since we're going to use a spTMS protocol, the
688 only parameter we may be interested in is the **voluntary marker** duration. This option allows
689 you to place digital markers of a chosen duration (max 9 different lengths) at any moment. This
690 may be useful in case you need to discriminate between different stimuli's onset (e.g., neural,
691 joy, or fearful faces). Remember that voluntary markers always get delivered through BNC3.
692 Its syntax is:

```
693 s.write(b"SET,MRKN,X\n")
```

694 In this case N is the preset number (between 1 and 9), and X is the marker length. Recalling the
695 three stimuli example, we may want three voluntary markers of different duration (e.g., 3, 5,
696 and 7 ms) ease our analysis, later. We will write then:

```
697 s.write(b"SET,MRK1,3\n") # Preset 1: 3ms marker  
698 s.write(b"SET,MRK2,5\n") # Preset 2: 5ms marker  
699 s.write(b"SET,MRK3,7\n") # Preset 3: 7ms marker
```

700 For rTMS paradigms it is important to declare the number of pulses constituting each train,
701 triggered simultaneously from BNC1 and BNC2. Its syntax is:

```
702 s.write(b"SET,nPULSN,X\n")
```

703 In this case N is the preset number (between 1 and 9), and X is the number of pulses within the
704 train. For example, we may want to test the difference in delivering trains of 4, 5, or 6 pulses
705 prior to the stimulus occurrence. We will write then:

```
706 s.write(b"SET,nPULS1,4\n") # Preset 1: 4 pulses in each train  
707 s.write(b"SET,nPULS2,5\n") # Preset 2: 5 pulses in each train  
708 s.write(b"SET,nPULS3,6\n") # Preset 3: 6 pulses in each train
```

709 Additionally, in rTMS paradigms it is also important to declare the distance in ms between the
710 two pulses (IPI), delivered from BNC1 and then BNC2. Its syntax is:

```
711 s.write(b"SET,IPIN,X\n")
```

712 In this case N is the preset number (between 1 and 9), and X is the distance between pulses in
713 ms. For example, we may want an 80 ms interval in the first preset (4 pulses), a 100 ms interval
714 for the second one (5 pulses), and a 120 ms interval for the third one (6 pulses). We will write
715 then:

```
716 s.write(b"SET,IPI1,80\n") # Preset 1: 80ms distance  
717 s.write(b"SET,IPI2,100\n") # Preset 2: 100ms distance  
718 s.write(b"SET,IPI3,120\n") # Preset 3: 120ms distance
```

719 Our last step before calling the pulse command is declaring which TMS protocol we are going
720 to use. We can choose between spTMS, dcTMS, or rTMS. Since we're using the double pulse
721 one, we will write:

```
722 s.write(b"rTMS\n")
```

723 Now that we have declared all of the necessary stimulation parameters, we can finally trigger
724 the TMS device. You can independently trigger each of the declared protocols. These lines
725 must be copy-pasted within your code according to your stimulation and task requirements (es.
726 prior to a certain stimulus onset).

```
727 s.write(b"1\n") # Protocol 1 (one train of 4 pulses with a 80ms distance)  
728 s.write(b"2\n") # Protocol 2 (one train of 5 pulses with a 100ms distance)
```

```
729 s.write(b"3\n") # Protocol 3 (one train of 6 pulses with a 120ms distance)
```

730 We can also individually call for voluntary markers at any time using letters. Preset 1 (MRK1)
731 is paired with letter “A” – Consequently, preset 2 with “B”, 3 with “C”, 4 with “D”, 5 with
732 “E”, 6 with “F”, 7 with “G”, 8 with “H”, and 9 with “I”. Since we declared three different
733 markers (“MRK1,3”, “MRK2,5”, “MRK3,7”) we can trigger them using the letters “A”, “B”,
734 and “C”.

```
735 s.write(b"A\n") # Deliver a voluntary marker from preset 1 (3 ms)  
736 s.write(b"B\n") # Deliver a voluntary marker from preset 2 (5 ms)  
737 s.write(b"C\n") # Deliver a voluntary marker from preset 3 (7 ms)
```

738 The command code also includes a line to run in case you may reset any setting and return to
739 the mandatory signature. You can then declare once again the parameter settings. The
740 command is:

```
741 s.write(b"Z\n") # Return to the beginning of the setting phase
```

742 Once the task is completed you can definitely close the serial communication between the
743 Silicon Spike device and the experimental computer, thus avoiding any potential bug.

```
744 s.close()  
745 del s
```