2024

Trigger Related Manual for EEG device in ELB407

WITH AND WITHOUT STIMTRACKER

MINGROU GUO (PSY)

Table of Contents

1 With StimTracker *2*

1.1 Introduction 2

1.2 Hardware 2

1.2.1 Visual 3

1.2.2 Auditory 4

1.2.3 Microphone 6

1.2.4 use USB to send triggers 6

1.3 USB Trigger - Scripts 7

1.3.1 PsychoPy 7

1.3.2 Matlab 10

2 Without StimTracker *11*

2.1 Hardware 11

2.2 USB Trigger - Scripts 12

2.2.1 Matlab 12

2.2.2 Python? Failed. 13

1 With StimTracker

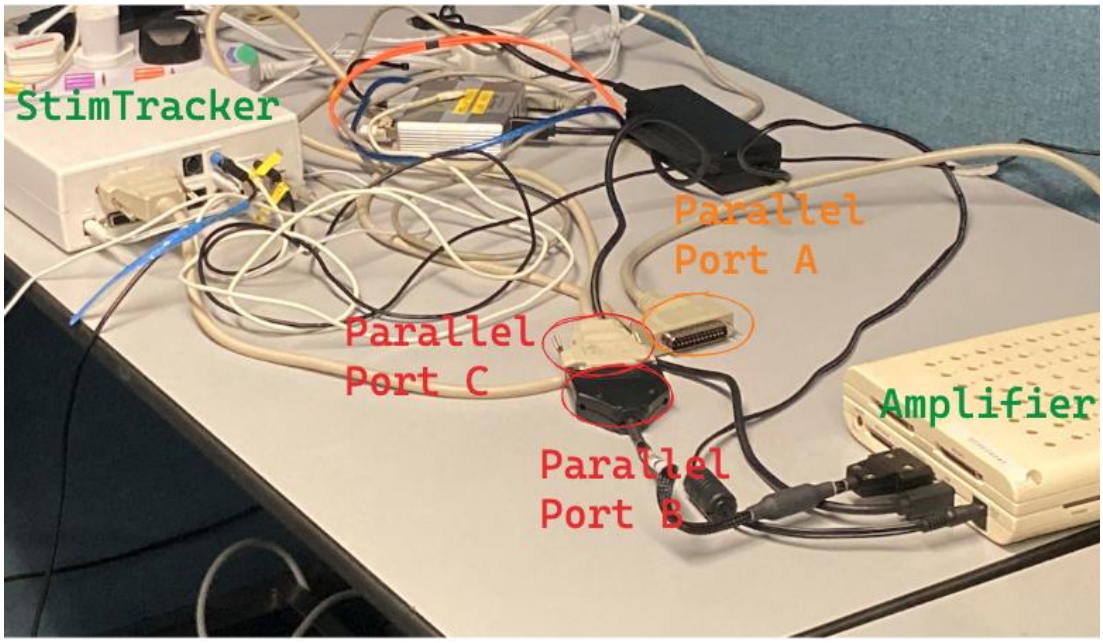
1.1 Introduction

- StimTracker is a box that can assist us in achieving accurate timing of triggers. We can receive triggers through USB (from the display computer to the recording computer) or via a sensor (from the sensor to the recording computer).

1.2 Hardware

- Press the button to turn on the box.

- Parallel Port A is connected to the display computer outside.
- Parallel Port B is connected to the amplifier.
- Parallel Port C is connected to the StimTracker Box.
- Therefore: If you only want to use the sensors to send triggers, you need to connect Parallel Port **C** and **B** together so that the StimTracker can send sensor-related triggers to the amplifier/EEG Recording Computer.

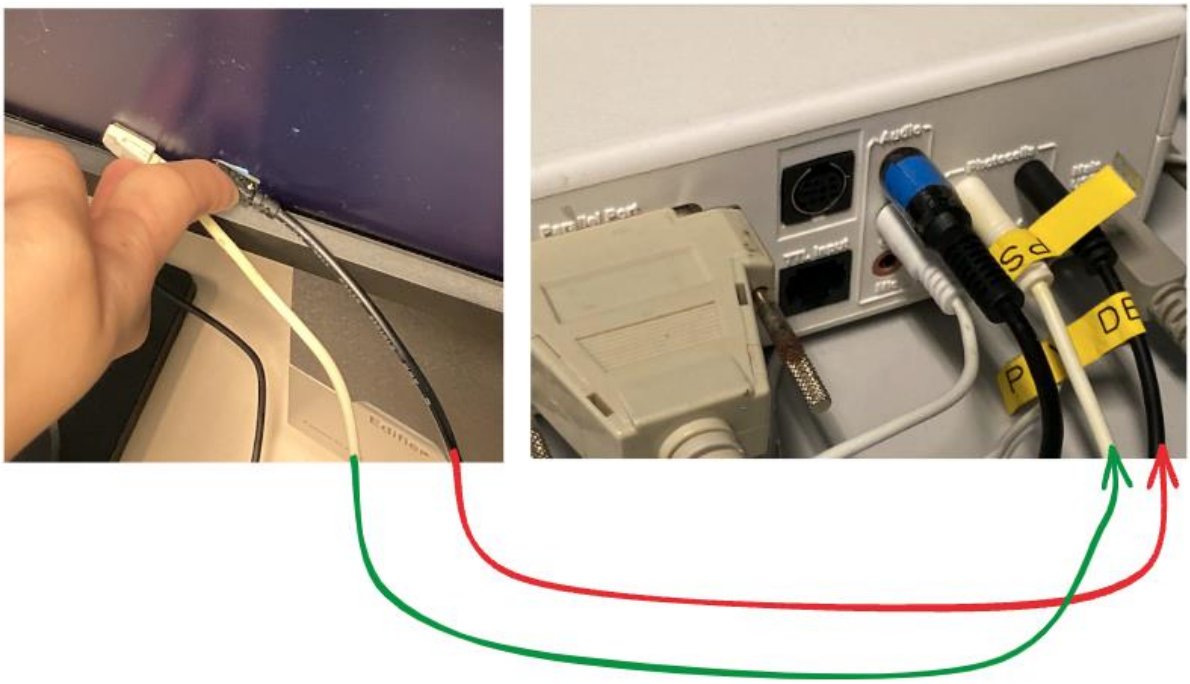


1.2.1 Visual

- Connection:

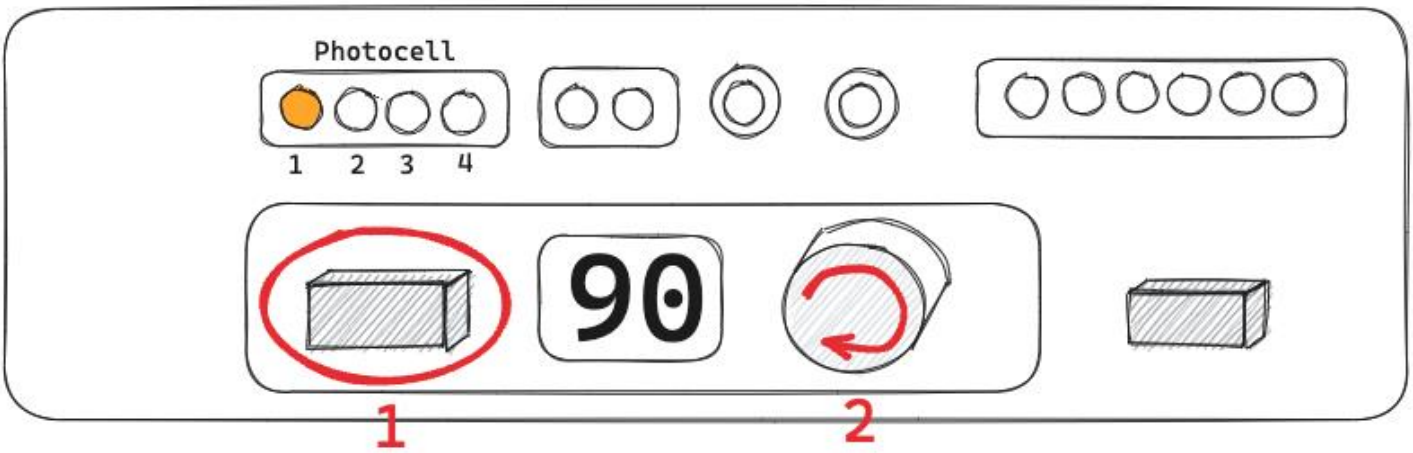
  - At the back of the box, connect the light sensors.

  - You can use up to 4 light sensors.



- In your script, remember to include a "flash" wherever you want to send a trigger. For example, you can use a 300ms-long small white square below.

- Setting the threshold for the light sensor: At the front of the box:
    - First, press button1 once to ensure the 1st photocell is on.
    - Then, use button2 to scroll and set the threshold to **>90**.
    - Repeat the same procedure for other light sensors (e.g., 2, 3, 4) if necessary
    - Each photocell will send different triggers to the amplifier (i.e., s1, s2, s3, s4).



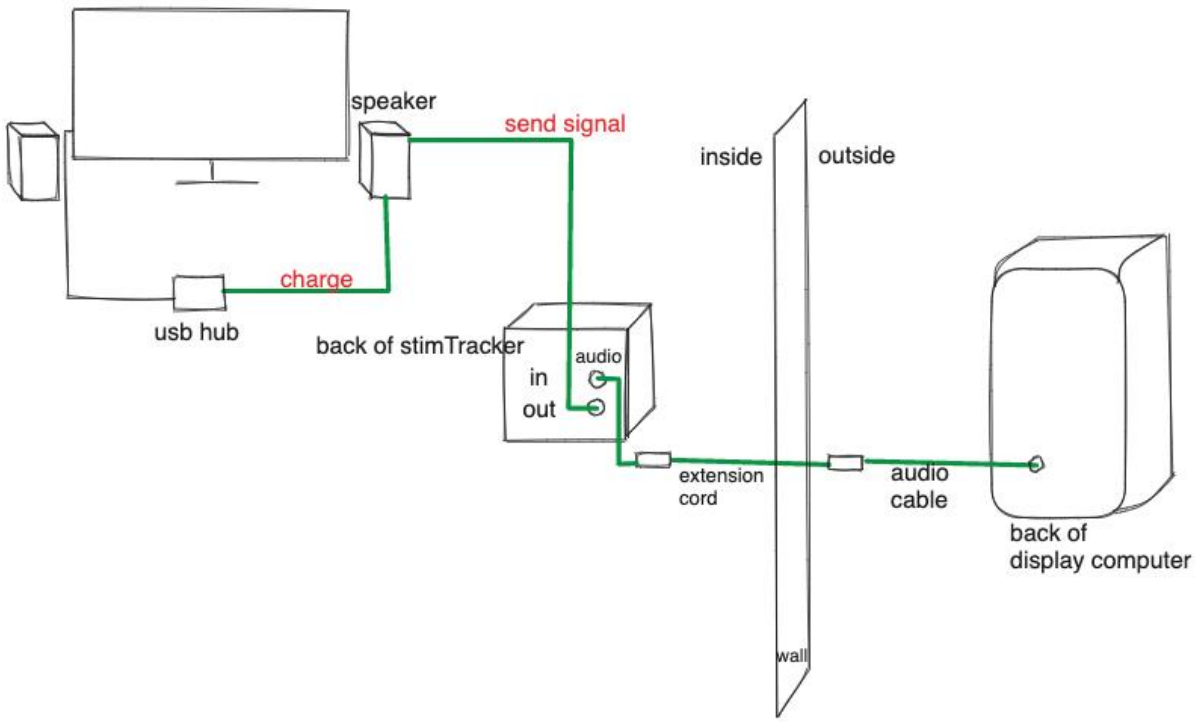
- dial test delay: ~30ms

1.2.2 Auditory

- Connection:

  - At the back of the box, connect the speaker (out).

  - At the back of the box, connect the display computer (in) via an extension cord.



- Setting the threshold:

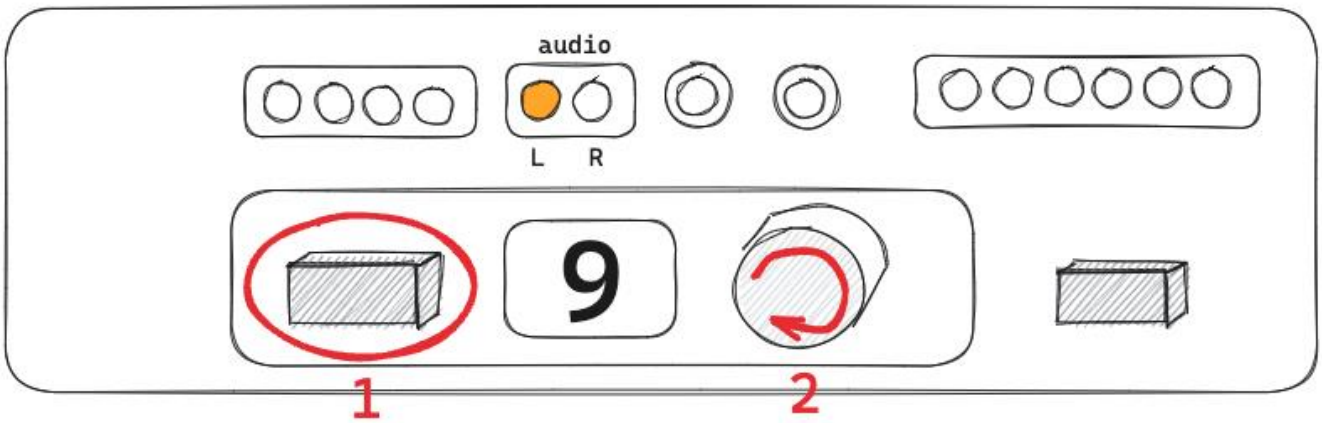
  - At the front of the box:

  - Press button1 several times to ensure the 1st audio bulb is on.

  - Scroll button2 to set the threshold to **<10**.

  - Repeat the same procedure for the R speaker if necessary.

  - The auditory trigger displayed on the recording computer is s8.

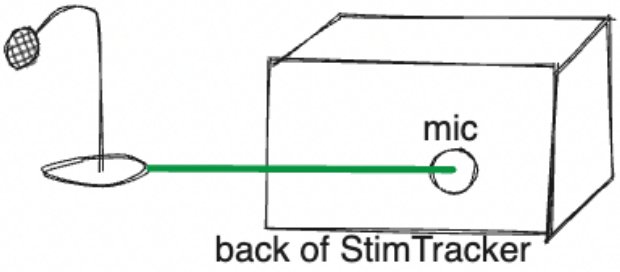


- dial test: delay ~10ms

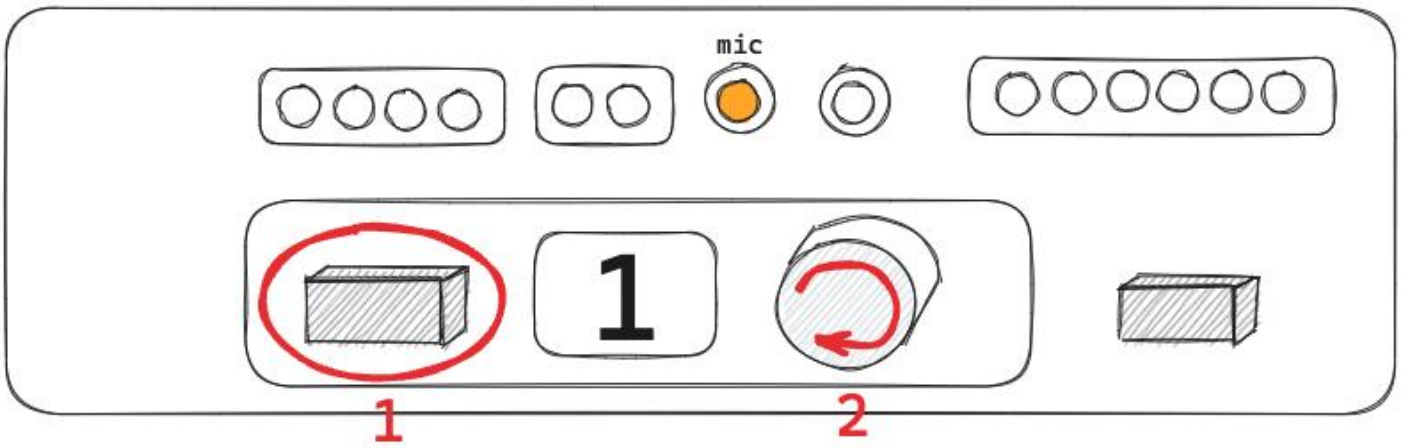
- Please use this set of speakers.

1.2.3 Microphone

- Connection: Directly connect the microphone and the StimTracker.



- **ATTENTION:** If you want to perform audio recording, please use another microphone. The current microphone is only for trigger sending purposes!
- Setting the threshold: At the front of the box:
  - Press button1 several times to ensure the microphone bulb is on.
  - Scroll button2 to set the threshold to **=1**.
  - The mic trigger displayed on the recording computer is `s16`.



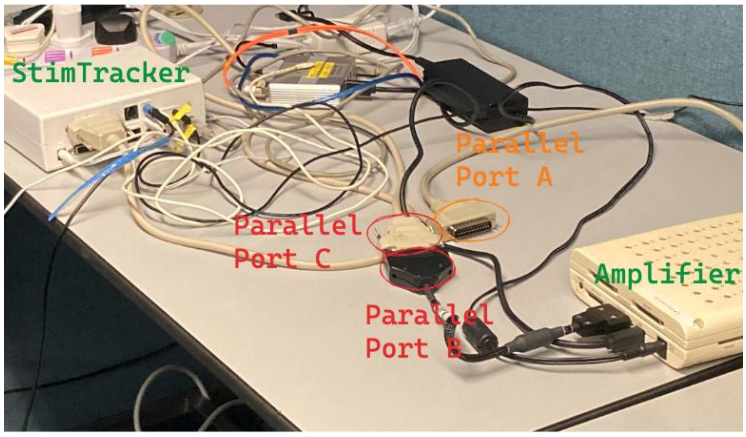
- dial test (between mic and audio): 10ms

1.2.4 use USB to send triggers

- Parallel Port A is connected to the display computer outside.
- Parallel Port B is connected to the amplifier.
- Parallel Port C is connected to the StimTracker Box.
- Therefore:

  - If you want to send triggers via StimTracker, you should connect Parallel Port **C** and **B** together.

- For Python codes, please refer to [USB Trigger - Scripts/PsychoPy](#).
- For MATLAB codes, please refer to [USB Trigger - Scripts/MATLAB](#).



1.3 USB Trigger - Scripts

1.3.1 PsychoPy

- ***trigger codes:*** It appears that with this method, we can only send a total of 7 different kinds of triggers.

in script	in EEG
1	32
2	64
3	96
4	128
5	160
6	192
7	224

1.3.1.1 Coder

- You can directly use the official sample triggers as provided in the official documentation, see [psychopy_pyxid_test](#) below

```
import pyxid2
import time
# get a list of all attached XID devices
devices = pyxid2.get_xid_devices()
dev = devices[0] # get the first device to use
print(dev)
dev.reset_base_timer()
dev.reset_rt_timer()
#dev._send_command('iuA1', 0)
```

```
#dev.enable_usb_output('K', True)
if dev.is_response_device():
    print ("Press a key!")
    while not dev.has_response():
        dev.poll_for_response()
    response = dev.get_next_response()
    print(response)
    dev.clear_response_queue()
dev.set_pulse_duration(300)

sleep_flash = .3
for bm in range(0, 16):
    mask = 2 ** bm
    print("activate_line bitmask: ", mask)
    #dev.activate_line(lines=[1,3,5,7,9,11,13,15])
    dev.activate_line(bitmask=mask)
    time.sleep(sleep_flash)
```

- When indicating the trigger code, replace `mask = 2 ** bm` with `mask =` followed by an integer ranging from `1` to `7`. The corresponding code in EEG will range from 32 to 224, with an interval of 32.
- The above conclusion has been tested using the revised script provided by Mingrou: psychopy_pyxid_test_GMR.

1.3.1.2 Builder

1. You can write your own code based on my example: oddball_trigger_3Apr_GMR.psyexp.

2. In the first routine of the experiment (e.g., *instruction*), you should add a *code* component (e.g., *code_8*) and copy the following codes to the *Begin Experiment* tab to establish a connection between Psychopy and the EEG system:

```
import pyxid2
import time
# get a list of all attached XID devices
devices = pyxid2.get_xid_devices()
dev = devices[0] # get the first device to use
print(dev)
dev.reset_base_timer()
dev.reset_rt_timer()
if dev.is_response_device():
    print ("Press a key!")
    while not dev.has_response():
        dev.poll_for_response()

    response = dev.get_next_response()
    print(response)
    dev.clear_response_queue()
dev.set_pulse_duration(1000)
sleep_flash = 1
```

3. If you are going to use sensor-induced triggers (visual/auditory/microphone), I strongly recommend sending a special event trigger to indicate the start of the experiment. This helps avoid confusion with the sensor triggers. Therefore, in the first routine of the experiment, you can add the following code in the *Begin Routine* tab within the same *code* component as the previous *code* segment:

```
mask = 3 #trigger: 96
print("activate_line bitmask: ", mask)
#dev.activate_line(lines=[1,3,5,7,9,11,13,15])
dev.activate_line(bitmask=mask)
# time.sleep(sleep_flash)
```

- Here, I have written `mask = 3`, which will send a trigger `s96` to the EEG system. I have also included the same code in the *Begin Routine* tab of the *code component* for the last routine (e.g., *end*) to indicate the end of the experiment. Please note that my other events/sensors do not send this code. If you plan to use this code in your main experiment, make sure to adjust the begin/end experiment code accordingly.
- **ATTENTION**: Please remember to remove or comment out the `time.sleep(sleep_flash)` line. Do not copy this line from the official sample. Otherwise, Psychopy may not display the following routines, although it will still send the required triggers continuously. 😵

4. Alright, now we will send the trigger for our first trial. In the *code component* (e.g., *code_6*) of this routine (e.g., *standard*), you can write the code in the *Begin Routine* tab as follows:

```
condition = allTrials[triali]
if condition == 2:
    continueRoutine = False
else:
    continueRoutine = True
    triali += 1
    ### trigger s32 for standard condition
    mask = 1 #trigger: 32
    print("activate_line bitmask: ", mask)
    dev.activate_line(bitmask=mask)
```

- **ATTENTION**: Please make sure to place the trigger-related codes and the paradigm-related codes **within** the **same** *code component* to avoid the possibility of ghost triggers. Mixing them in different code components can lead to unintended triggers being sent.
- the code for trigger sending is:

```
mask = 1 #trigger: 32
print("activate_line bitmask: ", mask)
dev.activate_line(bitmask=mask)
```

- You can simply copy the code and modify the mask as needed.

- I have included the code within an if-else loop of the paradigm design because I want to send different triggers for different conditions. You can implement similar logic based on your requirements.
- Once again, remember to remove or comment out the `time.sleep(sleep_flash)` line.
- Just for your reference, when I have another condition (e.g., *deviant*), I use similar code as mentioned above, but with a different `mask` value.

```
condition = allTrials[triali]
if condition == 1:
    continueRoutine = False
else:
    continueRoutine = True
    triali += 1
    ### trigger s64 for deviant condition
    mask = 2 # trigger: 64
    print("activate_line bitmask: ", mask)
    dev.activate_line(bitmask=mask)
```

5. Here is similar code that indicates the ending of the experiment. You can place it in the *Begin Routine* tab of the *Code Component* (e.g., *code_9*) of the last routine (e.g., *end*):

```
mask = 3 #trigger: 96
print("activate_line bitmask: ", mask)
#dev.activate_line(lines=[1,3,5,7,9,11,13,15])
dev.activate_line(bitmask=mask)
# time.sleep(sleep_flash)
```

1.3.2 Matlab

- **ATTENTION**: Please ensure you are using a version after 2019a for the MATLAB script to work properly.
- Official script: matlab_output_sample
- Mingrou's tested script:
    - matlab_output_sample_GMR
    - Use device_for2019b in your working directory (remember to include this .mat file in the working directory).

```
clc
clear device
load('device_for2019b.mat');
setPulseDuration(device, 1000)
write(device,sprintf("mh%c%c", 255, 0), "char")
pause(2)
function byte = getByte(val, index)
        byte = bitand(bitshift(val,-8*(index-1)), 255);
end
function setPulseDuration(device, duration)
        write(device,sprintf("mp%c%c%c%c",getByte(duration,1),getByte(duration,2), ...
        getByte(duration,3),getByte(duration,4)), "char")
end
```

- To define the trigger number in the MATLAB script, you can modify the following line:
  - `write(device,sprintf("mh%c%c", 255, 0), "char")`
  - You can change the `255` to any integer value between `0` and `255`. (ATTENTION! Note that `0` means you are turning off the trigger.)

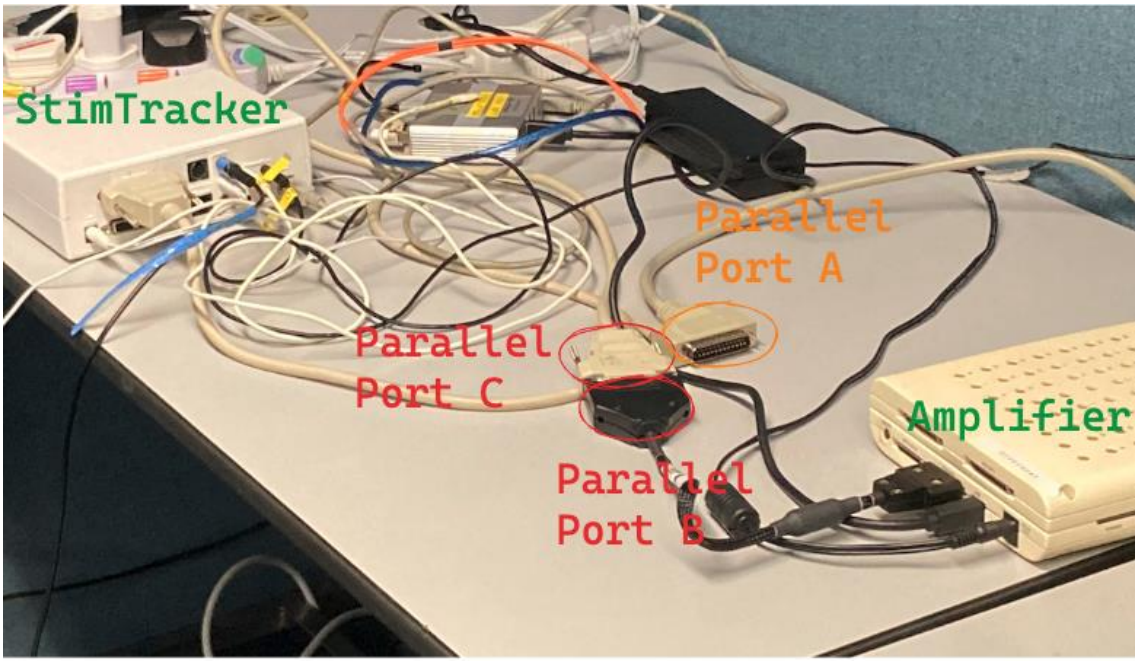
2 Without StimTracker

2.1 Hardware

- Parallel Port A is connected to the display computer outside.
- Parallel Port B is connected to the amplifier.
- Parallel Port C is connected to the StimTracker Box.
- Therefore:

  - If you want to send triggers directly from the display computer to the amplifier, you should connect Parallel Port **B** and **A** together.

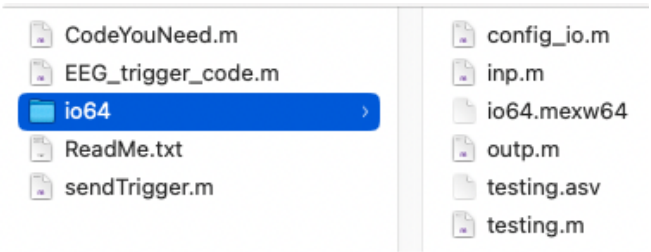
  - For MATLAB script, please refer to Without StimTracker/MATLAB.



2.2 USB Trigger - Scripts

2.2.1 Matlab

- **ATTENTION!** Please note that you should use a MATLAB version older than 2014a for the mentioned demo script to work properly.
- Remember to set the path (including subfolders) for the folder named cuhk Trigger, which contains the files shown in the image:



- In the demo script oddball_cogent_v1_3_2_2, the key settings are as follows:

1. At the beginning of the experiment: You can simply copy the following text:

```
config_io;
ioObj=io64;
status=io64(ioObj);
address=hex2dec('03BC');
outp(address,0);
```

2. for each stimulus/response:

2.1 Before:

```
trigger_stimuli=2;
outp(address,trigger_stimuli); % turn on the trigger
```

You can define the `trigger_stimuli` variable with values ranging from `0` to `255` (excluding 0, as it is reserved for turning off the trigger).

2.2 After:

```
outp(address,0);% turn off the trigger
```

2.2.2 Python? Failed.

(be aware of version issue!)
(python failed)