MEDUSA©: An Innovative Software Ecosystem to Accelerate BCI and Cognitive Neuroscience Experimentation

Dr. Eduardo Santamaría-Vázquez (eduardo.santamaria.vazquez@uva.es)
Dr. Víctor Martínez-Cagigal (victor.martinez.cagigal@uva.es)
Msc. Sergio Pérez-Velasco (sergio.perezv@uva.es)

Note: all the resources used in this workshop are freely available at the following GitHub repository:

https://github.com/medusabci/gbcic_2024_medusa_workshop

Contents

Module 1: Configure MEDUSA© platform

Task 1.1: Stream a fake LSL with the signal generator

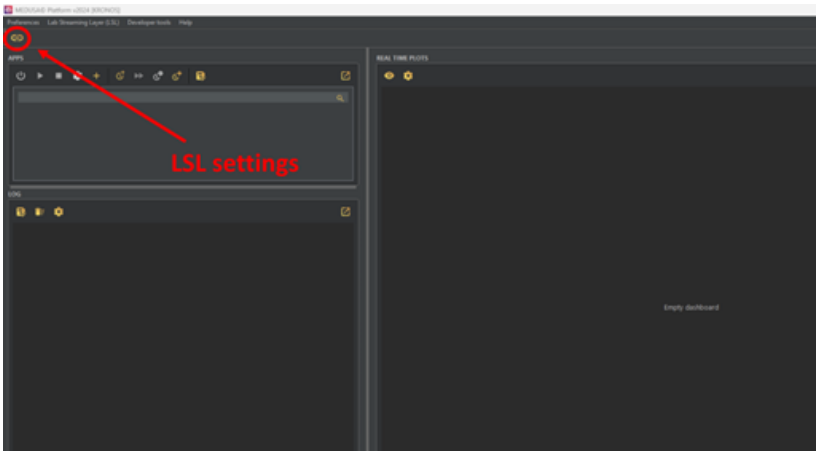
1. Run the latest version of Signal Generator.
2. Navigate through the installer to install the program.

Once installed, open Signal Generator to generate the fake signal:

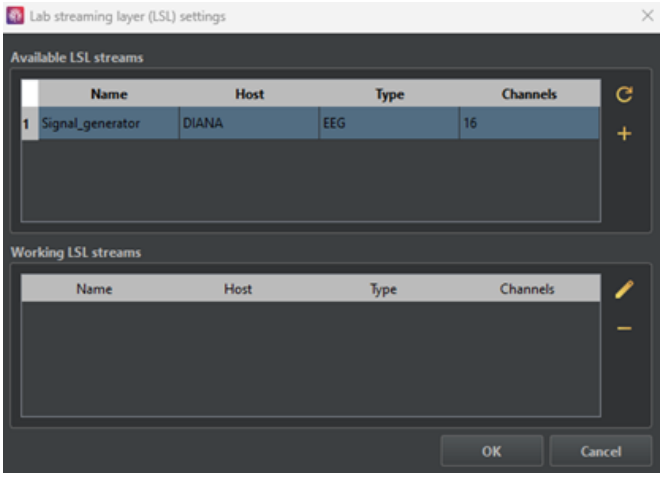
1. Customize the stream name “EEG_artificial_*YOUR_SURNAME*”.
   Number of channels: 9
   Name of channels: C3;C4;CZ;F3;F4;F7;F8;Ch1;Ch2
   Uncheck the option of “Real-time pink noise generation”
2. Press Start to stream the fake signal through LSL.
3. Press Stop to stop the transmission whenever you want.

Task 1.2: Connect MEDUSA© platform to the LSL signal

1. Double click on MEDUSA© platform (gbcic_2024).
2. Click the button to configure the LSL settings



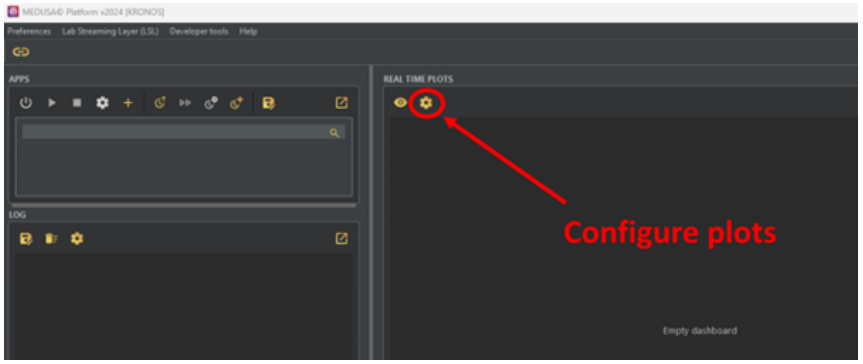
3. In Available LSL streams, select your LSL signal (the one we generated before), and press the button + to indicate you want to use it.



4. Configure the stream settings. Here we can change back the name of the last 2 channels to their original “FP1”, and “FP2”. Press OK when finished.
5. In this step, you should see your LSL stream marked as Working LSL stream.

Task 1.3: Configure plots panel

1. Inside the panel REAL TIME PLOTS, press the button resembling a gear to configure the visualization.

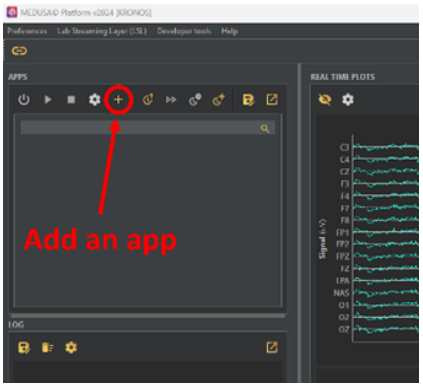


2. In this configuration, you can customize the number of different plots to be displayed in real-time by adding and removing elements inside the grid.
3. As an example, we are going to create three different visualizations: temporal, spectral and topography.
    a) Fix the grid size to 8×8.
    b) Press the + symbol to create the first element and drag and drop its corner to occupy half of the screen.
    c) Then, double-click the element we just created to configure it. Select **TimePlotMultichannel** to create a temporal multi-channel visualization (as an EEG).
    d) Press the + symbol to create the second element and drag and drop its corner to occupy the left side of the remaining screen.
    e) Double-click the element we just created and select **PSDPlot** to create a single-channel spectral visualization of the power spectral density (PSD).
    f) Press the + symbol to create the third element and drag and drop its corner to occupy the rest of the screen.
    g) Double-click the element we just created and select **TopographyPlot** to create a topography visualization of the relative power of the selected frequency band.
    h) Press OK to apply the changes.
4. Press the button resembling an eye to visualize the charts.

Module 2: Learn to work with apps

Task 2.1: Install an app: “Recorder”

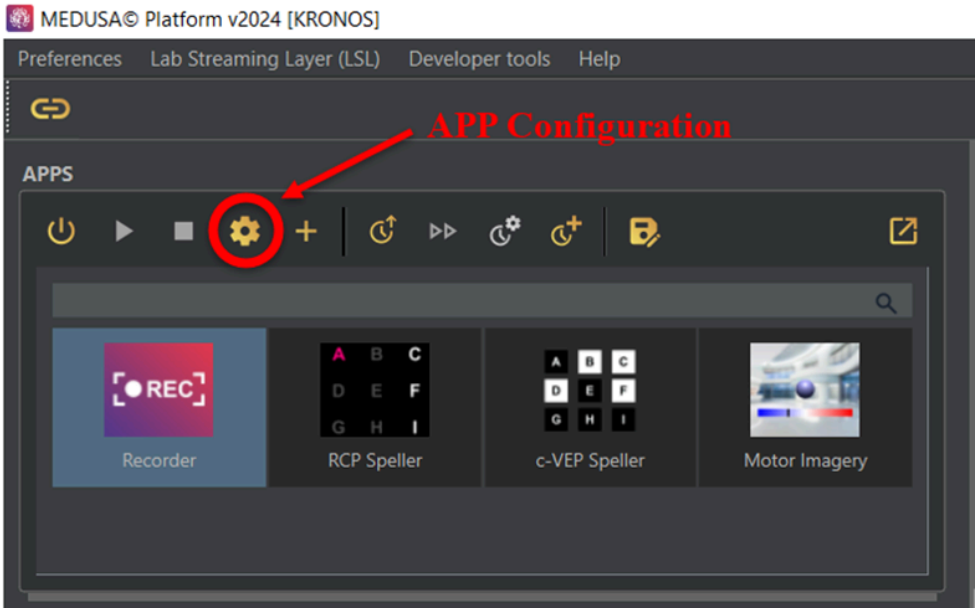
1. In MEDUSA© Platform, click on the button inside the APPLICATIONS panel.



2. Find the downloaded “rec.app” file and select it.
3. MEDUSA© Platform will install the app and link it to the logged account. When installed, the icon for the app will be displayed in the APPS panel.

Task 2.2: Experiment with the “Recorder”

1. Select the “Recorder” app and click on the configuration button.



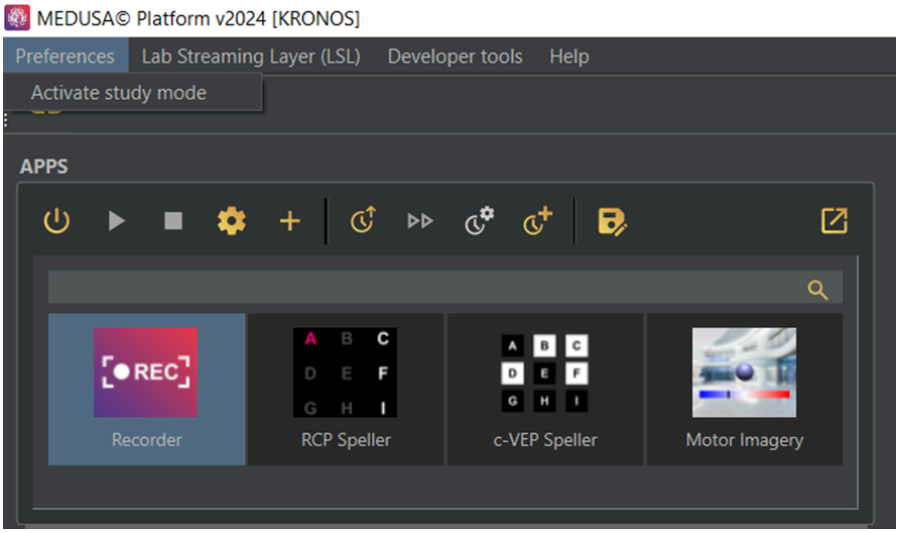
2. The settings of the app have now been prompted. We will change the duration of the recording plan of “eyes_open” and “eyes_closed” from 120 to 5 seconds.
3. Now SAVE the configuration with the name “workshop_rec” by clicking the Save button.

4. Close the settings by clicking on the Done button.
5. Launch the APP by clicking the power button of the APPs panel.
6. When the APP is loaded, run the experiment by clicking the PLAY button.
7. You can add custom markers of events by clicking the “B”, “M”, and “N” keys during the recording.
8. When the experiment ends, click the STOP button.
9. You will be prompted to save the signal. We will save it in the default folder with the default name.

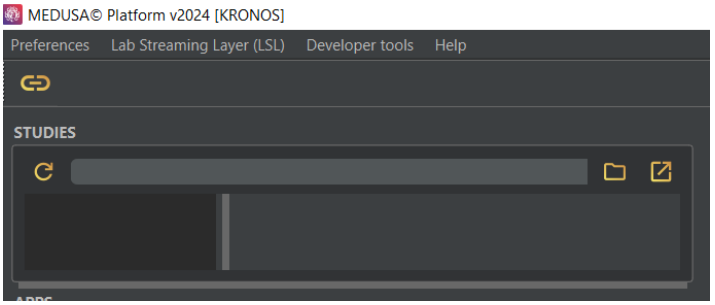
Module 3: Advanced functions for experimentation

Task 3.1: Select a study

1. First you will need to activate the study mode by clicking Preferences ⇒ Activate study mode



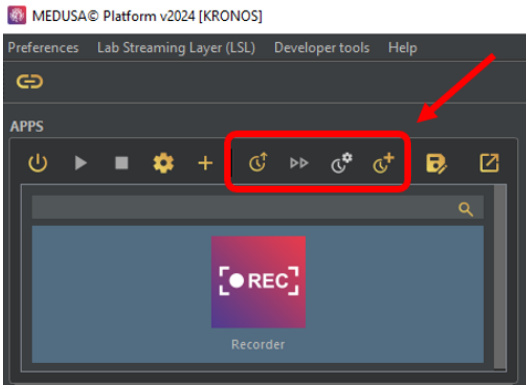
2. Now we can navigate through the STUDIES panel



3. Now here you can select a custom direction of your computer where previously you have configured folders for your participants and sessions. We will create a new folder called “workshop_study” and select it.

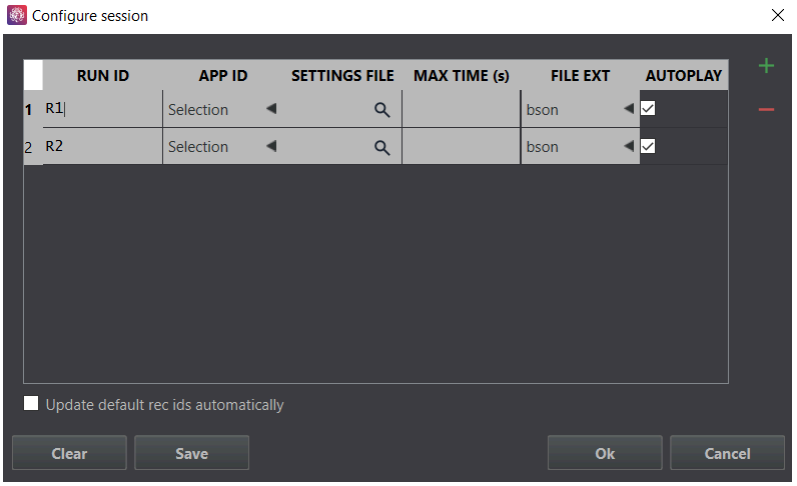
Task 3.2: Create a session

1. We will work with the Session related options located in the APPs panel.



2. We will create a new session by clicking the button that has a **+** symbol on it.

3. You will be prompted with the configure session panel. Here we will click on the **+** button two times to add two different runs.



4. Now select as APP ID the Recording app, and the settings file the previous file we saved as “workshop_rec.json”.
5. Uncheck the AUTOPLAY option of the second recording. It is checked by default.
6. Launch the session by clicking the double PLAY button of the session icons.

Module 4: ERP-based paradigms

Task 4.1: Configure the "RCP Speller" app

1. Select the RCP Speller app and click on the gear button to access its settings.
2. We are going to modify the following settings:

   Number seqs.: **3** (This is the number of times that each row and column will be highlighted)

   Target: **B; C; I** (These are the targets to follow during the calibration)
3. Press Done to close the window with these changes saved.

Task 4.2: Run the paradigm: calibration and test modes

1. Launch the App, wait for it to load, and start the experiment.
2. When the experiment finishes save the recording with the default name.
3. Now we need to access the settings of the RCP Speller app. Here you will open the Train models panel. Now we load the previous file by clicking on browse files and selecting our previous recording.
4. We move to the ERP MODEL tab. We click on "TRAIN COMMAND DECODING MODEL" and proceed with this default configuration of Settings of the rLDA model. We will train a classifier by clicking on "Train". We will save the model with the name "workshop_rlda_model" by clicking on "Save model". We now close the Train models panel.
5. We will now select in the "ERP model" this model by clicking on the Browse button.
6. Change the mode to "Test" and click "Done2 to close the window with these settings configured.
7. Launch the App, wait for it to load, and start the test run.

Task 4.3: [Google Colab] Processing ERP signals

Follow the instructions of the jupyter notebook: https://colab.research.google.com/github/medusabci/medusa-tutorials/blob/master/erp_spellers_tut.ipynb

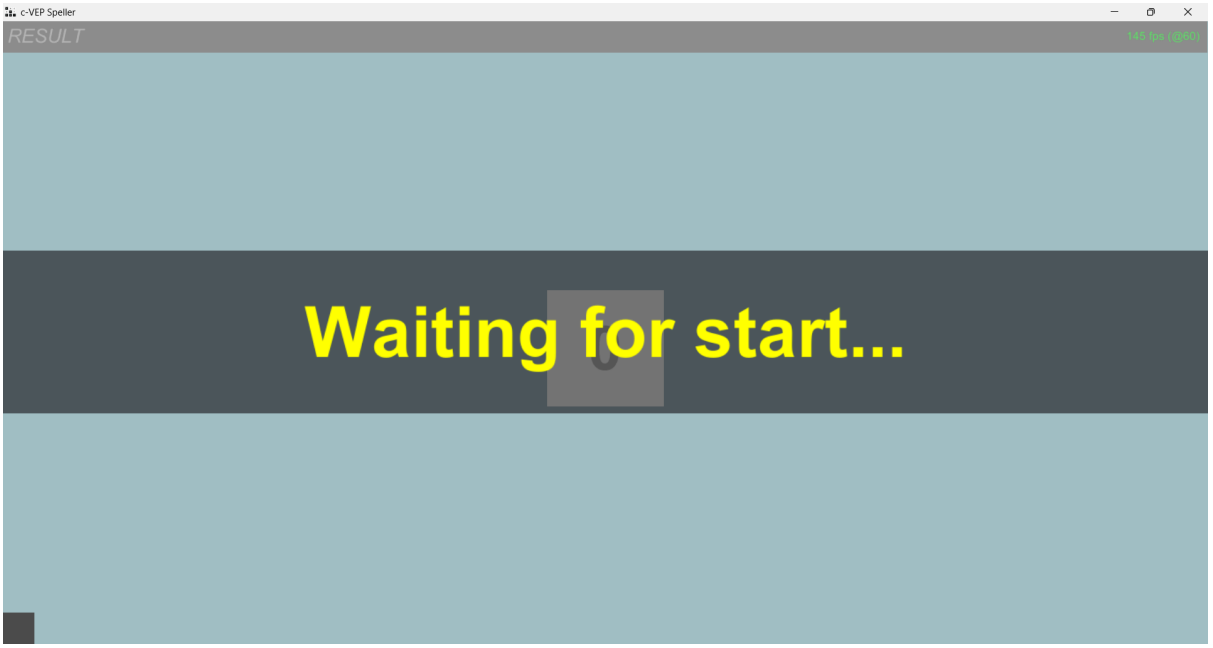
Module 5: c-VEP-based paradigms

Task 5.1: Configure the "c-VEP Speller" app

1. Select the "c-VEP Speller" app and click on the gear button to access its settings.
2. We are going to modify some settings:
    a. Change the train trials to 2 (a trial is composed of 10 cycles, or repetitions of the same code)
    b. Change the target FPS to 120 Hz (this means that the code will be displayed at that frequency, so it's faster than 60 Hz).
    c. On "Encoding and matrix", click on "Update encoding". Observe the code that we are going to use, a binary m-sequence of 63 samples (bits). The app also shows the encoding of the 16 commands, with temporarily shifted versions of the original m-sequence.
    d. Let's change the "Number of rows" to 10 and the "Number of columns" to 4. We would like to have a total of 40 commands in our application. Press "Update encoding" again. What happened? It seems that our m-sequence of 63 bits cannot encode all of those commands, we would need to use a longer code.
    e. Specify 4 rows and 4 columns again and press "Update encoding".

Task 5.2: Run the paradigm: calibration and online modes

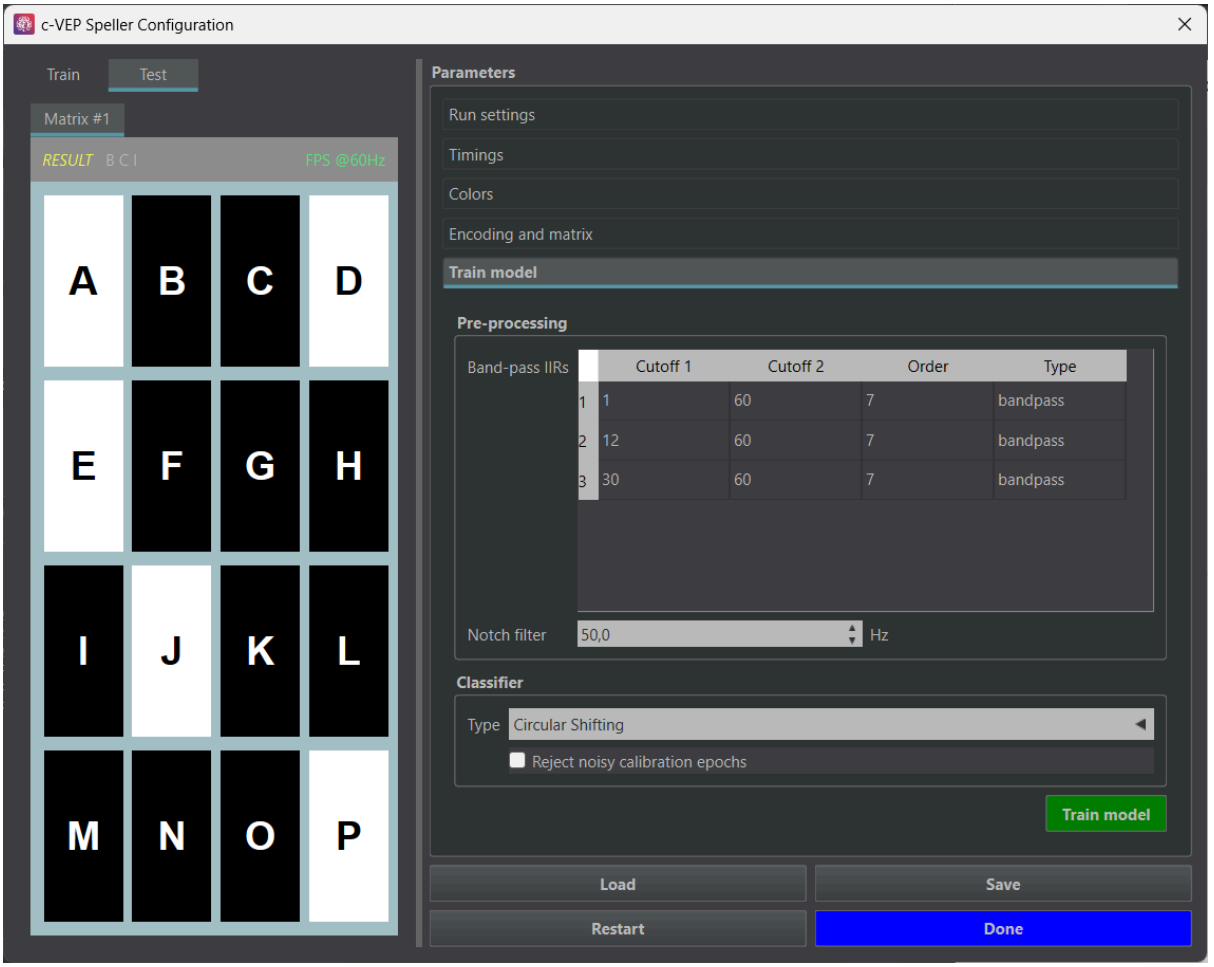
3. Everything is ready for the calibration, so press on "Done".
4. Launch the app by clicking on the power button within the apps panel.
5. When the app is launched and shows "Waiting for start…", press the play button in MEDUSA Platform to start the stimulation.



6. As shown, each trial begins by highlighting the target command (which displays the original m-sequence) in magenta. Then, the trial continues for 10 cycles or repetitions of the same m-sequence. When the calibration is finished, the app shows "Run

finished". Do you remember how long it took each trial? It must have been exactly 525 ms (i.e., 63 bits / 120 Hz).

7. Press on the stop button in MEDUSA Platform.
8. Save the recording file in a safe place, we are going to use it now.
9. Once the calibration is finished, we must train a model to use it in online mode. To do that, open the configuration again by clicking on the gear button.
10. Go down to "Train model", we are going to train the reference processing pipeline for c-VEPs.
    a. Instead of using a single band-pass filter, we are going to use a filter bank composed of 3 infinite impulse response (IIR) filters. To do that, right click inside the band-pass IIRs table and click on "Add row" twice.
    b. Specify three filters between 1-60 Hz, 12-60 Hz, and 30-60 Hz. This configuration has been demonstrated to work well when using a monitor rate of 120 Hz (as we are doing now).
    c. Uncheck the "Reject noisy calibration epochs". This will deactivate the automatic rejection algorithm over noisy epochs, as we are not interested anymore when using a fake LSL signal.



11. Click on "Train model" and select the previously saved file.
12. After training, save the model in a safe place.

13. Click on “Run settings” again and see how the “C-VEP model” path has been automatically updated with our model. We just need to change the mode to “Online” and press on “Done”.
14. In online mode, the BCI system must use the model to decode the epochs in real-time to predict where the user is looking at. Press the power button in MEDUSA Platform to launch the app in online mode.
15. Now, the paradigm is displaying all the possible commands, which will flash according to the shifted versions of our m-sequence. When the “Waiting for start…” message appears, you can click on the play button to start the flashings.

16. See how the different commands are being randomly selected (we are not wearing any EEG cap!). You can stop the flashings whenever you want by clicking on the stop button in MEDUSA Platform, and save the file afterwards.

Note: in a real application, we probably do not want to wait 10 cycles to send a selection, how would you modify the configuration (in online mode), to wait only for 3 cycles?

Task 5.3: [Google Colab] Processing c-VEP signals

17. We are going to learn how to process c-VEP signals, follow the instructions of the jupyter notebook:

https://colab.research.google.com/github/medusabci/medusa-tutorials/blob/master/cvep_spellers_tut.ipynb

Task 5.4: Other apps: “P-ary c-VEP Speller”, “c-VEP Keyboard”

18. We have other applications that use c-VEP control signals. Feel free to launch them and check the differences between them!

Module 6: MI-based paradigms

Task 6.1: Configure the "Motor Imagery" app

1. Select the Motor Imagery app and click on the gear button to access its settings.
2. We are going to modify the time settings: Change the Trials to 4.
3. Press Ok to close the window with these changes saved.

Task 6.2: Run the paradigm: calibration and feedback mode

1. Launch the App, wait for it to load, and start the experiment.
2. When the experiment finishes save the recording with the default name.
3. Now we need to access the settings of the MI app. Here you will load the previous file by clicking on browse files and select our previous recording.
4. With the parameters selected on the "Model settings" window we will train a classifier by clicking on "Train model". We will save the model with the name "workshop_csp_model". This model will be automatically loaded in the "Load classifier model" window.
5. We will now go to the "Scenario settings" and change the "Provide feedback" option to "Real time". Press Ok to save the changes and launch the app.

Task 6.3: [Google Colab] Processing Motor Imagery signals

Follow the instructions of the jupyter notebook:
https://colab.research.google.com/github/medusabci/medusa-tutorials/blob/master/motor_imagery_analysis_tut.ipynb